Extracting Information from Informal Communication

by

Jason D. M. Rennie

B.S. Computer Science
Carnegie Mellon University, 1999
S.M. Electrical Engineering and Computer Science
Massachusetts Institute of Technology, 2001

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

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2007

Author .....
Department of Electrical Engineering and Computer Science
January 23, 2007

Certified by .....
Tommi Jaakkola
Associate Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

Extracting Information from Informal Communication

by

Jason D. M. Rennie

Submitted to
the Department of Electrical Engineering and Computer Science
January 23, 2007
in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Abstract

This thesis focuses on the problem of extracting information from informal communication. Textual informal communication, such as e-mail, bulletin boards and blogs, has become a vast information resource. However, such information is poorly organized and difficult for a computer to understand due to lack of editing and structure. Thus, techniques which work well for formal text, such as newspaper articles, may be considered insufficient on informal text. One focus of ours is to attempt to advance the state-of-the-art for sub-problems of the information extraction task. We make contributions to the problems of named entity extraction, co-reference resolution and context tracking. We channel our efforts toward methods which are particularly applicable to informal communication. We also consider a type of information which is somewhat unique to informal communication: preferences and opinions. Individuals often expression their opinions on products and services in such communication. Others' may read these "reviews" to try to predict their own experiences. However, humans do a poor job of aggregating and generalizing large sets of data. We develop techniques that can perform the job of predicting unobserved opinions. We address both the single-user case where information about the items is known, and the multi-user case where we can generalize opinions without external information. Experiments on large-scale rating data sets validate our approach.

Thesis Supervisor: Tommi Jaakkola
Title: Associate Professor of Electrical Engineering and Computer Science

Contents

Acknowledgements

I owe a great deal of gratitude to a number of people, without whom I may never have been able to make it to MIT and complete the PhD program.

First, I'd like to thank my parents, Charles & Louise Rennie for nurturing me, teaching me the values of respect and hard work and giving me the tools to succeed. Mom, I will never forget the fact that you gave up your career in order to raise my brother and me, that you gave a part of yourself so that we could be raised in a caring, loving and safe environment. Dad, I will never forget how hard you worked to provide for us and to provide me with the opportunities to learn and succeed. Mom & Dad, I could have never made it this far without all that you gave to me. For that, I am eternally grateful.

I'd like to thank my brother, Kevin, for sharing his life with me, the celebrations and sorrows, for providing that intimate human bond that is so scarce and so hard to develop. For giving me a piece of my humanity, I thank you, Kevin.

Arkadiy, Tonya, and Leo, I'd like to thank you for allowing me to be part of your family, for sharing your love with me and introducing me to some wonderful traditions. With my thesis finished, we can spend even more time arguing over politics and religion!

I'd like to thank Tommi, my advisor, for taking me in as a naïve first-year grad student, mentoring me, gently guiding me and teaching me everything I know about machine learning. You gave me the confidence and knowledge I needed to turn my dreams into reality.

I'd like to thank Nati Srebro for teaching me about his work on matrix factorization and using the trace norm for supervised learning, and for allowing me to take part in that work. Without Nati's guidance and teachings, my chapter on preference learning would not be what it is today.

I'd like to thank John Barnett, my officemate and friend, for numerous discussions that helped shape what this work is today. I owe him a particular deal of gratitude for helping me in my quest to evaluate the trace norm distribution normalization constant, $\int \exp(-\lambda \|X\|_\Sigma) dX$. I'd also like to thank Prof. Alan Edelman for writing the class notes which allowed us to eventually complete the quest.

I'd like to thank my committee members, Professors Michael Collins and Tomaso Poggio for guidance during the thesis writing process. Prof. Collins, thank you for taking the time to provide me detailed comments on the entire thesis draft. Prof. Poggio, thank you for providing me the opportunity to give guest lectures in your Learning Theory class. Those lectures helped give me the confidence and practice I needed to deliver my thesis defense presentation.

I'd like to thank Prof. Tom Mitchell for taking a risk on me as a freshman undergrad to help develop homework assignments for his Machine Learning class. That summer of 1996 was when I learned about Decision Trees and Naive Bayes. It was then that I had my first real taste of machine learning and I've felt that it has been an inspiration for me ever since.

I'd like to thank the Cora group, Kamal Nigam, Kristie Seymore, and, especially Andrew McCallum, who involved me in my first research project and helped me develop the ideas that led to my first published work.

I'd like to thank everyone that I've known through the years at CSAIL and the AI Lab, including Harald Steck, Chen-Hsiang Yeang, Brooke Cowan, Greg Marton, David Sontag, Claire Monteleoni, Martin Szummer, Tony Jebara, Annie Lawthers, Luis-Perez Breva, Bryan Russel, Lilla Zollei, Polina Golland, Chris Stauffer, Kinh Tieu, Mihai Badoiu, Piotr Indyk, Amir Globerson, Kai Shih, Yu-Han Chang, Jaime Teevan, Mike Oltmans,

Jacob Eisenstein, Ryan Rifkin, Ali Mohammad, Harr Chen, Rodney Daughtry and everyone whose names I am momentarily forgetting. Thank you for sharing a part of your life with me. These were wonderful, at times trying, yet ultimately rewarding 7.5 years of my life and I will never forget them.

I'd like to thank Mark Stehlik for helping to guide me through my undergraduate education, for being one of the most amazing people I have ever met and for being so kind as to marry my wife and I. We couldn't have asked for a better person to lead us into our lives as adults.

Lastly, I'd like to thank the person who's life has most touched me during these years that I've been working toward my degree. When I embarked on my graduate studies, we were just beginning our lives together. I didn't realize how challenging the Ph.D. would be. When I felt inadequate, you reminded me of my achievements. When I needed to get away, you took me to places near and far. When I felt uninspired, you helped remind me of applications and ideas which I had forgotton. When I felt that I couldn't organize and plan my own work, you acted as my manager, helping me set deadlines and work through the parts that I didn't want to do. Without you, this thesis would have never been born. This is for you, Helen, my love.

Chapter 1

Introduction

This thesis began as an investigation of ways of extracting information from informal communication. By "informal communication", we mean unedited textual communication which has at most minimal formal structure. Some examples include e-mail, mailing lists, bulletin boards, blogs and instant messages. We think this kind of communication is interesting for two reasons: (1) the volume of it which is available, and (2) the quickness with which it can reflect current events or new information. With the advent of the Internet and digital communications networks, text has become increasingly popular as a means of communication. As such, textual communication has become less formal and the range of information available in such communication has become extremely broad. Another trend has been the increased ease with which individuals can produce textual communication. Whereas access to the Internet was once quite limited and required specialized knowledge, now individuals can publish content via a web browser or even a cellular phone. The result of this trend is that informal communication can provide specialized, up-to-date information beyond that of more formal sources of communication (such as newspapers, guides and "portal" web sites). However, informal communication has obvious drawbacks: it is poorly organized and more difficult to understand than structured or well-edited text. So, locating and extracting information in which we might be interested can be quite difficult. Yet, the volume and timeliness of such information suggests to us that the extra effort may be worthwhile.

One possible approach to this task of extracting information from informal communication is to construct a system which at least partially solves the problem. This would require substantial domain-specific knowledge and engineering effort. We feel that before such a system will be valuable, advancements need to be made on the core information extraction problems. So, instead of building a system, we focus our efforts on tackling what we feel are key sub-problems of the information extraction task. For each sub-problem, we identify a deficiency in existing techniques' ability to handle informal communication and provide an alternate approach or attempt to advance the state-of-the art. We feel that this will allow us to make lasting contributions to this problem.

We consider four sub-problems, devoting a chapter to each. Our depth of analysis and contribution varies, from a relatively small extension of an existing co-reference resolution algorithm (Chapter 3), to development of and large-scale evaluation of a new framework for learning user preferences (Chapter 5). We use restaurant discussion boards as a running example. Restaurant discussion boards exhibit many of the benefits and drawbacks that we find in general informal communication. They provide a wealth of up-to-date information

on local restaurants. But, they can be so unwieldy and free-form so as to make finding information difficult.

We begin with one of the most fundamental problems in information extraction: named entity extraction. People, places, locations, organizations, etc. almost always play a role in information that we want to extract. One type of information we find on restaurant discussion boards is opinions about specific restaurants, e.g. "French Laundry was outstanding." Before we can extract the opinion, it is helpful, if not necessary, to identify "French Laundry" as the name of a (restaurant) entity. Identification of such names is the first step toward being able to extract information about them. An entity (particularly after it has been mentioned), may be referred to in many different ways—with the full name, using an abbreviation, or via a pronoun or descriptive phrase. To be able to extract all of these information about an entity, we must be able to resolve all these various mentions—we must perform "co-reference resolution", which is the association of multiple mentions of the same entity. If we can solve these two tasks, named entity extraction and co-reference resolution, we will be able to identify and resolve all explicit entity mentions. However, sometimes information is provided indirectly, without explicit mention of an entity. For example, in reviewing a restaurant, someone might say, "The swordfish was excellent," which is a comment on the food served at a particular restaurant. Association of this comment with the restaurant requires that we be able to track context. We must be able to follow the "topic" of conversation. A final sub-problem that we address involves adding value to the morsels of information that we extract from text. Whereas formal communication tends to focus on factual information, informal communication often is filled with expressions of opinions and preferences. For example, restaurant boards are typically filled with user reviews of restaurants. Individually, such reviews and opinions are of limited value. But, collectively, they can be used to characterize differences between restaurants (or other named entities) and may also be used to predict unobserved opinions—whether an individual will like a restaurant she hasn't experienced yet.

1.1 Chapter Summaries

Most work on named entity extraction has focused on formal (e.g. newspaper) text. As such, systems tend to rely heavily on titles ("Mr."), keywords ("Inc."), capitalization and punctuation. However, capitalization and punctuation are noisy in informal communication. And, titles and keywords are used relatively rarely in informal communication, if they are used at all. Some named entity types (e.g. restaurant names) do not have keywords or titles. One aspect of names that is not fully utilized is that they are often involved in the "topic" of discussion. As such, words in names are often distributed like topic-oriented or informative words. If we can characterize the distribution of topic-oriented words, then we can use this as an additional feature for extracting named entities. Our contribution is exactly that: a new score which estimates a word's "informativeness" or "topic-orientedness". The score captures two aspects which we believe to be typical of the distribution of topic-oriented words: modality and rareness. Experiments indicate that our score can be used to improve NEE performance. Details can be found in Chapter 2.

Two categories of co-reference resolution algorithms have emerged. The first treats each noun phrase mention as referring to a single other mention; learning involves training a classifier to identify the antecedent for each noun phrase mention. The other framework treats co-reference resolution as a clustering problem; mentions are grouped together which

have high average "similarity". We view neither approach as being the answer. The classification approach treats each mention as referring to exactly one other mention (if any). Pronouns and other non-proper nouns do typically refer to other mentions in this way, but reference for proper nouns is less constrained. The classification approach also has the disadvantage of being greedy, making locally optimal reference decisions. The clustering approach requires that each mention in a cluster have (relatively) high average "similarity" with the other mentions in that cluster. This reflects how proper nouns tend to co-refer (string similarity), but is less appropriate for other nouns, which heavily depend on context and locality. Our contribution is a new co-reference resolution algorithm which is a hybrid of these two approaches. We extend a probabilistic clustering-style algorithm to utilize the clustering approach for proper nouns and a classification approach for other nouns. Details can be found in Chapter 3.

Topic or context changes in formal or structured text are often indicated by formatting or markup. Not so with informal communication, where word meaning may be the only clue that context has changed. Thus, we treat the problem of tracking context in informal communication as a sentence clustering problem. One theory for how text is generated within a topic is that it corresponds to a low-dimensional subspace of the probability simplex. Neither of the two popular clustering frameworks, mean/centroid and spectral/normalized cut, can discover low-dimensional subspaces in noisy data. Our contribution is the development of a new clustering framework which simultaneously identifies cluster assignments and the subspace of variation corresponding to each cluster. Details can be found in Chapter 4.

We address two related preference learning problems. Individuals typically express their opinions as partial orderings or ratings. Yet, we think that limited attention has been paid to algorithms which learn ordered categories. First, we consider the problem where a single user rates items and feature information is available on each of the items which might be rated. This is known as ordinal regression, which is an generalization of binary classification to multiple, ordered classes. We introduce a loss function which extends large margin classification theory: our loss function bounds the ordinal classification error. Our contribution is a set of experiments which show that it greatly outperforms other loss functions used for ordinal regression. Details can be found in Section 5.2. The second problem we address is when we have multiple users, but no information about the items. This is known as collaborative filtering. Most approaches to this problem have utilized a rank constraint to force the model to uncover similarities between users and items. However, a rank constraint yields poor solutions due to local minima which are introduced. We instead utilize a soft trace norm penalty for regularization which encourages a low-rank solution without the creation of local minima. We contribute a new way to optimize the trace norm which allows us to scale the framework to large collaborative filtering problems. Experiments on two large collaborative filtering data sets validate our approach. Finally, we show how to extend our preference learning framework in various ways. Details can be found in Section 5.3.

Chapter 2

Named Entity Extraction

2.1 Introduction

We are interested in the problem of extracting information from informal, written communication. The web is filled with information, but even more information is available in the informal communications people send and receive on a day-to-day basis. We call this communication informal because structure is not explicit and the writing is not fully grammatical. Web pages are highly structured—they use links, headers and tags to mark-up the text and identify important pieces of information. Newspaper text is harder to deal with. Gone is the computer-readable structure. But, newspaper articles have proper grammar with correct punctuation and capitalization; part-of-speech taggers show high accuracy on newspaper text. In informal communication, even these basic cues are noisy—grammar rules are bent, capitalization may be ignored or used haphazardly and punctuation use is creative. There is good reason why little work has been done on this topic: the problem is challenging and data can be difficult to attain due to privacy issues. Yet, the volume of informal communication that exists makes us believe that trying to chip away at the information extraction problem is a useful endeavor.

Restaurants are one subject where informal communication is highly valuable. Much information about restaurants can be found on the web and in newspaper articles. Zagat's publishes restaurant guides. Restaurants are also discussed on mailing lists and bulletin boards. When a new restaurant opens, it often takes weeks, or months before reviews are published on the web or in the newspaper (Zagat's guides take even longer). However, restaurant bulletin boards contain information about new restaurants almost immediately after they open (sometimes even before they open). They are also "up" on major changes: a temporary closure, new management, better service or a drop in food quality. This information is difficult to find elsewhere.

Here, we address the first step in extracting information about restaurants (or other entities) from informal communication. Before we can identify and extract facts about restaurants, we must first be able to identify and resolve restaurant names. In this chapter, we address the problem of identifying restaurant names, which is a specialization of the more general problem, "named entity extraction" (NEE).

Systems for performing named entity extraction (NEE) are commonly evaluated on newspaper and newswire articles. As such, they typically rely heavily on capitalization and punctuation information, as well as certain keywords, such as "Company" and "Mr.", which are used almost exclusively for organization and people names (Bikel et al., 1999).

In informal communication, such information is often unavailable or noisy. Capitalization and punctuation are inconsistent and noisy. And, there is no keyword which consistently identifies restaurant names. So, features which are effective for extracting organization and people names from newspaper text will have limited value in extracting (e.g.) restaurant names from informal communication.

In this chapter, we develop a new "feature" which will aid us in identifying restaurant names in informal communication. One aspect of names which is not typically exploited is the fact that each name occurs rarely, but is usually mentioned multiple times when it does occur. This fits a pattern of occurrence which is associated with topic-oriented or "informative" words. Though not all words which fit the topic-oriented occurrence pattern are names, knowledge of whether a word fits the topic-oriented occurrence pattern increases our certainty as to whether a word is (part of) a name. This information is especially valuable for the case where traditional NEE features are insufficient for making the determination. In order to utilize the occurrence distribution for a word, we develop a real-valued "score" which quantifies the degree to which the occurrence pattern fits a topic-oriented one. This score is then provided to the NEE system as an additional feature for each word token.

It is well known that topic-oriented/informative words have "peaked" or "heavy-tailed" frequency distributions (Church & Gale, 1995b). Many informativeness scores have been introduced, including Inverse Document Frequency (IDF) (Jones, 1973), Residual IDF (Church & Gale, 1995a), x^{I} (Bookstein & Swanson, 1974), the z-measure (Harter, 1975) and Gain (Papineni, 2001). Only x^{I} makes direct use of the fit of a word's frequency statistics to a peaked/heavy-tailed distribution. However, x^{I} does a poor job of finding informative words. We introduce a new informativeness score that is based on the fit of a word's frequency statistics to a mixture of 2 Unigram distributions. We find that it is effective at identifying topic-centric words. We also find that it combines well with IDF. Our combined IDF/Mixture score is highly effective at identifying informative words. In our restaurant extraction task, only one other informativeness score, Residual IDF, is competitive. Using Residual IDF or our combined IDF/Mixture score, our ability to identify restaurant names is significantly better than using capitalization, punctuation and part-of-speech information alone. In more formal or structured settings, informativeness may be of marginal use, but here we find it to be of great value.

2.2 Informativeness Measures

Inverse document frequency (IDF) is an informativeness score that was originally introduced by Jones (Jones, 1973). It embodies the principle that the more rare a word is, the greater the chance it is relevant to those documents in which it appears. Specifically, the IDF score for a word, w, is

$$\mathrm{IDF} = -\log \frac{(\text{docs with } w)}{(\text{total \# docs})}. \tag{2.1}$$

The IDF score has long been used to weight words for information retrieval. It has also been used with success in text classification (Joachims, 1998; Rennie et al., 2003). Recently, Papineni (Papineni, 2001) showed that the IDF score can be derived as the optimal classification weight for a special self-classification problem using an exponential model and a generalized form of likelihood. In short, IDF has seen much success and has theoretical justification. However, it is a weak identifier of informative words.

Since the introduction of IDF, many other informativeness scores have been introduced. Bookstein and Swanson (Bookstein & Swanson, 1974) introduce the x^I measure for a word w,

$$x^I(w) = f_w - d_w, \tag{2.2}$$

where f_w is the frequency of word w and d_w is the document frequency of word w (number of documents in which w occurs). Informative words tend to exhibit "peaked" distributions with most occurrences coming in a handful of documents. This score makes sense at the intuitive level since, for two words with the same frequency, the one that is more concentrated will have the higher score. However, this score has a bias toward frequent words, which tend to be less informative.

Harter (Harter, 1975) noted that frequency statistics of informative or "specialty" words tend to fit poorly to a Poisson distribution. He suggested that informative words may be identified by observing their fit to a mixture of 2 Poissons ("2-Poisson") model; he introduced the z-measure as a criterion for identifying informative words. The z-measure, introduced earlier by (Brookes, 1968), is a general measure between two distributions. It computes the difference between means divided by square-root of the summed variances:

$$z = \frac{\mu_1 - \mu_2}{\sqrt{\sigma_1^2 + \sigma_2^2}}, \tag{2.3}$$

where μ_1 and μ_2 are the means of the two distributions and σ_1^2 and σ_2^2 are their variances. Harter found that this measure could be used to identify informative words for keyword indexing.

Twenty years later, Church and Gale (Church & Gale, 1995a) noted that nearly all words have IDF scores that are larger than what one would expect according to an independence-based model (such as the Poisson). They note that interesting or informative words tend to have the largest deviations from what would be expected. They introduced the Residual IDF score, which is the difference between the observed IDF and the IDF that would be expected:

$$\text{Residual IDF} = \text{IDF} - \widehat{\text{IDF}}. \tag{2.4}$$

The intuition for this measure is similar to that of Bookstein and Swanson's x^I-measure. Words that are clustered in few documents will tend to have higher Residual IDF scores. However, whereas x^I has a bias toward high-frequency words, Residual IDF has the potential to be largest for medium-frequency words. As such, it serves as a better informativeness score. In our experiments, we find that Residual IDF is the most effective individual informativeness score.

Recently, Papineni (Papineni, 2001) showed that IDF is the "optimal weight of a word with respect to minimization of a Kullback-Lieber distance." He notes that the weight (IDF) is different from the importance or "gain" of a feature. He suggests that the gain in likelihood attained by introducing a feature can be used to identify "important" or informative words. He derives the gain for a word w as

$$\text{Gain}(w) = \frac{d_w}{D}\left(\frac{d_w}{D} - 1 - \log\frac{d_w}{D}\right), \tag{2.5}$$

where d_w is the document frequency of word w and D is the total number of documents. Extremely rare and extremely common words have low gain. Medium-frequency words have higher gain. A weakness of this measure is that it relies solely on document frequency—it does not take account for "peaked-ness" of a word's frequency distribution.

These informativeness measures represent a variety of approaches to identifying informative words. Only Harter's z-measure directly makes use of how a word's frequency statistics fit a heavy-tailed mixture distribution. Yet, our study indicates that the z-measure is a poor identifier of informative words. In the next section, we introduce a new measure based on a word's fit to a mixture distribution.

2.3 Mixture Models

It is a given that topic-centric words are somewhat rare. But we think that they also exhibit two modes of operation: (1) a high frequency mode, when the document is relevant to the word, and (2) a low (or zero) frequency mode, when the document is irrelevant. A simple unigram model captures this behavior poorly since it places nearly all of its probability mass close to the expected value. In contrast, a mixture of two unigrams has two modes, one for each of the two unigram models. Thus, a word that occurs rarely in some documents and frequently in others can be modeled accurately with a mixture of unigrams, whereas a simple unigram would model such a word poorly. We can evaluate how well a model fits data by looking at the likelihood after we fit parameters. The ratio of likelihoods gives us a suitable measure for comparing the two models.

Consider the modeling of a single vocabulary word. At each position in a document, there are only two possibilities: occurrence, and absence. The unigram model draws a word at each position independently of all other positions, so the marginal likelihood of a single word (under the unigram model) is a binomial:

$$p_{\text{uni}}(\vec{h}|\vec{n},\theta) = \prod_i \theta^{h_i}(1-\theta)^{(n_i-h_i)}, \tag{2.6}$$

where h_i is the number of occurrences of the word, n_i is the length of the document, and θ is the chance of occurrence. In other words, the unigram treats each word like a biased coin where θ is the chance of heads—a "head" represents an occurrence, and a "tail" represents an absence. Consider the following four short "documents":

$$\{\{HHH\},\{TTT\},\{HHH\},\{TTT\}\}$$

A unigram/binomial model fits this data extremely poorly. The maximum likelihood parameter is $\bar{\theta} = 0.5$, which yields a likelihood of 2^{-12}. Though each "word" occurs half the time overall, there is a switching nature to the data—either the word occurs throughout the document, or it never occurs. This data is better modeled by a two-step process—one which randomly selects between two unigrams/binomials to model the document. This is known as a mixture. The likelihood of a mixture of two binomials is:

$$p_{\text{mix}}(\vec{h}|\vec{n},\lambda,\phi_1,\phi_2) = \prod_i \left(\lambda\phi_1^{h_i}(1-\phi_1)^{(n_i-h_i)} + (1-\lambda)\phi_2^{h_i}(1-\phi_2)^{(n_i-h_i)}\right). \tag{2.7}$$

Here, the maximum likelihood parameters are $\lambda = 0.5$, $\phi_1 = 1$, $\phi_2 = 0$ and the data likelihood is 2^{-4}. In effect, the mixture model makes 4 binary decisions (one per document)

whereas the unigram makes 12 binary decisions (one per word). The additional parameters of the mixture allow for a improved modeling of the data. When data exhibits two distinct modes of behavior, such as with our coin example, the mixture will yield a much higher data likelihood than the simple unigram.

Now we are ready to introduce our new informativeness score. For each word, we find maximum-likelihood parameters for both the unigram and mixture models. Our "Mixture score" is then the log-odds of the two likelihoods:

$$s_{\text{mix}} = \log \frac{p_{\text{mix}}(\vec{h}|\vec{n}, \bar{\lambda}, \bar{\phi}_1, \bar{\phi}_2)}{p_{\text{uni}}(\vec{h}|\vec{n}, \bar{\theta})}. \tag{2.8}$$

We use a ratio because we are interested in knowing the comparative improvement of the mixture model over the simple unigram. And, the log-odds ratio grounds the score at zero. The mixture is strictly more expressive than the simple unigram, so the score will be non-negative.

2.4 Expectation-Maximization

We use Expectation-Maximization (EM) to maximize the likelihood of the mixture model. A detailed understanding of EM is not essential for appreciation of this work, so we avoid it here. We refer the reader to Appendix A and Dempster et al. (1977) for more information. EM uses a bound to iteratively update model parameters to increase likelihood. Since likelihood as a function of mixture parameters is not convex, the maximum EM finds may be local. To increase our chances of finding a global maximum, we use two starting points: (1) one slightly offset from the unigram model, and (2) one "split" model where the first unigram component is set to zero and the second component and the mixing parameter (λ) are set to otherwise maximize the likelihood of the data. We found that this worked well to find global maxima—extensive random sampling never found a higher likelihood parameter setting.

EM can be viewed as a particular optimization algorithm. Compared to general-purpose gradient descent algorithms, such as Conjugate Gradient (see (Shewchuk, 1994) for a tutorial), EM tends to be slow and inefficient. It served our purpose for the experiments presented later in this chapter, but may not be appropriate for work involving larger data sets. Appendix B provides the necessary gradient calculations for the mixture of two binomials model used in our work.

2.5 Finding Restaurants

We think that the Mixture score can serve as an effective term informativeness score. To evaluate the correctness of our belief, we use the task of identifying restaurant names in posts to a restaurant discussion bulletin board. We treat each thread as a document and calculate various informativeness scores using word-thread statistics. Restaurants are often the topic of discussion and tend to be highly informative words. The task of identifying them serves as a good test ground for any measure that claims to rate informativeness. We collected posts from the board and hand-labeled them. The next section details our findings.

Token	Score	Rest.
sichuan	99.62	31/52
fish	50.59	7/73
was	48.79	0/483
speed	44.69	16/19
tacos	43.77	4/19
indian	41.38	3/30
sour	40.93	0/31
villa	40.36	10/11
tokyo	39.27	7/11
greek	38.15	0/20

Token	Score	Rest.
sichuan	2.67	31/52
ribs	2.52	0/13
villa	2.36	10/11
tokyo	2.36	7/11
penang	2.17	7/9
kuay	1.92	0/7
br	1.92	6/7
khao	1.92	4/7
bombay	1.92	6/7
strike	1.77	0/6

Table 2.1: Top Mixture Score (left) and Residual IDF (right) Tokens. Bold-face words occurred at least once as part of a restaurant name.

2.5.1 The Restaurant Data

We used as a test-bed posts from a popular restaurant bulletin board. The maintainers of the site moderate the board and lay out a set of ground rules for posting. The people who post are not professional restaurant reviewers. They simply enjoy eating and discussing what they have eaten. Information about restaurants can be found in the discussions that ensue. Major metropolitan areas each have their own bulletin board; other boards are grouped by region.

We collected and labeled six sets of threads of approximately 100 posts each from a single board (615 posts total). We used Adwait Ratnaparkhi's MXPOST and MXTERMINATOR[1] software to determine sentence boundaries, tokenize the text and determine part-of-speech. We then hand-labeled each token as being part of a restaurant name or not. Unlike a more typical NEE annotation, we did not explicitly identify start/end tokens, so if two restaurant names appeared in sequence without any separating punctuation or words, the labels did not provide any distinction between the two names. Labeling of the 56,018 tokens took one person about five hours of time. 1,968 tokens were labeled as (part of) a restaurant name. The number of restaurant tokens per set ranged from 283 to 436. We found 5,956 unique tokens. Of those, 325 were used at least once as part of a restaurant name. We used a separate set of data for developing and debugging our experiment code.

2.6 Informativeness Filtering

Here we explore how the various measures serve as informativeness filters. First, we consider the density of restaurant tokens in the top-ranked words. Both Gain and IDF serve as poor informativeness filters, at least with respect to restaurant names—only occasional restaurant tokens are found in words ranked highest by Gain and IDF. The x^I-measure ranks some restaurant tokens highly—five of the top 10 words occur at least once as part of a restaurant. However, these tokens appear as a part of a restaurant name somewhat rarely. None of the top 30 x^I words occur as part of restaurant names at least 50% of the time. The z-measure serves as a reasonably good informativeness filter—three of the top 10 words occur as restaurant tokens and nine of the top 30 words occur in restaurant names

[1] http://www.cis.upenn.edu/~adwait/statnlp.html

Rank	Token	Rest.
1	sichuan	31/52
4	speed	16/19
8	villa	10/11
9	tokyo	7/11
21	zoe	10/11
22	penang	7/9
23	pearl	11/13
26	dhaba	8/13
29	gourmet	23/27
30	atasca	9/10

Rank	Token	Rest.
1	sichuan	31/52
3	villa	10/11
4	tokyo	7/11
5	penang	7/9
7	br	6/7
8	khao	4/7
9	bombay	6/7
12	aroma	5/6
14	baja	3/6
16	mahal	5/6

Table 2.2: Top Mixture Score (left) and Residual IDF (right) Restaurant Tokens (50%+ restaurant usage)

at least 50% of the time. Both the mixture score and Residual IDF have high densities of restaurant tokens in their top ranks. Table 2.1 shows the top 10 words ranked by the Mixture score and Residual IDF. For both measures, seven of the top 10 words are used at least once as part of a restaurant name. Table 2.2 shows, for each measure, the top 10 words used a majority of the time in restaurant names. Most of the top-ranked Residual IDF words occur a majority of the time in restaurant names. Fewer top Mixture score words are majority used in restaurant names, but those that are occur more often than the top Residual IDF words. Top-ranked words give only a partial view of the effectiveness of an informativeness filter. Next, we look at average and median scores and ranks across our entire corpus.

