Efficient Models for Relational Learning

Ajit P. Singh
May 30, 2008

Thesis Committee:
Geoff Gordon, CMU
Tom Mitchell, CMU
Christos Faloutsos, CMU
Pedro Domingos, University of Washington

Abstract

The primary difference between propositional (attribute-value) and relational data is the existence of relations, or links, between entities. Graphs, relational databases, sets of tensors, and first-order knowledge bases are all examples of relational encodings. Because of the relations between entities, standard statistical assumptions, such as independence of entities, is violated. Moreover, these correlations should not be ignored as they provide a source of information that can significantly improve the accuracy of common machine learning tasks (e.g., prediction, clustering) over propositional alternatives. A current limitation in relational models is that learning and inference are often substantially more expensive than propositional alternatives. One of our objectives is the development of models that account for uncertainty in relational data while scaling to very large data sets, which often cannot fit in main memory. To that end, we propose representing relational data as a set of tensors, one per relation, whose dimensions index different entity types in the data set. Each tensor has a low-dimensional approximation, where they share a low-dimensional factor for each shared entity-type. For the case of matrices, we refer to this model as collective matrix factorization.

While existing techniques for relational learning assume a batch of data, we propose exploring extensions to active and mixed initiative learning, where the learning algorithm can query its environment (typically a human user) about relationships between entities, the creation of new predicates, and relationships between predicates themselves. It is our belief that the expressiveness of relational representations will allow for more efficient interaction between the learner and its environment, as well as leading to better predictive models for relational data. Efficiency refers not only to computational efficiency, but also to the efficiency of data collection in active learning scenarios. To support the claim that our models are efficient, we propose exploring three problems: predicting user's ratings of movies with side information, topic models for text using fMRI images of neural activation on words, and mixed initiative tagging of e-mail and other information used by personal information managers—e.g., tasks from todo lists, recently edited files, and calendar entries.

Contents

1 Introduction

The standard propositional view of machine learning assumes a set of records, or entities, consisting of attribute-value pairs for a fixed set of attributes. Records are assumed to be independent draws from an underlying event distribution. Relational data additionally contains predicates (relations) between entities, which can provide additional information: relations are correlated, knowing the value of some relations an entity participates in provides information about the likely values of unobserved relations. However, in relational data, entities are no longer independent and identically distributed, and often are not even de Finetti exchangeable, as is assumed in hierarchical models [26, ch. 5]. A relational learner is one that exploits both attributes and relations.

To illustrate the non-exchangeability of relational data consider the following example: the data consists of a documents $\times$ words matrix X, whose entries X_{dw} are the number of times word w appeared in document d. Let Z_{dw} be the underlying random variable corresponding to X_{dw}. X is exchangeable if the distribution of $Z = \{Z_{dw}\}_{d,w}$ is the same as the distribution of the random variables after permuting the rows and columns: $\{Z_{\pi_1(d)\pi_2(w)}\}_{d,w}$. That is, there is no information in the relative position of two documents in X, which is reasonable since we know nothing more about a document d than what is represented in X. Now consider adding another documents $\times$ documents matrix Y to the data, where an entry $Y_{dd'}$ indicates whether two documents were written by the same author. Links between the documents mean that permuting the order of documents in X will change the distribution of Z, precisely because the links between documents provide side information that is not preserved under a random permutation of documents in X. In short, Y provides an additional source of information about documents, violating the exchangeability assumption on X.

Relational learning is important because there has been an explosion in the number and variety of relational data sets, especially those that are loosely typed and semi-structured, such as web pages, social networks, and annotated or tagged content. A number of domains that involve mixing information from different sources can be viewed as relational problems, such as the movie rating task described in Section 1.1. A prima facie case for the prevalence of relational data can be made by considering the ways in which such data are encoded:

- **Graphs**: Nodes typically correspond to entities and relations to edges. The nodes may have attributes, including a special attribute indicating the object type or label. Such data can equivalently be represented as a matrix.
- **Tensors**: An arity-k relation can be encoded as a tensor, where each dimension indexes possible values for one argument in the relation. Our work on relational models has focused on arity-two relations, matrices.
- **Predicate Logic**: Entities correspond to constants, relations are predicates, and attributes are unary predicates whose value is the value of the attribute. Logical formulae express the connections between relations. In relational learning, the most common choices for predicate logics are subsets of typed first-order probabilistic logic.
- **Relational Databases**: Entities correspond to records, grouped by type into tables. Relations are represented by foreign keys between tables[1].

Relational data has structure between entities. One could attempt to exploit this structure by converting it to features of entities, where the resulting model on attributes can be expressed in propositional logic—e.g., Bayesian networks, linear separators. However, our interest is in developing models drawn from more expressive languages. One can consider relational analogues of common machine learning tasks: prediction, clustering, ranking, etc. We further stand on the belief that since relational representations allow one to reason about both entities and relations between them, it opens up new possibilities for active learning, where the user can provide feedback about both entities and relations, e.g., predicate invention, hierarchies of categories, and cluster membership. We propose addressing the following three questions:

Q1 How should one predict and cluster relational data, taking into account that relations are typically observed for only a small fraction of possible combinations of entities; that entities in the test and training sets can differ; and that the correlations between relations are unknown ? (Learning under Relational Uncertainty)

Q2 Can the above techniques be scaled to deal with large data sets, which may contain a large number of relations, a large number of entities, and a large number of observed relations between entities, where the data does not necessarily fit into main memory ? (Computationally Efficient Relational Models)

Q3 How can we acquire and incorporate user knowledge while learning a relational model ? Can a more expressive relational representation allow one to request less feedback from the user than simpler propositional representations ? (Data Efficient Relational Models, i.e., Relational Active Learning)

[1]The relational calculus that underlies most databases is a predicate logic, a subset of typed first order logic. We contend that the popularity of relational databases, and its influence on statistical relational learning, merits special consideration.

1.1 Motivation

For concreteness, we introduce an example of relational data for the task of augmented collaborative filtering [3]. There are four types of entities: users, movies, genres, and actors. Data consists of observed values of the following relations: Rating($user, movie$) $\in \{1, 2, \ldots, R\}$, the rating a user assigns to a movie; IsGenre($movie, genre$) $\in \{0, 1\}$, whether or not a movie falls under the given genre; and HasRole($actor, movie$) $\in \{0, 1\}$, whether or not an actor has a role in the movie. Augmented collaborative filtering is the task of predicting the rating a user assigns to a movie, using additional information about movies, genres, and actors.

We believe that the relations are correlated, knowing the genres of a movie and the actors in it provides additional information about the rating a user assigns to it. However, prediction and clustering of relational data (Q1) is more complicated than its propositional analogue for several reasons:

- Relations can have heterogeneous values—e.g., ratings are on an ordinal scale, whereas the other relations are binary.
- Relations can have heterogeneous structure—relations can be one-to-one, one-to-many, or many-to-many. Moreover, the number of relations an entity participates in is often not constant. In the movie rating example, the number of ratings for each movie follows an approximate power law distribution.
- Correlations between the relations are unknown—e.g., how much do IsGenre and HasRole influence ratings ? In the coarsest sense, this is an example of schema uncertainty.
- The entities in the training set are not the only ones that can exist—e.g., new movies, actors, and users can be added to the data set. The number of entities is not known a priori (domain uncertainty). In some scenarios, two constants may refer to the same entity, such as when one person has two distinct user logins on the system (identity uncertainty).
- Relations are sparse and partially observed—e.g., in Rating($user, movie$) only a small fraction of the movies are rated by any single user.

One of the arguments for relational representations is that the learning algorithm has access to richer concept classes than is available to a propositional learner. The same argument applies to human feedback in relational active learning. A richer representation language allows more complicated questions to be posed to a user, such as assigning a new predicate to a group of entities or inferring a hierarchy of predicates.

2 Survey

The defining characteristic of relational data is the statistical dependence between entities, which can be encoded in several ways: propositionalization, which converts the relations into attributes; graphs, where entities and observed relations map to nodes and edges; tensors, where each dimension indexes an entity type and the entries correspond to observed values of the relation; and first-order logic, where entities and relations map to constants and predicates.

2.1 Propositionalization

The vast majority of learning algorithms work with propositional, attribute-value, data. Propositionalization [41] takes a relational data set, creates new attributes for each entity that encapsulate the dependencies induced by relations, and then produces a single table where each entity is represented by a record consisting of attribute-value pairs[2]. If the records in the new data set are close to independent, then one can legitimately use standard attribute-value learning algorithms like Bayesian networks or logistic regression.

A common form of propositionalization for data represented in first-order knowledge bases creates binary features for an entity by searching over rules, first-order literal conjunctions. Each rule corresponds to a feature. If the conjunction evaluates to true for an entity, then the corresponding binary feature is true for that entity. For example,

$$f_1(m) \leftarrow IsGenre(m, Comedy) \wedge (\exists u\ Rated(u, m) \geq 4)$$

is a binary feature for movies that is true for a comedy rated four stars or higher by any user. LINUS [45] and RSD [46] are examples of rule-based propositionalization. In many cases the propositionalization is coupled to a particular model, using propositionalization internally without producing tabular data, e.g., structural logistic regression [64]. An advantage of

[2]Propositionalization also refers to an inference technique in first-order logic that grounds all clauses with all possible combinations of constants. Inference reduces to checking whether the propositional form is satisfiable. We disambiguate the two meanings by referring to the inference technique as propositionalized inference, or by the algorithm used to determine satisfiability (e.g., DPLL [20, 21], WalkSAT [73]).

propositionalization is that, once the data are converted to tabular form, one can feed it to a wide variety of attribute-value learners. Propositionalization makes it relatively easy to encode user knowledge in the form of new features. The disadvantages are that (i) we can only reason about individual entities, instead of relations and entities; (ii) the space of rules is enormous, computational considerations necessitate a bias towards short rules; (iii) many features are relevant only to a subset of the entities, and so covering most entities requires many weakly relevant features; (iv) finding a set of attributes that make the entities exchangeable is difficult.

A common form of propositionalization in databases is denormalization, where multiple tables in a database are joined into a single one. If any of the relationships in the original schema are one-to-many or many-to-many, then records are repeated many times in the join, which can skew the marginal distribution of an attribute. In the augmented collaborative filtering example, Rating($user, movie$) and HasGenre($movie, genre$) are many-to-many relations. Were the relations joined, the frequency of a particular genre would depend on how often movies of a particular genre are rated. If many of the rated movies are action films, then the marginal distribution of genres would beskewed towards them, regardless of how common action films are among the set of movies. This problem is also known as relational autocorrelation [35].

