Factor Graphs for Relational Regression

Sumit Chopra
Courant Institute of Mathematical Sciences
New York University
sumit@cs.nyu.edu

Trivikaraman Thampy
Department of Economics
New York University
thampy@nyu.edu

John Leahy
Department of Economics
New York University
john.leahy@nyu.edu

Andrew Caplin
Department of Economics
New York University
ac1@nyu.edu

Yann LeCun
Courant Institute of Mathematical Sciences
New York University
yann@cs.nyu.edu

Technical Report: TR2007-906

Abstract

Traditional methods for supervised learning involve treating the input data as a set of independent, identically distributed samples. However, in many situations, the samples are related in such a way that variables associated with one sample depend on other samples. We present a new form of relational graphical model that, in addition to capturing the dependence of the output on sample specific features, can also capture hidden relationships among samples through a non-parametric latent manifold. Learning in the proposed graphical model involves simultaneously learning the non-parametric latent manifold along with a non-relational parametric model. Efficient inference algorithms are introduced to accomplish this task. The method is applied to the prediction of house prices. A non-relational model predicts an "intrinsic" price of the house which depends only on its individual characteristics, and a relational model estimates a hidden surface of "desirability" coefficients which links the price of a house to that of similar houses in the neighborhood.

1 Introduction

A majority of supervised learning algorithms assume that the input data is composed of a set of independent and identically distributed samples. With most methods, the model is fed with the data of a single sample which is processed independently of the other samples. However in a number of real-world problems the data inherently possesses a *relational structure*. The value of variables associated with each sample not only depends on features specific to that sample, but also on the features and variables of other related samples. An example is the problem of web page classification. Web pages that are linked to each other are more likely to be on the same topic. Exploiting the link structure between web pages may improve the accuracy of topic classification.

Another task that is inherently relational is predicting the price of real estate properties. Each house is described by a vector X of features that are specific to that house, such as the number of bed-

rooms, number of bathrooms, etc. The price of a house is clearly a function of these features: given a particular locality, a large 5 bedroom, 3 bathroom mansion will be more expensive than a smaller 2 bedroom, 1 bathroom house. In addition, the price of a house is also influenced by the features of the neighborhood in which it lies: given two houses of similar characteristics, the one that is located in an up-market locality will be more expensive than the one located in a poor locality. Some of these neighborhood features are measurable, such as the quality of the local schools, median house hold income etc. However most of the features, that make one particular neighborhood "desirable" to live in as compared to the others, are very difficult to measure directly, and are merely reflected in the prices of houses in that area. As the mantra of North-American real estate indicates ("Location, location, location"), one must take into account the price of similar neighboring houses when predicting the price of a particular house. Unfortunately, the "desirability" of a location is not directly measurable and hence must be treated as a latent variable. It must be estimated as part of the learning process, and efficiently inferred for unseen samples.

This paper introduces a general framework for doing regression in relational data using factor graphs. In a relational factor graph, a single graph models the entire collection of samples. The relations between samples are captured by factors that connect variables associated with multiple samples. An architecture of the relational factor graph is presented that allows efficient inference algorithms for continuous variables with relational dependencies. The method is applied to the problem of real estate price prediction and compared with other standard methods that have been conventionally used for this particular problem. From the analysis of the experimental results we conclude that the method proposed if far superior that any of the previous methods and it indeed learns a "desirability" manifold of the neighborhoods of the area that is reflective of the real world situation