Score	Avg. Rank	Med. Rank
Mixture	**505**	**202**
z	526	300
x^I	563	326
RIDF	858	636
Gain	2461	1527
Baseline	2978	2978
IDF	4562	5014

Table 2.3: Average and Median Restaurant Token Ranks (lower is better)

So far, we have considered the upper-tail of informativeness scores; we have done our counting over unique words, thus overweighting rare ones. Here, we compile statistics across the full set of data and count each restaurant token occurrence separately. For each informativeness score, we compute the score for each unique word and rank the words according to score. Then, for each of the 1,968 tokens labeled as (part of) a restaurant name, we determine the token's rank. We compute both the average and median ranks of the restaurant tokens. Table 2.3 gives the average and median ranks of restaurant words for the various informativeness scores. The Mixture score gives the best average and median rank. The z-measure and x^I-measure give slightly worse rankings. Residual IDF and Gain

are better than the baseline[2], while IDF yields worse rankings than the baseline. The average and median rank give us a good feel for how well a score works as a filter, but not necessarily as a feature in a natural language system. Next, we discuss an evaluation that may better reflect performance on a real task.

Score	Avg. Score	Med. Score
RIDF	**3.21**	2.72
IDF	1.92	**3.90**
Mixture	2.49	2.89
z	1.10	1.09
Gain	1.00	1.22
Baseline	1.00	1.00
x^I	0.33	0.00

Table 2.4: Average and Median Relative Scores of Restaurant Tokens

Now we consider the average and median *score* of restaurant tokens. For each of the 1,968 tokens labeled as (part of) a restaurant name, we compute the informativeness score. We then take an average or median of those scores. We divide by the average or median score across all 56,018 tokens to attain a "relative" score. We do this so that absolute magnitude of the informativeness score is irrelevant; i.e. multiplication by a constant has no effect. Table 2.4 shows average and median relative scores for restaurant tokens. Of note is the fact that informativeness scores that produce good average/median ranks do not necessarily produce good average/median scores (e.g. z and x^I). Residual IDF gives the best average relative score; IDF gives the best median relative score. The Mixture score gives the second-best average relative score and second-best median relative score.

Only IDF, Residual IDF (RIDF) and the Mixture score appear to provide unique value for identifying informative words. The other scores, the z-measure, the x^I measure and Gain do not appear to provide value beyond what we can achieve with IDF, RIDF and Mixture. x^I and the z-measure share similarities to the Mixture score, but appear strictly less effective. Gain seems severely handicapped by the fact that it only uses document frequency information. For the remainder of this work, we focus on the three scores that show some uniqueness in their ability to identify informative words: IDF, Residual IDF and the Mixture score.

2.6.1 Are Mixture and IDF Independent?

To this point, both Residual IDF and the Mixture Score appear to be excellent informativeness scores. Both have a high density of restaurant tokens in their highest ranks; for both measures, average/median ranks/scores are much better than baseline. IDF, however, ranks restaurant words poorly, but yields the best median relative score. Since IDF seems so different from the other two scores, we postulate that it might work well in combination.

We look at how well correlated the scores are. If two scores are highly correlated, there is little use in combining them—their combination will be similar to either score individually. However, if two scores are uncorrelated, then they are measuring different

[2]Baseline average and median rank are what would be expected from a score that assigns values randomly. Note that there are 5,956 unique words; 2,978 is half that number.

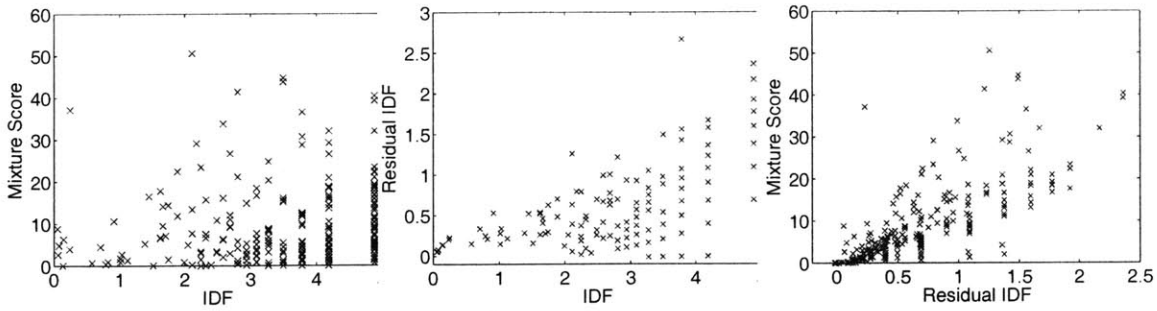

Figure 2-1: Scatter plots comparing pairs of the IDF, Residual IDF and Mixture scores. Only words that appear at least once within a restaurant name are plotted. RIDF/Mixture shows a high degree of correlation. IDF/RIDF shows some correlation. IDF/Mixture shows relatively little correlation.

sorts of information and may produce a score in combination that is better at identifying informative words than either score individually.

First, we consider a very simple test on our restaurant data set: how much overlap is there in highly-rated restaurant words? For each of the scores, we choose a threshold that splits the restaurant words (approximately) in half. We then count the number of restaurant words that score above both thresholds. For scores that are independent of each other, we would expect the joint count to be about half of the individual count. Table 2.5 gives the individual and joint statistics. The Mixture/RIDF and IDF/RIDF combinations both show a substantial degree of dependence. This is not the case for Mixture/IDF. If the Mixture and IDF scores were independent, we would expect a joint count of $176 * 170/325 = 92$, almost exactly the joint count that we do observe, 93. This gives us reason to believe that the Mixture and IDF scores may be uncorrelated and may work well in combination.

Condition	Restaurant
Mixture > 4.0	176/325
IDF > 4.0	170/325
RIDF > 0.5	174/325
Mix > 4.0 and IDF > 4.0	93/325
Mix > 4.0 and RIDF > 0.5	140/325
IDF > 4.0 and RIDF > 0.5	123/325

Table 2.5: Number of restaurant tokens above score thresholds.

Our test provides evidence that the IDF and Mixture scores are independent, but it does not exclude the possibility that there are pockets of high correlation. Next, we consider more traditional measures. Figure 2-1 shows scatter plots of the pairs of scores. Residual IDF (RIDF) and Mixture show a high degree of correlation—knowledge of RIDF is very useful for attempting to predict Mixture score and vice versa. IDF and RIDF show correlation, at least partially reflecting the fact that IDF bounds RIDF. IDF and Mixture show little relation—there is no clear trend in the Mixture score as a function of IDF. These observations are reflected in correlation coefficients calculated on the data, shown in Table 2.6. IDF and Mixture are practically uncorrelated, while the other score pairs show substantial correlation.

That the IDF and the Mixture scores would work well together makes sense intuitively.

Score Names	Correlation Coefficient
IDF/Mixture	-0.0139
IDF/RIDF	0.4113
RIDF/Mixture	0.7380

Table 2.6: Correlation coefficients for pairs of the IDF, Residual IDF and Mixture scores on restaurant words. IDF and Mixture are effectively uncorrelated in the way they score restaurant words.

Token	Score	Restaurant
sichuan	376.97	31/52
villa	197.08	10/11
tokyo	191.72	7/11
ribs	181.57	0/13
speed	156.25	16/19
penang	156.23	7/9
tacos	153.05	4/19
taco	138.38	1/15
zoe	134.23	10/11
festival	127.39	0/14

Table 2.7: Top IDF*Mixture Score Tokens

They capture very different aspects of the way in which we would expect an informative word to behave. IDF captures rareness; the Mixture score captures a multi-modal or topic-centric nature. These are both aspects that partially identify informative words. Next we investigate whether a combination score is effective for identifying informative words.

2.6.2 Combining Mixture and IDF

We use the relaxation of conjunction, a simple product, to combine IDF and Mixture. We denote this by "IDF*Mixture." Table 2.7 shows the top 10 tokens according to the IDF*Mixture score. Eight of the top 10 are used as restaurant names. Worth noting is

Rank	Token	Restaurant
1	sichuan	31/52
2	villa	10/11
3	tokyo	7/11
5	speed	16/19
6	penang	7/9
9	zoe	10/11
12	denise	5/8
16	pearl	11/13
19	khao	4/7
21	atasca	9/10
23	bombay	6/7

Table 2.8: Top IDF*Mixture Score Restaurant Tokens (50%+ restaurant usage)

that the other two words ("ribs" and "festival") were topics of discussions on the restaurant bulletin board. Table 2.8 gives the ranks of the top 10 tokens that were used regularly in restaurant names. Compared to the Mixture score, restaurant tokens more densely populate the upper ranks. Ten of the top 23 tokens are regularly used as restaurant names. The trend continues. 100 of the top 849 IDF*Mixture tokens are regularly used in restaurant names, while 100 of the top 945 Mixture tokens are regularly used in restaurant names. However, Mixture catches up and and surpasses IDF*Mixture (in terms of restaurant density) as we continue down the list. This explains why Mixture has better average and median ranks (next paragraph).

Score	Avg. Rank	Med. Rank
Mixture	**507**	**202**
IDF*Mixture	682	500
RIDF	858	636
IDF	4562	5014

Table 2.9: Average and Median Restaurant Token Ranks

Score	Avg. Score	Med. Score
IDF*Mixture	7.20	**17.15**
$RIDF^2$	**7.54**	7.40
$Mixture^2$	4.61	8.35
IDF^2	2.31	15.19

Table 2.10: Average and Median Relative Scores of Restaurant Tokens. Note that a superscript indicates that the score is raised to the given power.

Here we give rank and relative score averages for IDF*Mixture. Table 2.9 gives the average and median ranks like before. Mixture still leads, but IDF*Mixture is not far behind. Table 2.10 gives the average and median relative scores. The relative score is affected by scale, so we compare against squared versions of IDF, Mixture and Residual IDF. IDF*Mixture achieves the best median and is a close second for average relative score. IDF*Mixture appears to be a better informativeness score than either IDF or the Mixture score and very competitive with Residual IDF. In the next section, we describe the set-up for a "real" test: a named entity (restaurant name) extraction task.

2.7 Named Entity Detection

So far, we have focused on filtering. In this section, we consider on the task of detecting restaurant names. We use the informativeness scores as features in our classifier and report on how accurately restaurants are labeled on test data.

2.7.1 Performance Measures

The F-measure (van Rijsbergen, 1979), is commonly used to measure performance in problems where negative examples outnumber positive examples. See Table 2.11 for notation. We use the F1-measure ("F1"), which equally weights precision, $p = \frac{tp}{tp+fp}$, and recall,

		classification	
		+1	−1
true	+1	tp	fn
label	−1	fp	tn

Table 2.11: The contingency table for the binary classification problem. 'tp', 'fn', 'fp', and 'tn' are the numbers of true positives, false positives, false negatives and true negatives, respectively.

$r = \frac{\text{tp}}{\text{tp+fn}}$:

$$\text{F1}(p, r) = \frac{2pr}{p + r}. \tag{2.9}$$

F1 varies as we move our classification threshold along the real number line. To eliminate any effects of selecting a particular threshold, we report the maximum F1 score attained over all threshold values. We call this "F1 breakeven" in reference to a similarity it shares with precision-recall (PR) breakeven (Joachims, 1997); the F1 breakeven tends to occur when precision and recall are nearly equal.

We avoid the use of PR breakeven because it is somewhat ill-defined—there may not be any setting for which precision and recall are equal. When the classifier cannot differentiate within a set of examples, it may not be possible to make changes in precision and recall sufficiently small to find a breakeven point. This is not generally an issue in Joachims' domain, text categorization, unless the documents are extremely short. Hence, Joachims simply uses a null value for PR breakeven when a breakeven point cannot be found. Our case is a bit different. We are categorizing individual word tokens. Though we use a number of different features to differentiate the tokens, many tokens have identical feature vectors, which makes it impossible for the classifier to differentiate between them. Thus, it seemed clear that some "hack" would be necessary for us to utilize PR breakeven for evaluation. Instead, we chose to use a metric (F1 breakeven) which is better "behaved".

2.7.2 Significance

Given two classifiers evaluated on the same test sets, we can determine whether one is better than the other using paired differences. We use the Wilcoxon signed rank test (Wilcoxon, 1945); it imposes a minimal assumption—that the difference distribution is symmetric about zero. The Wilcoxon test uses ranks of differences to yield finer-grained distinctions than a simple sign test. We apply the Wilcoxon test by comparing the F1 breakeven of two different classifiers on pairs of random splits of the data.

We use the one-sided upper-tail test, which compares the zero-mean null hypothesis, $H_0 : \theta = 0$, against the hypothesis that the mean is greater than zero, $H_1 : \theta > 0$. We compute a statistic based on difference ranks. Let z_i be the i^{th} difference—the F1 breakeven difference on the i^{th} random split of the data. Let r_i be the rank of $|z_i|$. Let ψ_i be a an indicator for z_i:

$$\psi_i = \begin{cases} 1, & \text{if } z_i \geq 0, \\ 0, & \text{if } z_i < 0. \end{cases} \tag{2.10}$$

The Wilcoxon signed rank statistic is:

$$T^+ = \sum_{i=1}^{n} z_i \psi_i. \tag{2.11}$$

Upper-tail probabilities for the null hypothesis are calculated for each possible value[3]. We reject H_0 (and accept H_1) if the probability mass is sufficiently small. We use $\alpha = 0.05$ as the threshold below which we declare a result to be significant. Table 2.12 gives the upper-tail probabilities for a subset of the possible values of T^+. Values of 19 and higher are significant at the $\alpha = 0.05$ level.

x	$P_0(T^+ \geq x)$
17	.109
18	.078
19	.047
20	.031
21	.016

Table 2.12: Upper-tail probabilities for the null hypothesis.

2.7.3 Experimental Set-Up

We used 6-fold cross-validation for evaluation: for each of the six sets, we used the other five sets as training data for our classifier[4]. No data from the "test" set was ever used to select parameters for the corresponding classifier. However, since the test set for one fold is used in the training set for another, we note that our significance calculations may be overconfident.

For classification, we used a regularized least squares classifier (RLSC) (Rifkin, 2002) and used a base set of features like those used in (Bikel et al., 1999). Current, next and previous parts-of-speech (POS) were used, along with current-next POS pairs and previous-next POS pairs. We included features on the current, previous and next tokens indicating various types of location, capitalization, punctuation and character classes (firstWord, lastWord, initCap, allCaps, capPeriod, lowerCase, noAlpha and alphaNumeric). Unlike HMMs, CRFs or MMM networks, RLSC labels tokens independently (like an SVM does). We believe that using a better classifier would improve overall classification scores, but would not change relative performance ranking.

2.7.4 Experimental Results

Table 2.13 gives the averaged performance measures for six different experimental settings:

- Baseline: base features only
- IDF: base features, IDF score
- Mixture: base features, Mixture score

[3]Values are from Table A.4 of Hollander and Wolfe (1999).

[4]To select a regularization parameter, we trained on four of the five "training" sets, evaluated on the fifth and selected the parameter that gave the best F1 breakeven.

	F1 brkevn
Baseline	55.04%
IDF	55.95%
Mixture	55.95%
RIDF	57.43%
IDF*RIDF	58.50%
IDF*Mixture	**59.30%**

Table 2.13: Named Entity Extraction Performance

	Base.	IDF	Mix	RIDF	IDF*RIDF
IDF	20	-	-	-	-
Mixture	18	15	-	-	-
RIDF	20	19	19	-	-
IDF*RIDF	20	18	18	16	-
IDF*Mixture	21	21	21	21	15

Table 2.14: T^+ Statistic for F1 Breakeven. Each entry that is 19 or higher means that the score to the left is significantly better than the score above. For example, IDF*Mixture is significantly better than RIDF.

- RIDF: base features, Residual IDF
- IDF*RIDF: base features, IDF, RIDF, IDF^2, $RIDF^2$, IDF*RIDF
- IDF*Mixture: base features, IDF, Mixture, IDF^2, $Mixture^2$, IDF*Mixture

These numbers are lower than what is typically seen for named entity extraction for a number of reasons. The primary reason may be the specific task: extracting restaurant names. Restaurant names vary widely. Unlike company and person names, there are no commonly used prefixes (e.g. "Mr.") or suffixes ("Inc.") which easily identify restaurant names. Another reason for the low scores may be the domain—as we have already noted, capitalization and punctuation features are noisy in informal communication and so are not as reliable for identifying names. Finally, named entity extraction systems typically use word tokens as features and predict labels jointly; we do not use word tokens as features and use a classifier which predicts the label for each word token independently of other labels. All of these distinctions may contribute to the atypical error rates.

Table 2.14 gives the Wilcoxon signed rank statistic for pairs of experimental settings. IDF and the Mixture score both yield small improvements over baseline. The improvement for IDF is significant. Residual IDF serves as the best individual informativeness score, yielding a significant, 2.39 percentage-point improvement over baseline and significant improvements over both IDF and Mixture. The IDF*Mixture score yields further improvement, 4.26 percentage-points better than baseline and significantly better than IDF, Mixture and Residual IDF. For completeness, we compare against the IDF*RIDF score (the product of IDF and Residual IDF scores). IDF*Mixture yields the larger average F1 breakeven, but we cannot say that the difference is significant.

These results indicate that the IDF*Mixture product score is an effective informativeness criterion; it is better than Residual IDF and competitive with the IDF*RIDF product score.

The IDF*Mixture product score substantially improves our ability to identify restaurant names in our data.

2.8 Summary

We introduced a new informativeness measure, the Mixture score, and compared it against a number of other informativeness criteria. We conducted a study on identifying restaurant names from posts to a restaurant discussion board. We found the Mixture score to be an effective restaurant word filter. Residual IDF was the only other measure found to be competitive. We found that the Mixture score and IDF identify independent aspects of informativeness. We took the relaxed conjunction (product) of the two scores, IDF*Mixture, and found it to be a more effective filter than either score individually. We conducted experiments on extracting named entities (restaurant names). Residual IDF performed better than either IDF or Mixture individually, but IDF*Mixture out-performed Residual IDF.

Chapter 3

Co-reference Resolution

In the previous chapter, we discussed the task of extracting named entities from informal communication. We argued that traditional approaches for named entity extraction (NEE), which have been highly effective on newspaper text, are insufficient for NEE on the noisy, and (at times) ungrammatical text found in informal communication. We proposed a new measure for identifying informative words, which, when paired with traditional NEE techniques, improved extraction performance. In this chapter, we explore an essential next step in the information extraction process. Once named entities have been identified, we must resolve mentions of the entity. Many different names may be used to refer to the same entity—abbreviations, shortened forms and context-sensitive descriptions, as well as pronouns. To properly extract and organize the information contained in text, we must resolve all of these mentions. This task of resolving named entity mentions is known as co-reference resolution.

Co-reference resolution (CRR) is essentially a clustering problem. The goal is to group together noun phrase mentions which refer to the same entity. However, it is not like clustering documents or images. In a typical clustering problem, a distance measure between objects is given or assumed and the relation between objects is symmetric. Many clustering algorithms take a symmetric distance or similarity matrix as input and do not consider any special structure in the data. However, in co-reference resolution there is not symmetry amongst the mentions. In the case of pronouns and other non-proper noun phrases, the mention is not linked to all co-referent mentions, but rather has a specific antecedent. That is, for such mentions, chain or tree structure links the mention to its entity. Sentence structure, locality and context tend to play a large role in determining linkage. Any CRR algorithm must balance this structure against that of proper noun phrases, which tend to have the symmetric structure usually assumed with clustering algorithms.

Others have addressed these issues, but not to our satisfaction. In particular, it seems that no one has developed a framework for co-reference resolution based on what we see as the hybrid nature of the task. Named entity mentions can be divided into two categories: (1) proper nouns, and (2) non-proper nouns. The proper nouns tend to have a symmetric relationship and fit into the traditional clustering scenario quite well. For proper nouns, the content of the string is usually more important than the locality of the mention. In contrast, non-proper nouns (including pronouns) are strongly dependent on context and location for resolution and tend to point to a specific prior mention, its "antecedent." We think this hybrid nature requires a framework that treats proper and non-proper nouns differently.

The algorithm proposed by McCallum and Wellner (2005) comes close to meeting our

requirements. They learn a similarity measure between mentions, but do not recognize the hybrid nature of the clustering task. Their algorithm performs exceptionally well on proper nouns. But, we feel there is still room for improvement on the overall task. Our contributions in this chapter are (1) an explanation of the hybrid nature of CRR, (2) a new classification-based CRR algorithm which finds an antecedent for each NP mention, and (3) a hybrid CRR algorithm which combines our "antecedent" model with McCallum and Wellner's Model 3 to yield a "hybrid" CRR algorithm which we feel provides the proper balance of viewpoints for the co-reference resolution task.

3.1 Motivation: Why Co-Reference Resolution?

Consider the task of collecting information about entities. For example, we might want to collect opinions on movies, or names of recently opened restaurants. Now, imagine an oracle which can identify short text excerpts which provide the desired information (e.g. "I loved it!" "I ate there yesterday, the day they opened."). But, these excerpts are only informative in the context of the rest of the document. Mentions of entities in which we have an interest may not be specific and may need to be resolved. How an entity is mentioned in text can vary widely. Usually its full name is mentioned at least once. This "full name" mention serves as a connection to the actual entity. But, later mentions may use an abbreviated form, a contextual description or even a pronoun to refer to the entity. When a person reads a document with multiple mentions of an entity, (s)he must link together the mentions using the structure and context of the document. This task of joining mentions that refer to the same entity is known as co-reference resolution. Without it, our access to information is limited, even in the idealistic case described above. Some form of co-reference resolution is necessary whenever the full name of the entity is not used.

3.2 Views of Co-reference Resolution

There are two main views of the co-reference resolution problem. The first view treats CRR as a problem of finding the antecedent for each noun phrase (NP) mention. This view assumes that there is an initial "root" mention, followed by successive mentions, each of which refer to a specific, earlier mention. The referent is known as the "antecedent;" the referring mention is known as the "anaphor." Considering each mention as a node in a graph, the anaphor-antecedent relations for a text, represented as directed edges, form a forest, where each tree corresponds to a single entity. There is one "root" mention/node for each tree, corresponding to the first mention of the entity. This is the only node of a tree without an outward edge; other mentions/nodes have a single outward edge, each pointing to its antecedent. The inference problem is a certain type of classification problem: for each mention except for the first, identify the mention to which it refers. Aone and Bennett (1995) trained a decision tree to perform binary classification where each positive example was an anaphor-antecedent pair. Various features were defined and extracted to help the decision tree identify co-referent pairs. In labeling new data, the antecedent was determined to be the one in which the decision tree had the highest confidence. Soon et al. (2001) and Ng and Cardie (2002) used similar set ups, also using antecedent-anaphor pairs for training and binary decision trees for learning. But, no matter details of the implementation, this "classification" approach is handicapped by the fact that it makes local decisions, and thus cannot deal with certain important situations. McCallum and Wellner (2005) provide an

apt example. Consider the reference chain ("Mr. Powell" → "Powell" → "she"). Locally, it looks reasonable. But, it clearly can't be correct due to the gender mismatch.

The second view of co-reference resolution is as a clustering problem. Each noun phrase mention is a data item and the goal is to cluster together mentions that refer to the same entity. As with the classification approach, a feature vector is created/extracted for each mention pair. But, unlike the classification approach, a distance measure is established based on the feature vector and a global clustering measure is optimized to determine how mentions should be grouped. Cardie and Wagstaff (1999) hand-crafted a distance measure and used a simple, iterative clustering algorithm to group mentions. McCallum and Wellner (2005) took the approach further—they used labeled training data to learn a distance measure and performed clustering by optimizing an objective on the cluster labels. Thus, their procedure was completely automated and could adapt to different languages or writing styles. Experiments conducted by McCallum and Wellner indicated excellent performance for proper noun resolution and performance that compared well to state-of-the-art for the overall resolution task.

In this work, we propose a hybrid model of co-reference resolution which applies the clustering framework to the problem of resolving proper noun mentions, and applies the classification framework to the problem of resolving non-proper noun mentions. This hybrid model utilizes Model 3, as well as a classification-based antecedent model which we have developed. Next, we provide a detailed introduction to CRR and representations of reference that we will use for discussion of the models.

3.3 The Co-Reference Resolution Problem

Given a text document, we assume that a natural language parser (or similar program) can identify the noun phrases (NPs) and tell us whether each is a proper or non-proper NP. We use a vector, $\mathbf{x}$, to designate the set of NPs in a text. To establish a complete ordering on NPs, we first order them according to their starting position in text, then according to their location in a parse tree. Thus, an "outer" NP will always come before an "inner" NP in our ordering. We use x_i to designate the i^{th} NP in the the text.

An entity is a person, place, company, institution, etc. Anything that can be named can be an entity. Each noun phrase serves as a reference to an entity. In some cases (almost exclusively when the NP is proper), the reference is *direct*—no other information from the text is needed to identify the entity. In other cases, the reference is *indirect*—document context and other NPs are necessary to determine the ultimate reference. The primary computational task in resolving each NP mention to its corresponding entity is that of grouping or clustering together co-referent mentions. The final step of linking each cluster to its corresponding entity is often unnecessary, as the set of proper NP mentions of an entity is usually sufficient identification for computational purposes.

There are two ways of representing co-referring NPs, each of which will be used in our later discussion. The first takes a clustering viewpoint, simply assigning an identifier to each mention corresponding to the entity to which it refers. In this case, we use an integer-valued vector to represent references over a set of mentions, $\mathbf{y} \in \{1, \ldots, n\}^n$. Since we do not deal with the entities themselves, the integer values in this representation are arbitrary; customarily, we use the index of the first mention as the cluster identifier. This information can be equivalently represented as the upper-triangular portion of a binary-valued matrix, $Y \in \{0, 1\}^{n \times n}$. Here, entry i, j is "on" ($Y_{ij} = 1$) iff x_i and x_j are co-referent. Note that not

all values are allowable. If mentions x_i and x_j are co-referent ($Y_{ij} = 1$), and mentions x_j and x_k are co-referent (Y_{jk}) are co-referent, then mentions x_i and x_k must also be co-referent ($Y_{ik} = 1$). I.e. three indicator variables forming a ring or triangle cannot sum to 2, i.e. $Y_{ij} + Y_{jk} + Y_{ik} \neq 2$. This representation will be used in the discussion of Model 3 (§ 3.4).

The second representation assumes that each mention either directly refers to the entity or refers to it indirectly, via a chain of mentions. In the case of an indirect reference, the mention is assumed to refer to an earlier mention in the text, known as its antecedent. A mention that refers directly to the entity will be said to have a "null" antecedent. As with the clustering representation, the antecedent information can be represented as the upper-triangular portion of a binary matrix. However, the constraints on, and the meaning of the entries is different. Entry i, j ($i < j$) is "on" ($Y_{ij} = 1$) if the antecedent for x_j is x_i. Each mention has at most one antecedent, so each column-sum of the matrix is bounded by unity. I.e. $\sum_{i \leq j} Y_{ij} \leq 1$. This representation will be used for the discussion of the antecedent model (§ 3.5).

3.4 Model 3

Here we give an introduction to McCallum and Wellner's model for co-reference resolution, "Model 3."

Let $\mathbf{x} = \{x_1, \ldots, x_n\}$ denote a set of mentions. Let $y_{ij} \in \{0, 1\}$, $1 \leq i < j \leq m$, be indicator variables: $y_{ij} = 1$ iff x_i and x_j refer to the same entity, $y_{ij} = 0$ otherwise. The indicator variables are not independent; if $y_{ij} = y_{jk} = 1$ ($i < j < k$), then we must have $y_{ik} = 1$. In other words, the sum of indicator values in a triangle cannot be two: $y_{ij} + y_{jk} + y_{ik} \neq 2$. McCallum and Wellner define a likelihood distribution over the indicator variables,

$$P(\mathbf{y}|\mathbf{x}; \lambda) = \frac{1}{Z(\mathbf{x}, \lambda)} \exp\left(\sum_{i,j,l} \lambda_l f_l(x_i, x_j, y_{ij})\right), \tag{3.1}$$

where the f_l, $l \in \{1, \ldots, m\}$, are feature functions defined on mention pairs and indicator values, λ_l is the weight associated with feature function f_l, and variable settings where $y_{ij} + y_{jk} + y_{ik} = 2$ are not allowed. The partition function is a sum over feasible values of the indicator variables,

$$Z(\mathbf{x}, \lambda) = \sum_{\mathbf{y}|y_{ij}+y_{jk}+y_{ik} \neq 2} \exp\left(\sum_{i,j,l} \lambda_l f_l(x_i, x_j, y_{ij})\right). \tag{3.2}$$

Given labeled training data ($\mathbf{x}$ and $\mathbf{y}$), we learn weights $\{\lambda_l\}$ by maximizing the likelihood, (3.1). Often, likelihood is maximized via EM or gradient ascent. However, in this case, the normalization constant requires a sum over exponentially many terms, so such traditional algorithms are not practical. Instead, McCallum and Wellner use perceptron updates as described by Collins (2002) in order to maximize likelihood.