2.2 Matrices and Tensors

A k-ary relation can be encoded as a tensor, where each dimension indexes possible values for one argument in the relation. For example Rating($user, movie$) is typically encoded as a matrix, where each row corresponds to a user and each column to a movie. Representing a single relation as a matrix is common in collaborative filtering; bag-of-word models, where the rows are documents, the columns words, and the entries word counts; and microarrays, where the rows are people, the columns genes, and the entries gene expression levels.

Relational learning in matrix representations is usually framed in terms of matrix factorization with side information. When the side information is a set of labels for an entity type, the problem is known as supervised or semi-supervised matrix factorization. Support vector decomposition machines [63] is a semi-supervised model that assumes the labels are binary and the features are real-valued. A similar model was proposed by Zhu et al. [90], using a once-differentiable variant of the Hinge loss. Labels can be replaced by arbitrary matrices, which is the approach used by collective matrix factorization (Section 3.1.2). Other approaches that incorporate information from two different matrices include supervised LSI [87], which factors both the data and label matrices under squared loss; an extension of principal components analysis to the two matrix case [88]; and an extension of pLSI to incorporate links between documents [16].

Matrix bi-clustering, which simultaneously clusters entities that correspond to rows and columns, can be extended to a set of related matrices [50, 51, 52]. Early work on this model uses a spectral relaxation specific to squared loss [50], while later generalizations to regular exponential families [52] use EM. An equivalent formulation in terms of regular Bregman divergences, Long et al. [51], uses iterative majorization [48, 82] as the inner loop of an alternating projection algorithm. An improvement on Bregman co-clustering accounts for systematic biases, block effects, in the matrix [1]. In tensor clustering or factorization, each relation may connect multiple entity types. For example, Banerjee et al. propose a model for Bregman clustering in matrices and tensors [2, 3].

2.3 First-Order Logic

First-order logic (FOL) provides a particularly expressive language for relational models. The earliest approach that uses FOL for relational learning is inductive logic programming [59, 44], which does not contain a probabilistic treatment of uncertainty. First-order probabilistic logic (FOPL) allows for uncertainty over logical statements, and in some cases allow for other kinds of uncertainty, e.g., in the number of objects. A full treatment of first-order probabilistic logics is beyond the scope of this proposal[3]. Instead we focus on two approaches that have especially informed our thinking on the topic: Probabilistic Relational Models [23, 27], and Markov Logic Networks [67].

2.3.1 Inductive Logic Programming

Inductive logic programming (ILP) is the task of inducing first-order rules from observed relations encoded as statements in first-order predicate logic. Since ILP is inextricably tied to logic programming, we first review its basic concepts.

In first-order predicate logic, atoms consist of a predicate symbol and its arguments—e.g., HasRole($actor, movie$), Rating($user, movie, value$). Atoms are also referred to as literals, which can be divided into positive and negative literals. A positive literal is one whose logical value is true; a negative literal is one whose logical value is false. Unlike literals in propositional logic, here literals can have arguments. If a literal's arguments are replaced by constants (entities) we refer to it as grounded. A clause is a disjunction of literals where free variables are universally quantified—e.g.,

$$\forall a \forall m \forall u \forall v \ \left(\text{HasRole}(a, m) \vee \text{Rating}(u, m, v)\right).$$

[3] We refer interested readers to Milch and Russell [54], which contains a particularly clear discussion of FOPLs.

A rule $H \leftarrow B$ consists of a head H, a literal, and body B, a clause. If the body clause is true, then consequently so is the literal in the head. Facts are rules where the body is empty, and the head is a grounded literal. A logic program is the conjunction of a set of rules. Prolog [40, 85, 10] is one of the more popular languages for logic programming, its rules are equivalent to Horn clauses, where at most one literal is positive: $(L_H \leftarrow L_1 \wedge L_2 \wedge \ldots \wedge L_n) \equiv (\neg L_1 \vee \neg L_2 \vee \ldots \vee \neg L_n \vee L_H)$. Logic programs are deductive, new facts about entities are deduced from existing rules and facts. Equivalently, deduction can be viewed as ascertaining whether a grounded literal is true or not.

Inductive logic programming is the task of learning a set of rules that describes a target relation H. The inputs are positive facts, observed values of the target relation; negative facts, either explicit negative literals or implicitly defined via a closed world assumption; and background knowledge in the form of rules, which typically involves both the target relation and other non-target relations. A logic program consists of the background knowledge plus the learned rules, and should entail all the positive facts (completeness) and none of the negative facts (consistency). The learned logic program is the model for the target relation, and predicting the value of unobserved relations between entities is deduction, inference in the logic program.

Following [44], we distinguish ILP systems that are based on generalization from those based on specialization. Generalization starts with a clause in the data, finds a rule that entails the clause, and then generalizes it. Common generalization techniques are essentially inverses of standard deductive techniques, such as inverse resolution, which inverts the SLD-resolution technique used by Prolog. The first generalization technique for ILP is MARVIN [69, 70]. Most of the theoretical foundations for inverse resolution were established with CIGOL [58]. Other generalization techniques for ILP include GOLEM [59] and Aleph [81]. Specialization techniques are the relational analogue of decision trees for binary prediction, where the root is the target relation H, and each path in the tree represents the body of a rule whose head is H. Adding a child to node corresponds to adding a clause to the body of a rule, and the stopping techniques are often the same as those used for decision trees. Notable examples of specialization systems for ILP include MIS [74] and FOIL [65]. A variant of ILP that can model uncertainty is stochastic logic programming [57], which generalizes stochastic context-free grammars to include predicates.

2.3.2 Probabilistic Relational Models

A Probabilistic Relational Model (PRM) can be viewed as a probabilistic extension of a frame-based system, or as a generalization of Bayesian networks to relational databases. The latter formulation is likely more familiar to the reader. A table of data for a Bayes net consists of attributes; a database schema consists of multiple tables, whose columns correspond to descriptive attributes and foreign key relations. A Bayesian network is a factored representation over attributes; a PRM is a factored representation over descriptive attributes. Much of the additional complexity of a PRM arises from the fact that an entity in one table can participate in a relation with multiple entities in another table[4]. Thus far we have assumed that only the descriptive attributes are uncertain, and that the foreign keys for each database record are known. Extensions of PRMs to handle reference and existence uncertainty address this limitation, allowing one to predict whether two entities participate in a given relationship. Once the training database is augmented with additional test records, a PRM can be converted into a Bayesian network over individual entities. Each attribute of an entity corresponds to one node in the grounded Bayesian network, and inferring the value of unknown attributes for an entity reduces to MAP/MPE inference. Parameter estimation involves counting the number of records that match a particular assignments to attributes, which can be implemented as SQL counting queries. Since records are neither independent nor exchangeable, we cannot simply split training and test data apart.

2.3.3 Markov Logic Networks

A PRM is a template for creating Bayesian networks that are distributions over descriptive attributes. A Markov Logic network [67] (MLN) is a template for creating Markov networks that are distributions over possible worlds given a first-order knowledge base. A MLN consists of a set of non-negatively weighted first-order clauses representing the observed relations and domain knowledge. Given a set of entities (constants) one can construct a Markov network by creating a Markov network node for each possible grounding of a literal, with edges between grounded literals that occur in the same clause. The clause weights correspond to edge weights in a pairwise Markov network, yielding a Gibbs distribution over possible worlds. This removes the troublesome step of maintaining acyclicity of grounded Bayesian networks in PRMs, but parameter estimation, estimating the weights, does not have a closed form. One must resort to numerical optimization techniques like conjugate gradient, or an approximation like pseudo-likelihood. The problem is essentially the same as learning the parameters in a Markov network: the log-normalizer in the Gibbs distribution requires inference to compute the gradient. Once the first-order knowledge base is populated with additional test records, the MLN can be converted to a Markov network over grounded literals, and inference on Markov networks can be used to estimate the

[4] If all the relations in a data schema are one-to-one, the data is exactly equivalent to a single denormalized table, and a PRM is a Bayes net.

value of a grounded literal. One can represent a PRM using an MLN.

2.4 Graphs

Graphs provide a natural representation for untyped relational data: entities correspond to nodes, and relations to edges or hyper-edges. We focus our attention on attributed graphs, where nodes can have labels and/or attributes. One can propagate information directly over the graph representation of relational data. There is no clear division between test and training data, and some view belief propagation of the grounded networks formed by PRMs as a graph propagation algorithm [25]. If the edges are typed by the relation they represent, then frequent subgraph mining on attributed graphs (e.g., Tong et al. [83]) can be viewed as finding clauses in the relational domain [19]. Clustering nodes in attributed graphs has been used for entity resolution [4], where combining measures of graph similarity and attribute similarity improve performance on an entity resolution task. Our work on absorbing random walks (Section 3.2) is a random walk on a specific kind of attributed graph.

3 Completed Work

Our primary contribution to relational learning is collective matrix factorization [75, 77]. However, much of our future work also draws upon problems and techniques that we have explored previously: information cascades on graphs [49]; absorbing random walks for collaborative filtering [78]; structure learning and inference under parameter uncertainty in Bayesian Networks [76, 84]; and sensor placement [30, 42], which is an example of active learning in spatial statistics.

3.1 Collective Matrix Factorization

The simplest relational data set, that is not simply propositional, is a single arity-n relation $\phi(x_1, \ldots, x_n)$ where $\phi \rightarrow \{\text{False}, \text{True}\}$. We generalize our presentation to allow ϕ to range over other sets of values, e.g., $\mathbb{R}$, $\mathbb{Z}$. A single relation can be represented by an n-dimensional tensor, where each dimension indexes the possible values of an argument of ϕ. Each entry of the tensor corresponds to the value of ϕ for a particular instantiation of its arguments, or grounding. Predicting the value of an entry in the tensor corresponds to predicting the value of a relation for a given grounding. We restrict our discussion to arity-two relations, which can be represented by matrices. Proposed work includes generalizations to arity-k relations (Section 4.1).

Matrix factorization is a common technique for predicting entries of a matrix, given the observed entries. Our contributions include (i) a unified view of matrix factorization, which subsumes a number of existing techniques; (ii) collective matrix factorization, where a set of relations that share arguments are modeled as a set of matrix factorizations with parameter sharing; (iii) an alternating projection algorithm for collective matrix factorization, with a tractable Newton-based projection; (iv) a novel application of stochastic approximation to collective matrix factorization, which allows one to efficiently learn models when the matrices contain many observations.