Given fixed weights (λ) and a set of mentions ($\mathbf{x}$), we learn indicator values (and hence, a clustering of the mentions) again by maximizing the likelihood, (3.1). In this case, the partition function, $Z(\mathbf{x}, \lambda)$, is constant, so we can ignore it for optimization. What remains is a classic multi-way min-cut/max-flow problem, or equivalently, a classic maximum intra-

cluster similarity clustering objective. We discuss this equivalence in detail in the following subsections.

3.4.1 Minimum Cut Equivalence

Likelihood maximization for Model 3 is equivalent to a generalization of the "minimum cut" problem. Given a graph with edge weights and two special nodes, one identified as the "source" and one as the "sink," the minimum cut is the set of edges with smallest summed weight such that any path from source to sink must pass through one of the selected edges. In the inference problem for Model 3, there are no "special" nodes. Rather, the goal is to find the set of edges (the "cut") with minimum summed edge weight which separate the nodes into "clusters." As suggested by its name, if two nodes are separated by an edge in the cut, then any path between the nodes must include at least one edge in the cut. Next, we give the mathematical details for the minimum cut interpretation of Model 3.

If the "cut" is empty, then the log-likelihood of the model is

$$\log P(C = \emptyset|\mathbf{x}) = \sum_{i,j,k} \lambda_k f_k(x_i, x_j, 1) - \log Z(\mathbf{x}, \lambda). \tag{3.3}$$

where C is the set of edges in the "cut." If C is not empty, we get

$$\log P(C|\mathbf{x}) = \sum_{i,j,k} \lambda_k f_k(x_i, x_j, 1) + \sum_{(i,j)\in C} \sum_k \lambda_k \Big[f_k(x_i, x_j, 0) - f_k(x_i, x_j, 1) \Big] - \log Z(\mathbf{x}, \lambda). \tag{3.4}$$

The maximum likelihood objective maximizes this second quantity. However, we can subtract the empty cut likelihood (3.3) without harm since it is constant. Thus,

$$\arg\max_C \log P(C|\mathbf{x}) = \arg\max_C \sum_{(i,j)\in C} \sum_k \lambda_k \Big[f_k(x_i, x_j, 0) - f_k(x_i, x_j, 1) \Big], \tag{3.5}$$

In other words, maximum likelihood is equivalent to finding the minimum cut on the graph with edge weights $W_{ij} = \sum_k \lambda_k \Big[f_k(x_i, x_j, 1) - f_k(x_i, x_j, 0) \Big]$.

3.4.2 Maximum Inter-Cluster Similarity Equivalence

Similarly, we can show that maximum likelihood is equivalent to maximizing inter-cluster similarity on the graph with particular edge weights. Let E be the set of edges connecting objects within the same cluster; i.e. two objects are in the same cluster if and only if the edge between them, e, is in E. If E is empty, then the log-likelihood of the model is

$$\log P(E = \emptyset|\mathbf{x}) = \sum_{i,j,k} \lambda_k f_k(x_i, x_j, 0) - \log Z(\mathbf{x}, \lambda). \tag{3.6}$$

If E is not empty, we get

$$\log P(E|\mathbf{x}) = \sum_{i,j,k} \lambda_k f_k(x_i, x_j, 0) + \sum_{(i,j)\in E} \sum_k \lambda_k \Big[f_k(x_i, x_j, 1) - f_k(x_i, x_j, 0) \Big] - \log Z(\mathbf{x}, \lambda). \tag{3.7}$$

The maximum likelihood objective maximizes this second quantity. However, we can subtract the empty set likelihood (3.6) without harm since it is constant. Thus,

$$\arg\max_E \log P(E|\mathbf{x}) = \arg\max_E \sum_{(i,j)\in E} \sum_k \lambda_k \Big[f_k(x_i, x_j, 1) - f_k(x_i, x_j, 0) \Big]. \tag{3.8}$$

In other words, maximum likelihood is equivalent to maximum inter-cluster similarity on the graph with edge weights $W_{ij} = \sum_k \lambda_k \Big[f_k(x_i, x_j, 1) - f_k(x_i, x_j, 0) \Big]$, the same weights as those found for the Minimum Cut Equivalence.

3.5 An Antecedent Model of Co-Reference Resolution

Here we introduce a conditional model for co-reference resolution that embodies the classification viewpoint used by Aone and Bennett (1995), Soon et al. (2001) and Ng and Cardie (2002). We assume that each noun phrase mention has at most one antecedent. Either that antecedent is a mention that occurs earlier in the text, or there is no antecedent—an antecedent cannot come after the mention and a mention cannot be its own antecedent. Consider a graph where the nodes are the mentions of a text and two nodes have an edge between them if one is the antecedent of the other. Then, two mentions are "co-referent" if their corresponding nodes are connected, i.e. if they are linked via a series of antecedent relations. This is known as an "antecedent" viewpoint because inference involves identifying the antecedent for each mention; it is known as a "classification" viewpoint because, for the n^{th} mention, inference involves selecting between one of n possibilities.

One of our goals here is to construct an antecedent-based model which is compatible with McCallum and Wellner's Model 3. Though our antecedent model can be used alone, we see it's most valuable application as part of a hybrid model involving Model 3. We discuss this in more detail in § 3.6.

Our antecedent model is essentially a product of softmax[1] models. It can also be seen as modification of Model 3. Model 3 uses binary indicator variables, $y_{ij} \in \{0, 1\}$, to represent whether two mentions refer to the same entity. For our antecedent model, we use the binary indicator variables to represent antecedent relationships. A positive value value ($y_{ij} = 1$, $i < j$) means that x_i is the antecedent of x_j; i.e. mention x_j refers to mention x_i. Two mentions refer to the same entity if an antecedent chain connects them. Each mention has at most one antecedent, so we require $\sum_{i<j} y_{ij} \leq 1$. Thus, if the $\{y_{ij}\}$ are viewed as edges on a graph, two mentions refer to the same entity iff there is a path of edges connecting their corresponding nodes.

Recall the distribution over label indicator variables that McCallum and Wellner define,

$$P(\mathbf{y}|\mathbf{x}; \lambda) \propto \exp\left(\sum_{i,j,l} \lambda_l f_l(x_i, x_j, y_{ij}) \right). \tag{3.9}$$

Ignoring the variable constraints that McCallum and Wellner impose, this is simply a product of Logistic Regression models, one per mention pair. Let λ be the column vector of weights; let $\mathbf{g}(x_i, x_j)$ be the column vector of feature differences for mention pair (x_i, x_j),

[1] Softmax is the generalization of Logistic Regression to multiple categories.

$g_l(x_i, x_j) = f_l(x_i, x_j, 1) - f_l(x_i, x_j, 0)$. Then, we can rewrite the model as

$$P(\mathbf{y}|\mathbf{x}; \lambda) = \prod_{i,j} \frac{\exp(y_{ij}\lambda^T \mathbf{g}(x_i, x_j))}{1 + \exp(\lambda^T \mathbf{g}(x_i, x_j))}. \tag{3.10}$$

Note that the y_{ij} in the numerator simply selects between two unnormalized probability values (1 for $y_{ij} = 0$, and $\exp(\lambda^T \mathbf{g}(x_i, x_j))$ for $y_{ij} = 1$). In our antecedent model, each $y_{ij} = 1$ setting corresponds to an antecedent relation. Each mention may have at most one antecedent relation, so the normalization of our model must reflect this. The indicator variables for each mention, $\{y_{1,j}, y_{2,j}, \ldots, y_{j-1,j}\}$ are tied, so the model must normalize each such group of variables jointly. Note that the unnormalized probability of an indicator variable being "off" ($y_{ij} = 0$) is unity, so to calculate the unnormalized probability of an event, we need only account for variables that are "on" ($y_{ij} = 1$). Thus, the unnormalized probability of x_j having no antecedent is unity. The unnormalized probability of x_j having x_i ($i < j$) as its antecedent is $\exp(\lambda^T g(x_i, x_j))$. The joint antecedent model is

$$P(\mathbf{y}|\mathbf{x}; \lambda) = \prod_{j} \frac{\prod_{i<j} \exp(y_{ij}\lambda^T \mathbf{g}(x_i, x_j))}{1 + \sum_{i<j} \exp(\lambda^T \mathbf{g}(x_i, x_j))}. \tag{3.11}$$

This can be viewed as a product of classification models, one per NP mention. Each individual classification model is a generalization of logistic regression to multiple classes (softmax), where each preceding mention acts as a "class." We include a special "null" mention to indicate the case that an NP is the first mention of an entity.

Like Model 3, learning and inference for the antecedent model are performed via maximum likelihood, (3.11). Also like Model 3, inference for the antecedent model is efficient. However, whereas exact parameter learning for Model 3 scales exponentially and involves a non-convex objective, parameter learning for the antecedent model is efficient and involves a convex objective. In the next two sub-sections, we provide the details of learning and inference for the antecedent model.

3.5.1 Learning

In our antecedent model, the set of weights that establishes the similarity function between mentions must be learned. We proceed by finding the setting of the weights that maximizes the conditional likelihood of the indicator variables given the mentions, (3.11). We minimize the negative log-likelihood, a convex quantity, as it provides a simpler form for optimization. The gradient with respect to the weight vector is

$$\frac{\partial(-\log P)}{\partial \lambda} = \sum_{j} \left[\frac{\sum_{i<j} g(x_i, x_j) \exp(\lambda^T g(x_i, x_j))}{1 + \sum_{i<j} \exp(\lambda^T g(x_i, x_j))} - \sum_{i<j} y_{ij} g(x_i, x_j) \right], \tag{3.12}$$

which is expected feature count minus empirical feature count, the typical gradient form of exponential family models. The normalization constant for the antecedent model is efficient (quadratic in the number of mentions) to compute. So, learning can be performed using traditional optimization techniques, such as the method of Conjugate Gradients (Shewchuk, 1994).

Limited Training Data

The above discussion assumes that we are given antecedent information in our training data. While it is common for a specific antecedent to be identified for each mention in training data, it is also reasonable for each mention to simply be assigned an identifier corresponding to the entity to which it refers. In this case, antecedent relations must be inferred in order to learn the weight vector for our model. We do this by altering our objective. Instead of maximizing the chance that our model selects the right antecedent, we maximize the chance that each mention's path of antecedent links terminates at the correct "root" mention.

As noted earlier, in the antecedent representation of co-reference, there is a distinct root mention for each entity. The root mention is always the first mention of that entity. Given a cluster label for each mention, we can trivially identify the root for each mention. Then, the probability that a mention's antecedent chain ends at its root can be easily defined in terms of antecedent probabilities. Let $R(i,j)$ be the probability that the root for mention x_j is x_i $(i < j)$. Then,

$$R(i,j) = P(y_{\cdot j} = 0|\mathbf{x};\lambda)\delta_{i=j} + \sum_{i \leq k < j} P(y_{kj} = 1|\mathbf{x};\lambda)R(i,k). \tag{3.13}$$

Let $r(j)$ be the index of the root for the cluster with which mention j is associated. Then, the likelihood that we maximize when antecedent information is not given is

$$P(\mathbf{y}|\mathbf{x};\lambda) = \prod_j R(r(j),j). \tag{3.14}$$

As with the earlier objective, this can be optimized efficiently using traditional optimization techniques. However, due to the fact that we are solving a hidden-variable problem, the objective cannot be written as a convex program.

3.5.2 Inference

Given a learned weight vector and mentions in a document, we infer antecedents which maximize the conditional likelihood, (3.11). Due to the structure of the likelihood, we can treat the inference problem for each mention as a separate, local problem. We simply select the antecedent which yields the largest (unnormalized) likelihood. Note that the "null" antecedent (which has an unnormalized likelihood of unity) is one of the possibilities.

3.6 Hybrid Model

3.6.1 Motivation

Our antecedent model groups together mentions which either (1) have an antecedent relationship (i.e. one mention is a clear reference to the other), or (2) are linked via antecedent relationships. Pronouns and other non-proper nouns tend to be linked in this way—by directly linking to another mention. Sentence structure, locality and context play important roles and it is often impossible to determine whether two distant non-proper noun mentions are related without following the chain of antecedent relations which link them. A disadvantage of the antecedent model is that it cannot enforce sex and number consistency.

Model 3 treats co-reference resolution as a clustering problem—a mention is included in

a cluster if its average similarity with other mentions in the cluster is relatively high.Model 3 works especially well for proper nouns as most pairs of proper nouns which refer to the same entity have some clear similarity, such as string overlap. Model 3 can enforce sex and number consistency, but may dismiss a mention from a cluster if it only shows high similarity to a single mention in the cluster. Hence, performance on pronouns and other non-proper nouns since such mentions may suffer.

Next, we discuss a hybrid model which utilizes the clustering viewpoint for resolution of proper nouns and the classification viewpoint for resolution of non-proper nouns. We utilize both Model 3, as well as our antecedent model to create a hybrid model of co-reference resolution.

3.6.2 Details

For our hybrid model of co-reference resolution, we divide the set of noun phrases ($\mathbf{x}$) into two groups. The first, which we denote A, is the set of proper noun phrases; the second, denoted B, is the set of non-proper noun phrases. It is common for an automatic parser to provide the proper vs. non-proper distinction, so we assume this information is given. We use Model 3 to resolve references among proper nouns (model A), and our antecedent model to determine antecedents for non-proper nouns (model B).

These models act largely independently of each other. For inference, model A operates completely in a stand-alone fashion—given the weight vector, it does not use any information about the non-proper nouns in order to cluster the proper nouns. Model B makes use of the result of proper noun inference, but only in a limited fashion. To ensure consistency, we use the result of proper noun inference to "fill in" any missing information in the feature vector. For example, if the prospective antecedent is ambiguous in its gender (e.g. "Powell"), other mentions in its cluster are used to determine gender (e.g. "Mr. Powell"). In practice, we run inference for model A first, to determine the proper noun clusters. Then, we run model B inference, one mention at a time, updating cluster attributes as necessary and utilizing cluster attributes to fill-in any feature information not immediately apparent from a mention pair.

Learning involves the setting of the weight vector to maximize conditional likelihood. Using the chain rule, we can write down the conditional likelihood as a product of the two, individual model likelihoods

$$P(\mathbf{y}|\mathbf{x};\lambda) = P^A(\mathbf{y}^A|\mathbf{x}^A;\lambda^A)P^B(\mathbf{y}^B|\mathbf{x},\mathbf{y}^A;\lambda^B), \tag{3.15}$$

where superscripts are used to distinguish between proper (A) and non-proper (B) parts. Since different features tend to be used for clustering proper nouns vs. resolution of non-proper nouns, we have designated separate weight (and hence feature) vectors for each model. In this case, learning is performed independently for each of the two models. We may also use a single weight (and feature) vector for the entire model, thus necessitating that we use an approximate optimization technique for the entire model (rather than just model A). Note the dependence of model B on the proper NP labels (y^A); this is due to its use of them to "fill in" feature information.

3.7 Discussion

Recent work in co-reference resolution has given evidence of a community divided over how to approach the problem. One body of work, (Aone & Bennett, 1995; Soon et al., 2001; Ng & Cardie, 2002), views each mention as either directly referring to the entity, or else referring to the entity indirectly via a chain of antecedents. In this case, the key task for each mention is to either (1) identify its antecedent, or (2) determine that it has none (i.e. it refers directly to the entity). This is essentially a classification task and has been approached as such by this work. A second body of work, (Cardie & Wagstaff, 1999; McCallum & Wellner, 2005), views mentions that refer to the same entity as being part of a cluster. No assumptions are made about how mentions are related and mentions are treated uniformly. Clusters are inferred which optimize a clustering criterion which maximizes intra-cluster similarity, and minimizes inter-cluster similarity.

Our contribution in this chapter was to explain the differences between these two approaches, identify their strengths and weaknesses and provide a unified algorithm for co-reference resolution. We noted that the clustering viewpoint is well-suited to the task of grouping proper nouns, and that the classification/antecedent viewpoint is well-suited to the task of grouping pronouns and other non-proper nouns. We developed a probabilistic, conditional classification/antecedent model of co-reference resolution which shares similarities with McCallum and Wellner's (2005) clustering model for co-reference resolution. Finally, we provided specification of how to combine these two models to achieve a hybrid model which applies the most appropriate viewpoint to each type of NP mention.

Chapter 4

Tracking Context

4.1 Introduction

So far in this thesis, we have described methods for identifying named entities and resolving multiple mentions of the same named entity. In this chapter, we continue the process of developing methods for extracting information from textual data, especially informal communication. Of importance in extracting opinions, descriptions, etc. is to be able to identify transition points—where the author changes topic. This can be especially difficult in e-mail, bulletin board posts and other informal text because traditional cues such as paragraph breaks and titles/headings are used less frequently, or are used incorrectly. Instead of relying on traditional cues, which may be noisy or nonexistent, we instead use word statistics to attempt to identify changes in topic. We approach this as a text clustering problem. The text document provides an ordering of text segments (e.g. sentences). Our goal is to identify points where there is a substantial change in topic. To do this, we perform clustering on segments with the constraint that two segments may be in the same cluster only if all intermediate segments are also in that cluster.

Many have addressed the text clustering task before. Elkan (2006) uses a heavy-tailed text frequency distribution to model text for clustering. Bekkerman et al. (2005) simultaneously clusters documents, words and authors utilizing pairwise interactions between the three data types to determine associations. McCallum et al. (2004) use a derivative of the Latent Dirichlet Allocation (Blei et al., 2002) model to cluster e-mail messages, incorporating author information into the multinomial model. A variety of algorithms find clusters which minimize a normalized cut of a similarity graph of the items to be clustered. Meila and Shi (2001) show the correspondence between this view and spectral clustering algorithms. Ng et al. (2002) provide a straightforward spectral clustering algorithm and an analysis of the conditions under which the algorithm can be expected to work well.

The goal in any clustering model is to group together similar items. There are two common approaches to this. The first is to identify each cluster with a "mean" or "centroid" representing the cluster and then to associate each point with the nearest mean/centroid. The quintessential example of this approach is k-means. k-means is, in fact, equivalent to MAP estimation of a Gaussian mixture model with identity covariance matrix. Many other mixture models used for clustering or semi-supervised learning[1] are also mean/centroid

[1]Semi-supervised learning can be viewed as clustering where the clusters are seeded by a small number of "labeled" examples.

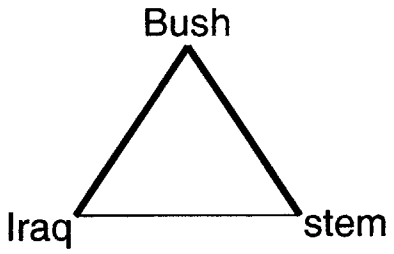

Figure 4-1: A 2-simplex, points of which represent parameters of a trinomial. Here, the three possible outcomes (words) are "Bush", "Iraq" and "stem". Two topics—the Iraq war and funding of stem cell research—are represented by two bold edges of the simplex. Note the intersection at "Bush".

algorithms, such as a Gaussian mixture (without covariance constraints), a mixture of Factor Analyzers (Ghahramani & Hinton, 1996), a mixture of Naive Bayes models (Nigam, 2001), and Latent Dirichlet Allocation (Blei et al., 2002). The second main approach to clustering is known as "spectral clustering" or "normalized cut". Here, the points are represented as nodes on a graph; edge weights, which represent similarities between points, are computed as a function of the data. Normalized cut finds a segmentation of the points which minimizes a normalized inter-cluster similarity. Meila and Shi (2001) show the correspondence between this and "spectral clustering". Spectral clustering algorithms tend to group together points which are well connected. An advantage is that any prior knowledge of the data may be incorporated into the similarity function. However, neither mean/centroid nor spectral approaches may be effective for a data set where points for each cluster are drawn from a low-dimensional subspace. If the subspaces for two clusters intersect, then neither approach can successfully untangle points from the two clusters.

Consider two political topics: one pertaining to the Iraq war and another pertaining to funding of stem cell research. Both are likely to include mentions of President Bush, but "Iraq" and "stem" are likely unique to their corresponding topic. If we consider term distributions over a vocabulary limited to three relevant keywords (Bush, Iraq, and stem), then, as depicted in Figure 4-1, the two topics represent subspaces of the 2-simplex which intersect at "Bush".

Here, we develop an algorithm which can recover clusters where points for each cluster are drawn from a low-dimensional sub-space. The algorithm we present is related to mean/centroid algorithms. It establishes a mean/centroid for each cluster. However, its bias for recovering clusters is atypical. Mean/centroid algorithms generally select centroid locations which minimize a (weighted) average distance and make cluster assignments purely based on the distance function. In contrast, our algorithm is sensitive to the rank of the subspace spanned by the member points. Given points equidistant from the cluster centroid, it is easier to add a point which lies in the subspace already spanned by member points than to add a point outside of the subspace. Thus, our algorithm will prefer to cluster together points which lie in a low-dimensional subspace. Our clustering algorithm can generally be thought of as a way of automatically uncovering subspaces or linear manifolds from a data set.

4.2 Model Overview

Here we introduce our new clustering algorithm, along with the mathematical details of how it works and the intuitions for understanding its behavior.

Trace Norm Clustering (TNC) can be thought of as a mean/centroid algorithm. And, like many mean/centroid algorithms, TNC can be viewed as a model-based algorithm. We can write-down an explicit joint probability for the data and parameters; we use MAP estimation to set parameters for the model. Before providing the mathematical details, we provide an overview of the structure of the model.

4.2.1 Structure

The model consists of three parts: (1) a likelihood, (2) a centroid prior, and (3) a low-rank prior. Here we provide an overview of the model.

Likelihood The likelihood links each data point with a set of parameters which will "represent" that data point in the priors. The likelihood is applied to each data point individually, so it cannot capture interactions between data points. This task is left to the priors. A simple exponential model can work well, such as the multinomial for text data, or an identity-covariance Gaussian distribution for continuous data. Other likelihoods may also be used. The only hard limitation is that parameters of the likelihood must be continuously-valued; ideally, a subspace of parameters of the likelihood would correspond to some meaningful intuition for how data is generated—the subspaces depicted in Figure 4-1 correspond to likelihoods from which certain words are never generated. The likelihood function allows us to try to find a low-dimensional fit for a group of points which is not perfect. If a set of points nearly fit into a low-dimensional subspace, we can utilize a set of parameters which do fit into the low-dimensional subspace and simply incur some loss in likelihood for the imperfect fit between data points and parameters. Thus, the likelihood provides us with a mechanism for trading off fit-to-the-data with our goal of finding clusters corresponding to low-dimensional subspaces.

Centroid Prior As with all mean/centroid algorithms, we learn a set of centroids, one per cluster. Due to the interaction with the low-rank prior, which will be described next, we must penalize distance to the origin of each of the centroids to avoid the trivial clustering of one-point-per-cluster.

Low-Rank Prior The unique aspect of our model is a prior over likelihood parameter which encourages a low rank solution. Consider an assignment of points to clusters. Our goal is to discourage an assignment which groups together points which are a poor fit to a low-dimensional subspace. Intuitively, we want a prior which encourages each cluster to have few dimensions. But, this sounds easier than it really is—we can trivially satisfy our stated criteria of fit-to-the-data and low-dimension-per-cluster by assigning each point to its own cluster. To seemingly complicate matters, we also prefer a prior/criterion which is smooth, or, ideally, concave. To find a prior which will allow us to find an appropriate trade-off between data fitness and low-dimensionality clusters, we turn to the rank minimization problem (RMP). The rank of a matrix is the number of basis directions needed to represent all rows or columns of the matrix—it's "dimensionality". The generic RMP is to minimize the rank of a matrix within a convex set. However, the RMP is NP-hard. Fazel (2002)

discusses various heuristic approaches, including the trace norm, which tends to prefer low-rank solutions. The trace norm of a matrix is the sum of its singular values. It is a convex function of a matrix; minimizing the trace norm within a convex set is a convex optimization problem, making it easy to solve compared to the RMP. We utilize the trace norm in the creation of our parameter prior. We do so by stacking likelihood parameter vectors into a matrix for each cluster. We then calculate the trace norm of this stacked parameter matrix and compute the prior as a function of the trace norm.

4.2.2 Clustering

Now that we have provided an overview of the Trace Norm Clustering model, we provide an overview of how that model is used to cluster data points. The simplest description of our clustering procedure is that we find a clustering and parameters which maximize the joint model probability. This involves both optimizing the assignment of points to clusters, as well as optimizing centroids and other likelihood parameters. Due to our choice of priors, as long as the likelihood is a concave quantity, optimization of the parameters is relatively easy—it can be written as a convex optimization problem and there are fast methods for optimizing a tight bound on the objective. A discussion of such optimization can be found in Rennie and Srebro (2005a). What makes our algorithm computationally challenging is the selection between different assignments of data points to clusters. The default way to determine the optimal assignment is to evaluate each assignment individually. This approach has exponential time complexity. We can find approximate solutions efficiently by constructing clusters in a top-down or bottom-up fashion. However, there is another computational bottleneck to our model—calculation of the normalization constant for our low-rank prior. The constant need not be computed for every assignment of points to clusters—only those which differ in terms of their set of cluster sizes. But, computing a single normalization constant may require exponential time. To make our clustering algorithm practical, we focus on efficient ways of computing the normalization constant.

4.2.3 Definitions

Before discussing the details of our model, it is helpful to provide notation and mathematical definitions for our model. We assume that there are a number of sources, l, which generate data.

Centroids We assume that each cluster/source generates a centroid, μ_l, which helps to define how data from that source is generated. The centroid can be represented in terms of a radius $r \equiv \|\mu_l\|_2$ and a set of angles. We are not concerned with the direction of the centroid and only care about its distance from the origin. So, our centroid prior is a $m \times 1$ trace norm distribution (which is, in fact, a gamma distribution over the radius value),

$$P(\mu_l|\lambda) \propto e^{-\lambda\|\mu_l\|_\Sigma}, \tag{4.1}$$

where $\lambda > 0$ controls the scaling of the distribution, and the vector μ_l is treated as an $m \times 1$ matrix. Note that the trace norm of a $m \times 1$ matrix is simply the L_2 (Euclidean) length of the vector. We assume that λ is fixed and known: it is an input to the generative process.

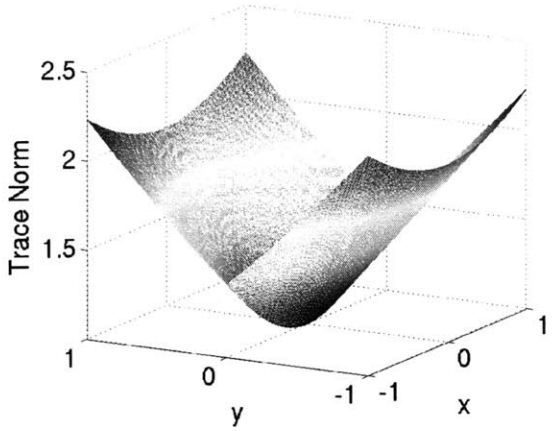

Figure 4-2: Trace norm of $X(x, y) = [1\ 0;\ x\ y]$. The x- and y-axes represent parameters of $X(x, y)$. the z-axes represents the trace norm of $X(x, y)$, $\|X(x, y)\|_\Sigma$. Note that $X(x, 0)$ is a rank one matrix, whereas $X(0, y)$ is a rank two matrix for $y \neq 0$.

Low-rank Parameters We assume that each source generates a matrix of parameters, θ_l, via a trace norm distribution,

$$P(\theta_l|\lambda) \propto \exp(-\lambda\|\theta_l\|_\Sigma), \tag{4.2}$$

where, again, λ, controls the scaling of the prior. Details of this distribution are discussed in § 4.3. We use θ_{li} to denote the i^{th} row of θ_l and θ_{lij} to denote the (i, j) position of θ_l.

Likelihood We assume that there is a common likelihood function which uses a vector of parameters to generate data points. Each data point to be generated has its own vector of parameters, which is a combination of a centroid (common to all points in the same cluster) and a row of the low-rank matrix (unique to the point). Let X_{li} be a vector which represents the i^{th} point generated from cluster l. Then, the distribution which generates X_{li} is

$$P(X_{li}|\mu_l + \theta_{li}). \tag{4.3}$$

In other words, $\mu_l + \theta_{li}$ is the parameter vector that we use to parameterize the likelihood for generation of X_{li}.

4.3 Trace Norm Distribution

A major part of our clustering model is our choice of a low-rank prior for modeling of the likelihood parameters. We utilize the trace norm to give us a prior which encourages a low-rank parameter matrix. A difficulty with this choice is that the normalization constant for the prior is not easy to compute. The normalization constant is needed to compare different clusterings of the data. We discuss various approximations for the normalization constant.

Trace Norm

The trace norm of a matrix, denoted $\|X\|_\Sigma$, is the sum of its singular values. The trace norm of a column (or row) matrix is simply the Euclidean length (L_2-norm) of the matrix (treated as a vector). The trace norm was introduced by Fazel et al. (2001) as a surrogate for rank in the rank minimization problem. Srebro et al. (2005) utilized the trace norm for regularization in supervised learning.

To gain intuition for the 2×2 case, we consider the matrix $X(x,y) = \begin{bmatrix} 1 & 0 \\ x & y \end{bmatrix}$. The trace norm is rotationally-invariant (i.e. $\|RX\|_\Sigma = \|X\|_\Sigma$, R is a rotation matrix) and commutative with scalar multiplication (i.e. $c\|X\|_\Sigma = \|cX\|_\Sigma$, c is a scalar). Hence, $X(x,y)$ allows us to fully explore trace norm values of 2×2 matrices by varying the two parameters, x and y. Figure 4-2 plots $\|X(x,y)\|_\Sigma$ as a function of x and y. We can see from this plot that the trace norm encourages a low-rank solution. Consider using the trace norm as a negative log-prior. MAP estimation involves minimizing a sum of the trace norm and a negative log-likelihood (aka loss function). Consider optimizing x and y. Consider any point with $x = 0$. If the negative log-likelihood has a non-zero derivative for x, then we can decrease the overall objective by changing x due the smooth derivative of the trace norm with respect to x. This is not the case for y. Consider any point with $y = 0$. If the negative log-likelihood has a non-zero derivative with respect to y, it will not necessarily be advantageous to change y from 0 due to the non-continuous derivative of the trace norm with respect to y; the gradient of the negative log-likelihood (wrt y) must be at least as large as the gradient of the trace norm (wrt y) at $y = \pm\epsilon$ in order to force a move away from $y = 0$. Also, note that the gradient of the trace norm with respect to y is always at least as large as the gradient with respect to x—minimization of the trace norm will prefer to decrease the magnitude of y first, unless constraints or another objective component (e.g. loss function) force it to do otherwise.

Now, consider the general $m \times n$ case. Again, we will observe a discontinuous gradient and larger magnitude gradient for directions which would change the rank of the matrix. As discussed in the 2×2 case, this will tend to a yield low rank solution since substantial "force" is needed from constraints or the loss function to counter this tendency to decrease the rank.

Definition

The Trace Norm Distribution (TND) is the parameter prior for our model which encourages low-rank/low-dimension clusters. It is applied to the set of parameters corresponding to the set of points in each cluster. A single parameter, λ, controls the strength of the prior.

Minimization of the trace norm in combination with other objectives tends to yield a low-rank solution; we formulate the prior to take advantage of this fact. We define the Trace Norm Distribution as

$$P_{m\times n}(X|\lambda) \propto \exp(-\lambda\|X\|_\Sigma), \tag{4.4}$$

where $X \in \mathbb{R}^{m\times n}$. I.e. the distribution is over matrices of size $m \times n$. The distribution is normalized by integrating over matrices of fixed size,

$$Z_{m\times n}(\lambda) = \int_{\mathbb{R}^{m\times n}} \exp(-\lambda\|X\|_\Sigma)dX. \tag{4.5}$$

Thus, the negative log-prior will be a multiple of the trace norm (plus a normalizing constant), $-\log P_{m\times n}(X|\lambda) = \lambda\|X\|_\Sigma - \log Z_{m\times n}(\lambda)$. Thus, MAP estimation will minimize a sum of the negative log-likelihood and a constant multiple of the trace norm. We use the TND as the parameter prior for each cluster. The probability is large when the rows or columns of X lie in a low-dimensional space and the magnitude of the matrix is low. The probability is small when the rows and columns of X are close to perpendicular and the magnitude of the matrix is high. λ controls the scaling of the distribution.

Comparison to Exponential Distribution

A cursory understanding of the distribution might lead one to mistakenly believe that it is the product of exponential distributions, one for each singular value, since the trace norm is the sum of singular values. If so, sampling from the trace norm distribution would be straightforward: (1) sample the singular values independently from an exponential distribution, (2) arrange in a diagonal matrix, Σ, and (3) multiply on the left by a random matrix with orthonormal columns, U, and (4) multiply on right by a random with orthonormal rows, V^T. Alas, this is not correct as the volume of points increases as we move away from the origin. Thus, additional terms appear in the likelihood in the change of variables to the singular value decomposition. A simple example might aid intuition. Consider the two-dimensional Gaussian,

$$p(x, y) \propto \exp(-(x^2 + y^2)/2). \tag{4.6}$$

It is tempting to believe that the radius $r = \sqrt{x^2 + y^2}$ is Normally distributed. However, the change of variables to polar coordinates introduces an extra r term,

$$p(r, \theta) \propto re^{-r^2/2}. \tag{4.7}$$

This is a direct result of the fact that the surface (circumference) of a circle is a linear function of the radius. Large radii are more likely than they would be if radius were distributed as a Normal. As we will see in section 4.4, in rewriting our trace norm distribution in terms of singular values, additional terms are introduced which make clear that the singular values are neither independent, nor individually distributed as an exponential.

Partitioning

Determining the best clustering for our model involves selecting between assignments of data points to clusters which maximize the joint probability as a function of the parameters. To gain intuition for how our model behaves, it will be helpful to have some notation for an assignment of data points to clusters. We will use "partitioning" to describe the assignment of points to clusters and will use notation which is typical for this terminology. Let m be the total number of data in all clusters. Let $\mathbb{R}_m = \{1, \ldots, m\}$. Then, we define a partitioning of the data points as a function,

$$\rho : \mathbb{R}_m \to \mathbb{R}_m, \tag{4.8}$$

which takes a data point index and returns a cluster index. We also define the inverse function which takes a cluster index and returns the set of data point indices corresponding

to that cluster,

$$\varrho : \mathbb{R}_m \to \mathcal{P}(\mathbb{R}_m), \tag{4.9}$$

where $\mathcal{P}(\cdot)$ denotes the power set, or the set of all possible subsets of the argument. We will use $|\varrho(i)|$ to denote the size of cluster i. Note that $i \in \varrho(\rho(i))$.

Lambda

To provide further intuition about our model, we provide some discussion on the impact of λ. We focus on how λ affects the impact of the trace norm prior on the optimal clustering.

To gain intuition for the effect of λ, we consider the case that all parameters (θ, μ) are fixed and that the only remaining choice to be made is the partitioning. To simplify further, we consider only two scenarios: a single cluster vs. two clusters. Let $\theta \in \mathbb{R}^{m \times n}$ be the full matrix of low-rank parameters. Let θ_1 and θ_2 be a row-wise partitioning of those parameters. Then, the partitioning into two clusters will be preferred over the single cluster if

$$P(\theta_1|\lambda)P(\theta_2|\lambda) > P(\theta|\lambda). \tag{4.10}$$

Equivalently, the split will be preferred if

$$\log Z_{m\times n} - \log Z_{m_1\times n} - \log Z_{m_2\times n} > \lambda(\|\theta_1\|_\Sigma + \|\theta_2\|_\Sigma - \|\theta\|_\Sigma), \tag{4.11}$$

where m is the total number of data points, m_1 and m_2 are the number of data points in each of the two clusters ($m = m_1 + m_2$), and n is the dimensionality of the space in which the data points lie. The trace norm is sub-additive (see, *e.g.*, Corollary 3.4.3 of (Horn & Johnson, 1991)),

$$\|\theta\|_\Sigma \leq \|\theta_1\|_\Sigma + \|\theta_2\|_\Sigma. \tag{4.12}$$

The fact that the log-normalization constants are sub-additive is an immediate corollary. Continuing from (4.12) (note the change in direction of the inequality),

$$\int e^{-\|\theta\|_\Sigma} d\theta \geq \int e^{-\|\theta_1\|_\Sigma} d\theta_1 \int e^{-\|\theta_2\|_\Sigma} d\theta_2, \tag{4.13}$$

$$\log Z_{m\times n} \geq \log Z_{m_1\times n} + \log Z_{m_2\times n}. \tag{4.14}$$

Thus,

$$\lambda^* = \frac{\log Z_{m\times n} - \log Z_{m_1\times n} - \log Z_{m_2\times n}}{\|\theta_1\|_\Sigma + \|\theta_2\|_\Sigma - \|\theta\|_\Sigma} \tag{4.15}$$

is the point at which neither partitioning is preferred. For $\lambda < \lambda^*$, the split into two clusters will yield a larger prior; for $\lambda > \lambda^*$, the single cluster will have a larger prior. Note that λ^* is non-negative. We can draw clear conclusions from this observation. As λ decreases, finer partitionings are preferred, and in the limit as $\lambda \to 0$, no partitioning will be preferred over the one-point-per-cluster partitioning. Conversely, as λ increases, we will tend to prefer a few, large clusters. As $\lambda \to \infty$, no partitioning will be preferred over the single cluster solution. By changing λ we can control the granularity of clustering. Note

that the intuition here is similar to supervised learning where we minimize a loss function plus a regularization term. Here, the trace norm prior corresponds to the regularization term. For small λ, the loss function dominates and we prefer parameters which provide a close fit to the data. For large λ, the regularization dominates and we prefer parameters which satisfy the regularization term (such as a low-rank parameter matrix in the case of the trace norm).

4.4 Normalizing Constant

As mentioned earlier, computation of the normalization constant of the trace norm distribution cannot be computed exactly and efficiently; yet, it is essential for clustering—it tells us how to compare trace norm values for different size partitionings. The normalization constant for the Trace Norm Distribution is an integral over matrices of size $m \times n$. In this section, we show how use a change of variables to rewrite the integral into a form in which it can be integrated analytically. However, this form is computationally inefficient and may scale exponentially as m or n increases. Thus, we also discuss two ways of efficiently calculating estimates of this quantity.

Singular Value Decomposition

The Singular Value Decomposition of a matrix $X \in \mathbb{R}^{m \times n}$ (wlog $m > n$) is a product of three matrices, $U \in V_{m,n}$, $\Sigma \in \text{diag}(\mathbb{R}^n)$, and $V \in V_{n,n}$, where $V_{m,n}$ is the Stiefel manifold[2] (i.e. $U^T U = I$, $V^T V = I$). The SVD is $X = U\Sigma V^T$. Note that since Σ is diagonal, and that U and V are orthogonal, the number of entries in X is equal to the number of free entries in U, Σ, and V. We assume that the singular values (diagonal elements of Σ) are ordered. If the singular values are unique, $\sigma_1 > \cdots > \sigma_n$, then SVD is unique up to a change in signs of matching columns of U and V.

Recall that the normalization constant is an integral of the exponentiated negative trace norm over matrices of size $m \times n$. A change of variables to the SVD simplifies the trace norm computation (to the sum of diagonal entries of Σ), but our variable of integration is incorrect,

$$Z_{m \times n}(\lambda) = \int_{\mathbb{R}^{m \times n}} \exp(-\lambda \|X\|_\Sigma) dX = \int_{\mathbb{R}^{m \times n}} \exp\left(-\lambda \sum_{i=1}^{n} \sigma_i\right) dX. \tag{4.16}$$

Next, we calculate the proper change of variables from X to $U\Sigma V$.

Jacobian for the Change of Variables

To change variables to the SVD, we must calculate the Jacobian. Following the derivation given in Edelman (2005a), we get

$$dX^\wedge = \prod_{i<j \leq n} (\sigma_i^2 - \sigma_j^2) \prod_{i \leq m} \sigma_i^{m-n} (d\Sigma)^\wedge (V^T dV)^\wedge (H^T dU)^\wedge. \tag{4.17}$$

where H is an $m \times m$ orthogonal matrix whose first n columns are identical to U, and $\wedge$ denotes the wedge product.

[2]See Edelman (2005b) for a discussion of the Stiefel manifold.

The change of variables introduces two new terms. First, there is a σ_i^{m-n} factor, which is similar to the extra r introduced in the Gaussian change of variables to radial coordinates. These account for the extra density arising from integrating over the surface of a sphere. When combined with the exponential term, these yield the form of a gamma distribution PDF. Second, there is a term that goes to zero with the difference between singular values. Thus, singular values are clearly not independent of each other. The trace norm favors matrices where the singular values are separated. Indeed, it is this coupling between singular values which makes the distribution difficult to integrate, as we discuss next.

Trace Norm Distribution Integral

Applying the change of variables to our integral, we get

$$Z_{m\times n}(\lambda) = \frac{1}{2^n}\int \prod_{i<j\leq n}(\sigma_i^2-\sigma_j^2)\prod_{i\leq n}\sigma_i^{m-n}e^{-\lambda\sigma_i}(d\Sigma)^\wedge(V^TdV)^\wedge(H^TdU)^\wedge. \tag{4.18}$$

Recall that except for a measure zero set, the SVD is unique up to a sign, hence the initial 2^{-n} factor. There is no intermingling between Σ, U and V, so what we really have is the product of three separate integrals,

$$\frac{1}{2^n}\int \prod_{i<j\leq n}(\sigma_i^2-\sigma_j^2)\prod_{i\leq n}\sigma_i^{m-n}e^{-\sigma_i}(d\Sigma)^\wedge \int(V^TdV)^\wedge \int(H^TdU)^\wedge. \tag{4.19}$$

Note that $\int(V^TdV)^\wedge$ and $\int(H^TdU)^\wedge$ are volumes of the Stiefel manifold. Edelman (2005b) provides the Stiefel manifold volume,

$$\mathrm{Vol}(V_{m,n}) = \prod_{i=m}^{m-n+1} A_i = \prod_{i=m}^{m-n+1}\frac{2\pi^{i/2}}{\Gamma\left(\frac{i}{2}\right)}, \tag{4.20}$$

where A_i is the surface area of the sphere in $\mathbb{R}^i$ of radius 1. The remaining integral over singular values,

$$\int_0^\infty \cdots \int_0^{\sigma_{n-1}} \prod_{i<j\leq n}(\sigma_i^2-\sigma_j^2)\prod_{i\leq n}\sigma_i^{m-n}e^{-\sigma_i}d\sigma_n\ldots d\sigma_1, \tag{4.21}$$

can be computed analytically. Using maxima[3], we can compute this for matrix sizes of 10×5 and 40×4 in a few minutes of CPU time. However, the non-constant limits of integration, as well as the explosion of terms makes evaluation intractable for large n or m. Large n presents a problem because $\prod_{i<j}(\sigma_i^2-\sigma_j^2)$ yields a polynomial with a number of terms which is exponential in n; simply expanding this product is infeasible for large n. For small n, we can still face a problem because of large m, as each stage of the integration yields an incomplete gamma function of order m. Integrating the incomplete gamma function doubles the number of terms involved, resulting in a total number of terms of $O(2^m)$. In the next sections, we discuss ways of approximating the integral for large matrices.

[3] http://maxima.sourceforge.net/

Sampling

The form of the integral over singular values (4.21) suggests a sampling scheme. Define $g(x|a,\theta) \equiv \frac{x^{a-1}e^{-x/\theta}}{\Gamma(a)\theta^a}$ as the gamma distribution pdf. Furthermore rewrite the integral so that it is symmetric with respect to the variables by taking absolute values of the difference terms and using constant limits of integration,

$$J \equiv \int_0^\infty \cdots \int_0^\infty \prod_{i<j\leq n} |\sigma_i^2 - \sigma_j^2| \prod_{i=1}^n \sigma_i^{m-n} e^{-\sigma_i} d\sigma_n \ldots d\sigma_1. \tag{4.22}$$

Clearly the integral can be computed as an expectation over a product of gamma distributions of a product of difference between squared singular values. We can approximate the integral by taking an average of weighted samples of singular values. Define $\alpha = m - n + 1$ and $\theta = 1/\lambda$. Define the following sampling scheme:

- for $t = 1$ to k:
  - Sample $\{\sigma_1, \ldots, \sigma_n\}$ iid from $g(\sigma_i|\alpha,\theta)$
  - Let $z_t = \prod_{i<j} |\sigma_i^2 - \sigma_j^2|$.

Then, our estimate of the integral is $J \approx \frac{\Gamma^m(\alpha)\theta^{m\alpha}}{k} \sum_{t=1}^k z_t$. Note that as the number of samples goes to infinity, so does the approximation tend toward the correct answer, $J = \lim_{k\to\infty} \frac{\Gamma^m(\alpha)\theta^{m\alpha}}{k} \sum_{t=1}^k z_t$.

Variational Bound

Another approach to approximating the singular value integral is to find an upper bound. Here we describe such a method developed in conjunction with John Barnett (personal communication, March 2006). Note that any definite integral can be reformulated as an optimization problem:

$$\int_A f(x)dx = \min_{g:g\geq f} \int_A g(x)dx, \tag{4.23}$$

assuming that f and g are in a space of integrable functions. Restricting the space of functions to those of a parametric form, one obtains an upper bound,

$$\int_A f(x)dx \leq \min_\theta \int_A g(x;\theta)dx. \tag{4.24}$$

A suitable upper bound depends on finding a suitable class of parametric functions.

A logarithmic bound is used to replace the product of differences. Since the logarithm is concave, we can bound it with the tangent line at any point a, $\log x \leq \frac{x}{a} - 1 + \log a$. So,

$$\begin{aligned}\prod_{i<j}(\sigma_i^2 - \sigma_j^2) &\leq \exp\left[\sum_{i<j} \frac{\sigma_i + \sigma_j}{a_{ij}} + \frac{\sigma_i - \sigma_j}{b_{ij}} - 2 + \log a_{ij}b_{ij}\right] \\ &= \left(\prod_{i<j} a_{ij}b_{ij}\right) e^{-n(n-1)} e^{\sum_{i=1}^n d_i\sigma_i},\end{aligned} \tag{4.25}$$

where $d_i = \sum_{j=i+1}^{n} \left(\frac{1}{a_{ij}} + \frac{1}{b_{ij}}\right) + \sum_{j=1}^{i-1} \left(\frac{1}{a_{ji}} - \frac{1}{b_{ji}}\right)$. This gives us an upper bound which is log-linear in the singular values, $\{\sigma_i\}$.

To eliminate the incomplete gamma functions, rewrite the integral using step functions so that our limits of integration do not depend on the variables themselves. Applying our first bound, (4.25), and ignoring constant quantities, the singular value integral becomes

$$\int_0^\infty \cdots \int_0^\infty \prod_{i=1}^{n} \sigma_i^{m-n} e^{-(1-d_i)\sigma_i} u(\sigma_{i-1} - \sigma_i) d\sigma_n \cdots d\sigma_1, \tag{4.26}$$

where $\sigma_0 \equiv \infty$ and $u(\cdot)$ is the Heaviside step function. Using an exponential bound on the step function, $u(x) \leq e^{\beta x}$, we obtain a tractable bound. Continuing from (4.26), we substitute the exponential bound to attain a bound on the full integral,

$$\leq \prod_{i=1}^{n} \int_0^\infty \sigma_i^{m-n} e^{-(1-d_i+\beta_{i+1}-\beta_i)\sigma_i} d\sigma_i, \tag{4.27}$$

where $\beta_1 \equiv \beta_{n+1} \equiv 0$. The integral inside the product is the normalization constant for a gamma distribution, which is well-known and easy to evaluate,

$$\int_0^\infty \sigma_i^{m-n} e^{-(1-d_i+\beta_{i+1}-\beta_i)\sigma_i} d\sigma_i = \prod_{i=1}^{n} \frac{\Gamma(m-n+1)}{(1-d_i+\beta_{i+1}-\beta_i)^{m-n+1}}. \tag{4.28}$$

Adding back the constant terms we ignored earlier, one arrives at a bound on the singular value integral,

$$\leq \left(\prod_{i<j} a_{ij} b_{ij}\right) \prod_{i=1}^{n} \frac{\Gamma(n-m+1) e^{-n(n-1)}}{(1-d_i+\beta_{i+1}-\beta_i)^{m-n+1}}. \tag{4.29}$$

This bound holds for all $a_{ij}, b_{ij} \geq 0$ and for all $\beta_i \geq 0$. One finds the best bound by optimizing over these parameters.

Note that this approach can also be used to obtain a lower bound on the normalization constant.

Evaluation

Table 4.1 gives approximate and exact log-values of the trace norm distribution normalization constant for various values of m and n. Our ability to compute the integral exactly is quite limited—we were not able to compute exact values for $m > n > 5$. We tested the two approximate methods we have described: sampling and the variational bound. The sampling method seems to be moderately accurate, especially for $m > n$. However, sampling may not scale well and we can observe inconsistencies even for this small evaluation. Sampling returns a larger value for a 16×8 size matrix than it does for a 16×16 size matrix. Yet, it can be shown that the value for the 16×16 case is larger than the 16×8 case. The variational bound was less accurate over the sizes we tested. However, it scales better and might be valuable in conjunction with a (yet developed) lower bound to provide an accurate, scalable estimate.

		m			
n	method	2	4	8	16
2	exact	.405	3.27	16.2	54.6
	sampling	.406	3.27	16.3	54.6
	variational	1.86	4.35	17.2	55.6
4	exact		8.22	31.3	106
	sampling		8.13	31.3	106
	variational		17.4	36.1	110
8	sampling			56.9	205
	variational			118	229
16	sampling				186
	variational				672

Table 4.1: Values of the logarithm of the Trace Norm Distribution normalization constant.

4.5 Clustering Algorithm

Now that we have approximate methods for efficiently calculating the normalization constant of the Trace Norm Distribution, we can turn to the task of efficiently finding an approximately optimal clustering of the data points. As noted earlier, to find the optimal clustering, we must evaluate all possible assignments of data points to clusters. This is computationally infeasible for even relatively small numbers of data points. The application we are considering, sentence clustering, yields fewer possible assignments as a function of the number of data points, but still an exponential number. So, we consider two greedy partitioning methods: (1) a bottom-up approach and, (2) a k-means-like approach.

4.5.1 Bottom-up clustering

Bottom-up clustering is a typical method for finding a cluster assignment. We begin with a one-point-per-cluster assignment and consider all possible cluster merges. We select the merge which yields the largest joint probability according to our model.

There are two possible approaches we can take for bottom-up clustering. The first is to take λ as a fixed value and perform merges as long as they increase the joint probability of our model. This will differ from a typical bottom-up approach in that the model will typically select a stopping-point before all points have been merged together into a single cluster. Another approach is to slide λ from $0 \rightarrow \infty$ in order to achieve a full range of possible clusterings (from one-point-per-cluster to a single cluster). To do this, we begin with each point in its own cluster. Then, for each pair of clusters, we calculate the λ value at which the model would be indifferent about merging them, as in (4.15). We merge the pair of clusters with the lowest λ-indifference value. After merging, we recalculate indifference values and repeat until we are left with a single cluster. An advantage of this approach is that it gives us with a full hierarchy of the data, providing us with additional association information which a simple clustering does not provide.

4.5.2 k-flats

An alternative approach is to use iterative reassignment in the style of k-means. However, the intuition behind the algorithm is sufficiently different to warrant a new name, hence

n_1	n_2	d_1	d_2	TNC Accuracy	Baseline Acc.
4	4	2	2	80%	0.12%
4	4	3	3	40%	0.12%
6	2	4	2	52%	1.8%

Table 4.2: Accuracy of Trace Norm Clustering on Simulated Data

"k-flats". Whereas k-means attempts to find k centroids which provide the best fit to the data, k-flats attempts to find best-fit centroids and low trace norm ("flat") sub-spaces. We initialize the algorithm with a random assignment of points to clusters. Each round, we select a random data point and consider assigning to each of the other clusters; we also consider establishing a new cluster for the point under consideration. We evaluate each possibility for the selected point and choice the one that yields the largest joint model probability. We repeat this procedure for a pre-determined number of rounds or until the assignment becomes stable.

4.6 Simulated Data

Here we merely begin to evaluate our Trace Norm Clustering framework. We use simulated, real-valued data and use a trivial likelihood function, effectively applying our trace norm prior directly to the data itself. We force all centroids to the origin, effectively eliminating the centroid component from our model. I.e. this is a pure test of the model's ability to recover data generated in a low-dimensional subspace. So that we could make computations without approximation, we limited ourselves to 8 data points, each embedded in $\mathbb{R}^5$. We generated points from two low-dimensional standard Normals, then used our TNC model to find the most likely partitioning for a range of λ values. We conducted 100 trials for each scenario and report the fraction of times our model selected the correct assignment of all points for some value of λ. Note that this is a difficult task: if the algorithm does not select the correct assignment of all points for some value of λ, the entire trial is considered incorrect (no partial credit). Table 4.2 provides the results. n_1 and n_2 are the number of data points generated for each of the two clusters. d_1 and d_2 is the dimensionality of the standard Normal that was used to generate the points in each of the two clusters. "TNC Accuracy" denotes the number of trials for which TNC made the correct assignment (for some value of λ). "Baseline Acc." gives the accuracy for an algorithm which assigns points uniformly at random to one of the two clusters (ensuring the appropriate number per cluster). The generated points in each cluster were (jointly) randomly projected into $\mathbb{R}^5$. We used a standard Normal to generate the projection matrix for each cluster. Note that since both clusters have mean zero, any mean-based clustering algorithm would likely fail. Spectral and normalized cut algorithms would also likely fail at this task due to the intersection of the subspaces at the origin. Trace Norm Clustering is able to recover the clusters with regularity, vastly outperforming the baseline algorithm. Of particular note is the fact that it is able to recover the 6/2 clustering at all. Since the cluster of 2 is being projected from a 2-D standard Normal, the points are effectively iid from a 5-D standard Normal. I.e. they are only related to each other in how they are different from the cluster of 6.

4.7 Discussion

We have introduced a new framework for clustering based on associating points which lie together in a low-dimensional space. We use the trace norm as our basis, which provides a smooth penalty on the dimensionality of data. Incorporation of the trace norm into a distribution allowed us to compare partitionings of various sizes. Normalization of this distribution is computationally difficult, so we discussed approximation methods that would be useful for clustering of large data sets. Experiments on simulated data indicate that our framework is able to uncover low-rank data partitionings.

Chapter 5

Preference Learning

5.1 Introduction

So far, this thesis has focused on direct extraction of information (identifying and resolving restaurant names; determining the extent of opinions in text). In this chapter, we assume that information about user opinions/preferences has already been extracted. Here, we will describe methods for the synthesis of that information to predict unobserved user preferences.

The preference learning problem is common to many domains. Two popular applications are ratings of books and movies. Users give discrete ordinal ratings (e.g. 1 through 5 "stars") and we would like to be able to predict ratings for unobserved user/item pairs. The task of predicting restaurant preferences is similar. However, when extracting opinions from bulletin boards, an additional step must be taken to translate written opinions into discrete ordered ratings. We do not address the problem of extracting the direction (positive/negative) and magnitude of the opinion, as many before us have worked on this problem (e.g. Wiebe, 1994; Pang & Lee, 2004). However, we do address the issue that users express themselves in different ways. The methods we describe allow each user to have his/her own rating system. The rating "style" and/or number of rating levels may vary by user. Thus, the translation system may focus on identifying the user's summary opinion and translating the key words/adjectives to an ordinal scale. Any uniqueness in the words chosen by the user to express him/herself will be handled by the preference learning system.

We begin this chapter with a discussion of the Ordinal Regression (OR) problem, where we assume that there are a collection of items, each with an associated feature vector, some of which have an associated rating. There is a single user, and the goal is to predict ratings for the items which have not been rated. We simplify the problem to one of learning a user weight vector which represents, as best as possible, his/her preferences for the items. For our restaurant bulletin-board example, the feature vector might include who owns the restaurant, the number of seats, the type of cuisine, as well as specific food and/or drink items from the menu. The user weight vector would then reflect the direction (positive/negative) and magnitude of the user's feelings toward each of these aspects. Two important characteristics of OR are that (1) the user may not treat the ratings in a uniform fashion, and (2) mistakes may vary in their magnitude. We describe techniques that address both of these concerns. We also give a detailed discussion of the construction of loss functions for OR and their relation to loss functions for binary classification. Finally, we provide an experimental

comparison of the discussed techniques.

When there are multiple users, a simple solution would be to learn weight vectors independently via OR. However, this ignores the fact that the problems are related. When two (or more) users rate the same item, they implicitly provide information about how their preferences relate. This can, in turn, be used to deduce information about the items. Such information may not be otherwise available—there may be aspects of items which are difficult to quantify (such as the dramatic component of a movie) or impossible to access (such as the done-ness and seasoning of a dish consumed by a certain restaurant customer).