3.1.1 Unified View of Matrix Factorization

There are a wide variety of matrix factorization algorithms that fall under the rubric of maximum likelihood matrix factorization (MLMF): singular value decompositions, non-negative matrix factorization, probabilistic latent semantic indexing, exponential family PCA[5], etc. All maximum likelihood matrix factorizations can be differentiated by the following choices:

1. Data weights $W \in \mathbb{R}_+^{m \times n}$.
2. Prediction link $f : \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^{m \times n}$.
3. Hard constraints on factors, $U, V \in \mathcal{C}$.
4. Weighted divergence or loss between X and $\hat{X} = f(UV^T)$, $\mathcal{D}(X||\hat{X}, W) \geq 0$.
5. Regularization penalty, $\mathcal{R}(U, V) \geq 0$.

Learning the model $X \approx f(UV^T)$ reduces to the following optimization:

$$(U^*, V^*) = \underset{(U,V) \in \mathcal{C}}{\operatorname{argmin}} \left(\mathcal{D}(X||f(UV^T), W) + \mathcal{R}(U, V) \right). \tag{1}$$

Data weights allow one to scale the relative importance of entries in the matrix. By convention $W_{ij} = 0$ is equivalent to ignoring the contribution of the corresponding data matrix entry in the loss, thus providing a mechanism for excluding missing values. Prediction links allow non-linear relations between parameters $\Theta = UV^T$ and responses $\hat{X} = f(UV^T)$, the same way that the prediction link in generalized linear models [53] extends linear regression. We focus on generalized

[5] Citations for these methods are in Table 1.

Bregman divergences (Definition 1), which subsumes divergences based on exponential families—e.g., Gaussian/Square-Loss, Poisson/I-Divergence, as well as the hinge loss with a margin and smoothed variants thereof. Constraints on the factors allow for non-negativity, orthogonality, and other useful constraints. Constraining each row of U to sum to 1 corresponds to a clustering constraint: U_{ij} is the probability that the i^{th} entity belongs to latent class j. The divergence $\mathcal{D}$ is a measure of how close the estimate is to the training data X. Since there are k parameters for each entity in the data set, and potentially very few observed relations between them, we regularize the factors. A few representative examples of maximum likelihood matrix factorization are presented in Table 1.

Examples of matrix factorizations that are not MLMF include Bayesian matrix factorization, which produces a distribution over factors. Nonparametric Bayesian matrix factorization additionally averages over the embedding dimension k. MLMF models represent a generative distribution over entries of a matrix, but do not form a generative distribution over entities. This makes adding new entities (rows or columns) to the model statistically perilous [86]. Hierarchical priors, such as those used in latent Dirichlet allocation [5], induce a generative distribution over entities.

3.1.2 Collective Matrix Factorization as a Relational Model

The previous section established a wide class of models for a single arity-two relation. However, our interest is focused on domains where there are several relations that share arguments. For example, augmented collaborative filtering, where the relations Rating($user, movie$) and HasGenre($movie, genre$) both provide information about movies.

The input data consists of a set of related matrices on types $\mathcal{E}_1, \ldots, \mathcal{E}_t$. In the above example $\mathcal{E}_1$, $\mathcal{E}_2$ and $\mathcal{E}_3$ correspond to users, movies, and genres respectively. The number of entities of type $\mathcal{E}_i$ is denoted n_i. A relation between two types is denoted $\mathcal{E}_i \sim \mathcal{E}_j$. The schema is the set of relationships between types: $E = \{(i,j) : \mathcal{E}_i \sim \mathcal{E}_j, i < j\}$. In the above example the relations are $\mathcal{E}_1 \sim \mathcal{E}_2 \sim \mathcal{E}_3$. The matrices themselves are denoted $X^{(ij)}$ where each row corresponds to an element of type $\mathcal{E}_i$ and each column to an element of type $\mathcal{E}_j$[6]. Each data matrix is factored under the model $X^{(ij)} \approx f^{(ij)}(\Theta^{(ij)})$ where $\Theta^{(ij)} = U^{(i)}(U^{(j)})^T$ for low rank factors $U^{(i)} \in \mathbb{R}^{n_i \times k_{ij}}$. Where it is clear from the argument, we drop the superscript notation on f. The embedding dimensions are $k_{ij} \in \mathbb{N}$, where k is the largest one. Collective matrix factorization allows per-matrix reconstructions to use different subsets of the columns of $U^{(i)}$ for reconstructing different matrices that involve $\mathcal{E}_i$. This allows for the addition of bias terms, and local effects like block-models. For notational simplicity, we do not discuss bias terms or block models.

Collective matrix factorization models each relation as a matrix factorization, sharing factor $U^{(i)}$ among the reconstruction of matrices that index type $\mathcal{E}_i$. The individual matrix reconstructions are based on Section 3.1.1, where the loss is a weighted decomposable Bregman divergence:

Definition 1 *For a closed, proper, convex function $F : \mathbb{R} \to \mathbb{R}$ the weighted decomposable Bregman divergence between matrices Θ and X is*

$$\mathbb{D}_F(\Theta||X, W) = \sum_{ij} W_{ij} \left(F(\Theta_{ij}) + F^*(X_{ij}) - X_{ij}\Theta_{ij} \right), \tag{2}$$

where F^* is the convex conjugate of F:

$$F^*(\mu) = \sup_{\Theta \in \operatorname{dom} F} \left[\Theta \circ \mu - F(\Theta) \right].$$

If F is additionally differentiable, $\nabla F = f$, and $W_{ij} = 1$, Equation 2 is equivalent to the standard definition [11, 12]:

$$\mathbb{D}_F(\Theta||X, W) = \sum_{ij} F^*(X_{ij}) - F^*(f(\Theta_{ij})) - \nabla F^*(f(\Theta))(X_{ij} - f(\Theta_{ij})) = D_{F^*}(X||f(\Theta))$$

We restrict ourselves to decomposable losses, which can be represented as the sum of terms over the elements of X. Bregman divergences encompass the most commonly used losses, such as squared loss and KL-divergence, but the sequel generalizes to any twice-differentiable and decomposable loss. The total reconstruction loss on the set of matrices is the weighted sum of the losses for each reconstruction:

$$\mathcal{L}_u = \sum_{(i,j) \in E} \alpha^{(ij)} \mathbb{D}_F(\Theta^{(ij)}||X^{(ij)}, W^{(ij)})$$

[6] If there are multiple relations between two types, we disambiguate them using the notation $\mathcal{E}_i \sim_u \mathcal{E}_j$ and $X^{(ij,u)}$ for $u \in \mathbb{N}$.

Table 1: Examples of maximum likelihood matrix factorization. $\mathrm{dom}\, X_{ij}$ describes the types of values allowed in the data matrix, where $1 \circ X = 1$ means that the entries of X are parameters of a discrete distribution. That is, some techniques factor a data matrix and others a normalized data matrix. All the losses are decomposable, and f is an element-wise function on matrices. Unweighted matrix factorizations are those where $W_{ij} = 1$.

Method	$\mathrm{dom}\, X_{ij}$	Link $f(\theta)$	Loss $\mathcal{D}(X\|\hat{X} = f(\Theta), W)$	W_{ij}
SVD [28]	$\mathbb{R}$	θ	$\|\|W \odot (X - \hat{X})\|\|^2_{Fro}$	1
W-SVD [24, 79]	$\mathbb{R}$	θ	$\|\|W \odot (X - \hat{X})\|\|^2_{Fro}$	≥ 0
k-means [31]	$\mathbb{R}$	θ	$\|\|W \odot (X - \hat{X})\|\|^2_{Fro}$	1
k-medians	$\mathbb{R}$	θ	$\sum_{ij} \|W_{ij}\left(X_{ij} - \hat{X}_{ij}\right)\|$	1
L_1-SVD [37]	$\mathbb{R}$	θ	$\sum_{ij} \|W_{ij}\left(X_{ij} - \hat{X}_{ij}\right)\|$	≥ 0
pLSI [33]	$\mathbf{1} \circ X = 1$	θ	$\sum_{ij} W_{ij}\left(X_{ij} \log \frac{X_{ij}}{\hat{X}_{ij}}\right)$	1
NMF [47]	$\mathbb{R}_+$	θ	$\sum_{ij} W_{ij}\left(X_{ij} \log \frac{X_{ij}}{\Theta_{ij}} + \Theta_{ij} - X_{ij}\right)$	1
ℓ_2-NMF [61, 47]	$\mathbb{R}$	θ	$\|\|W \odot (X - \hat{X})\|\|^2_{Fro}$	1
Logistic PCA [71]	$\{0, 1\}$	$(1 + e^{-\theta})^{-1}$	$\sum_{ij} W_{ij}\left(X_{ij} \log \frac{X_{ij}}{\hat{X}_{ij}} + (1 - X_{ij}) \log \frac{1-X_{ij}}{1-\hat{X}_{ij}}\right)$	1
E-PCA [18]	many	many	decomposable Bregman	1
$\mathrm{G^2L^2M}$ [29]	many	many	decomposable Generalized Bregman[a]	1
MMMF [80]	$\{1, \ldots, R\}$	min-loss[b]	$\sum_{r=1}^{R} \sum_{ij:X_{ij} \neq 0} W_{ij} \cdot h(\Theta_{ij} - B_{ir})$[c]	1
Fast-MMMF [66]	$\{1, \ldots, R\}$	min-loss	$\sum_{r=1}^{R} \sum_{ij:X_{ij} \neq 0} W_{ij} \cdot h_\gamma(\Theta_{ij} - B_{ir})$	1