In section 5.3, we discuss Collaborative Filtering (CF), the name given to this task of predicting unobserved ordinal ratings based on a sparse set of user-item ratings. Unlike OR, we assume that no item information is given a priori. Rather, we use the overlapping preferences to generalize to unseen user/item pairs. Whereas OR is limited by the identifiable features of the items, CF is limited by the overlap of ratings provided for learning.

One approach to the CF problem is to learn a low rank matrix which approximates the observed user/item rating matrix. The idea is that though there are many users and items, there are only a handful of *factors* which determine the ratings. Limiting the rank of the learned matrix encourages the predictions to be based on the factors that influence ratings rather than any noise which may have corrupted the ratings. Applying ideas from our Ordinal Regression discussion (user-specific "styles" and a loss function which accounts for the mistake magnitude) yields a viable CF framework. However, a lingering difficulty is that the space of rank-constrained matrices is not convex[1]. As a result, optimization over rank-constrained matrices will generally face local minima problems. This, in effect, makes the problem ill-defined, since very different solutions may appear to be optimal or near-optimal. An alternative approach to the CF problem is to allow an unlimited rank matrix, but to apply a regularization penalty which encourages a low-rank solution. The idea of using the trace norm for this purpose was recently introduced as "Maximum Margin Matrix Factorization" (MMMF). MMMF poses no constraint on the reconstructed rating matrix, so the search space is convex. Also, the trace norm is a convex function, so as long as the loss function used to measure dissimilarity between the true and reconstructed rating matrix is convex, the result is a convex optimization problem with a well-defined solution. Interestingly, MMMF is a generalization of the soft-margin Support Vector Machine. We describe how to scale MMMF to large collaborative filtering problems (Fast MMMF), then evaluate Fast MMMF against a variety of CF algorithms and find that Fast MMMF yields the best generalization performance.

MMMF was originally formulated assuming that no outside feature information is available for the items and users. However, the Fast MMMF optimization method that we describe lends itself to a hybrid algorithm where given features/weights can be incorporated alongside features/weights learned from the ratings. This can be viewed as a synthesis of ordinal regression (OR) and collaborative filtering (CF). In OR, we relied upon given feature vectors for the items to generalize; in CF, we relied upon the overlapping user ratings to generalize. Our hybrid approach, which we describe in section 5.4, uses both sources of information to better predict unobserved ratings.

A typical assumption in both ordinal regression (OR) and collaborative filtering (CF) is that there is no pattern to the rating observations; it is assumed that the selection of user/item pairs corresponding to "observed" ratings is made via a uniform distribution (over users and items). A quick analysis of one's own movie-watching habits will likely reveal that

[1]To see this, note that the convex combination of two rank-one matrices can yield a rank-two matrix.

this is not the case. Some people follow certain actors or actresses; others have a favorite director or writer; others limit themselves to certain genres. And, it is rare to find someone who is completely unaffected by movie marketing and promotion. The process by which a user selects movies to see and rate is known as a Missingness Mechanism (MM). In the context of OR and CF, the MM can be viewed as a binary classification problem (whether the user saw and rated the movie) layered on top of the rating prediction problem. We describe a MM extension to MMMF in section 5.5.

Though we do not address such problems here, we note that preference learning and collaborative filtering techniques are not limited to users/item pairs. The techniques we describe may be applied to general tuples (e.g. triples). This might be useful for modeling sports performance, where many factors combine to yield the observed performance. Our methods might also be applied to a scenario where interactions are between objects from a common set, such as in social networking, trust networks or predicting the results of head-to-head competitions.

5.2 Ordinal Regression

In many systems, users specify preferences by selecting, for each item, one of several rating "levels", e.g. one though five "stars". When learning to predict further preferences, these rating levels serve as target labels (responses). This type of discrete ordered labels differs from more standard types of target labels encountered in other machine learning problems: binary labels (as in classification tasks), discrete, unordered labels (as in multi-class classification tasks), and continuous real-valued labels (as in typical regression tasks). Rating levels are discrete with a finite number of possibilities, like class labels in multi-class classification. However, unlike a standard multi-class classification setting, the labels are ordered—a rating of "three stars" is between a rating of "two stars" and a rating of "four stars".

Two obvious approaches for handling discrete ordinal labels are (1) treating the different rating levels as unrelated classes and learning to predict them as in a multi-class classification setting, and (2) treating them as a real-valued responses and using a standard regression setting with a loss function such as sum-squared error. However, neither of these reflects the specific structure of discrete ordinal labels.

We view fitting rating levels as a regression problem with discrete ordered labels. We view this as a generalization of binary classification, which can be seen as a degenerate case in which only two levels, "positive" and "negative," are available. As is common for binary classification, we learn a real-valued *predictor*, $z(x)$, to minimize some *loss*, $\text{loss}(z(x); y, \theta)$, on the target labels and thresholds. Whereas the single threshold in binary classification is often made implicit in the loss function by including a "bias" term in the predictor, we must make the thresholds explicit for ordinal regression to allow for multiple thresholds. Common choices for the binary classification loss function are the logistic loss (as in logistic regression), and the hinge loss (distance from the classification margin) used in the Support Vector Machine. Here, we consider various generalizations to these loss functions suitable for multiple-level discrete ordinal labels.

5.2.1 Related Work

Probabilistic approaches

McCullagh (1980) introduces the Proportional Odds model, which can be viewed as the basis of the Ordinal Regression algorithms we present in this section. The Proportional Odds model learns a linear map from examples to the real number line and uses a set of thresholds to define segments corresponding to the ordinal rating values. McCullagh focuses on a specific model which is similar to what we call immediate-thresholds with a Logistic Regression MPF. However, Proportional Odds cannot be described as a sum of MPFs, and so cannot easily be extended to an all-thresholds loss function. We provide a more technical discussion of Proportional Odds at the end of this section (§ 5.2.7). Other statistical models, such as the adjacent category odds model (Clogg & Shihadeh, 1994) and the continuation ratio model (Feinberg, 1980), vary in how they define the likelihood or cumulative likelihood of a label (Fu & Simpson, 2002), but otherwise are largely similar to Proportional Odds.

Chu and Ghahramani (2004) introduce a Gaussian Processes approach to Ordinal Regression. The core of their model can be seen as identical to Proportional Odds except for the substitution of the sigmoid function for the Gaussian CDF in the likelihood. The resulting loss penalizes margin violations in an approximately quadratic fashion, possibly leading to the model being overly sensitive to outliers. We provide a more detailed discussion at the end of this section.

Rennie and Srebro (2005b) introduce the Ordistic model, which differs from other probabilistic models in that each label class is represented in terms of a mean value, rather than as segments of the real number line defined by thresholds.

Loss Threshold-based approaches

Shashua and Levin (2003) suggest a generalization to the Support Vector Machine (SVM): the single margin constraints (for each observation) of standard SVMs are replaced with a pair of margin constraints on the thresholds bounding the "correct" region (the region corresponding to the target label).

When slack is allowed, Shashua and Levin's approach can be seen as regularized regression with a specific generalization to the hinge loss, which we describe in Section 5.2.5 as the *immediate-thresholds* generalization of the hinge loss. In Section 5.2.5 we discuss a different generalization, the *all-thresholds* generalization, where constraints (and slack) are considered for all $K-1$ thresholds—not only those immediately bounding the correct region. We argue that such a generalization better penalizes predictions which violate multiple thresholds and present experimental evidence supporting this argument. We also discuss how other margin penalty functions, such as the logistic, and modified least squares, can also be generalized in the same way.

Other approaches

Crammer and Singer (2002) provide a perceptron-style online update algorithm for Ordinal Regression. They map examples to the real line and use thresholds similar to McCullagh. On-line updating is done by treating each threshold as representing a separate, but related binary classification problem. Updates for multiple thresholds are combined to make a single update per example. Their online algorithm performs updates to minimize absolute error (difference between correct label and predicted label). The error that they are minimizing

is equivalent to the all-thresholds loss function (§ 5.2.5) when a sign agreement margin penalty function is used.

We briefly mention another approach suggested for handling discrete ordinal ratings. Herbrich et al. (2000) suggest extracting from the rating levels binary comparison relationships on the rated items and thus mapping the problem to a partial ordering problem. Herbrich et al. then study a generalized SVM for learning from binary comparison relationships. A drawback of this approach is the number of order constraints on T items with observed labels can be of order T^2, even though the original input to the problem (the observed labels) is only linear in T.

Ranking

Ordinal Regression (or more generally, "rating") can be viewed as a specific type of partial ordering, where each example is placed into one of a number of equivalence sets and the equivalence sets are given a total ordering. This relation is evident in work by Cohen et al. (1998), where a weighting of expert opinions is used to yield a ranking on items. Each expert specifies a rating of the items (mapping of the items into totally ordered equivalence sets); since each expert has his/her own rating style/system, the authors are able to use the overlapping ratings to establish a (nearly) total ranking of the items.

5.2.2 Specific contribution

The main contribution of this section is to study, in a systematic way, various loss functions for (discrete) ordinal regression. Reflecting the fact that ordinal regression is a generalization of binary classification, we separate the choice of a loss function for ordinal regression into two parts: (1) the selection of a *margin penalty function* (MPF), and (2) the selection of a *construction*, or particular combination of MPFs. Since our main interest is in how to handle discrete ordinal labels, we focus on regularized linear prediction in a simple learning setting, which we clarify in Section 5.2.3. In Section 5.2.4 we review various binary classification loss functions, their corresponding margin penalty functions and we discuss their properties. In Section 5.2.5 we present the immediate-thresholds and all-thresholds constructions, each of which can utilize any of the MPFs discussed in section 5.2.4. In Section 5.2.6 we compare the various methods through experiments using the various convex MPFs, and compare them also to standard multi-class and sum-squared-error regression approaches.

5.2.3 Preliminaries

Since our main object of interest is how to handle discrete ordinal labels, we focus on a simple learning setting in which we can demonstrate and experiment with various loss functions. We are given a training set $(x_i, y_i)_{i=1\ldots n}$ of n rated items, where for each item, $x_i \in \mathbb{R}^d$ is a feature vector describing the item and $y_i \in \{1, \ldots, l\}$ is the rating level for the item. We want to predict ratings that the user would give to future items. We do so by learning a *prediction mapping*, $z(x) : \mathbb{R}^d \to \mathbb{R}$, and set of thresholds, $\theta \in \mathbb{R}^l$, such that for an item with feature vector x, $z(x)$ corresponds as well as possible to the appeal of the item. We investigate different *loss functions*, $\text{loss}(z; y, \theta)$, for measuring the goodness of the correspondence between the mapping, $z(x)$, the target rating level, y, and the thresholds, θ.

In this paper, we focus on L_2-regularized linear prediction, where $z(x; w) = w^T x$ is a linear function of x, parameterized by a weight vector $w \in \mathbb{R}^d$. We seek a linear predictor

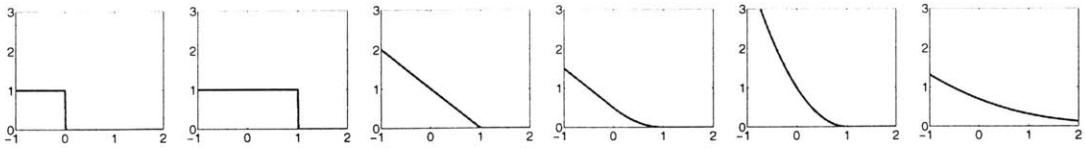

Figure 5-1: Different margin penalty functions (left to right): (1) sign agreement, (2) margin agreement, (3) hinge, (4) smooth hinge, (5) modified least squares, (6) logistic regression.

that minimizes a trade-off between the overall training loss and the squared (Euclidean) norm of the weights:

$$J(w, \theta) = \sum_i \text{loss}(w^T x_i; y_i, \theta) + \frac{\lambda}{2}\|w\|_2^2, \tag{5.1}$$

where λ is a trade-off parameter set using (e.g.) cross-validation. Note that by using $l = 2$ and substituting the appropriate loss function, we can arrive at the minimization objective for a variety of linear predictor binary classification models (e.g. SVM, logistic regression, least squares).

5.2.4 Margin Penalty Functions

The ordinal regression objective (5.1) with $l = 2$ is a common choice for binary classification. Sign values, $y \in \{-1, +1\}$, are typically used in place of the ordinal values for convenience. The per-example loss is typically defined in terms of a non-negative, real-valued function, what we will call the margin penalty function (MPF),

$$\text{loss}(z; y, \theta) = \text{MPF}(y(z - \theta)). \tag{5.2}$$

Input to the MPF is the real-valued predictor, shifted and oriented so that zero corresponds to the decision, a positive value corresponds to a correctly classified example and a negative value corresponds to an incorrectly classified example. As such, MPFs are usually monotonic, non-decreasing functions.

Many well-known binary classifiers can be defined in this way. For the remainder of this section, we discuss various MPFs and their corresponding binary classifiers. Note that a binary label is inferred for an example by taking the sign of the linear predictor after the threshold is subtracted: $\hat{y}(x) = w^T x - \theta$.

As we will discuss in section 5.2.5, MPFs are instrumental in our development of loss functions for ordinal regression. In particular, they can be "plugged" into threshold-based constructions to achieve ordinal regression loss functions. We go into some detail regarding aspects of these margin penalty functions (MPFs), as understanding them will be valuable for our later discussions.

Sign Agreement Our objective in binary classification is to be able to correctly predict a binary label. The simplest conceivable binary classifier is one that selects the weight vector, w, and threshold, θ, so as to minimize the number of errors on the training data. The

corresponding MPF is the negation of a slight variation of the Heaviside[2] step function,

$$\mathrm{MPF}_{\mathrm{Sign}}(z) = -H(z) = \begin{cases} 0 & \text{if } z > 0 \\ 1 & \text{if } z \leq 0 \end{cases} \tag{5.3}$$

This is also known as *zero-one error* and is the quantity that we would like to minimize on future, unobserved examples. However, using this simple loss function is problematic for several reasons:

1. It is insensitive to the magnitude of z, and thus the magnitude of w. Regularizing w is therefore meaningless, as shrinking w towards zero would yield the same error, but with a regularization term approaching zero.

2. It is not convex, and minimizing it is a difficult (in fact, NP-hard) optimization problem.

3. It is not continuous, let alone differentiable, and so even local optimization is difficult.

Margin The first problem can be addressed by requiring not only that z predict y correctly, but that it does so with a margin:

$$\mathrm{MPF}_{\mathrm{Margin}}(z) = \begin{cases} 0 & \text{if } z \geq 1 \\ 1 & \text{if } z < 1 \end{cases} \tag{5.4}$$

The corresponding loss function is sensitive to the magnitude of z, and therefore also to the magnitude of w. Summing this loss function corresponds to counting the number of violations of the constraints $y(w^T x - \theta) \geq 1$. Rewriting these constraints as $y\left(\frac{w^T}{\|w\|_2}x - \frac{\theta}{\|w\|_2}\right) \geq \frac{1}{\|w\|_2}$, we can interpret $\frac{1}{\|w\|_2}$ as a geometrical margin around the separating hyperplane, specified by its normal $\frac{w}{\|w\|_2}$. Minimizing the loss plus L_2 regularizer $\|w\|_2^2$ (5.1) can therefore be interpreted as maximizing the separation margin $M = \frac{1}{\|w\|_2}$ while minimizing the number of training points not classified correctly with a margin of at least M. Note that the Margin MPF retains the second and third drawbacks of the Sign Agreement MPF.

Hinge Using the Margin MPF (5.4) for binary classification might be worthwhile, but it (and its corresponding loss) are neither convex nor continuous. A common approach to large-margin classification is therefore to use the *Hinge* MPF,

$$\mathrm{MPF}_{\mathrm{Hinge}}(z) = (1 - z)_+ = \begin{cases} 0 & \text{if } z \geq 1 \\ 1 - z & \text{if } z < 1 \end{cases} \tag{5.5}$$

The corresponding Hinge loss is minimized for soft-margin Support Vector Machine (SVM) classification. In the context of SVMs, the Hinge loss is usually written as a sum over margin violations ξ_i included in the constraints $y(w^T x - \theta) \geq 1 - \xi_i$.

An important property of the Hinge MPF is that it is an upper bound on the Sign Agreement MPF (5.3); thus, the Hinge loss is a bound on the number of training data errors. Thus, any generalization error bounds on the Hinge loss on unseen data also provide

[2]The Heaviside function is typically defined with $H(0) = 0.5$.

a bound on the number of misclassifications on said unseen data, which is the true object of interest.

Since the Hinge MPF is convex, we can easily analyze the minimum Hinge loss solution easily. Ignoring any regularization term, the minimum is achieved when the sum of gradients of the Hinge losses is zero. Margin violations of positive and negative examples correspond to gradients of -1 and $+1$ (respectively). So, without regularization, the hinge MPF yields a solution where equal numbers of positive and negative examples incur margin violations, $y(w^T x_i - \theta) \leq 1$. Note that the gradient is discontinuous—points on the margin can be considered to have either a 0 or ± 1 gradient.

Other MPFs share properties of the hinge, such as convexity, continuity and sensitivity to w, but are easier to minimize since they have a smooth derivative. We describe these alternate MPFs presently.

Smooth Hinge loss The *Smooth Hinge* MPF, introduced by Rennie and Srebro (2005a), is an approximation to the Hinge MPF that is easier to minimize:

$$\text{MPF}_{\text{Smooth}}(z) = \begin{cases} 0 & \text{if } z \geq 1 \\ (1-z)^2/2 & \text{if } 0 < z < 1 \\ 0.5 - z & \text{if } z \leq 0 \end{cases} \tag{5.6}$$

The gradient of the Smooth Hinge is identical to that of the hinge except for $z \in (0, 1)$. As such, the minimum Smoothed Hinge loss solution is similar to the minimum Hinge loss solution. However, whereas the Hinge solution equates the number of positive and negative examples which violate the margin, the Smooth Hinge solution counts an example in the margin (but on the correct side of the decision boundary) as a partial example, so examples are weighted according to the severity of the margin violation.

The Smooth Hinge is not a bound on zero-one error, but twice the Smooth Hinge is. Since the appropriate trade-off parameter (from eqn. 5.1) is problem-specific, the scale of the MPF is irrelevant. I.e. if $\exists \alpha$ with $1 < \alpha < \infty$ s.t. $\alpha \text{MPF}(z) > \text{MFP}_{\text{Sign}}(z)\ \forall z$, then the binary classification loss corresponding to $\text{MPF}(z)$ is effectively a bound on zero-one error.

Modified Least Squares Zhang and Oles (2001) suggest a different MPF with a smooth derivative:

$$\text{MPF}_{\text{MLS}}(z) = \begin{cases} 0 & \text{if } z \geq 1 \\ (1-z)^2 & \text{if } z < 1 \end{cases} \tag{5.7}$$

The modified least squares MPF is much more sensitive to the magnitude of violations. Whereas the Smooth Hinge MPF is only sensitive to magnitude within the margin ($z \in (0, 1)$), and the Hinge MPF is insensitive to the degree of violation, the Modified Least Squares MPF is continuously sensitive to a violation of any magnitude. Adding a single outlier in the training data can severely affect the Modified Least Squares solution, whereas a single outlier will have a limited affect on the Hinge and Smooth Hinge solutions.

Logistic Regression Logistic Regression can be described in terms of the objective and loss functions we have discussed. The conditional likelihood of an example under the Logistic

Figure 5-2: Shown are ordinal regression loss functions with the hinge MPF and two different constructions: (left) immediate-thresholds, and (right) all-thresholds. Both correspond to an example with label $y = 4$. Note that the slope of the immediate-thresholds loss remains constant outside the label segment, whereas for all-thresholds, the slope increases with each margin/threshold.

Regression model is

$$P(y|x) = \frac{1}{1 + \exp(-y(x^T w - \theta))}, \tag{5.8}$$

where $y \in \{-1, +1\}$. The per-example loss corresponding to (5.1) is the negative log conditional likelihood, $-\log P(y|x)$. The corresponding MPF is

$$\mathrm{MPF}_{\mathrm{LR}}(z) = \log(1 + e^{-z}). \tag{5.9}$$

The L_2 regularizer as in (5.1) corresponds to maximum a-posteriori (MAP) estimation with a Gaussian prior on the weight vector w.

Like the Hinge MPF, the sensitivity of the Logistic Regression MPF to outliers is limited since the magnitude of the gradient of the MPF is bounded by 1. However, the Logistic Regression MPF has a "perfectionist" tendency in that it decreases continuously as $z \to \infty$. I.e. a solution is encouraged where (correctly classified) examples are as distant as possible from the decision boundary.

5.2.5 Ordinal Regression Loss Functions

To extend binary loss to the case of discrete ordinal regression, we introduce $K-1$ thresholds $\theta_1 < \theta_2 < \cdots < \theta_{K-1}$ partitioning the real line to K segments. The exterior segments are semi-infinite, and for convenience we denote $\theta_0 = -\infty$ and $\theta_K = +\infty$. Each of the K segments corresponds to one of the K labels and a predictor value: a value $\theta_{y-1} < z < \theta_y$ (the yth segment) corresponds to a rating of y. This generalizes the binary case, where we use a single threshold separating the real line into two semi-infinite segments: (1) $z < \theta$, corresponding to negative labels $y = -1$, and (2) $z > \theta$, corresponding to positive labels. Often in binary classification, a threshold of zero is used, but a bias term is added to weight vector, in effect, performing the same role as the threshold. The $K - 1$ thresholds replace the single bias term/threshold.

We restrict our attention to a class of two-part loss functions. Each loss function is composed of: (1) a margin penalty function (MPF), and (2) a threshold based construction. Each loss function is simply a sum of copies of the specified MPF. The threshold-based construction specifies the orientation, number and locations of the MPFs.

Beyond Feature-Based Learning

We focus on linear predictors using explicit features, but we note that the methods discussed here can also be "kernelized," as in Support Vector Machines. Both the immediate-thresholds and all-thresholds constructions with a hinge loss are generalizations of the SVM and can be stated as quadratic programs to which optimization techniques typically employed in learning SVMs apply. In fact, Shashua and Levin (2003) introduced the immediate-thresholds construction in the context of SVMs.

Immediate-Thresholds

Immediate-thresholds combined with the Hinge MPF was originally introduced by Shashua and Levin (2003) under the name "fixed margin". Our description here is slightly more general in that we detach the "construction" from the MPF.

For the immediate-thresholds construction, we consider, for each labeled example (x, y), only the two thresholds defining the "correct" segment (θ_{y-1}, θ_y), and penalize violations of these thresholds:

$$\text{loss}_{\text{I}}(w^T x; y, \theta) = \text{MPF}(w^T x - \theta_{y-1}) + \text{MPF}(\theta_y - w^T x), \tag{5.10}$$

where $z(x) = w^T x$ is the predictor output for the example. Figure 5-2 (left) gives an example visualization for $y = 4$. Note that if MPF is an upper bound on the sign agreement MPF, then the immediate-threshold loss is an upper bound on the misclassification (zero-one) error. The immediate-threshold loss is ignorant of whether multiple thresholds are crossed.

When a convex MPF is combined with immediate-thresholds, it yields a convex loss function. The Hinge MPF minimum loss solution equates the number of optimistic predictions (rating too high) with the number of pessimistic predictions (rating too low).

All-Thresholds

All-thresholds was originally introduced by Srebro et al. (2005) (briefly in §6), and later published by Chu and Keerthi (2005). Our descriptions here are slightly more general in that we detach the "construction" from the MPF and consider a range of MPFs, not only the hinge function (as in both Srebro et al. (2005) and Chu and Keerthi (2005)).

In a simple multi-class classification setting, all mistakes are equal; there is no reason to prefer one incorrect label over another. This is not true in ordinal regression. Since the labels are ordered, some mistakes are better than others. For example, if someone adores a movie ("5 stars"), it is better that the system predict a moderately high rating ("4 stars") than a low rating ("1 star"). This is reflected in the evaluation criterion used for ordinal regression. It is typical to use absolute error—the absolute difference between the true and predicted labels—as the evaluation criterion. Immediate-thresholds does not bound absolute error and the immediate-threshold gradient does not reflect the absolute error criterion.

We introduce a loss function, all-thresholds, that bounds mean absolute error. To bound absolute error, we must introduce a MPF at each threshold. A sum of Sign Agreement MPFs, one located at each threshold, and oriented appropriately, is equal to the absolute error. In order for the loss to bound absolute error, we must place an MPF at each threshold and use an MPF which bounds the Sign Agreement MPF. To achieve the proper orientations

	Multi-class Test MAE	Imm-Thresh Test MAE	All-Thresh Test MAE	
Mod. Least Squares	0.7486	0.7491	**0.6700**	(1.74e-18)
Smooth Hinge	0.7433	0.7628	**0.6702**	(6.63e-17)
Logistic	0.7490	0.7248	**0.6623**	(7.29e-22)
	Multi-class Test ZOE	Imm-Thresh Test ZOE	All-Thresh Test ZOE	
Mod. Least Squares	0.5606	0.5807	**0.5509**	(7.68e-02)
Smooth Hinge	0.5594	0.5839	**0.5512**	(1.37e-01)
Logistic	0.5592	0.5699	**0.5466**	(2.97e-02)

Table 5.1: Mean absolute error (MAE) and zero-one error (ZOE) results on MovieLens. For each construction/loss and error type, we selected the regularization parameter with lowest validation error. Numbers in parentheses are p-values for all-threshold versus the next best construction. As a baseline comparison, simple sum-squared-error (L2) regression achieved test MAE of 1.3368 and test ZOE of 0.7635.

of MPFs, we define $s(j;y) = \begin{cases} -1 & \text{if } j < y \\ +1 & \text{if } j \geq y \end{cases}$. Then the all-threshold loss is

$$\text{loss}_{\text{A}}(w^T x; y, \theta) = \sum_{j=1}^{l-1} \text{MPF}\Big(s(j;y)(\theta_j - w^T x)\Big). \tag{5.11}$$

Figure 5-2 (right) gives an example visualization for $y = 4$ and the Hinge MPF. Note that the magnitude of the slope increases with each threshold margin crossing.

When a convex MPF is used for all-thresholds, the result is a convex loss function. The Hinge MPF minimum loss solution balances a the number of optimistic margin violations with the number of pessimistic margin violations. Whereas the immediate-thresholds solution only accounts for immediate margin violations, the all-thresholds solution counts margin violations for all-thresholds.

Learning Thresholds

Fitting an ordinal regression models involves fitting the parameters of the predictor, e.g. $z(x) = w^T x$, as well as the thresholds $\theta_1, \ldots, \theta_{l-1}$. Learning the thresholds, rather than fixing them to be equally spaced, allows us to capture the different ways in which users use the available ratings, and alleviates the need for per-user rating normalization. In a setting in which multiple users are considered concurrently, e.g. in collaborative prediction, a different set of thresholds can be learned for each user.

5.2.6 Experiments

To determine the appropriateness of the different constructions discussed earlier, we conducted experiments on a well-known collaborative filtering data set. We implemented the two threshold-based constructions discussed in Section 5.2.5 (combined with a variety of MPFs), as well as multi-class classification and sum-squared error regression to compare against.

We used the "1 Million" MovieLens data set for evaluation. The data set contains 1,000,209 rating entries, made by 6040 users on 3952 movies. Similar to the work of Crammer and Singer (2002) and Shashua and Levin (2003), we used the ratings of the top 200 users to predict the ratings of the remaining users. I.e. each of the "remaining" users' ratings were predicted as a weighted sum of the top 200 users' ratings. To deal with "missing" ratings, we subtracted the user's mean rating and filled-in empty values with zero. We used the remaining 5840 users for evaluation. For each user, we used one randomly selected rating for testing, another for validation and the remaining ratings for training. We limited our experiments to the top 2000 movies to ensure a minimum of 10 ratings per movie. This gave us test and validation sets each of 5,840 ratings and a training set of 769,659 ratings.

For each method (combination of construction method and MPF), and range of values of the regularization parameter λ, we fit weight and threshold vectors for each user by minimizing the convex objective (5.1) using conjugate gradient descent[3]. We calculated mean absolute error (MAE) and zero-one error (ZOE) between predicted and actual ratings on the validation set and used the regularization parameter with the lowest validation set MAE/ZOE for test set evaluation.

Table 5.1 shows test MAE and ZOE for various constructions and MPFs. Across all MPFs, all-threshold yields the lowest MAE. The MAE differences between all-threshold and the next-best construction (multi-class or imm-thresh depending on loss function) are highly significant according to a non-parametric, two-tailed binomial test—the largest p-value is 6.63e-17. Interestingly, all-threshold also yields lower ZOE, though the comparison with multi-class classification (the next-best construction) is not conclusive (p-values $\in (0.03, 0.14)$)[4]. Performance of the immediate-threshold construction seems poor, performing roughly the same as multi-class for MAE and worse than multi-class for ZOE.

Results indicate that the choice of construction is more important than threshold penalty function—all-threshold with the worst-performing penalty function yields lower MAE and ZOE than the best non-all-threshold construction. However, it appears that the logistic loss tends to work best; in particular, the differences in MAE between logistic and other penalty functions (for the all-threshold construction) are significant at the $p = 0.01$ level (largest p-value is 9.52e-03) according to the two-tailed binomial test.

5.2.7 Related Models

Before concluding, we provide an extended discussion of McCullagh's Proportional Odds model (1980) and Chu and Ghahramani's Gaussian Processes model, comparing them to the loss-based constructions we have discussed above.

Proportional Odds

McCullagh's Proportional Odds model (1980) assumes that:

- Examples are represented as d-dimensional real-valued feature vectors, $x \in \mathbb{R}^d$,

[3]Note that any simple gradient descent method (such as Conjugate Gradients) requires that the objective has a smooth gradient. Thus, we restricted ourselves to MPFs with a smooth gradient. Optimization of the Hinge MPF requires (e.g.) a sub-gradient optimization method.

[4]The difference between all-thresholds and immediate-thresholds is significant for ZOE—the largest p-value is 7.84e-6 (not shown).

- Each example has an underlying score, defined by the dot-product between the feature vector and a weight vector, $w \in \mathbb{R}^d$,
- The real number line is segmented via a set of threshold values, $-\infty \equiv \theta_0 < \theta_1 < \theta_2 < \cdots < \theta_{l-1} < \theta_l \equiv +\infty$, and
- Each example is associated with a discrete, ordinal label, $y \in \{1, \ldots, l\}$,
- Each label, y, is associated with a segment of the real number line, (θ_{y-1}, θ_y).

These assumptions match those that that we use for development of our models. Define $g(z) \equiv \frac{1}{1+e^{-z}}$, the sigmoid function. Proportional Odds defines the cumulative likelihood of an example being associated with a label less-than-or-equal-to j (for $j \leq l-1$) as the sigmoid function,

$$P_{\text{PO}}(y \leq j|x) = g(\theta_j - w^T x) = \frac{1}{1 + \exp(w^T x - \theta_j)}. \tag{5.12}$$

By definition, $P(y \leq l|x) = 1$. Note that $P(y \leq 1|x) \equiv P(y = 1|x)$. We calculate other likelihoods by taking cumulative differences,

$$\begin{aligned} P_{\text{PO}}(y = j|x) &= P_{\text{PO}}(y \leq j|x) - P_{\text{PO}}(y \leq j-1|x) \qquad \text{for } j \geq 2, \\ &= \frac{1}{1 + \exp(w^T x - \theta_j)} - \frac{1}{1 + \exp(w^T x - \theta_{j-1})}. \end{aligned} \tag{5.13}$$

Parameters (w, θ) are learned via maximum likelihood. One can "modernize" Proportional Odds by introducing a Gaussian prior on the weight vector (w), which would serve to regularize the predictor. Also, a non-linear predictor can be added to Proportional Odds without increasing the memory footprint via the kernel trick (Vapnik, 1995).

One might also include a (e.g. Gaussian) prior for the weight vector, which would correspond to the L_2-norm penalty commonly used in SVMs and other classification algorithms.

Comparison with Proportional Odds The "loss" function for immediate-thresholds with a Logistic Regression MPF is

$$\begin{aligned} \text{Loss}_{\text{IM}}(j|x) &= \log\left[1 + \exp(w^T x - \theta_j)\right] + \log\left[1 + \exp(\theta_{j-1} - w^T x)\right] & (5.14) \\ &= -\log\left[g(\theta_j - w^T x)\right] - \log\left[g(w^T x - \theta_{j-1})\right] & (5.15) \end{aligned}$$

Note that we can artificially reproduce this from Proportional Odds by treating each $y = j$ event as two events: $y \leq j$ and $y > j - 1$. Treating these two events as independent, we get an (unnormalized) likelihood of

$$P_{\text{IM}}(y = j|x) \propto \frac{1}{\left[1 + \exp(w^T x - \theta_j)\right]\left[1 + \exp(\theta_{j-1} - w^T x)\right]}, \tag{5.16}$$

which corresponds[5] exactly with the IM loss. Note that we can similarly artificially reproduce All-Thresholds (w/ Logistic Regression MPF) by splitting the $y = j$ event into $l - 1$ events.

[5]The loss function for a probabilistic model is the negative log probability.

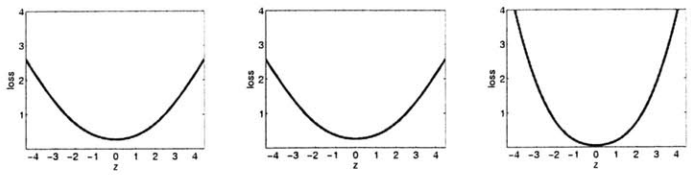

Figure 5-3: Loss functions for Proportional Odds (left), Immediate Thresholds (center), and Gaussian Processes (right). The x-axis is the predictor output, $z \equiv w^T x$; the y-axis is the loss (negative log-likelihood). The thresholds are fixed at $\theta_j = 2$ and $\theta_{j-1} = 2$.

Figure 5-3 compares the loss functions for Proportional Odds and immediate-thresholds using $z \equiv w^T x$ as a single parameter (as well as Gaussian Processes, discussed in section 5.2.7). The two plots (for PO and IM) look strikingly similar. In fact, they are identical save for a small constant difference of 0.0185 (PO loss is larger). Why is this? Consider the difference between loss functions:

$$\Delta \equiv \text{Loss}_{\text{IM}}(j|x) + \log P(y = j|x) = \log\left[g(\theta_j - w^T x) - g(\theta_{j-1} - w^T x)\right] - \log\left[g(\theta_j - w^T x)\right] - \log\left[g(w^T x - \theta_{j-1})\right]. \quad (5.17)$$

Now, consider the partial derivative of the difference with respect to w,

$$\frac{\partial \Delta}{\partial w} = 1 - \frac{\exp(\theta_{j-1} - w^T x)}{1 + \exp(\theta_{j-1} - w^T x)} - \frac{\exp(w^T x - \theta_{j-1})}{1 + \exp(w^T x - \theta_{j-1})} = 0 \quad (5.18)$$

A change in the weight vector, w, does not affect the difference in loss functions. This explains the near-identical graphs in Figure 5-3. Now, consider the partial derivative with respect to a threshold, θ_j,

$$\frac{\partial \Delta \text{Loss}}{\partial \theta_j} = \frac{\exp(w^T x - \theta_j)}{\exp(w^T x - \theta_{j-1}) - \exp(w^T x - \theta_j)} = \frac{1}{\exp(\theta_j - \theta_{j-1}) - 1} \quad (5.19)$$

Note that $\theta_j \geq \theta_{j-1}$ so the partial derivative is non-negative. We see that the loss difference *is* affected by the threshold parameters. If we had compared Proportional Odds and immediate-thresholds with a different upper threshold (θ_j), we would have found a different constant difference between the loss functions.

We conclude that if the thresholds are fixed a priori, then the two models (Proportional Odds and immediate-thresholds) are equivalent. However, the models are affected differently by the settings of the threshold values.

Gaussian Processes

Chu and Ghahramani (2004) introduce a Gaussian Process framework for Ordinal Regression. Similar to Proportional Odds, they introduce a predictor function and a set of thresholds. They also use a probabilistic model which is normalized over the set of labels.

They posit an "ideal" likelihood[6],

$$P_{\text{ideal}}(y = j|w^T x) = \begin{cases} 1 & \theta_{j-1} < w^T x \leq \theta_j \\ 0 & \text{otherwise} \end{cases}, \tag{5.20}$$

and assume a Gaussian prior over the data point location to account for noise in the data. They integrate the "ideal" likelihood over the Gaussian prior to arrive at the likelihood,

$$P(y = j|x) = \int P_{\text{ideal}}(y = j|w^T x + \delta)\mathcal{N}(\delta; 0, \sigma^2)d\delta = \Phi(z_1) - \Phi(z_2), \tag{5.21}$$

where $z_1 = \frac{\theta_j - w^T x}{\sigma}$, $z_2 = \frac{\theta_{j-1} - w^T x}{\sigma}$, $\Phi(z) = \int_{-\infty}^{z} \mathcal{N}(z; 0, 1)dz$ and $\mathcal{N}(\delta; 0, \sigma^2)$ denotes a Gaussian PDF with random variable δ, zero mean and σ^2 variance.

Comparison with Proportional Odds Recall that the likelihood for Proportional Odds is the difference between two sigmoids (5.13). The likelihood for Chu and Ghahramani's model is the difference between two Gaussian CDFs. If we replace the Gaussian PDF in (5.21) with the derivative of the sigmoid, we arrive at the Proportional Odds likelihood,

$$P(y = j|x) = \int P_{\text{ideal}}(y = j|w^T x + \delta)g(\delta)(1 - g(\delta))d\delta \tag{5.22}$$

$$= g(\theta_j - w^T x) - g(\theta_{j-1} - w^T x). \tag{5.23}$$

I.e. at its core, Chu and Ghahramani's model is Proportional Odds with a swap of the sigmoid for a Gaussian CDF. As can be observed in Figure 5-3, Proportional Odds imposes an approximately linear penalty on margin violations, whereas Chu and Ghahramani's model imposes an approximately quadratic penalty. As a result, Chu and Ghahramani's model may be overly sensitive to outliers.

5.2.8 Ranking

Though we have focused on the task of rating (assigning a discrete ordinal value to each item), the techniques we use are readily applicable to the task of ranking (ordering a set of items). Note that any Ordinal Regression model trained according to our objective (5.1) can ran a set of items simply according to the predictor output: $\hat{w}^T x_i$, where $\hat{w}$ is the learned weight vector. Furthermore, if we are given ranked training data, we can apply an all-thresholds-type construction by interpreting each pair of examples as an example and threshold. An MPF is introduced for each example pair where the input to the MPF is the predictor output difference: $w^T(x_1 - x_2)$. When there are a large number of examples, it may be a waste to introduce an MPF for *every* pair; it may be sufficient to only introduce an MPF for pairs of examples within (say) 5 or 10 rank-values of each other.

5.2.9 Summary

Our motivation for this work was to be able to synthesize opinions and preferences expressed about items in informal communication, such as those expressed about restaurants on a

[6]For simplicity and consistency, we use a linear predictor here. Chu and Ghahramani's framework allows for general (non-linear) predictors.

restaurant bulletin board. We assumed that each item can be represented as attribute values and that opinions/preferences are given to us in terms of discrete, ordered values, which might correspond to different words used to describe the item. We detailed the construction of loss functions for this domain (ordinal regression) and recommended the use of one in particular (all-thresholds), which is well motivated, and performed very well in our experimental evaluation.

5.3 Collaborative Filtering

In the previous section, we conducted a study of preference learning based on two key assumptions: (1) preferences are given in terms of discrete, ordered ratings, and (2) we can encapsulate information about each of the items in a vector of attribute values (feature vector). The first assumption seems reasonable for our informal communication setting since adjectives and words used by a user to describe an item are, effectively, a personalized, discrete rating system. The second assumption seems too strong since it can be difficult or impossible to observe and/or quantify all aspects of items that may affect a user's rating. Consider the restaurant scenario. It may be easy to gather information on cuisine type, menu items, ownership, and staffing. But, service, decor and the balance & complexity of the food are all aspects which may affect a user's rating of a restaurant, yet cannot easily be quantified.

In this section, we discuss an alternate paradigm, "collaborative filtering", which eliminates assumption #2, but makes its own assumption—that preference information can be deduced by observing how multiple users rate a variety of items. In ordinal regression, each user is treated separately. Yet, when users express opinions on a common set of items, the overlap of opinions can be used to generalize to unseen user/item pairs—we do not need to collect features/attributes of the individual items. This is a natural extension of the experiments conducted in the previous section, where we used one set of user preferences as the attribute vectors for ordinal regression.

5.3.1 Introduction and Related Work

"Collaborative filtering" refers to the task of predicting preferences of users based on their preferences so far, and how they relate to the preferences of other users. For example, in a collaborative filtering movie recommendation system, the inputs to the system are user ratings on movies the users have already seen. Prediction of user preferences on movies they have not yet seen are then based on patterns in the partially observed rating matrix. The setting can be formalized as a matrix completion problem—completing entries in a partially observed data matrix Y. This approach contrasts with a more traditional feature-based approach, such as Ordinal Regression § 5.2, where predictions are made based on features of the movies (e.g. genre, year, actors, external reviews) and/or users (e.g. age, gender, explicitly specified preferences). Users "collaborate" by sharing their ratings instead of relying on external feature information.

A common approach to collaborative filtering is to fit a factor model to the data, and use it in order to make further predictions (Azar et al., 2001; Billsus & Pazzani, 1998; Hofmann, 2004; Marlin & Zemel, 2004; Canny, 2004). The premise behind a low-dimensional factor model is that there is only a small number of *factors* influencing decisions, and that a user's choices are determined by how the factors for a movie relate to each user's preferences. In

a linear factor model, movie factors and user preferences are each represented by a real-valued vector with one entry per factor. Thus, for m users and n items, the predicted ratings according to a k-factor model are given by the product of an $m \times k$ *preference matrix* U (each row representing a user's preferences) and a $n \times k$ *factor matrix* V (each row representing the factors, or features, of a movie). The rating matrices which admit such a factorization are matrices of rank at most k. Thus, training such a linear factor model amounts to approximating the observed ratings Y with a low-rank matrix X.

The low-rank matrix X that minimizes the sum-squared distance to a *fully observed* target rating matrix Y is given by the leading singular components of Y and can be efficiently found. However, in a collaborative filtering setting, only some of the entries of Y are observed, and the low-rank matrix X minimizing the sum-squared distance to the *observed* entries can not be computed in terms of a singular value decomposition. In fact, the problem of finding a low-rank approximation to a partially observed matrix is a difficult non-convex problem with many local minima, for which only local search heuristics are known (Srebro & Jaakkola, 2003).

Furthermore, especially when predicting discrete values such as ratings, loss functions other than sum-squared loss are often more appropriate: loss corresponding to a specific probabilistic model (as in pLSA (Hofmann, 2004) and Exponential-PCA (Collins et al., 2002)) or a bound on classification error such as the Hinge loss. Finding a low-rank matrix X minimizing loss functions other then squared-error is a non-convex optimization problem with multiple local minima, even when the the target matrix Y is fully observed[7].

Low-rank approximations constrain the dimensionality of the factorization $X = UV^T$, i.e. the number of allowed factors. Other constraints, such as sparsity and non-negativity (Lee & Seung, 1999), have also been suggested for better capturing the structure in Y, and also lead to non-convex optimization problems.

Recently, Srebro et al. (2005) suggested "Maximum Margin Matrix Factorization" (MMMF), constraining the *norms* of U and V instead of their dimensionality. Viewed as a factor model, this corresponds to constraining the overall "strength" of the factors, rather than their number. That is, a potentially infinite number of factors is allowed, but relatively few of them are allowed to be important. For example, when modeling movie ratings, there might be a very strong factor corresponding to the amount of violence in the movie, slightly weaker factors corresponding to its comic and dramatic value, and additional factors of decaying importance corresponding to more subtle features such as the magnificence of the scenery and appeal of the musical score.

Mathematically, constraining the norms of U and V corresponds to constraining the *trace norm*, or sum of singular values, of X. Interestingly, this is a convex constraint, and so finding a matrix X with a low-norm factorization minimizing any convex loss versus a partially (or fully) observed target matrix Y, is a convex optimization problem. This contrasts sharply with rank constraints, which are not convex constraints, yielding non-convex optimization problems as described above. In fact, the trace norm (sum of singular values) has also been suggested as a convex surrogate for the rank (number of non-zero singular values) in control applications (Fazel et al., 2001).

Fazel et al. (2001) show how a trace-norm constraint can be written in terms of a linear and semi-definite constraints. By using this form, Srebro et al. (2005) formulate

[7]The problem is non-convex even when minimizing the sum-squared error, but for this special case of minimizing sum-squared error over a fully observed target matrix, all local minima are global (Srebro & Jaakkola, 2003).

MMMF as semi-definite programming (SDP) and employ standard SDP solvers to find maximum margin matrix factorizations. However, such generic solvers are only able to handle problems with no more than a few tens of thousands of constraints, corresponding to about ten thousand observations (observed user-item pairs). This is far from the size of typical collaborative filtering problems, with thousands of users and items, and millions of observations.

In this section, we investigate methods for seeking a MMMF by directly optimizing the factorization $X = UV^T$. That is, we perform gradient-based local search on the matrices U and V. We call this strategy *Fast MMMF*. We show that the Fast MMMF objective is a bound on the original MMMF objective and that any solution to the Fast MMMF objective is a solution of the original MMMF objective. Though the Fast MMMF objective is not convex, empirical tests reveal no evidence of local minima. Using such methods, we are able to find MMMF solutions for a realistically-sized collaborative filtering data set, and demonstrate the competitiveness of MMMF versus other collaborative filtering methods.

5.3.2 Simultaneously Learning Weights and Features

Consider fitting an $m \times n$ target matrix Y with a rank-k matrix $X = UV^T$, where $U \in \mathbb{R}^{m \times k}$ and $V \in \mathbb{R}^{n \times k}$. If one of the matrices, say V, is fixed, and only the other matrix U needs to be learned, then fitting each row of the target matrix Y is a separate linear prediction problem. In the case that the entries of Y are discrete ordered values, then we will also learn a matrix of threshold values, $\theta \in \mathbb{R}^{m \times (l-1)}$, and fitting each row of Y is exactly an Ordinal Regression problem as discussed earlier. Each row of V serves as a "feature vector;" each row of U is a vector of preference weights. Their product yields a matrix $UV^T = X$ which is compared to the target matrix, Y, via a set of thresholds, θ, and an Ordinal Regression loss function (such as all-thresholds, § 5.2.5).

In collaborative filtering, U, V and θ are unknown and need to be estimated. This can be thought of as learning feature vectors (rows of V) for each of the columns of Y, enabling good linear prediction across all of the prediction problems (rows of Y) concurrently, each with a different linear predictor (row of U) and set of thresholds (row of θ). The features are learned without any external information or constraints which is impossible for a single prediction task. The underlying assumption that enables us to do this in a collaborative filtering situation is that the prediction tasks (rows of Y) are *related*, in that the same features can be used for all of them, though possibly in different ways.

Note that k bounds the rank of X. If $k < \min(m, n)$, learning of the approximation matrix X is rank-constrained. This is similar to limiting the number of features available for Ordinal Regression. A rank constraint is a common choice of regularization to limit the strength of the model in order to improve generalization. Instead, our framework for Ordinal Regression (§ 5.2) already has regularization in the form of a squared L_2 penalty (5.1), which encourages a solution where data are well separated from the thresholds.

Iterative Algorithm The symmetry of the collaborative filtering problem suggests a simple iterative solution. Allow U and V to be full rank, $k = \min(m, n)$, and initialize the parameters (U, V, and θ) to random (e.g. drawn from unit Normal) values. Learn U and θ while holding V fixed, effectively learning a Ordinal Regression (OR) model for each row of Y—each row of U (θ) corresponds to a weight (threshold) vector. Then, learn V while holding U and θ fixed. Here, the OR problems are tied and must be learned jointly, each row of V representing the feature vector for a single item across all OR problems. The

minimization objective for learning U and θ can be written as a set of OR objectives, or as a single, combined objective,

$$J_1(U,\theta) = \sum_{i,j \in S} \text{loss}(U_i V_j^T; Y_{ij}, \theta_i) + \frac{\lambda}{2}\|U\|_{\text{Fro}}^2, \tag{5.24}$$

where U_i is the i^{th} row of U, V_j is the j^{th} row of V, θ_i is the i^{th} row of θ, "loss" is one of the Ordinal Regression loss functions discussed in § 5.2.5, S is the set of observed (training) entries in Y, and $\|\cdot\|_{\text{Fro}}^2$ designates the squared Frobenius norm, which is simply a sum of squared entries of a matrix: $\|U\|_{\text{Fro}}^2 = \sum_{i,j} U_{ij}^2$. Optimization of V cannot be divided and must be learned jointly. The minimization objective for V is

$$J_2(V) = \sum_{i,j \in S} \text{loss}(U_i V_j^T; Y_{ij}, \theta_i) + \frac{\lambda}{2}\|V\|_{\text{Fro}}^2. \tag{5.25}$$

Each of these two objectives is simply a sum over users of the Ordinal Regression objective.

Note that for any loss function with a smooth derivative (including constructions based on the Modified Least Squares, Logistic Regression, and the Smooth Hinge MPFs), parameters (U, V and θ) for the above iterative algorithm may be learned via gradient descent style algorithms (such as Conjugate Gradients (Shewchuk, 1994) or L-BFGS (Nocedal & Wright, 1999)).

A disadvantage of the iterative algorithm is that it hinders the optimization procedure by separating the parameters. A primary challenge of optimization is to account for second and higher order effects. The iterative algorithm optimizes U and V separately, so any joint curvature in the objective function is not accounted for in the choice of descent directions. Such optimization may lead to a "zig-zag" phenomenon and exceedingly slow convergence, as discussed in §6.2 of Shewchuk (1994); see Figure 19 of Shewchuk (1994) for a visualization.

Joint Optimization We can remedy this issue by solving for U, V and θ jointly. We do this by optimizing what we call the "joint" objective (aka "Fast MMMF"). To jointly solve for U, V and θ, we simply combine regularization terms from the iterative objectives,

$$J(U,V,\theta) = \sum_{i,j \in S} \text{loss}(U_i V_j^T; Y_{ij}, \theta_i) + \frac{\lambda}{2}(\|U\|_{\text{Fro}}^2 + \|V\|_{\text{Fro}}^2). \tag{5.26}$$

Fast, first-order gradient descent algorithms, such as Conjugate Gradients and L-BFGS, will converge more quickly using this joint objective than using the aforementioned iterative algorithm due to the additional information available to the algorithm at each iteration.

One drawback with the joint objective (Fast MMMF) is that it cannot easily be kernelized. The kernelization typically employed in (e.g.) Support Vector Machines require that the "feature vector" be fixed. Here we are (in effect) simultaneously learning both features and weights. Thus, in order to learn kernelized features, we would need to calculate the gradient of the inverted kernel mapping. For polynomial kernels, this would not be difficult, but use of other kernels may prove difficult. Note that kernels may not be particularly useful in the context of MMMF. Whereas with linear binary prediction, the expression of the classifier is limited by the set of feature vectors, this is not the case with MMMF, where expression is only limited by the strength of the regularization.

Though we have presented the joint objective (5.26) as a simple generalization of Ordinal

Regression, solutions to the joint objective have an appealing property which is not clear from our development. The joint objective was originally introduced as a way to apply Maximum Margin Matrix Factorization to large data sets. In the next section, we detail this alternate development and discuss properties of the joint objective.

5.3.3 Maximum Margin Matrix Factorization

Maximum Margin Matrix Factorization (MMMF) was introduced by Srebro et al. (2005) as a framework for Collaborative Filtering. The all-thresholds, trace norm variant of MMMF uses a minimization objective closely related to the joint objective we have developed (5.26). However, the matrix (X) is not factorized and the trace norm of X is used in place of the sum of Frobenius norms of the factorized components,

$$J_{\text{MMMF}}(X, \theta) = \sum_{i,j \in S} \text{loss}(X_{ij}; Y_{ij}, \theta_i) + \lambda \|X\|_\Sigma, \tag{5.27}$$

where "loss" denotes an Ordinal Regression loss function (§ 5.2.5) and $\|X\|_\Sigma$ denotes the trace norm of X (the Σ subscript alluding to the fact that it is the *sum* of singular values). This objective cannot be optimized via simple gradient descent techniques. As observed by Srebro et al. (2005), the objective can be written as a semi-definite program (SDP). Modern SDP solvers can handle medium-size problems (e.g. a few thousand rating observations), but cannot cope with large data sets involving millions of rating observations. For this reason, the joint objective (5.26) has been suggested as an alternative to the original MMMF objective (5.27) (Rennie & Srebro, 2005a).

Trace Norm The trace norm of a matrix, $\|X\|_\Sigma$, is the sum of its singular values. See § 4.3 for additional background and intuition. The trace norm in the MMMF objective is used to encourage a low-rank solution. The trace norm penalizes the rank of X much in the same way that the hinge loss penalizes classification error and the L_1-norm penalizes non-zero weights. It is a convex function with a non-continuous derivative at each change in rank. One way which the trace norm can be defined is as the minimum over factorizations of X of the average Frobenius norm of the factors:

$$\|X\|_\Sigma = \min_{\substack{U,V \\ X=UV^T}} \frac{1}{2}(\|U\|_{\text{Fro}}^2 + \|V\|_{\text{Fro}}^2). \tag{5.28}$$

In other words, given a factorization of $X = UV^T$, the average Frobenius norm of the factors is a bound on the trace norm of X, $\|X\|_\Sigma \leq \frac{1}{2}(\|U\|_{\text{Fro}}^2 + \|V\|_{\text{Fro}}^2)$. Thus, our "joint" objective is actually a tight upper bound on the MMMF objective.

Consider substituting this trace norm definition into the MMMF objective:

$$J_{\text{MMMF}}(X, \theta) = \sum_{i,j \in S} \text{loss}(X_{ij}; Y_{ij}, \theta_i) + \frac{\lambda}{2} \min_{\substack{U,V \\ X=UV^T}} (\|U\|_{\text{Fro}}^2 + \|V\|_{\text{Fro}}^2). \tag{5.29}$$

To go from this to our joint objective, we must change parameters from X to U,V and eliminate the (now unnecessary) minimization over U,V in the regularization term.

The joint objective is a tight variational bound on the MMMF objective. The objectives are equal when $X = UV^T$ and U,V are the minimum average Frobenius norm factorization

of X. Generally speaking, $J_{\text{MMMF}}(UV^T, \theta) \leq J(U, V, \theta)$. We describe the bound as "variational" because additional parameters are introduced which make the objective easier to optimize.

5.3.4 Low-Rank Solutions

Here we discuss the reason that the MMMF and joint objectives tend to produce low-rank solutions. To begin, we discuss a simple setting involving linear prediction and L_1-norm regularization of the weight vector. The use of the trace norm in our collaborative filtering work is a generalization of this simple case. Whereas the L_1-norm encourages few non-zero parameter values, the trace norm encourages few non-zero singular values (i.e. low-rank).

L_1 Regularization for Linear Prediction

It is well known that L_1 regularization in linear prediction encourages a solution where many parameter values are null. A typical such objective is

$$J_{L_1}(w) = \sum_i \text{loss}(w^T x_i; y_i) + \lambda \|w\|_1, \tag{5.30}$$

where the sum of absolute values is the L_1 norm, $\|w\|_1 = \sum_j |w_j|$. If the loss function is convex, as is typical, then the minimum/solution is found where the gradient with respect to w is zero. The sub-differential of the L_1 norm at $w_j = 0$ is $[-1, +1]$, and it is ± 1 for all other values, $w_j \neq 0$. Let w^* denote a global minimum of J_{L_1}. In order that $w_j^* \neq 0$, it must be that the gradient of the loss with respect to w_j somewhere has magnitude $\geq \lambda$. Compare this to an L_2-norm regularized objective. The gradient of the L_2-norm is continuous, so any non-zero loss gradient is sufficient to offset the cost of a non-zero parameter value. The idea of using the L_1 norm to regularize supervised learning was introduced by Tibshirani (1996) as "Lasso".

Trace Norm

The trace norm, $\|X\|_\Sigma$ is an L_1 norm on the singular values of X. Let $W(w)$ be a diagonal matrix parameterized by w (values along the diagonal). Then, the following objective using trace norm regularization is equivalent to the L_1 objective (5.30),

$$J_\Sigma(w) = \sum_i \text{loss}(w^T x_i; y_i) + \lambda \|W(w)\|_\Sigma \equiv J_{L_1}(w). \tag{5.31}$$

When we apply the trace norm as regularization to an unconstrained matrix, as in the MMMF objective (5.27), it encourages few non-zero singular values in the same way that the L_1-norm encourages few non-zero weight values. See §5.1.7 of Fazel (2002) for additional discussion.

For additional intuition, we consider the trace norm of a 2×2 matrix. Note that the trace norm (1) is rotation invariant (i.e. $\|X\|_\Sigma = \|XR\|_\Sigma$ for $R \in O(n)$), and (2) the trace norm is associative with scalar multiplication (i.e. $\|cX\|_\Sigma = c\|X\|_\Sigma$). Thus, orientation and scale of one row of X is arbitrary. So, we only parameterize the second row:

$$X(x, y) = \begin{bmatrix} 1 & 0 \\ x & y \end{bmatrix}. \tag{5.32}$$

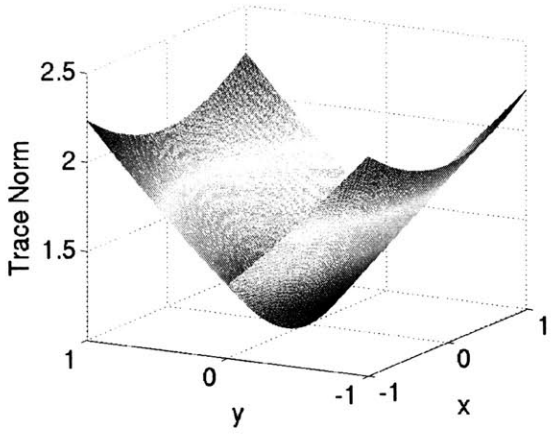

Figure 5-4: Shown is the trace norm of the matrix $X(x, y)$.