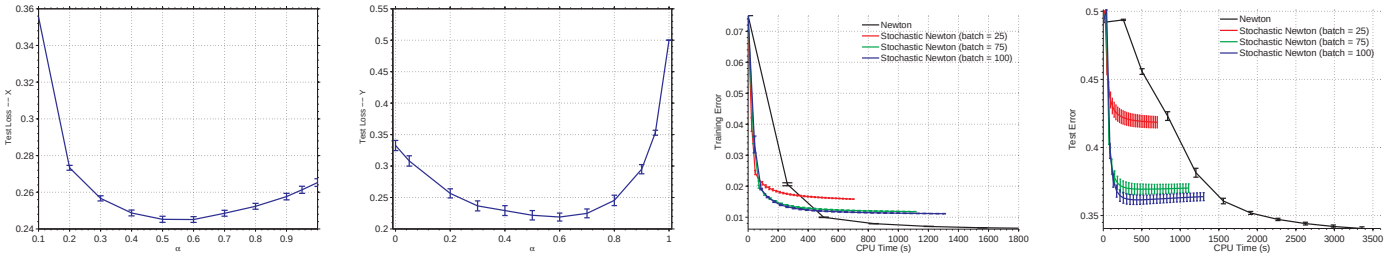
Method	Constraints U	Constraints V	Regularizer $\mathcal{R}(U, V)$	Algorithm(s)
SVD	$U^T U = I$	$V^T V = \Lambda$	—	Gaussian Elimination, Power Method
W-SVD	—	—	—	Gradient, EM
k-means	—	$V^T V = I$, $V_{ij} \in \{0, 1\}$[d]	—	EM
k-medians	—	$V^T V = I$, $V_{ij} \in \{0, 1\}$	—	Alternating
L_1-SVD	—	—	—	Alternating
pLSI	$\mathbf{1}^T U \mathbf{1} = 1$	$\mathbf{1}^T V = \mathbf{1}$	—	EM
NMF	$U_{ij} \geq 0$	$V_{ij} \geq 0$	—	Multiplicative
ℓ_2-NMF	$U_{ij} \geq 0$	$V_{ij} \geq 0$	—	Multiplicative, Alternating
Logistic PCA	—	—	—	Alternating (on lower bound)[e]
E-PCA	—	—	—	Alternating
$\mathrm{G^2L^2M}$	—	—	$\|\|U\|\|^2_{Fro} + \|\|V\|\|^2_{Fro}$	Alternating (Subgradient, Newton)[f]
MMMF	—	—	$tr(UV^T)$	Semidefinite Program
Fast-MMMF	—	—	$\frac{1}{2}(\|\|U\|\|^2_{Fro} + \|\|V\|\|^2_{Fro})$	Conjugate Gradient

[a] Bregman divergences and their generalization are explained in Definition 1.
[b] min-loss finds the integral rating in $\{1...R\}$ that minimizes the loss. The search involves testing whether a $X_{ij} \leq r$ for $r = 1 \ldots R$.
[c] h is the hinge loss. MMMF is ordinal regression, breaking down the prediction into a set of parallel hyperplanes representing $X_{ij} \leq r$ for $r = 1 \ldots R$. The per-row bias B_{ir} acts as margin for what is essentially a set of SVMs for each row of X.
[d] Each column of X correspond to a point, and each row of V corresponds to a hard cluster assignment of a point.
[e] A quadratic lower bound on the logistic PCA objective is proposed, which is optimized using alternating projections.
[f] For twice-differentiable losses, the Newton step is used. For non-differentiable losses (e.g., hinge loss) subgradient is proposed.

where $\sum_{(i,j) \in E} \alpha^{(ij)} = 1$. We regularize on a per-factor basis to mitigate overfitting:

$$\mathcal{L} = \sum_{(i,j) \in E} \alpha^{(ij)} \mathbb{D}_F(\Theta^{(ij)} || X^{(ij)}, W^{(ij)}) + \sum_{i=1}^{t} \mathcal{R}(U^{(i)}) \tag{3}$$

In our experiments we use ℓ_2-regularization: $R(U^{(i)}) = \lambda_i \sum_{pq} \left(U^{(i)}_{pq}\right)^2 / 2$. Learning a collective matrix factorization consists of finding factors $U^{(i)}$ that minimize $\mathcal{L}$.



(a) Ratings – Test error vs. α

(b) Genres – Test error vs. α

(c) Stoch. Newton – Training Loss vs. CPU Time

(d) Stoch. Newton – Test Loss vs. CPU Time

Figure 1: Figures (a-b) measure the test error (MAE) as a function of how strongly ratings are weighted over genres: $\alpha = 0$ corresponds to using only genres; $\alpha = 1$ corresponds to using only ratings. Figures (c-d) compares how quickly stochastic Newton and full Newton step reduce training and test error (log-loss).

We consider a variant of the augmented collaborative filtering example, where the goal is to predict whether or not a user rated a movie, rather than the actual rating. Predicting whether or not a user rated a movie allows us to control for the variation due to the distribution of 1-5 star ratings. The data is a subsample of the Netflix-IMDB dataset described in Section 4.4. There are $n_1 = 500$ users, $n_2 = 3000$ movies, and $n_3 = 21$ genres, and the median number of ratings per user is 60. The embedding dimension $k = 20$ is the same for both matrices. There is one degree of freedom controlling the relative importance of ratings and genres in the model: $\alpha^{(12)} = \alpha$, $\alpha^{(23)} = 1 - \alpha$. Figures 1(a) and 1(b) show the test error for rating and genre prediction under mean absolute error (MAE). For both tasks the optimal value of α is between 0 and 1: mixing the two relations improves prediction of both ratings and genres.

3.1.3 Efficient Maximum Likelihood via Alternating Projections

The parameter space for a collective matrix factorization is large, $O(k \sum_{i=1}^{t} n_i)$, and $\mathcal{L}$ is non-convex with multiple local optima[7]. One typically resorts to gradient methods and EM, as a direct Newton step is infeasible due to the number of parameters. Another approach is alternating projection, a.k.a. block coordinate ascent: iteratively optimize one factor $U^{(r)}$ at a time, fixing all the others. Ignoring terms that are constant with respect to the factors, the gradient of the objective with respect to one factor, $\nabla_r \mathcal{L} = \frac{\partial \mathcal{L}}{\partial U^{(r)}}$, is

$$\nabla_r \mathcal{L} \;=\; \sum_{(r,s):E} \alpha^{(rs)} \left(W^{(rs)} \odot \left(f^{(rs)} \left(\Theta^{(rs)} \right) - X^{(rs)} \right) \right) U^{(s)} + \nabla \mathcal{R}(U^{(r)}). \tag{4}$$

The gradient of a collective matrix factorization is the weighted sum of the gradients for each individual matrix reconstruction. If all the per-matrix losses are decomposable then the Hessian of $\mathcal{L}$ with respect to $U^{(r)}$ is block-diagonal, with each block corresponding to a row of $U^{(r)}$. For a single matrix the result is proven by noting that a decomposable loss implies that the estimate of $X_{i\cdot}$ is determined entirely by $U_{i\cdot}$ and V. If V is fixed then the Hessian is block diagonal. An analogous argument applies when U is fixed and V is optimized. For a set of related matrices the result follows immediately by noting that Equation 4 is a linear function of terms and the differential is a linear operator. Assume that the regularizers $\mathcal{R}(\cdot)$ are also decomposable. Differentiating the gradient of the loss with respect to $U_{i\cdot}^{(r)}$ yields the Hessian for the row:

$$\nabla^2_{r,i} \mathcal{L} = \sum_{(r,s):E} \alpha^{(rs)} \left(U^{(s)} \right)^T \operatorname{diag} \left(W_{i\cdot}^{(rs)} \odot f^{(rs)} \left(\Theta_{i\cdot}^{(rs)} \right) \right) U^{(s)} + \nabla^2 \mathcal{R}(U_{i\cdot}^{(r)}).$$

Newton's rule yields the step direction $\nabla_{r,i} \mathcal{L} \cdot [\nabla^2_{r,i} \mathcal{L}]^{-1}$. For the single matrix case, this approach is known as $\mathrm{G^2L^2M}$ [29]. No matter how large the matrices get, the Hessian remains a $k \times k$ matrix, where $k \ll \min(n_1, \ldots, n_t)$. The cost of a gradient update for $U_{i\cdot}^{(r)}$ is $O(k \sum_{j:\mathcal{E}_i \sim \mathcal{E}_j} n_j)$. The cost of Newton update for the same row is $O(k^3 + k^2 \sum_{j:\mathcal{E}_i \sim \mathcal{E}_j} n_j)$. If the matrix is sparse, n_j can be replaced with the number of entries with non-zero weight. Advantages to our alternating-Newton approach include:

- **Memory Usage**: A solver that optimizes over all the factors simultaneously needs to compute residual errors to compute the update. Even if the data X is sparse, the residuals $(f(\Theta) - X)$ rarely are. Our approach requires only

[7]The exception being SVD, which while being non-convex has one global minimum and saddle points [80].

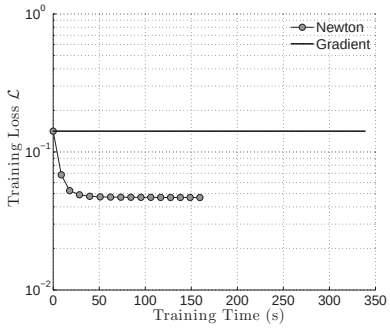


Figure 2: Gradient vs. Newton steps in alternating projection.

that we store one row or column of a matrix in memory, plus $O(k^2)$ memory to perform the update. This make out-of-core factorizations, where the data matrix cannot be stored in RAM, straightforward.

- **Simpler Updates**: Our approach reduces an optimization over data matrices into an optimization over data vectors. We exploit this reduction to incorporate constraints on the factors. In general, the form of our updates makes it easy to exploit a wide range of techniques from regression—*e.g.*, ℓ_1 regularization.

The primary disadvantage of our alternating-Newton approach is that its convergence behaviour is governed by the fact that it is block coordinate descent. Like all coordinate descent methods it has a tendency to get stuck in local optima that other methods may be able to avoid.

An advantage of reducing the per-factor updates to per-row updates is that we can easily take advantage of techniques from convex optimization. For example, equality constraints can be solved as an unconstrained optimization. The stochastic constraint $\sum_j U_{ij} = 1$ is an example of an equality constraint commonly used for matrix clustering or bi-clustering. Since the Newton step is based on a quadratic approximation to the objective, a null space argument (Boyd and Vandenberghe [8, ch. 10]) can be used to show that with a stochastic constraint the step direction d for row $U_{i\cdot}^{(r)}$ is the solution to

$$\begin{bmatrix} \nabla^2_{r,i}\mathcal{L} & 1 \\ 1^T & 0 \end{bmatrix} \begin{bmatrix} d \\ \nu \end{bmatrix} = \begin{bmatrix} -\nabla_{r,i}\mathcal{L} \\ 0 \end{bmatrix} \tag{5}$$

where ν is the Lagrange multiplier for the stochastic constraint. Equation 5 requires that the initial estimate of the row factors must be feasible—*i.e.*, each row of $U^{(r)}$ sums to one. The above technique is easily generalized to $p > 1$ linear constraints, yielding a Hessian of size $k + p$. Likewise, we can take advantage of techniques for ℓ_1-regularized regression [72] to get ℓ_1-regularized matrix factorization. Unlike the stochastic constraint, the resulting row update is a constrained optimization.

One may question whether the cost of maintaining even a $k \times k$ Hessian is worth the effort, especially since the Newton update is just an implementation of the projection, rather than an update on $\mathcal{L}$. Using the data described in Section 3.1.2 we consider the task of rating prediction using a Poisson link on the rating matrix and a logistic link on the genre matrix. From the same initial starting point, we measure the training loss of alternating maximization using a single Newton step for the projection vs. using a single gradient step for the projection. Again $k = 20$ for both matrices. Figure 2 shows that the Newton projection is clearly worthwhile, especially since computing it costs only a factor of k more than the gradient, plus the cost of inverting the $k \times k$ Hessian.

3.1.4 Stochastic Approximation

The alternating-Newton approach leads to the unusual situation where our primary concern is not the additional cost of computing and inverting the Hessian, but rather the cost of computing the gradient itself. Updating a factor $U^{(i)}$ involves estimating the value of any observed relation that entities of type $\mathcal{E}_i$ participate in. Computing the errors on a subset of the observed relations, chosen randomly at each iteration, is the basic idea behind stochastic approximation [6]. We generalize the gradient and Newton steps of the previous section to their stochastic approximation analogues.

Denote a sample of the data at iteration τ as $p_{s,\tau} \subseteq \{1, \ldots, n_s\}$. The sample gradient and Hessian are

$$\nabla_{r,i,\tau}\mathcal{L} = \sum_{(r,s)\in E} \alpha^{(rs)} \left(W^{(rs)}_{ip_{s,\tau}} \odot \left(f^{(rs)} \left(\Theta^{(rs)}_{ip_{s,\tau}} \right) - X^{(rs)}_{ip_{s,\tau}} \right) \right) U^{(s)}_{p_{s,\tau}\cdot} + \nabla\mathcal{R}(U^{(r)}_{i\cdot}) \tag{6}$$

$$\nabla^2_{r,i,\tau}\mathcal{L} = \sum_{(r,s)\in E} \alpha^{(rs)} \left(U^{(s)}_{p_{s,\tau}\cdot} \right)^T \operatorname{diag} \left(W^{(rs)}_{ip_{s,\tau}} \odot f^{(rs)} \left(\Theta^{(rs)}_{ip_{s,\tau}} \right) \right) U^{(s)}_{p_{s,\tau}\cdot} + \nabla^2\mathcal{R}(U^{(r)}_{i\cdot}). \tag{7}$$

Stochastic Newton simply replaces the gradient and Hessian by their sample analogues. While the gradient reduces the loss at each iteration, a sample estimate of the gradient only reduces the loss in expectation; Likewise, the Hessian is the best descent direction, but a sample estimate follows the best descent direction only in expectation. At any given iteration, stochastic approximation may increase or decrease the training error: line search cannot be used, but the asymptotically optimal step length for stochastic Newton is $\eta_\tau = 1/\tau$. Since alternating projection optimizes all the rows of $U^{(r)}$ at once, we reduce the variance of the estimated training loss by using different entity samples in each row of $U^{(r)}$. In practice this causes the training loss with respect to $U^{(r)}$ to decrease at almost every iteration of alternating projection. We sample data non-uniformly, without replacement, from the distribution induced by the data weights. That is, when updating a row $U^{(r)}_{i\cdot}$, the probability of drawing $X^{(rs)}_{ij}$ is proportional to $W^{(rs)}_{ij} / \sum_j W^{(rs)}_{ij}$. There is a compelling relational interpretation to sampling. Given a relation matrix, pick one entity whose parameters are to be updated, then sample observed relations involving that entity. The cost of the gradient update no longer grows linearly in the number of entities involved, but in the number of entities sampled. Another advantage of this approach is that when we sample one entity at a time, stochastic approximation yields an online algorithm, processing each entry in the matrix as it is added to the data set.

To satisfy convergence conditions, we use an exponentially weighted moving average of the Hessian,

$$\bar{q}_{\tau+1}(\cdot) = \left(1 - \frac{2}{\tau+1}\right) \bar{q}_\tau(\cdot) + \frac{2}{\tau+1} \hat{q}_{\tau+1}(\cdot), \tag{8}$$

where $\hat{q}_{\tau=1}$ is computed using Equation 7. We consider three properties of our stochastic Newton method, which together are sufficient for convergence [7]:

- *Local Convexity*: The loss must be locally convex around its minimum, which must be contained in its domain. In alternating projections the loss is convex for any Bregman divergence and, for regular divergences, has $\mathbb{R}$ as its domain. The non-regular divergences we consider, such as Hinge loss, also satisfy this property.

- *Uniformly Bounded Hessian*: The eigenvalues of the sample Hessians are bounded in some finite interval with probability 1, i.e., $\hat{q}$ must be invertible. The eigenvalue condition implies that the elements of $\bar{q}$ and its inverse are uniformly bounded.

- *Convergence of the Hessian*: There are two choices of convergence criteria for the Hessian. Either one suffices for proving convergence of stochastic Newton. (i) The sequence of inverses of the sample Hessian converges in probability to the true Hessian: $\lim_{\tau\to\infty} \bar{q}_\tau^{-1} = q^{-1}$. Alternately, (ii) the perturbation of the sample Hessian from its mean is bounded. Let $\mathcal{P}_{\tau-1}$ consist of the history of the stochastic Newton iterations: the data samples and the parameters for the first $\tau - 1$ iterations. Let $g_\tau = o_s(f_\tau)$ denote an almost uniformly bounded stochastic order of magnitude. The stochastic o_s-notation is similar to regular o-notation, except that we are allowed to ignore measure-zero events and $E[o_s(f_t)] = f_t$. The alternate convergence criteria is a concentration of measure statement:

  $$E[\bar{q}_\tau | \mathcal{P}_{\tau-1}] = \bar{q}_\tau + o_s(1/\tau).$$

  For Equation 8 this condition is easy to verify:

  $$E[\bar{q}_\tau | \mathcal{P}_{\tau-1}] = \left(1 - \frac{2}{\tau}\right) \bar{q}_{\tau-1} + \frac{2}{\tau} E[\hat{q}_\tau | \mathcal{P}_{\tau-1}]$$

  since $\mathcal{P}_{\tau-1}$ contains $\bar{q}_{\tau-1}$. Any perturbation from the mean is due to the second term. If $\hat{q}$ is invertible then its elements are uniformly bounded, and so are the elements of $\delta_\tau = E[\hat{q}_\tau | \mathcal{P}_{\tau-1}]$. Since δ_t has bounded elements and is scaled by $2/\tau$ it follows that the perturbation is $o_s(1/\tau)$.

We consider a larger instance of the task presented in Section 3.1.2, predicting whether a user rated a movie. There are $n_1 = 10000$ users, $n_2 = 2000$ movies, and $n_3 = 22$ genres. Because zeros are not missing values, there are $n_1 \cdot n_2 = 20M$ observations in the ratings matrix. We set the mixing coefficient $\alpha = 0.5$ and embedding dimension $k = 30$, learning collective matrix factorizations using full Newton steps and stochastic Newton steps with batch sizes of 25, 75, and 100 samples per row. Since there are only 22 genres, we use all of them. A comparison of training loss (log-loss) and test error (MAE) versus CPU time is presented in Figures 1(c) and 1(d). Initially, stochastic Newton outperforms the full Newton step. Many iterations of stochastic Newton can be performed in the time required for a full Newton step. Predicting the actual value of the rating is computationally simpler, since zeros in the ratings matrix correspond to missing values. For example, on a problem with $n_1 = 100000$ users, $n_2 = 5000$, and $n_3 = 21$ genres, containing 1.3M observed ratings, alternating projection with full Newton steps runs to convergence in 32 minutes on a single 1.6 GHz CPU.

3.2 Information flow on Graphs

Relational learning is fundamentally about exploiting correlations between different relations. One can also consider exploiting correlations within a single relation that is indexed over time—i.e., observed relationships between entities are timestamped, and the time when an observation appears is informative. We explored the issues of temporal correlation in the form of graph cascades in a viral marketing application [49] where the relation is represented as a graph. Nodes in the graph correspond to users, who purchase items from a large online retailer. After purchasing an item, a user has the option of recommending the item to their friends via e-mail. These recommendations are represented as directed, timestamped edges between nodes. Should one of the recipients of this e-mail purchase the item through the link provided, the recommender receives a discount on their purchase. The recipients of the e-mail also have a financial incentive for purchasing the item through the provided link. The data consists of 3.9M users, over 0.5M products in four categories, covering almost a two year period. Each edge is typed, containing the product being recommended and the time the purchase was made by the recommender. This work showed the existence of cascades on a real world data set. By counting frequent subgraphs over bounded time windows, we showed that several assumptions made in theoretical models of cascades, such as additivity of influence (more recommenders means a user is more likely to purchase), do not hold in practice.

Matrices are not the only representation for mixing different relations. In recommendation systems one may have additional information about users, instead of additional information about products. For example, we have considered the problem of collaborative filtering where one additionally has buddy lists for users of an online gaming service. The relations are Purchased($user, game$) and the symmetric relation Buddy($user, user$). The purchases form a bipartite graph between users and games, and the buddy relation augments the graph with undirected links between users. One could exploit the structure of this graph by computing a random walk over it, but we are only interested in generating recommendations of games to users, instead of just between any two nodes in the graph. Our approach is to compute an absorbing random walk [78]: replace the undirected links between users and items by directed links from users to items, making the items absorbing states. The walk starts at the user we want to generate recommendations for. With probability $\alpha \in (0, 1)$ a transition is made to a user, either uniformly at random or biased using other features of users, such as demographics. With probability $(1 - \alpha)$ a transition is made on a user-product edge, uniformly at random. At convergence all walks terminate in a product node, which induces a distribution over products. Since the walk is not ergodic, the distribution over products depends on which user you started at, the recommendations are personalized. The larger the value of α, the farther a walk goes out on the social network before falling into an absorbing state. The main argument for our approach is computational. A naïve MATLAB implementation, that approximates the absorbing random walk with a finite number of steps, scales easily to 900,000 users.

3.3 Graphical Models

Matrix factorization can be viewed as a two layer graphical model, and collective matrix factorization allows parameters to be shared across multiple graphical models. Much of our proposed work moves collective matrix factorization towards increasingly broader classes of graphical models, especially those with plate representations (c.f., hierarchical priors). Our prior work in graphical models includes contributions to both structure learning and parameter estimation.