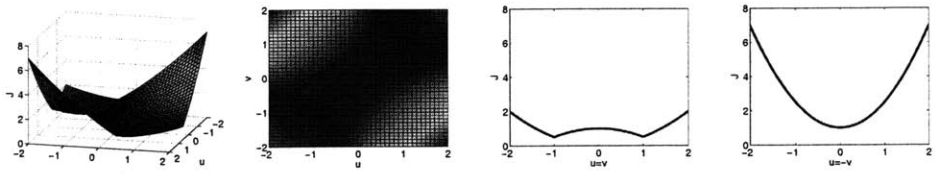

Figure 5-5: Visualization of the joint objective for a simple example from various viewpoints. From left-to-right, the first plot provides a general view of the surface; the second plot provides a top-down view, where colors indicate surface height, with red being the highest and dark blue being the lowest—note the two minima at $(1, 1)$ and $(-1, -1)$; the third plot shows values of the objective for $u = v$; the fourth plot shows values for $u = -v$.

We plot the trace norm of X in Figure 5-4 as a function of x and y. For any fixed $x = c$, the trace norm is approximately $\|X(c, y)\|_\Sigma \approx |y| + \sqrt{1 + c^2}$. For fixed $y = c$, the trace norm is approximately $\|X(x, c)\|_\Sigma \approx |c| + \sqrt{1 + x^2}$. When $y = 0$, the matrix has rank 1. When $y \neq 0$, the matrix has rank 2. The fact that trace norm behaves like an absolute value function as y changes (x held fixed) means that when the trace norm is used in a combined objective, a low rank solution is encouraged. Thus the trace norm is an effective regularizer for learning—it provides a continuous, convex penalty which discourages a high rank solution.

5.3.5 Properties of the Joint Objective

Now that we have established some intuition for the trace norm and described how it encourages a low-rank solution, we now turn to a discussion of the joint objective (5.26). We focus on stationary points, paying special attention to points where gradient descent optimization might terminate (minima).

A Simple Example

To build intuition, we begin with a simple example of the joint objective. Consider the scenario of binary classification on the real number line with the hinge as our loss function. There is only a single variable, $x \in \mathbb{R}$, which we parameterize as $x = uv$, $u, v \in \mathbb{R}$. The

joint objective is

$$J(u, v, \theta) = \sum_i (1 - y_i(uv - \theta))_+ + \frac{\lambda}{2}(u^2 + v^2) \tag{5.33}$$

To simplify further, consider the case that there is just a single, positively labeled ($y_1 = +1$) example, the regularization parameter is fixed at $\lambda = 1/2$ and the threshold is set to $\theta = 0$. Figure 5-5 provides a graphical depiction of this objective surface as a function of u and v. Note that there are exactly two minima, at $(1, 1)$ and $(-1, -1)$, both attaining the same globally minimal objective value, $J(1, 1, 0) = 0.5$.

Note that it is trivial to show that there are (at least) two global minima. Let $(\bar{U}, \bar{V})$ be a global minimum. Then $(-\bar{U}, -\bar{V})$ yields the same matrix $X = \bar{U}\bar{V}^T = (-\bar{U})(-\bar{V})^T$ without affecting the regularization term.

Non-Convex

An immediate corollary of the fact that the joint objective has multiple global minima is that it is not convex. This introduces questions of stationary points, and, especially, local minima. Recall that the joint objective is a variational bound on the (convex) MMMF objective. The joint objective is over-parameterized relative to the MMMF objective. What is the consequence of these additional parameters? Do they simply serve to create stationary points and multiple global minima? Or, do they also have the effect of creating local minima.

Multiple Global Minima

We have established that there are at least two global minima in the joint objective. Are there more? Generally speaking, the answer is: yes. And, we can describe those additional minima succinctly. First note that there are many factorizations which yield the same matrix product. Let (U, V) be a parameter setting of the joint objective. Then, any appropriately-sized invertible matrix, A, yields a parameter setting[8], (UA, VA^{-T}), with the same product: $X = UV^T = UAA^{-1}V^T$. This change does not affect the loss portion of the joint objective, but it may affect the regularization term. Given a single global minimum of the joint objective, we can attain other global minima via invertible transforms which do not affect the average Frobenius norm; i.e. via rotation. Given a global minimum, $(\bar{U}, \bar{V})$, and an appropriately-size rotation matrix, R, $(\bar{U}R, \bar{V}R)$ is a global minimum. Note that in our simple example, $R = [-1]$ is the only non-trivial rotation matrix. However, for more interesting cases, such as the 2×2 matrix, there is a continuum of rotation matrices, and thus a continuum of global minima.

Parameter Counting

Ignoring θ and λ, the MMMF objective has mn parameters (recall that $X \in \mathbb{R}^{m \times n}$). Wlog, assume $m \leq n$. Then, the joint objective has $m(m + n)$ parameters ($U \in \mathbb{R}^{m \times m}$ and $V \in \mathbb{R}^{n \times m}$). mn of the joint objective's parameters are accounted for in the determination of $X = UV^T$. This leaves m^2 parameters. The multiple global minima account for $m(m-1)/2$ of those parameters, which is the number of parameters required to define an $m \times m$ rotation matrix (strictly upper triangular matrix). Another m parameters (diagonal matrix) are

[8]The notation A^{-T} denotes the transpose of A^{-1}.

accounted-for by variations in the average Frobenius norm ($(\|U\|^2_{\text{Fro}} + \|V\|^2_{\text{Fro}})/2$). Note that by multiplying a column of U by $c > 0$ and multiplying the corresponding column of V by $1/c$, we can alter the average Frobenius norm without altering the matrix product, $X = UV^T$. This leaves us with $m(m-1)/2$ parameters (strictly lower triangular matrix) which are accounted-for by another rotation. We observed earlier that any $m \times m$ invertible matrix transform U and V (to UA and VA^{-T}) without affecting the product $UV^T = X$. The set of non-invertible matrices is a measure-zero set and so does not affect the number of parameters. Note that the singular value decomposition of a matrix ($X = USV^T$, where U, V are orthonormal and S is diagonal) is a decomposition into two rotation matrices (U and V) and one diagonal/magnitude matrix (S). I.e. this corresponds exactly with our discussion: the set of parameters which do not affect X are exactly the (m^2) parameters of an invertible matrix, corresponding to two rotation matrices and a diagonal matrix.

Normalized Optimization

Of note in our U-V parameterization is that we have twice the number of magnitude parameters that we need. Note that we can multiply the k^{th} column of U by c_k and the k^{th} column of V by $\frac{1}{c_k}$ without affecting the product $UV^T = X$. However, this scaling of the columns does affect the average Frobenius norm, which we define as $\text{Fro}(U,V) \equiv (\|U\|^2_{\text{Fro}} + \|V\|^2_{\text{Fro}})/2$. Can we eliminate this excess variability in scale? In fact, it can be done relatively easily. Consider parameterizing U and V with a column-scale vector, $c \in \mathbb{R}^m_+$. Let $\bar{U}$ and $\bar{V}$ represent some factorization of $X = \bar{U}\bar{V}^T$. Then, $U_{ik}(c) = c_k \bar{U}_{ik}$ and $V_{jk}(c) = \frac{1}{c_k}\bar{V}_{jk}$. We can modify the average Frobenius norm of $U(c)$ and $V(c)$ by changing c. Consider minimizing the average Frobenius norm:

$$\min_c \sum_k \left(c_k^2 \sum_i \bar{U}_{ik}^2 + \frac{1}{c_k^2} \sum_j \bar{V}_{jk}^2 \right) \tag{5.34}$$

Each column can be minimized independently. Note that $\arg\min_x x^2 a + \frac{b}{x^2} = \sqrt[4]{\frac{b}{a}}$. Hence, the optimum is $c_k^* = \sqrt[4]{\frac{\sum_j V_{jk}^2}{\sum_i U_{ik}^2}}$, which corresponds to equalizing the column norms. I.e. c_k^* is chosen so that the k^{th} column of $U(c)$ has the same L_2-norm as the k^{th} column of $V(c)$.

This observation suggests that we can eliminate the duplicate magnitude control in our parameterization. This can be done by parameterizing each row of U and V as a unit-length vector using radial coordinates and by creating a new vector of magnitude parameters which are represented as the diagonal of a diagonal matrix Σ.

An alternative, heuristic method for applying this knowledge (which does not require a reparameterization) is to simply normalize the columns of U and V before every iteration of gradient descent. Figure 5-6 provides an outline of the algorithm. This provides us with much of the benefit of a reparameterization without having to substantially alter code we have written for optimization of the joint objective. We find this heuristic technique to be a great improvement over regular optimization of the joint objective. In testing on the MovieLens data set, we find that, with random initialization of U and V, performance through approximately the first 50 iterations is comparable to regular optimization. After the first 50 iterations, we see the heuristic method making much larger decreases in the objective compared to regular optimization. All of our MMMF experiments in § 5.3.7 use this "normalized" optimization method.

- Select random initialization: U^0, V^0
- Repeat:
  1. Normalize $\tilde{U}^T$, $\tilde{V}^T$ so that each pair of k^{th} columns have the same length:

$$U_{ik}^T = \tilde{U}_{ik}^T \sqrt{\frac{\sum_j (V_{jk}^T)^2}{\sum_{i'} (U_{i'k}^T)^2}} \qquad V_{jk}^T = \tilde{V}_{jk}^T \sqrt{\frac{\sum_i (U_{ik}^T)^2}{\sum_{j'} (V_{j'k}^T)^2}}$$

  2. Perform iteration of gradient descent, yielding updated parameters: $\tilde{U}^T$, $\tilde{V}^T$

Figure 5-6: Outline of heuristic algorithm for "normalized" optimization of the joint objective.

Stationary Points

In our simple example (Figure 5-5), the origin is a stationary point. Generally, the joint objective may have a continuum of stationary points much like it may have a continuum of global minima. A factor in establishing stationary points of the joint objective is whether U and V have matching rank deficiency. Consider U,V such that $\exists x \neq 0$ with $Ux = Vx = 0$. U and V are rank deficient. Now, consider the gradient of the joint objective at (U, V). We find that the gradient of $J(U, V, \theta)$ with respect to U or V is zero in the direction of x. I.e. $\frac{\partial J(U,V,\theta)}{\partial U} x = \frac{\partial J(U,V,\theta)}{\partial V} x = 0$ (where $\frac{\partial J}{\partial U} \equiv \left[\frac{\partial J}{\partial U_{ik}}\right]$). This is due to the fact that adding a constant multiple of x to any row of U will not effect a change in $X = UV^T$. Without a corresponding change in V, x will remain in V's null space and the change to U will not affect the matrix product. I.e. a setting of U,V with matching rank deficiency is stationary with respect to the rank deficient portion. Gradient descent will find a minimum with respect to the non-rank-deficient portion of U and V, leaving the rank-deficient portion of U,V unchanged. This is an important issue for the initialization of gradient descent—U and V must be initialized to full-rank matrices. However, stationary points are unlikely to be a concern for optimization since rank deficient parameter settings represent a zero-measure set and stationary points are not attractive.

Local Minima

We now have some intuition and understanding of the joint objective. However, a lingering question remains: are there local minima? If so, we may not be any better off than we would have been by imposing a hard rank constraint. On the contrary, if local minima are nonexistent, then we have found an efficient method for finding Maximum Margin Matrix Factorizations. We proceed by conducting an empirical investigation.

Maximum Margin Matrix Factorization (Srebro et al., 2005) was originally proposed with the Hinge MPF. For relatively small problems (e.g. a few thousand rating observations), we can use an SDP solver to obtain the correct global solution to the original MMMF formulation (equation (5.27) where "loss" is all-thresholds with the Hinge MPF). However, since we optimize the joint objective with conjugate gradients, we cannot effectively optimize a loss involving the Hinge because it is not sufficiently smooth. Instead, we optimize a parameterized upper bound on the Hinge which becomes increasingly tight as the parameter

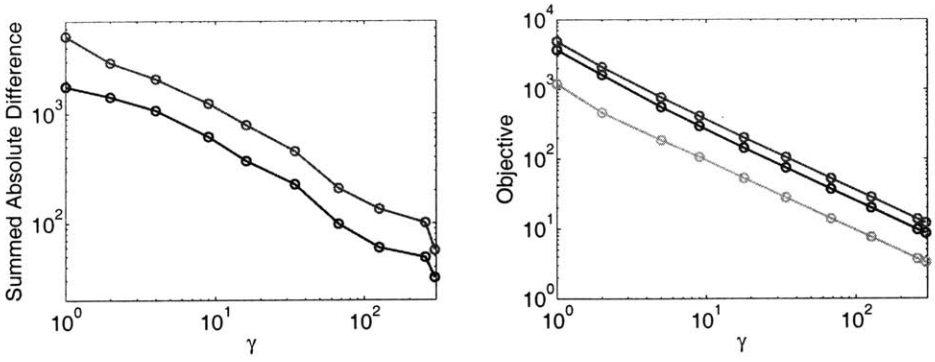

Figure 5-7: **(left)** Summed absolute difference between X (top) and Y (bottom) matrices between the joint/SGL solution and the MMMF/Hinge solution as a function of γ. A value of 10^2 for the Y matrix corresponds to a difference in at most 1% of entries. **(right)** Difference between SGL and Hinge objective values as a function of γ. The top two lines show the SGL objective values for the (top) SDP, and (middle) SGL solutions. The bottom line gives the Hinge objective value for the SGL solution. There is a clear trend toward zero as $\gamma \to 0$, suggesting that there are no local minima issues.

increases. The shifted generalized Logistic[9] (SGL) is the bound that we use on the Hinge,

$$\mathrm{MPF}_{\mathrm{SGL}}(z) = \frac{1}{\gamma}\log(1+\exp(\gamma(1-z))). \tag{5.35}$$

Note that as $\gamma \to \infty$, this function approaches $\mathrm{MPF}_{\mathrm{Hinge}}(z) = (1-z)_+$.

To test for local minima in the joint objective, we took a 100x100 subset of the MovieLens data set, found the solution to the MMMF Hinge-based objective using an SDP solver, then optimized via Conjugate Gradients using the SGL-based joint objective. We found that as we increased γ toward infinity, the solution found via Conjugate Gradients became increasingly close to the "correct" solution returned by the SDP solver. Figure 5-7 (left) shows differences in the solution matrices of the joint objective compared to MMMF solutions returned by the SDP solver. As γ increases, the differences in X and Y shrink toward zero. Numerical issues made it impossible for us to explore values of $\gamma > 300$, but the trend is clear—the difference tends to zero as $\gamma \to \infty$. Tests using a variety of regularization parameters and randomly drawn training sets were similar. Figure 5-7 (right) shows objective values compared to the "correct" solution. The top two lines give the SGL-based objective values for the (top) SDP/Hinge and (middle) Conjugate Gradients/SGL solutions. If the SGL solution were a local minimum, the corresponding objective value would likely be somewhere greater than the loss for the SDP/Hinge solution. The bottom line gives the Hinge-based objective value for the SGL solution. It tends toward zero, suggesting that, in the limit, the SGL solution will achieve the same global minimum as the SDP/Hinge solution.

Note that for our large-scale experiments (§ 5.3.7), we limit the sizes of U and V. This is equivalent to imposing low-rank constraints on U and V, which is certain to introduce local minima. However, the trace norm regularization seems to temper the local minima issue as our approach out-performs many rank-constrained approaches.

[9] Zhang and Oles (2001) and Zhang et al. (2003) discuss the generalized Logistic.

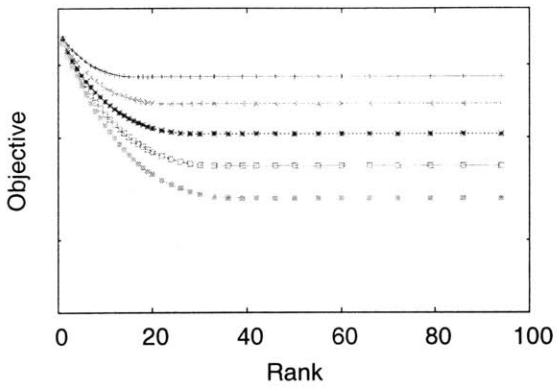

Figure 5-8: Objective value after learning U and V for various regularization values on a 100x100 subset of the MovieLens data set. The "rank" axis indicates the number of columns we used for U and V (the value of k). Each line corresponds to a different regularization constant (λ). Each point corresponds to separate, randomly initialized optimization.

5.3.6 Implementation Details

The original MMMF formulation does not directly impose any restriction on the rank of X. However, a matrix of size $m \times n$ cannot have rank larger than $k = \min(m, n)$. Furthermore, a convex combination of two $m \times n$ matrices, $X = \lambda X_1 + (1 - \lambda)X_2$, cannot have rank larger than k. So, in our optimization of the joint objective, it is sufficient to use U, V matrices of rank k; we gain no benefit from using larger-sized U, V matrices. However, for dealing with large data sets, even rank-k matrices may be too large due to real-world computational resource constraints. For our experiments in Section 5.3.7, we use values of $k \in \{100, 500\}$. This is certain to introduce local minima. While using a too-small value of k may lead to a sub-optimal solution, we found a wide range of values of k that yielded near-identical solutions on a sample data set. Figure 5-8 shows the objective value for various regularization values and rank-truncated U, V matrices on a subset of the MovieLens data set. Although X is 100x100, values of $k \in (20, 40)$ (depending on λ) achieve nearly the same objective value as $k = 100$. Trace norm regularization seems to temper the local minima issue as is seen here and in our large scale experiments (detailed in the next section) as our approach out-performs many rank-constrained approaches.

For optimization of U, V and θ, we used "normalized" optimization of the joint objective, as described in § 5.3.5. For gradient descent, we used the Polak-Ribière variant of Conjugate Gradients (Shewchuk, 1994; Nocedal & Wright, 1999) with the consecutive gradient independence test (Nocedal & Wright, 1999) to determine when to "reset" the direction of exploration. We used the Secant line search suggested by (Shewchuk, 1994), which uses linear interpolation to find an approximate root of the directional derivative. We found PR-CG to be reasonably fast, yielding matrix completion on a 30000x1648 EachMovie rating matrix (4% observed entries, using rank $k = 100$ U, V matrices) in about 5 hours of computation time (single 3.06Ghz Pentium 4 CPU). We note that other gradient-based algorithms, such as L-BFGS (Nocedal & Wright, 1999), may yield even faster optimization.

5.3.7 Experiments

Here we report on experiments conducted on the 1M MovieLens and EachMovie data sets.

Data and Setup

The EachMovie data set provides 2.6 million ratings for 74,424 users and 1,648 movies. There are six possible rating values, $\{1, 2, \ldots, 6\}$. As did Marlin, we discarded users with fewer than 20 ratings. This left us with 36,656 users. We randomly selected 30,000 users for the "weak generalization" set and used the remaining 6,656 users for the "strong generalization" set. The MovieLens data set provides 1 million ratings for 6,040 users and 3,952 movies. There are five possible rating values, $\{1, 2, \ldots, 5\}$. All users had 20 or more ratings, so we utilized all users. We randomly selected 5,000 users for the "weak generalization" set and used the remaining 1,040 users for the "strong generalization" set.

We mimic the setup used by Marlin (2004) and compare against his results. Marlin tested two types of generalization, "weak" and "strong." We conduct test on both types. "Weak generalization" is a single stage process which involves the learner filling-in missing entries of a rating matrix. "Strong generalization" is a two-stage process where the learner trains a model on one set of users and then is asked to make predictions on a new set of users. The learner is given sample ratings on the new set of users, but may not utilize those ratings until after the initial model is constructed.

As did Marlin, we repeat the user selection process three times for each data set. We randomly withheld one movie for each user to construct the test set. We compute Normalized Mean Absolute Error (NMAE) as Marlin describes. The normalization constant for MovieLens (5 rating values) is 1.6; the normalization constant for EachMovie (6 rating values) is 1.944.

Results

We present results on two algorithms based on the ideas we have described in this chapter. The first, which we dub "All-Thresholds", is a rank-constrained variant of MMMF. For All-Thresholds, we use the rank of U and V as a regularization parameter and optimize only the "loss" portion of the joint objective—the trace norm portion is dropped. The second is "Fast MMMF", or optimization of the joint objective. We evaluate a range of regularization parameters for Fast MMMF and All-Thresholds. To select the regularization parameter for each, we withhold one movie per user from the training data to construct a validation set and chose the regularization parameter with lowest validation error. To train the models, we run 100 iterations of gradient descent; we utilize the solution for one regularization parameter to initialize the next; we use a random initialization for the first regularization parameter.

Table 5.2 provides the results. Parenthetical comments indicate settings for each of the two algorithms. k denotes the rank of U and V for Fast MMMF; "SH" denotes use of the Smooth Hinge MPF; "Log" denotes use of the Logistic Regression MPF. Due to time constraints, we do not provide EachMovie results using the Logistic Regression MPF. We note that the rank chosen for All-Thresholds was typically quite small, $k \in \{3, 4\}$.

Our first observation is that All-Thresholds and Fast MMMF substantially and consistently outperform the algorithms that Marlin tested (URP and Attitude). Nearly all All-Thresholds/MMMF average errors are at least one standard deviation lower than the lowest URP or Attitude error—many are multiple standard deviations better. This suggests that the all-thresholds loss function is superior to the loss function (equivalents) for the algorithms that Marlin tested.

We observe that error sometimes decreases when we use larger rank U and V matrices

Algorithm	EachMovie Weak NMAE	EachMovie Strong NMAE
URP	.4422 ± .0008	.4557 ± .0008
Attitude	.4520 ± .0016	.4550 ± .0023
All-Thresholds (SH)	.4394 ± .0035	.4373 ± .0047
Fast MMMF ($k = 100$, SH)	**.4387** ± .0003	.4369 ± .0039
Fast MMMF ($k = 500$, SH)	**.4387** ± .0013	**.4259** ± .0017

Algorithm	MovieLens Weak NMAE	MovieLens Strong NMAE
URP	.4341 ± .0023	.4444 ± .0032
Attitude	.4320 ± .0055	.4375 ± .0028
All-Thresholds (SH)	.4153 ± .0028	.4185 ± .0099
All-Thresholds (Log)	.4145 ± .0059	.4201 ± .0096
Fast MMMF ($k = 100$, SH)	.4148 ± .0043	.4195 ± .0055
Fast MMMF ($k = 500$, SH)	.4111 ± .0026	.4191 ± .0066
Fast MMMF ($k = 500$, Log)	**.4090** ± .0039	**.4183** ± .0117

Table 5.2: Collaborative filtering experimental results on (top) EachMovie and (bottom) MovieLens. URP and Attitude are the best two performing algorithms from Marlin's experiments; his results are reproduced here for comparison. We report performance on "Fast MMMF", and "All-Thresholds", which is rank-constrained Fast MMMF without the (trace norm) regularization term. For Fast MMMF, k is the rank of the U and V matrices. We use "SH" to abbreviate the smooth hinge MPF; we use "Log" to abbreviate the Logistic Regression MPF. All-Thresholds and Fast MMMF select their regularization parameters (rank and λ, respectively) via a validation set taken from the training data. Marlin's results represent the lowest error across a range of regularization parameters (no validation evaluation).

(EachMovie Strong NMAE and MovieLens Weak NMAE). We presume that the larger U and V matrices allowed optimization to find a solution closer to the global optimum and lessened the chance of getting stuck in a local minimum. We found that Fast MMMF solutions selected via evaluation on the validation set were always of full rank (rank equal to k). This suggests that performance may improve further as we increase k. We were not able to test larger values of k due to memory limitations.

We observe that the Logistic Regression MPF performs slightly better than the Smooth Hinge MPF on MovieLens, mimicking the results seen for our Ordinal Regression experiments, though the difference here is smaller.

Finally, we observe that Fast MMMF performs as well as or better than All-Thresholds, indicating that use of the trace norm does provide practical benefit. For two out of the four experiments, Fast MMMF ($k = 500$) achieves at least one-standard-deviation lower error than All-Thresholds. Further experimentation will be needed to determine whether trace norm regularization provides a consistent benefit over using a rank constraint. Note that the inconsistent performance may be a result of our experimental conditions: we limit the ranks of U and V for Fast MMMF and for both algorithms, we limit gradient descent to 100 iterations. Experiments using full-rank U and V may show that the trace norm is superior.

5.3.8 Discussion

We began this section by noting that ordinal regression may be hindered by the fact that it is limited to using features which can be explicitly derived from the data. In many domains, such as restaurant reviews, such features cannot completely describe the items in question. We discussed collaborative filtering, a paradigm for the preference learning problem which does not require external item information. Instead, it makes use of the fact that the user rating tasks are related—information about one user's preferences can be transfered to another user via movies that both users rate. We presented a specific collaborative filtering algorithm, Maximum Margin Matrix Factorization (MMMF), which utilizes the best-performing all-thresholds loss function described earlier. Since the original formulation of MMMF cannot currently be applied to large data sets, we developed an alternate formulation utilizing a variational bound on the MMMF objective called "Fast MMMF". We found that this Fast MMMF was able to achieve a lower error rate on two large collaborative filtering problems than a variety of other collaborative filtering algorithms.

5.4 Hybrid Collaborative Filtering

So far, we have discussed one technique for preference learning which utilizes only "features" of the items, ignoring the fact that other users' ratings may be useful for learning (ordinal regression), and one which ignores "feature" information, and instead utilizes only the relations between users' ratings in order to learn and make predictions (collaborative filtering). It is clear that there are (at least) two sources of information we can use to learn preferences and make future rating predictions: (1) item features, and (2) other users' ratings. There is no reason that we cannot utilize both of these sources of information in an attempt to improve our ability to generalize.

Our development from ordinal regression to collaborative filtering provides an excellent foundation for establishing such a "hybrid" approach to preference learning. Recall the

joint objective:

$$J_{\text{Hybrid}}(U, V, \theta) = \sum_{i,j \in S} \text{loss}(U_i V_j^T; Y_{ij}, \theta_i) + \frac{\lambda}{2}(\|U\|_{\text{Fro}}^2 + \|V\|_{\text{Fro}}^2). \tag{5.36}$$

Our suggested collaborative filtering approach was to minimize the above objective utilizing one of the ordinal regression loss functions from section 5.2. We learned both user preference weights (rows of U) and feature vectors for the items (rows of V), as well as per-user thresholds. To attain a "hybrid" preference learning algorithm, we can simply split $V = [V_1\ V_2]$ into two parts: (1) "fixed" features which are given a priori (as in ordinal regression), and (2) feature variables which are learned during the optimization (as in collaborative filtering). The fixed features compose the first k_1 columns of V and are not modified during learning. The variable features compose the next k_2 columns of V and are learned. Preference weights are learned for both sets of features. I.e. U has $k_1 + k_2$ columns, k_1 weights per user for the fixed features, and k_2 weights per user for the learned features. The only changes that we need to make to our joint objective are to exclude V_1 from the optimization, and to pre-populate V_1 with the given, fixed features.

Note that this hybrid approach may also be applied to the original MMMF formulation. We treat it as a combination of one part ordinal regression with one part MMMF. We add together ordinal regression and MMMF outputs before feeding the value to the loss function, and we sum regularization terms in the objective to yield:

$$J_{\text{HybridMMMF}}(U, X, \theta) = \sum_{i,j \in S} \text{loss}(U_i V_j^T + X_{ij}; Y_{ij}, \theta_i) + \lambda \left(\frac{\|U\|_{\text{Fro}}^2}{2} + \|X\|_{\Sigma} \right), \tag{5.37}$$

where V is the matrix of "fixed" features and U are the corresponding user preference weights.

We also note that the Hybrid approach can be extended to user features. For example, a person's age, sex, location of residence, race and socioeconomic status may affect their preferences for movies, restaurants, etc. We can incorporate this information by adding extra columns to U which encode this fixed information about the person. Corresponding columns are added to V which are learned during the optimization. In short, fixed "weights" are added to U which define known characteristics of the users; corresponding "features" of the items are learned which best represent how the fixed weights affect user/item ratings.

An advantage of the Hybrid approach is that it takes advantage of both sources of information which may be available in a collaborative filtering task: (1) features of the items, and (2) overlapping ratings given by the users. As a result, the algorithm is able to dynamically adapt to whichever source of information proves better able to predict ratings.