A Bayesian network is a factored representation of a probability distribution on n variables. The factored representation is presented as a directed acyclic graph whose nodes correspond to variables. Maximum a posteriori structure learning is the task of finding the most likely directed acyclic graph, or structure, given training data. Without harsh restrictions, such as limiting oneself to directed trees, structure learning is NP-hard [15]. Moreover, naive enumeration is utterly impractical for more than $n = 6$ since there are $O\left(n!2^{\binom{n}{2}}\right)$ possible structures [68]. We have developed a dynamic programming algorithm that makes a memory-speed tradeoff: $O(n2^n)$ memory to reduce the time complexity to $O(n2^n)$ [76]. The dynamic programming algorithm exploits the fact that common scoring functions for Bayesian networks can

be decomposed into selecting the best set of parents for a variable, out of 2^{n-1} possible choices, or parent sets. We have developed a data structure, P-caches, that record all relevant information about the parent sets, pruning combinations that we can prove could not exist in the optimal Bayesian network. We have shown that, under BDeu, the memory usage of P-caches is far smaller in practice than the bound would suggest. Moreover, we can take advantage of in-degree constraints to reduce the memory complexity exponentially, even in the worst case. Together, the dynamic programming technique with P-caches allows us to learn the provably optimal structure for $n = 26$ variables in a few hours on a desktop machine.

Given the structure of a Bayesian network on discrete variables, a point estimate of the parameters, Θ, can be computed by counting how often certain assignments of the variables occur in the data set. Point estimates of the parameters ignore uncertainty due estimating from a finite data set, but inference algorithms like junction trees [43, 36] or variable elimination [89, 22] assume that we have point estimates, at least when the variables are discrete. If point estimates are replaced by distributions over the parameters, then a query $P(X|E = e, \Theta)$ is itself a distribution. We propose a novel change to variable elimination that allows us to compute the mean and variance of a query response distribution, $E_\Theta[P(X|E = e, \Theta)]$ and $Var_\Theta[P(X|E = e, \Theta)]$, in the same asymptotic time as point inference: $O(n \exp(w))$, where w is the induced tree-width of the chosen variable ordering. The query response, a distribution on the $[0, 1]$ interval, can be approximated by a Beta distribution whose parameters are estimated using the computed mean and variance of a query. This allows us to produce error bars on queries—e.g., $P(X|E = e) = 0.78 \pm 0.13$ with probability 0.95.

3.4 Active Learning

Our prior work on active learning has centered on spatial statistics rather than relational learning. Monitoring spatial phenomena, such as temperature or rainfall, is done by placement of a small number of sensors that accurately model the phenomena is a small area around it. To monitor the entire space one can perform regression using the sensor readings to predict the value of interest at uninstrumented locations. While one could discretize the space, and model the discrete sites using a multivariate Gaussian distribution, we instead use an alternate approach from spatial statistics and learn a Gaussian process model: a nonparametric generalization of multivariate Gaussians that allows for infinitely many variables. We consider the problem of sensor placement in Gaussian processes, choosing locations for a fixed number of sensors that maximizes the predictive accuracy of the Gaussian process [30, 42].

We argue that a good criterion for placing k sensors is the mutual information between the instrumented and uninstrumented locations, as it maximally reduces the entropy at the uninstrumented locations. However, finding the k locations that maximize mutual information is NP-complete. We show that, under easy to satisfy conditions, mutual information is submodular and approximately monotonic. Thus the greedy selection algorithm—place one sensor at a time, maximizing the mutual information of that one sensor given those already placed—is a polynomial time approximation scheme that is guaranteed to be within $(1 - 1/e)$ of the optimal solution. In practice, our algorithm outperforms many classical techniques from experimental design and sensor placement (i.e., A,D,E-optimality) under root mean squared error of the Gaussian process predictor. Moreover, the greedy approach is significantly faster to compute than the aforementioned alternatives.

Fundamentally, the sensor placement problem is active learning in a linear model. The goal is to select features that maximize the accuracy of a linear model, without knowing the values the feature takes in the data. We speculate that some of these ideas should generalize to matrix factorizations, which are bilinear models: $X = UV^T$ is linear if either U or V is fixed.

4 Proposed Work

We propose extending collective matrix factorization to address the three questions posed in the introduction (Q1-Q3) in three application domains: movie rating prediction, transfer learning for fMRI activation, and mixed initiative item tagging.

4.1 Topic 1: Learning under Relational Uncertainty

Collective matrix factorization addresses link uncertainty, predicting whether a relation occurs between entities, or alternately the value of a relation between two entities. Since attributes can be encoded as predicates, we also support attribute uncertainty. However, our approach has a relatively rigid structure: relations are encoded as matrices, and tying entire factors $U^{(i)}$ together may not always be warranted. We propose the following extensions to collective matrix factorization:

- Collective Tensor Factorization: The current model is limited to matrices, and thus arity-two relations. An extension to tensors would allow higher arity relations. This differs from Banerjee et al. [3] in that they optimize with respect

to Bregman information, the expected value of the Bregman divergence, with a clustering constraint; whereas we propose extending MLMF models to tensors.

- Hierarchical Priors: The current model makes a closed-world assumption on entities. Hierarchical priors yield a generative model on entities. This would also allow us to encompass the maximum likelihood versions of popular plate models, such as latent Dirichlet allocation [5], into our framework.

- Conditional Learning: If one is concerned with predicting the value of only one relationship, then learning a set of joint models over each matrix may be an inefficient use of parameters. In augmented collaborative filtering, predicting ratings given the genres, as opposed to modeling distributions over each relation, is an example of the problem we would consider.

- Structure Learning: Each matrix factorization can be viewed as a two-layer model, where each entry in a matrix is a mixture of all k latent variables. Structure learning in this domain involves selecting subsets of latent variables that model each entity.

There are two views of what a statistical relational model represents [54]: a distribution over possible worlds, or a plate model containing nested sets of indexed random variables. Adding per-matrix losses in collective matrix factorization is neither, it proposes a distribution over values for each grounding of a relation, where shared entities correspond to shared parameters across these distributions. A plate model defines a joint distribution over all the entities. Adding per-matrix losses is computationally convenient, and works well in practice, but lacks a principled justification. It would be elucidating to consider whether there is an equivalent representation of our model as a distribution over possible worlds, e.g., using Markov Logic. If we can represent our approach in Markov Logic, it would clarify exactly how first-order probabilistic logic can subsume matrix factorization. If it is not possible, the reasons why should clarify the representational limits of our approach.

We believe that there is a continuum of models between the commonly used propositional ones like linear regression and general models used in relational learning. Situating a wide variety of well-understood techniques, like matrix factorization and plate models, into a unified relational framework would ease the adoption relational learning by practitioners. A small example of this line of reasoning is the relationship we have drawn between standard regression and clustering techniques and matrix factorization and matrix bi-clustering, which could quite naturally be generalized to tensors. A recurring theme during our review of existing methods is the focus on domain specific variations of matrix factorization and plate models. Generalizing these problems is the first step to building flexible, reusable building blocks for a variety of relational domains. For example, the similarity between plate models and DAPER [32], a relational modeling language built on top of entity-relationship diagrams, is well-understood; but DAPER is a purely conceptual exercise that has never been implemented.

4.2 Topic 2: Computationally Efficient Relational Models

Attribute-value data sets can be large in two senses: many records and many attributes. Similarly, relational data can be large in the sense of containing many entities, large in the sense of containing many observed relations, or large in the sense of having many relations. We are interested in models that scale well in all three senses. The stochastic Newton step addresses scalability in the number of observed relations, but there are several other ways to improve scalability:

- Online and out-of-core learning: Online learning refers to the case where observed relations appear over time, during the learning process. Out-of-core learning refers to the case where the data cannot fit in system memory. The alternating projection algorithm described in Section 3.1 can deal with both cases, but its performance has yet to be verified.

- Random walks and Matrix Factorization: There is an established relationship between PageRank [62] and a model similar to collective matrix factorization on two related matrices (e.g., a document-word matrix and a interdocument link matrix) [17]. The critical difference is that the two matrices are stacked together, and the low-rank factors are added as pseudo-features to the data set. Recursively adding factors yields the relationship to PageRank. We believe that there is a relationship between more general cases of collective matrix factorization and random walks on graphs, which opens up the possibility of using approximation techniques on random walks to speed up learning in our model.

4.3 Topic 3: Relational Active Learning

An active learner is one that can pose queries, and use the responses it receives along with given data to learn a model. In propositional domains the only objects are entities, and so the only queries we can pose involve revealing the value

of attributes and class labels for a particular entity. In relational domains queries can involve the relations themselves, thus allowing for richer forms of feedback. In addition to requesting the value of an attribute or label for an entity, we want to have an algorithm that can ask for

- The value of a relation, which subsumes querying the value of an attribute.
- A clustering of entities, or conversely a request for the user to remove entities they feel do not belong in the group.
- Inductive biases, weights on rules such as

  $$\forall i\ \exists r \exists s\ \text{IsGenre}(movie_r, Drama) \wedge \text{IsGenre}(movie_s, Comedy) \wedge \text{Rating}(user_i, movie_r) > \text{Rating}(user_i, movie_s) \implies \\ \text{IsGenre}(movie_m, Drama) \wedge \text{IsGenre}(movie_n, Comedy) \wedge \text{Rating}(user_i, movie_m) > \text{Rating}(user_i, movie_n)$$

  That is, if a user rates one drama higher than one comedy, it is more likely that they will rate another drama higher than another comedy.

Clustering entities and relations simultaneously from data is known as statistical predicate invention [39, 38]. Our interest is in eliciting such relationships using both data and queries posed to a user.

4.4 Application 1: Movie Rating Prediction

This is an example of augmented collaborative filtering, which has been our primary test application for collective matrix factorization. The ratings come from the Netflix data set [60], augmented with movie and actor information from IMDB[8] [34]. There are $479,820$ users, $11,825$ movies, $220,579$ actors, and 28 genres in the database. The goal is to predict held-out ratings, where the users are known in advance. Since many users have made only a few ratings, there is a strong case to be made for hierarchical priors on users. Moreover, there is an approximate power law distribution on the number of ratings per movie. A few popular movies have most of the ratings. Folding-in new users and movies provides a relatively clear motivation for hierarchical priors. Since the primary task involves predicting values of Rating, there is a prima facie case for a conditional version of collective matrix factorization. Preliminary results suggest that the benefit of adding actors, even restricted to the most popular ones, provides little information beyond that already provided by genres.

The Netflix prize itself may provide a convenient way of comparing the performance of our approach to other techniques. However, the use of side information from IMDB appears to preclude competing under the terms of the contest. Moreover, we suspect that our efforts are better spent generalizing our model to handle a wide variety of scoring measures, rather than just root mean-squared error on link prediction, the Netflix prize objective.