5.5 A Missingness Mechanism for Collaborative Filtering

5.5.1 Introduction

A common choice in approaching the problem of collaborative filtering is to assume that the observations (ratings of item given by users) are independently and identically distributed (iid). Clearly, this is not the case. For movies, some people watch only heavily promoted movies, others follow certain actors or directors, some restrict their viewing to a certain genre, while still others limit their watching to movies from a certain era or about a certain

set of events. The process for other types of items is analogous—there are usually certain factors which influence the items which each user rates. Note that there are, in fact, two situations which may account for a missing rating: (1) a user not having the experience (watching a movie, eating at a restaurant), and (2) a user choosing not to rate the item once he/she has rated it. For example, a user may only rate movies which are memorable, or are different from what he/she expected. For our discussion here, we will ignore this distinction and focus on a model for missing ratings.

When a user rates a movie, he/she is providing two bits of information: (1) what he/she thinks of the movie, and (2) an example of movies he/she watches (and rates). Since our goal in collaborative filtering is to predict how the users will rate unobserved items, the pattern of (un)rated movies might seem irrelevant. However, the two processes (selecting items to rate, and user rating of the items) may be related. Usually, people do not select items randomly, but rather try to find items that they will like. Even if the selection and rating processes for a user are independent, it may be that the factors one user utilizes to select movies may overlap with the factors another user uses to rate movies. In either case, we can take advantage of *transfer*. The idea of transfer learning is that modeling two processes jointly may improve generalization for both sets of data since there is a larger pool of data to draw from. Transfer learning can be successful when the two processes are closely related, so that information about one process indirectly provides information about the other. We believe this to be the case for collaborative filtering. Consider the case of movies. Many people only watch and rate movies that they will like—they may exclude entire categories of movies that are unlikely to appeal to them. With this information about a user, one might conclude that the majority of unseen movies for that user will receive poor ratings. In contrast, a user might watch mainstream, heavily marketed movies to appease his/her friends even though his/her real interest lies in historical documentaries. The poor ratings generally given by the user may be due to his/her viewing habits and may not reflect an overly pessimistic view of movies in general. By introducing a missingness mechanism, we allow the model to use the simplest explanation to explain the data.

5.5.2 The Model

First, we develop a missingness mechanism for MMMF. Then, we show how this is easily extended to our joint objective (5.26) ("Fast MMMF"). Our key observation is that the selection process—what movies are "observed"—is a binary classification problem. We learn a set of real values (one per user/item) and per-user binary thresholds to minimize a binary classification loss, such as one of the margin penalty function discussed in section 5.2.4. To link the missingness model with the rating prediction model, we use as regularization the trace norm of a combined matrix, which includes the real-valued "attitudes" parameters from both models. It is via this combined regularization that we hope to achieve transfer. This regularization will encourage the selection of parameters where there is overlap in how items are rated and how they are selected for rating. The objective that we minimize is

$$J_{\mathrm{M^5F}}(X, B, \theta, \phi) = \sum_{i,j \in S} \mathrm{loss}(X_{ij}; Y_{ij}, \theta_i) + \sum_{i,j} \mathrm{lossBC}(B_{ij}; Z_{ij}, \phi_i) + \lambda \left\| \begin{bmatrix} X \\ B \end{bmatrix} \right\|_{\Sigma}, \quad (5.38)$$

where $Z_{ij} \equiv [[(i,j) \in S]]$ indicates[10] whether user i has rated item j, loss is the all-thresholds loss, lossBC is a margin penalty function and $\begin{bmatrix} X \\ B \end{bmatrix}$ represents the vertical concatenation of matrices X and B.

Computational Issues

Note that MMMF with the missingness mechanism is more computationally complex than ordinary MMMF. The number of parameters is approximately double and the number of supervised data points is much larger. A typical collaborative filtering problem may have approximately 5% of the ratings observed for training. Each observed rating has a corresponding ordinal regression loss, which is the sum of $l-1$ (typically $l=5$) margin penalty functions (MPFs). I.e. a typical MMMF scenario is a number of MPFs equal to 20% of the entries. The missingness mechanism introduces a binary loss for each user/item pair, or one MPF per entry. MMMF with the missingness mechanism yields a combined loss function which has a number of MPFs equal to 120% of the entries, or a six-fold increase in complexity of the loss function. If the complexity of the missingness mechanism is a bottleneck, one can obtain some of the transfer benefit with a reduced computational burden by (uniformly) sampling (without replacement) missingness observations for inclusion in the loss function. If the increase in the number of parameters is an issue, one may randomly exclude users from the missingness mechanism model.

Fast MMMF

The missingness mechanism is easily extended to the joint (Fast MMMF) formulation. A new matrix, $A \in \mathbb{R}^{m \times n}$ is introduced to represent the user weights pertaining to the chance that the user will rate an item. Utilizing the definitions from the MMMF formulation, we arrive at the updated minimization objective,

$$J_{\text{FastM}^5\text{F}}(U,V,A,\theta,\phi) = \sum_{i,j \in S} \text{loss}(U_i V_j^T; Y_{ij}, \theta_i) + \sum_{i,j} \text{lossBC}(A_i V_j^T; Z_{ij}, \phi_i) + \frac{\lambda}{2}(\|U\|_{\text{Fro}}^2 + \|A\|_{\text{Fro}}^2 + \|V\|_{\text{Fro}}^2). \quad (5.39)$$

5.6 Summary

We began with the goal of analyzing opinions about items (such as restaurants or movies) extracted from informal communication in order to predict unobserved opinions. This task can be separated into two parts: (1) identifying opinions in text and translating them to numerical ratings, and (2) using the observed ratings to predict unobserved opinions. We chose to focus on the second task and noted others who have dealt with the first task. The task of predicting user opinions of items can be called "preference learning".

We began our treatment of preference learning with a generalization of binary classification: "ordinal regression". We assumed that for each item, we were given a feature vector identifying various aspects of the item. The goal of learning was to find a vector of preference weights for the user that, when combined with the feature vectors, would minimize

[10] For consistency with our margin penalty function notation, we assume that the logic notation $[[\cdot]]$ returns ± 1 instead of the traditional 0/1 value.

a loss associated with the observed ratings. We used the standard L_2-norm regularization penalty to encourage generalization. We described various loss functions for ordinal regression, including one which is a bound on the average absolute error. the quantity that is commonly used to evaluate an ordinal regression system.

Our ordinal regression approach proved highly effective, but it ignored a valuable source of information: the overlap in ratings between users. Ordinal regression treats users independently even though the preference learning task is typically addressed to a set of users and a set of items. By simultaneously learning preferences for all users, we hoped to take advantage of transfer, the concept that jointly solving multiple related problems may yield better generalization. We used "collaborative filtering" to describe the task of predicting ratings where no information about the items are given, save for the observed ratings. We showed how to generalized our ordinal regression framework to this collaborative filtering scenario and observed that the regularization term for our "joint" objective was actually a tight bound on the trace norm, a convex quantity that penalizes rank much like the L_1-norm selects features in learning. This connection revealed that our framework could be viewed as a "fast" version of Maximum Margin Matrix Factorization (MMMF) which could be applied to large collaborative filtering data sets. Experiments showed our Fast MMMF algorithm to be highly effective.

To round-out our discussion of the preference learning problem, we described two extensions to Fast MMMF which are likely to further improve performance without severely affecting computational complexity of the algorithm. The first is a straightforward extension of the ordinal regression and collaborative filtering techniques we have described. Each of ordinal regression (OR) and collaborative filtering (CF) ignored some portion of information that is normally available for preference learning. We described a "hybrid" algorithm which combines OR and CF to take advantage of both item feature vectors and the transfer of information from other users' ratings. The second extension examines an aspect of preference learning that is often ignored: the process by which users experience and express opinions about items. Most work on preference learning assumes the selection of items for rating to be i.i.d. We introduced a "missingness mechanism" for (Fast) MMMF which models this process. Though the missingness mechanism does not directly affect predicted ratings, it uses the regularization term to transfer information between the rating and missingness tasks, thus allowing for improved generalization.

Appendix A

Expectation-Maximization

We show how to derive the Expectation-Mazimization (EM) algorithm for mixture models. In a general setting, we show how to obtain a lower bound on the observed data likelihood that is easier to optimize. For a simple mixture example, we solve the update equations and give a "canned" algorithm.

A.1 EM for Mixture Models

Consider a probability model with unobserved data, $p(x, y|\theta)$, where x represents observed variables and y represents unobserved varaibles. Expectation-Maximization (EM) is an algorithm to find a local maximum of the likelihood of the observed data. It proceeds in rounds. Each round, parameters are chosen to maximize a lower-bound on the likelihood. The lower-bound is then updated so as to be tight for the the new parameter setting.

Let $\theta^{(t)}$ be the current parameter setting. The log-likelihood of the observed data is

$$l(\theta^{(t)}) = \sum_i \log p(x_i|\theta^{(t)}) = \sum_i \log \sum_y p(x_i, y|\theta^{(t)}). \tag{A.1}$$

We want to find a new parameter setting, $\theta^{(t+1)}$, that increases the log-likelihood of the observed data. Let $Q(\theta, \theta^{(t)}) = l(\theta) - l(\theta^{(t)})$. Note that $p(y|x_i, \theta^{(t)}) = \frac{p(x_i, y|\theta^{(t)})}{\sum_{y'} p(x_i, y'|\theta^{(t)})}$. Consider the following manipulations which result in a lower bound on Q:

$$Q(\theta, \theta^{(t)}) = \sum_i \log \sum_y p(x_i, y|\theta) \tag{A.2}$$

$$= \sum_i \log \sum_y p(y|x_i, \theta^{(t)}) \frac{p(x_i, y|\theta)}{p(y|x_i, \theta^{(t)})} \tag{A.3}$$

$$= \sum_i \log E_{p(y|x_i, \theta^{(t)})} \frac{p(x_i, y|\theta)}{p(y|x_i, \theta^{(t)})} \tag{A.4}$$

$$\geq \sum_i E_{p(y|x_i, \theta^{(t)})} \log \frac{p(x_i, y|\theta)}{p(y|x_i, \theta^{(t)})} \tag{A.5}$$

$$= \sum_i \sum_y p(y|x_i, \theta^{(t)}) \log \frac{p(x_i, y|\theta)}{p(y|x_i, \theta^{(t)})} = L(\theta, \theta^{(t)}) \tag{A.6}$$

The inequality is a direct result of the concavity of the log function (Jensen's inequality). Call the lower bound $L(\theta, \theta^{(t)})$.

Consider the following (trivial) fact for an arbitrary function f. Let g be a lower bound on f such that for some $\overline{x}$, $g(\overline{x}) = f(\overline{x})$. Then $g(x) > g(\overline{x})$ implies $f(x) > f(\overline{x})$. In other words, if we find a new point, x, that increases g, then it also increases f. We have constructed L as a lower bound on Q in exactly this way. $L(\theta, \theta^{(t)})$ is a lower bound on $Q(\theta, \theta^{(t)})$ and $L(\theta^{(t)}, \theta^{(t)}) = Q(\theta^{(t)}, \theta^{(t)})$. Thus, we can increasing L rather than Q.

Note that maximizing $L(\theta, \theta^{(t)})$ with respect to θ does not involve the denominator of the log term. In other words, the parameter setting that maximizes L is

$$\theta^{(t+1)} = \arg\max_{\theta} \sum_i \sum_y p(y|x_i, \theta^{(t)}) \log p(x_i, y|\theta). \tag{A.7}$$

It is often easier to maximize $L(\theta, \theta^{(t)})$ (with respect to θ) than it is to maximize $Q(\theta, \theta^{(t)})$ (with respect to θ). For example, if $p(x_i, y|\theta)$ is an exponential distribution, $L(\theta, \theta^{(t)})$ is a convex function of θ. For some models, we can solve for the parameters directly, such as in the example discussed in the next section.

(Dempster et al., 1977) is the original Expectation-Maximization paper. (Salakhutdinov et al., 2003) discuss the convergence properties and suggest a hybrid algorithm that switches between EM and Conjugate Gradients based on an estimate of the "missing information."

A.2 A Simple Mixture Example

Consider a two-component mixture model where the observations are sequences of heads and tails. The unobserved variable takes on one of two values, $y \in \{1, 2\}$. Three parameters define the joint distribution, $\theta = \{\lambda, \phi_1, \phi_2\}$. λ is the probability of using component #1 to generate the observations. ϕ_1 is the probability of heads for component #1; ϕ_2 is the probability of heads for component #2. Let n_i be the length of observed sequence i; let h_i be the number of heads. Let $\lambda_1 = \lambda$, $\lambda_2 = 1 - \lambda$. The joint likelihood is

$$p(x_i, y|\theta) = \lambda_y \phi_y^{h_i} (1 - \phi_y)^{(n_i - h_i)}. \tag{A.8}$$

To maximize the observed data likelihood, we start from an initial setting of the parameters, $\theta^{(0)}$, and iteratively maximize the lower bound. Let

$$J(\theta, \theta^{(t)}) = \sum_i \sum_y p(y|x_i, \theta^{(t)}) \log p(x_i, y|\theta) \tag{A.9}$$

$$= \sum_i \sum_y p(y|x_i, \theta^{(t)}) \log \lambda_y \phi_y^{h_i} (1 - \phi_y)^{(n_i - h_i)} \tag{A.10}$$

Due to the structure of the function, we can solve for the optimal parameter settings by simply setting the partial derivatives to zero. Let $p_{1i} = p(y = 1|x_i, \theta^{(t)})$, $p_{2i} = p(y = 2|x_i, \theta^{(t)})$. The partial derivative of J with respect to λ is

$$\frac{\partial J}{\partial \lambda} = \frac{\sum_i p_{1i}}{\lambda} - \frac{\sum_i p_{2i}}{1 - \lambda} \tag{A.11}$$

Thus, the maximizing setting of λ_1 is $\lambda_1^* = \frac{\sum_i p_{1i}}{\sum_i p_{1i}+p_{2i}}$. The partial of J wrt ϕ_1 is

$$\frac{\partial J}{\partial \phi_1} = \frac{\sum_i p_{1i} h_i - \phi_1 \sum_i p_{1i} n_i}{\phi_1 (1 - \phi_1)} \tag{A.12}$$

Thus, the maximizing setting of ϕ_1 is $\phi_1^* = \frac{\sum_i p_{1i} h_i}{\sum_i p_{1i} n_i}$. Similarly, the maximizing setting of ϕ_2 is $\phi_2^* = \frac{\sum_i p_{2i} h_i}{\sum_i p_{2i} n_i}$. We set $\theta^{(t+1)} = (\lambda_1^*, \phi_1^*, \phi_2^*)$ and repeat. Figure A-1 gives a concise summary of the implementation of EM for this example.

The "canned" algorithms given in (Shewchuk, 1994) (Appendix B) provide useful criteria for determining convergence.

- Randomly choose an inital parameter setting, $\theta^{(0)}$.
- Let $t = 0$. Repeat until convergence.
  - Let $(\lambda_1, \phi_1, \phi_2) := \theta^{(t)}$, $\lambda_2 := 1 - \lambda_1$.
  - Let $p_{yi} := \frac{\lambda_y \phi_y^{h_i} (1-\phi_y)^{(n_i - h_i)}}{\sum_{y'} \lambda_{y'} \phi_{y'}^{h_i} (1-\phi_{y'})^{(n_i - h_i)}}$ for $y \in \{1, 2\}$, $i \in \{1, \ldots, m\}$.
  - Let $\lambda_1^* := \frac{\sum_i p_{1i}}{\sum_i p_{1i}+p_{2i}}$
  - Let $\phi_1^* := \frac{\sum_i p_{1i} h_i}{\sum_i p_{1i} n_i}$.
  - Let $\phi_2^* := \frac{\sum_i p_{2i} h_i}{\sum_i p_{2i} n_i}$.
  - Let $\theta^{(t+1)} := (\lambda_1^*, \phi_1^*, \phi_2^*)$.
  - Let $t := t + 1$.

Figure A-1: A summary of using the EM algorithm for the simple mixture example.

Appendix B

Gradients for a Two-Component Binomial Mixture Model

We observe many samples of a binomial. h_i and n_i are the numbers of heads and total flips for the i^{th} sample. For each sample, there are two binomials that could have produced it. Our goal is to learn the mixing parameter and parameters for the two binomials that are most likely to have produced the samples. Our model is:

$$p(D|\theta) = \prod_i \binom{n_i}{h_i} \left[\lambda \phi_1^{h_i}(1-\phi_1)^{n_i-h_i} + (1-\lambda)\phi_2^{h_i}(1-\phi_2)^{n_i-h_i}\right] \tag{B.1}$$

For numerical stability reasons, we don't simply maximize likelihood in order to find the best parameters. We re-parameterize using (unconstrained) natural parameters, and minimize the negative log-odds ratio of the mixture and (simple) binomial likelihoods. Let $g(x) = (1+\exp(-x))^{-1}$, the logistic function. We reparameterize as follows:

$$\lambda = g(u) \qquad \phi_1 = g(v_1) \qquad \phi_2 = g(v_2) \tag{B.2}$$

Let $\phi^* = \frac{\sum_i h_i}{\sum_i n_i}$ be the maximum-likelihood (simple) binomial parameter. We define the following ratios which appear in the log-likelihood ratio:

$$r_{11} = \frac{\phi_1}{\phi^*} \qquad r_{12} = \frac{1-\phi_1}{1-\phi^*} \qquad r_{21} = \frac{\phi_2}{\phi^*} \qquad r_{22} = \frac{1-\phi_2}{1-\phi^*} \tag{B.3}$$

We define the following quantities which are useful for writing the log-odds and its derivatives:

$$p_{i1} = r_{11}^{h_i} r_{12}^{n_i-h_i}, \tag{B.4}$$

$$p_{i2} = r_{21}^{h_i} r_{22}^{n_i-h_i}, \tag{B.5}$$

$$z_i = \lambda p_{i1} + (1-\lambda)p_{i2}, \tag{B.6}$$

Then the log-odds is simply

$$l(D|\theta) = -\sum_i \log(z_i) \tag{B.7}$$

Note that $\frac{\partial g(x)}{\partial x} = g(x)(1 - g(x))$. The partial derivatives are:

$$\frac{\partial l}{\partial u} = -\sum_i \frac{\frac{\partial \lambda}{\partial u} p_{i1} + \frac{\partial (1-\lambda)}{\partial u} p_{i2}}{z_i} = -\lambda(1-\lambda) \sum_i \frac{p_{i1} - p_{i2}}{z_i} \tag{B.8}$$

$$\frac{\partial l}{\partial v_1} = -\sum_i \frac{\lambda}{z_i} \frac{\partial p_{i1}}{\partial v_1} = -\sum_i \frac{\lambda p_{i1}}{z_i} (h_i - \phi_1 n_i) \tag{B.9}$$

$$\frac{\partial l}{\partial v_2} = -\sum_i \frac{(1-\lambda)}{z_i} \frac{\partial p_{i2}}{\partial v_2} = -\sum_i \frac{(1-\lambda) p_{i2}}{z_i} (h_i - \phi_2 n_i) \tag{B.10}$$

One can use general-purpose optimization software to solve for (locally) maximum-likelihood parameters.

Bibliography

Agresti, A., Mehta, C. R., & Patel, N. R. (1990). Exact inference for contingency tables with ordered categories. *Journal of the American Statistical Association, 85*, 453–458.

Aone, C., & Bennett, S. (1995). Evaluating automated and manual acquisition of anaphora resolution strategies. *Proceedings of the 33rd conference on Association for Computational Linguistics.*

Azar, Y., Fiat, A., Karlin, A. R., McSherry, F., & Saia, J. (2001). Spectral analysis of data. *ACM Symposium on Theory of Computing* (pp. 619–626).

Bekkerman, R., El-Yaniv, R., & McCallum, A. (2005). Multi-way distributional clustering via pairwise interactions. *Proceedings of the 22nd International Conference on Machine Learning (ICML).*

Bikel, D. M., Schwartz, R. L., & Weischedel, R. M. (1999). An algorithm that learns what's in a name. *Machine Learning, 34*, 211–231.

Billsus, D., & Pazzani, M. J. (1998). Learning collaborative information filters. *Proceedings of the 15th International Conference on Machine Learning* (pp. 46–54). Morgan Kaufmann, San Francisco, CA.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2002). Latent dirichlet allocation. *Advances in Neural Information Processing Systems 14.*

Bookstein, A., & Swanson, D. R. (1974). Probabilistic models for automatic indexing. *Journal of the American Society for Information Science, 25*, 312–318.

Brookes, B. C. (1968). The measure of information retrieval effectivenss proposed by Swets. *Journal of Documentation, 24*, 41–54.

Canny, J. (2004). Gap: A factor model for discrete data. *SIGIR '04: Proceedings of the 27th Annual International Conference on Research and Development in information Retrieval* (pp. 122–129). ACM Press.

Cardie, C., & Wagstaff, K. (1999). Noun phrase coreference as clustering. *Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora* (pp. 82–89).

Chu, W., & Ghahramani, Z. (2004). *Gaussian processes for ordinal regression* (Technical Report). University College London.

Chu, W., & Keerthi, S. S. (2005). New approaches to support vector ordinal regression. *Proceedings of the 22nd International Conference on Machine Learning.*

Church, K. W., & Gale, W. A. (1995a). Inverse document frequency (IDF): A measure of deviation from poisson. *Proceedings of the Third Workshop on Very Large Corpora* (pp. 121–130).

Church, K. W., & Gale, W. A. (1995b). Poisson mixtures. *Journal of Natural Language Engineering.*

Clogg, C. C., & Shihadeh, E. S. (1994). *Statistical models for ordinal variables.* SAGE Publications.

Cohen, W. W., Schapire, R. E., & Singer, Y. (1998). Learning to order things. *Advances in Neural Information Processing Systems 10.*

Collins, M. (2002). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. *Proceedings of EMNLP.*

Collins, M., Dasgupta, S., & Schapire, R. E. (2002). A generalization of principal components analysis to the exponential family. *Advances in Neural Information Processing Systems 14.*

Crammer, K., & Singer, Y. (2002). PRanking with ranking. *Advances in Neural Information Processing Systems 14.*

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 series B, 39,* 1–38.

Edelman, A. (2005a). Jacobians of matrix transforms (with wedge products). http://web.mit.edu/18.325/www/handouts.html. 18.325 Class Notes: Finite Random Matrix Theory, Handout #3.

Edelman, A. (2005b). Volumes and integration. http://web.mit.edu/18.325/www/handouts.html. 18.325 Class Notes: Finite Random Matrix Theory, Handout #4.

Elkan, C. (2006). Clustering documents with an exponential-family approximation of the dirichlet compound multinomial distribution. *23rd International Conference on Machine Learning.*

Fazel, M. (2002). *Matrix rank minimization with applications.* Doctoral dissertation, Stanford University.

Fazel, M., Hindi, H., & Boyd, S. P. (2001). A rank minimization heuristic with application to minimum order system approximation. *Proceedings of the American Control Conference* (pp. 4734–4739).

Feinberg, S. E. (1980). *The analysis of cross-classified categorical data.* MIT Press.

Fu, L., & Simpson, D. G. (2002). Conditional risk models for ordinal response data: simultaneous logistic regression analysis and generalized score tests. *Journal of Statistical Planning and Inference, 108,* 201–217.

Ghahramani, Z., & Hinton, G. E. (1996). *The EM algorithm for mixtures of factor analyzers* (Technical Report CRG-TR-96-1). Department of Computer Science, University of Toronto.

Harter, S. P. (1975). A probabilistic approach to automatic keyword indexing: Part I. On the distribution of specialty words in a technical literature. *Journal of the American Society for Information Science, 26*, 197–206.

Herbrich, R., Graepel, T., & Obermayer, K. (2000). Large margin rank boundaries for ordinal regression. In *Advances in large margin classifiers*, 115–132. MIT Press.

Hofmann, T. (2004). Latent semantic models for collaborative filtering. *ACM Transactions on Information Systems, 22*, 89–115.

Hollander, M., & Wolfe, D. A. (1999). *Nonparametric statistical methods.* John Wiley & Sons.

Horn, R. A., & Johnson, C. R. (1991). *Topics in matrix analysis.* Cambridge University Press.

Joachims, T. (1997). *Text categorization with support vector machines: Learning with many relevant features* (Technical Report LS-8 Report 23). Computer Science Department, University of Dortmund.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. *Proceedings of the Tenth European Conference on Machine Learning.*

Johnson, V. E., & Albert, J. H. (1999). *Ordinal data modeling.* Springer.

Jones, K. S. (1973). Index term weighting. *Information Storage and Retrieval, 9*, 619–633.

Kramer, S., Widmer, G., Pfahringer, B., & Groeve, M. D. (2001). Prediction of ordinal classes using regression trees. *Fundamenta Informaticae, 47*, 1–13.

Lee, D. D., & Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature, 401*, 788–791.

Marlin, B. (2004). Collaborative filtering: A machine learning perspective. Master's thesis, University of Toronto, Computer Science Department.

Marlin, B., & Zemel, R. S. (2004). The multiple multiplicative factor model for collaborative filtering. *Proceedings of the 21st International Conference on Machine Learning.*

McCallum, A., Corrada-Emmanuel, A., & Wang, X. (2004). *The author-recipient-topic model for topic and role discovery in social networks: Experiments with enron and academic email* (Technical Report UM-CS-2004-096). University of Massachusetts, Department of Computer Science.

McCallum, A., & Wellner, B. (2005). Conditional models of identity uncertainty with application to noun coreference. *Advances in Neural Information Processing Systems 17* (pp. 905–912). Cambridge, MA: MIT Press.

McCullagh, P. (1980). Regression models for ordinal data. *Journal of the Royal Statistical Society, Series B (Methodological), 42*, 109–142.

Meila, M., & Shi, J. (2001). A random walks view of spectral segmentation. *Proceedings of the Eighth International Workshop on Artificial Intelligence and Statistics.*

Ng, A. Y., Jordan, M. I., & Weiss, Y. (2002). On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems 14.*

Ng, V., & Cardie, C. (2002). Improving machine learning approaches to coreference resolution. *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL).*

Nigam, K. (2001). *Using unlabeled data to improve text classification.* Doctoral dissertation, Carnegie Mellon University.

Nocedal, J., & Wright, S. J. (1999). *Numerical optimization.* Springer-Verlag.

Pang, B., & Lee, L. (2004). A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. *Proceedings of the ACL.*

Papineni, K. (2001). Why inverse document frequency. *Proceedings of the NAACL.*

Rennie, J. D. M., Shih, L., Teevan, J., & Karger, D. R. (2003). Tackling the poor assumptions of naive bayes text classifiers. *Proceedings of the Twentieth International Conference on Machine Learning.*

Rennie, J. D. M., & Srebro, N. (2005a). Fast maximum margin matrix factorization for collaborative prediction. *Proceedings of the 22nd International Conference on Machine Learning.*

Rennie, J. D. M., & Srebro, N. (2005b). Loss functions for preference levels: Regression with discrete ordered labels. *Proceedings of the IJCAI Multidisciplinary Workshop on Advances in Preference Handling.*

Rifkin, R. (2002). *Everything old is new again: A fresh look at historical approaches in machine learning.* Doctoral dissertation, Massachusetts Institute of Technology.

Salakhutdinov, R., Roweis, S., & Ghahramani, Z. (2003). Optimization with EM and expectation-conjugate-gradient. *Proceedings of the Twentieth International Conference on Machine Learning (ICML-2003).*

Shashua, A., & Levin, A. (2003). Ranking with large margin principle: Two approaches. *Advances in Neural Information Processing Systems 15.*

Shewchuk, J. R. (1994). An introduction to the conjugate gradient method without the agonizing pain. http://www.cs.cmu.edu/~jrs/jrspapers.html.

Soon, W. M., Ng, H. T., & Lim, D. C. Y. (2001). A machine learning approach to coreference resolution of noun phrases. *Computational Linguistics, 27,* 521–544.

Srebro, N., & Jaakkola, T. (2003). Weighted low rank approximation. *20th International Conference on Machine Learning.*

Srebro, N., Rennie, J. D. M., & Jaakkola, T. S. (2005). Maximum margin matrix factorization. *Advances in Neural Information Processing Systems 17.*

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society B, 58.*

Tutz, G. (2003). Generalized semiparametrically structured ordinal models. *Biometrics, 59.*

van Rijsbergen, C. J. (1979). *Information retireval.* London: Butterworths.

Vapnik, V. (1995). *The nature of statistical learning theory.* Springer-Verlag.

Wiebe, J. (1994). Tracking point of view in narrative. *Computational Linguistics, 20,* 233–287.

Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics, 1,* 80–83.

Zhang, J., Jin, R., Yang, Y., & Hauptmann, A. G. (2003). Modified logistic regression: An approximation to svm and its applications in large-scale text categorization. *Proceedings of the 20th International Conference on Machine Learning (ICML).*

Zhang, T., & Oles, F. J. (2001). Text categorization based on regularized linear classification methods. *Information Retrieval, 4,* 5–31.