4.5 Application 2: Relational Learning using Functional MRI Activation

Functional MRI (fMRI) measures neural activation in regions of the brain. There is a correspondence between spatial patterns of neural activation and the concept being thought about by the subject. In text modeling there is a similar correspondence between patterns of word co-occurrence and the semantic categories (topics) of words. A recent study [56] showed that correlations between these two data sources can be used to predict brain activation for an arbitrary concrete noun. In the above study, fMRI images are collected for people presented with a semantically coherent word—e.g., cat, house, cup, chisel. This data has been augmented with word co-occurrence counts extracted from the Google 5-gram data set [9]—i.e., the n-gram counts measure how frequently two words occur within five words of each other.

We propose applying collective matrix factorization to this domain, using the data layout in Figure 3. There are two tasks we would like to explore:

- fMRI Image Prediction: Predict the fMRI image for a new person given that we know the word presented, and the co-occurrences for that word.
- Word Clustering: Given fMRI images and the categories that each image corresponds to, predict how frequently two words co-occur. fMRI images may also provide a novel (albeit impractical) source of side information for topic modeling in text. If the topics (word clusters) change significantly when fMRI activations are added in, it would provide some evidence of a mismatch between how words are associated in a topic model vs. how words are associated in the brain.

The fMRI image prediction task has a well-defined evaluation metric from Mitchell et al. [56]. Measuring the quality of a clustering is invariably a fraught issue, since there are many definitions of cluster quality. We believe that link prediction and word similarity ranking measures should provide a believable quantitative measure.

[8] We acknowledge Jon Ostlund for his work merging the Netflix and IMDB data sets.

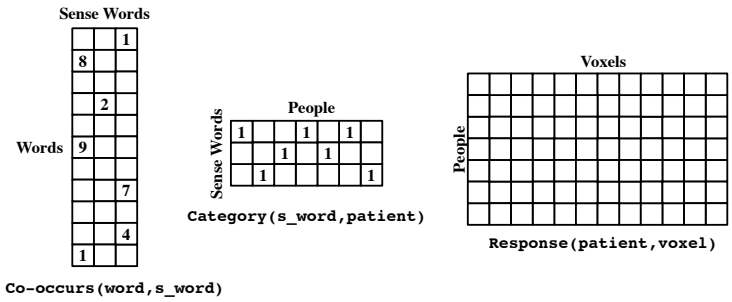


Figure 3: fMRI and word co-occurrence data as a set of related matrices. Voxels are real-valued.

4.6 Application 3: Mixed Initiative Item Tagging

Personal information management is fundamentally about maintaining relations between disparate types of information: e-mail messages, contacts, files, web pages, tasks on todo lists, calendar entries, etc. While the main technique used for finding information in these systems is keyword search, one can envision alternatives that formulate and refine searches using a combination of keywords, the type of information, and relations between the types. An example of such a system is FELDSPAR [14, 13], which allows users to graphically construct queries of the form "Where are the (type) related to (type) related to the (type): (keywords) ?", e.g., "Where are the folders related to files attached to emails containing the words 'KDD Submission 2008' ?". The related-to relation is really a set of different predicates, one for each combination of the seven types supported by the system.

Even restricting our initial consideration to e-mail yields an interesting relational problem. An e-mail inbox consists of messages, senders, attachments, and most importantly folder tags. Folder tags are a generalization of folders, where a message can belong to multiple folders. We allow for relationships between folder tags. For example, a tag for "submitted papers" for e-mails regarding papers under review, and sub-tags "KDD submission" and "ECML submission" containing messages specific to each conference. We propose considering the problem of tagging e-mails that incorporates user-created tags and assignments to messages, but which can also request intervention by the user. The input is a user's e-mail inbox, along with existing folder tags. The goal is to suggest folder tags for messages where the interactions between the user and system include:

- The user assigning tags to an e-mail, either of their own volition or at the request of the system.
- The user removing e-mails which do not belong from an automatically generated folder tag. The clusters can contain both e-mails and other tags. For example a tag for "submitted papers" may include both e-mails regarding the submission, and the sub-tags "KDD submission" and "ECML submission".
- The user placing tags in a hierarchy, either by request or of their own volition.

We refer to this problem as mixed initiative tagging, since the user can provide feedback either at the request of the learning algorithm or of their own initiative. The problem becomes even more interesting when additional data types, such as tasks on a todo list, are incorporated: one could include actions that correspond to linking e-mails to projects or specific items on a todo list.

The closest propositional alternative is presented in Mitchell et al. [55], which builds a frame-based representation of activities, a collection of related entities, such as contacts, emails, and meetings, using propositional clustering based on keywords. The clusters are refined post-hoc by finding disconnected components in the social graph formed by e-mails sent between contacts.

4.7 Timeline

- **May 2008**: Thesis Proposal Writing
- **June–August 2008**: Complete large scale and online variants of augmented collaborative filtering
- **June–September 2008**: Joint fMRI/word modeling
- **September 2008–March 2009**: Mixed Initiative Item Tagging
- **April 2009**: Thesis Writing
- **May 2009**: Thesis Defense

5 Conclusion

The history of machine learning is littered with approaches that are based on expressive, deterministic, logic-based representations, that have seen few practical applications. In the move from deterministic to probabilistic models, we have restricted ourselves to propositional data and propositional models. Propositional representations work well when one has a fixed and relatively complete view of an entity in terms of its attributes. Domains where one has many types of entities—where much of the information is in links between entities rather than their attributes—begs for something more expressive than propositional models. Since such domains often contain less direct information about any single entity than the propositional case, it seems wise to exploit the relations between a large number of entities: hence our interest in computationally efficient relational models. Collective matrix factorization is our first step in this direction.

Focusing on prediction and clustering, even on very large scale data sets, is only the beginning of the story. While attribute-value representations provide a solid framework for reasoning about uncertainty, extracting information from, or verifying hypotheses using, human intervention is laborious. We believe that relations, being far more expressive than attributes, will allow for simpler and more natural forms of information elicitation.

Finally, we stand on the position that for any of these models to influence practitioners and potential users of machine learning techniques, we must show that our models give us something we didn't have before. In the case of movie rating prediction, a better quality of rating predictions by using side information. In the case of the fMRI domain, a way of confirming the validity of the similarity measures induced by topic models using fMRI images. In the case of personal information management, a way to exploit passively collected user actions and actively elicited information to organize and find e-mails, files, todo lists, etc.

References

[1] D. Agarwal and S. Merugu. Predictive discrete latent factor models for large scale dyadic data. In P. Berkhin, R. Caruana, and X. Wu, editors, *KDD*, pages 26–35, New York, NY, USA, 2007. ACM.

[2] A. Banerjee, S. Merugu, I. S. Dhillon, and J. Ghosh. Clustering with Bregman divergences. *J. Mach. Learn. Res.*, 6:1705–1749, 2005.

[3] A. Banerjee, S. Basu, and S. Merugu. Multi-way clustering on relation graphs. In *SDM*. SIAM, 2007.

[4] I. Bhattacharya and L. Getoor. Collective entity resolution in relational data. *ACM Trans. Knowl. Discov. Data*, 1(1):5, 2007.

[5] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

[6] L. Bottou. Online algorithms and stochastic approximations. In D. Saad, editor, *Online Learning and Neural Networks*. Cambridge UP, Cambridge, UK, 1998.

[7] L. Bottou and Y. LeCun. Large scale online learning. In S. Thrun, L. K. Saul, and B. Schölkopf, editors, *NIPS*. MIT Press, 2003.

[8] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge UP, 2004.

[9] T. Brants and A. Franz. Web 1T 5-gram corpus, version 1. Electronic, Sept. 2006.

[10] I. Bratko. *PROLOG Programming for Artificial Intelligence*. Addison-Wesley Longman, Boston, MA, USA, 1990.

[11] L. Bregman. The relaxation method of finding the common points of convex sets and its application to the solution of problems in convex programming. *USSR Comp. Math and Math. Phys.*, 7:200–217, 1967.

[12] Y. Censor and S. A. Zenios. *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford UP, 1997.

[13] D. H. Chau, B. Myers, and A. Faulring. Feldspar: A system for finding information by association. CHI 2008 Workshop on Personal Information Management: PIM 2008, apr 2008.

[14] D. H. Chau, B. Myers, and A. Faulring. What to do when search fails: finding information by association. In M. Czerwinski, A. M. Lund, and D. S. Tan, editors, *CHI*, pages 999–1008. ACM, 2008.

[15] D. M. Chickering. Learning Bayesian networks is NP-complete. In *Learning from Data*. Springer, 1996.

[16] D. Cohn and T. Hofmann. The missing link–a probabilistic model of document content and hypertext connectivity. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *NIPS*. MIT Press, 2000.

[17] D. Cohn, D. Verma, and K. Pfleger. Recursive attribute factoring. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *NIPS*, Cambridge, MA, 2006. MIT Press.

[18] M. Collins, S. Dasgupta, and R. E. Schapire. A generalization of principal component analysis to the exponential family. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *NIPS*. MIT Press, 2001.

[19] D. J. Cook and L. B. Holder. Graph-based data mining. *IEEE Intelligent Systems*, 15(2):32–41, 2000.

[20] M. Davis and H. Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, 1960.

[21] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, 1962.

[22] R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artif. Intell.*, 113(1-2):41–85, 1999.

[23] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *IJCAI*, pages 1300–1309, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[24] K. R. Gabriel and S. Zamir. Lower rank approximation of matrices by least squares with any choice of weights. *Technometrics*, 21(4):489–498, 1979.

[25] A. Galstyan and P. R. Cohen. Empirical comparison of "hard" and "soft" label propagation for relational classification. In H. Blockeel, J. Ramon, J. W. Shavlik, and P. Tadepalli, editors, *ILP*, volume 4894 of *Lecture Notes in Computer Science*, pages 98–111. Springer, 2007.

[26] A. Gelman, J. B. Carlin, and H. S. Stern. *Bayesian Data Analysis*. CRC Press, 2nd edition, 2004.

[27] L. Getoor, N. Friedman, D. Koller, and A. Pfeffer. Learning probabilistic relational models. *Relational Data Mining*, pages 307–335, 2001.

[28] G. H. Golub and C. F. V. Loan. *Matrix Computions*. John Hopkins UP, 3rd edition, 1996.

[29] G. J. Gordon. Generalized^2 linear^2 models. In S. Becker, S. Thrun, and K. Obermayer, editors, *NIPS*. MIT Press, 2002.

[30] C. Guestrin, A. Krause, and A. P. Singh. Near-optimal sensor placements in Gaussian processes. In L. D. Raedt and S. Wrobel, editors, *ICML*, pages 265–272. ACM, 2005.

[31] J. Hartigan. *Clustering Algorithms*. Wiley, 1975.

[32] D. Heckerman, C. Meek, and D. Koller. Probabilistic entity-relationship models, PRMs, and plate models. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*, chapter 7, pages 201–238. MIT Press, 2008.

[33] T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR*, pages 50–57. ACM, 1999.

[34] Internet Movie Database Inc. IMDB alternate interfaces. `http://www.imdb.com/interfaces`, Jan. 2007.

[35] D. Jensen and J. Neville. Linkage and autocorrelation cause feature selection bias in relational learning. In C. Sammut and A. G. Hoffmann, editors, *ICML*, pages 259–266, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[36] F. V. Jensen, S. L. Lauritzen, and K. G. Olesen. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quaterly*, 4:269–282, 1990.

[37] Q. Ke and T. Kanade. Robust l_1 norm factorization in the presence of outliers and missing data by alternative convex programming. In *CVPR (1)*, pages 739–746. IEEE Computer Society, 2005.

[38] C. Kemp, J. B. Tenenbaum, T. L. Griffiths, T. Yamada, and N. Ueda. Learning systems of concepts with an infinite relational model. In *AAAI*. AAAI Press, 2006.

[39] S. Kok and P. Domingos. Statistical predicate invention. In Z. Ghahramani, editor, *ICML*, volume 227 of *ACM International Conference Proceeding Series*, pages 433–440, New York, NY, USA, 2007. ACM.

[40] R. Kowalski. Predicate logic as programming language. In *Proceedings IFIP Congress, Stockholm*, 1974.

[41] S. Kramer, N. Lavrač;, and P. Flach. *Propositionalization approaches to relational data mining*, chapter 11, pages 262–286. Springer-Verlag New York, Inc., New York, NY, USA, 2000.

[42] A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *J. Mach. Learn Res.*, 9:235–284, 2008.

[43] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *J. Roy. Statist. Soc. Ser. B*, 50(2):157–224, 1988.

[44] N. Lavrac and S. Dzeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, New York, 1994.

[45] N. Lavrac, S. Dzeroski, and M. Grobelnik. Learning nonrecursive definitions of relations with linus. In *EWSL '91*, pages 265–281, London, UK, 1991. Springer-Verlag. Proceedings of the European Working Session on Machine Learning.

[46] N. Lavrac, F. Zelezny, and P. Flach. *RSD: Relational Subgroup Discovery through First-Order Feature Construction*, volume 2583/2003 of *Lecture Notes in Computer Science*, pages 149–165. Springer Berlin, 2003.

[47] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. In T. K. Leen, T. G. Dietterich, and V. Tresp, editors, *NIPS*. MIT Press, 2001.

[48] J. D. Leeuw. Block relaxation algorithms in statistics, 1994.

[49] J. Leskovec, A. Singh, and J. M. Kleinberg. Patterns of influence in a recommendation network. In W. K. Ng, M. Kitsuregawa, J. Li, and K. Chang, editors, *PAKDD*, volume 3918 of *Lecture Notes in Computer Science*, pages 380–389. Springer, 2006.

[50] B. Long, Z. M. Zhang, X. Wú;, and P. S. Yu. Spectral clustering for multi-type relational data. In W. W. Cohen and A. Moore, editors, *ICML*, pages 585–592, New York, NY, USA, 2006. ACM Press.

[51] B. Long, Z. M. Zhang, X. Wu, and P. S. Yu. Relational clustering by symmetric convex coding. In Z. Ghahramani, editor, *ICML*, volume 227 of *ACM International Conference Proceeding Series*, pages 569–576, New York, NY, USA, 2007. ACM.

[52] B. Long, Z. M. Zhang, and P. S. Yu. A probabilistic framework for relational clustering. In P. Berkhin, R. Caruana, and X. Wu, editors, *KDD*, pages 470–479, New York, NY, USA, 2007. ACM.

[53] P. McCullagh and J. Nelder. *Generalized Linear Models*. Chapman and Hall: London., 1989.

[54] B. Milch and S. J. Russell. First-order probabilistic languages: Into the unknown. In S. Muggleton, R. P. Otero, and A. Tamaddoni-Nezhad, editors, *ILP*, volume 4455 of *Lecture Notes in Computer Science*, pages 10–24. Springer, 2006.

[55] T. M. Mitchell, S. H. Wang, Y. Huang, and A. Cheyer. Extracting knowledge about users' activities from raw workstation contents. In *AAAI*. AAAI Press, 2006.

[56] T. M. Mitchell, S. V. Shinkareva, A. Carlson, K.-M. Chang, V. L. Malave, R. A. Mason, and M. A. Just. Predicting human brain activity associated with the meanings of nouns. *Science*, 2008. Preprint.

[57] S. Muggleton. Stochastic logic programs. In L. De Raedt, editor, *Proc. 5th Intl. Workshop on Inductive Logic Programming*, page 29. Department of Computer Science, Katholieke Universiteit Leuven, 1995.

[58] S. Muggleton and W. L. Buntine. Machine invention of first order predicates by inverting resolution. In *ML*, pages 339–352, 1988.

[59] S. Muggleton and C. Feng. Efficient induction of logic programs. In *ALT*, pages 368–381, 1990.

[60] Netflix. Netflix prize dataset. `http://www.netflixprize.com`, Jan. 2007.

[61] P. Paatero and U. Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics,*, 5:111–126, 1994.

[62] L. Page, S. Brin, R. Motwani, and T. Winograd. The PageRank citation ranking: Bringing order to the web. Technical report, Stanford University, 1998.

[63] F. Pereira and G. Gordon. The support vector decomposition machine. In W. W. Cohen and A. Moore, editors, *ICML*, pages 689–696, New York, NY, USA, 2006. ACM Press.

[64] A. Popescul and L. Ungar. Structural logistic regression for link analysis. KDD Workshop on Multi-Relational Data Mining, 2003.

[65] J. Quinlan. Learning logical definitions from relations. *Mach. Learn.*, 5(3):239–266, 1990.

[66] J. D. M. Rennie and N. Srebro. Fast maximum margin matrix factorization for collaborative prediction. In L. D. Raedt and S. Wrobel, editors, *ICML*, pages 713–719, New York, NY, USA, 2005. ACM Press.

[67] M. Richardson and P. Domingos. Markov logic networks. *Mach. Learn.*, 62(1-2):107–136, 2006.

[68] R. Robinson. Counting labelled acyclic digraphs. In *New Directions in the Theory of Graphs*, pages 239–273. Academic Press, 1973.

[69] C. Sammut. *Learning concepts by performing experiments*. PhD thesis, University of New South Wales, Australia, 1981.

[70] C. Sammut and R. Banerji. *Learning concepts by asking questions*, pages 167–192. Morgan Kaufman, 1986.

[71] A. I. Schein, L. K. Saul, and L. H. Ungar. A generalized linear model for principal component analysis of binary data. In C. M. Bishop and B. J. Frey, editors, *AISTATS*. Society for Artificial Intelligence and Statistics, Jan. 2003.

[72] M. Schmidt, G. Fung, and R. Rosales. Fast optimization methods for L1 regularization: A comparative study and two new approaches. In J. N. Kok, J. Koronacki, R. L. de Mántaras, S. Matwin, D. Mladenic, and A. Skowron, editors, *ECML*, volume 4701 of *Lecture Notes in Computer Science*, pages 286–297. Springer, 2007.

[73] B. Selman, H. Kautz, and B. Cohen. Local search strategies for satisfiability testing. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 26:521–532, 1996.

[74] E. Shapiro. *Algorithmic Program Debugging*. MIT Press, 1983.

[75] A. Singh and G. Gordon. Relational learning via collective matrix factorization. Submitted KDD, 2008.

[76] A. Singh and A. W. Moore. Finding optimal Bayesian networks by dynamic programming. Technical Report CMU-CALD-05-106, Carnegie Mellon University, 2005.

[77] A. P. Singh and G. G. Gordon. Collective matrix factorization. Submitted ECML/PKDD, 2008.

[78] A. P. Singh, A. Gunawardana, C. Meek, and A. C. Surendran. Recommendations using absorbing random walks. In *North East Student Colloquium on Artificial Intelligence (NESCAI)*, 2007.

[79] N. Srebro and T. Jaakola. Weighted low-rank approximations. In T. Fawcett and N. Mishra, editors, *ICML*. AAAI Press, 2003.

[80] N. Srebro, J. D. Rennie, and T. S. Jaakkola. Maximum-margin matrix factorization. In *NIPS*, 2004.

[81] A. Srinivasan. The aleph manual. Technical report, Computing Laboratory, Oxford, 2000. `http://web.comlab.ox.ac.uk/oucl/research/areas/machlearn/Aleph/`.

[82] P. Stoica and Y. Selen. Cyclic minimizers, majorization techniques, and the expectation-maximization algorithm: a refresher. *Signal Processing Magazine, IEEE*, 21(1):112–114, Jan 2004.

[83] H. Tong, C. Faloutsos, B. Gallagher, and T. Eliassi-Rad. Fast best-effort pattern matching in large attributed graphs. In P. Berkhin, R. Caruana, and X. Wu, editors, *KDD*, pages 737–746. ACM, 2007.

[84] T. van Allen, A. Singh, R. Greiner, and P. Hooper. Quantifying the uncertainty of a belief net response: Bayesian error-bars for belief net inference. *Artif. Intell.*, 172(4-5):483–513, Mar. 2008.

[85] D. H. D. Warren, L. M. Pereira, and F. Pereira. Prolog–the language and its implementation compared with Lisp. *SIGPLAN Not.*, 12(8):109–115, 1977.

[86] M. Welling, C. Chemudugunta, and N. Sutter. Deterministic latent variable models and their pitfalls. In *SDM*. SIAM, 2008.

[87] K. Yu, S. Yu, and V. Tresp. Multi-label informed latent semantic indexing. In R. A. Baeza-Yates, N. Ziviani, G. Marchionini, A. Moffat, and J. Tait, editors, *SIGIR*, pages 258–265, New York, NY, USA, 2005. ACM.

[88] S. Yu, K. Yu, V. Tresp, H.-P. Kriegel, and M. Wu. Supervised probabilistic principal component analysis. In T. Eliassi-Rad, L. H. Ungar, M. Craven, and D. Gunopulos, editors, *KDD*, pages 464–473. ACM, 2006.

[89] N. Zhang and D. Poole. A simple approach to Bayesian network computations. In *Proc. of the 10th Canadian Conf. on AI*, 1994.

[90] S. Zhu, K. Yu, Y. Chi, and Y. Gong. Combining content and link for classification using matrix factorization. In W. Kraaij, A. P. de Vries, C. L. A. Clarke, N. Fuhr, and N. Kando, editors, *SIGIR*, pages 487–494, New York, NY, USA, 2007. ACM Press.