1.1 Relational Graphical Models

A number of authors have recently proposed architectures and learning algorithms that make use of relational structure within the data, with applications to web page labeling, and topic classification for scientific publication. Most of the early techniques are based on the idea of influence propagation. This involves iteratively refining the prediction of a sample by propagating information from other samples that are connected to it through the relationship graph [6, 2, 11, 17, 15, 3, 9, 4]. Probabilistic Relational Models (PRMs) were introduced in [12, 7] as an extension of Bayesian networks to relational data. A PRM is a single Bayesian network over the entire data that specifies the joint probability distribution over all the attributes and samples. Learning algorithms for PRMs have originally been generative [19]. Discriminative extensions, called Relational Markov Networks (RMN) have later been proposed [18]. Unlike PRMs, RMNs are undirected discriminative models that can capture cyclic dependencies between samples. RMNs have been applied to tasks in which the variables to be predicted (the *answer variables*) are discrete, and the inter-sample dependencies act on the labels. When the relational graph has cycles, the partition function of the conditional distribution of labels given the inputs is intractable and one must resort to approximate methods, such as loopy belief propagation. Furthermore, most RMNs presented in the literature are parametric models with a log-likelihood function that is linear in the parameters that are subject to learning. Though the problem of predicting house prices has a long history in economics [1, 5, 8, 10, 14, 16], very little has been done in the direction of actually predicting the house prices. Most models focus on estimating the model parameters efficiently and precisely and on index construction.

The class of models introduced in this paper is different in several ways: 1. it pertains to *relational regression problems* in which the answer variable is continuous (e.g. the price of a house); 2. it allows inter-sample dependencies through hidden variables (which may also be continuous), as well as through the answer variable; 3. it allows log-likelihood functions that are non-linear in the parameters, which leads to non-convex loss functions but are considerably more flexible; 4. it allows the use the non-parametric models for the relational factors of the factor graph, while providing an efficient inference algorithm for new samples; 5. it eliminates the intractable partition function problem through appropriate design of the relational and non-relational factors of the factor graph.

Relational graphical models can be seen as a special kind *structured-output model* in which the entire training set is taken as input. Probabilistic structured-output models are sometimes marred by intractable partition functions over the answer variables or latent variables. The Energy-Based Learning approach [13] advocates circumventing the problem by: 1. minimizing an energy function (un-normalized negative log probability) over the latent variables instead of marginalizing the

distribution over them; 2. training the model using alternative loss functions that do not involve an intractable log partition function that involve summing over all configurations of answer variables. The approach followed in this paper is somewhat similar in spirit.

2 The Relational Factor Graph

We are given a set of N training samples, each of which is described by a sample-specific feature vector X^i and an answer Y^i to be predicted. We will use the collective notation $\mathbf{X} = \{X^i,\ i = 1 \dots N\}$, $\mathbf{Y} = \{Y^i,\ i = 1 \dots N\}$ to denote the entire collection of input and answer variables. In the case where there are no hidden variables, the Relational Factor Graph (RFG) is defined by an *energy function* of the form

$$E(W, \mathbf{Y}, \mathbf{X}) = E(W, Y^1, \dots, Y^N, X^1, \dots, X^N), \tag{1}$$

in which W is the set of parameters to be estimated by learning. This energy function can be used to derive a conditional distribution over the answer variables using a Gibbs distribution

$$P(\mathbf{Y}|\mathbf{X}, W) = \frac{e^{-\beta E(W,\mathbf{Y},\mathbf{X})}}{\int_{\mathbf{Y}' \in \mathcal{Y}} e^{-\beta E(W,\mathbf{Y}',\mathbf{X})}}, \tag{2}$$

where $\mathcal{Y}$ is the set of all possible combinations of answer variables, and β is a suitably chosen positive constant.

Most applications in the domain of relational learning require the use of hidden (latent) variables. Let us denote by $\mathbf{Z}$ the set of latent variables associated with the problem. Then the energy associated with the system in this case is given by

$$E(W, \mathbf{Z}, \mathbf{Y}, \mathbf{X}) = E(W, \mathbf{Z}, Y^1, \dots, Y^N, X^1, \dots, X^N). \tag{3}$$

The conditional distribution in this case can be obtained by marginalizing over the latent variables to give

$$P(\mathbf{Y}|\mathbf{X}, W) = \int_{\mathbf{Z}' \in \mathcal{Z}} P(\mathbf{Y}, \mathbf{Z}'|\mathbf{X}, W) = \int_{\mathbf{Z}' \in \mathcal{Z}} \frac{e^{-\beta E(W,\mathbf{Z}',\mathbf{Y},\mathbf{X})}}{\int_{\mathbf{Y}' \in \mathcal{Y}, \mathbf{Z}'' \in \mathcal{Z}} e^{-\beta E(W,\mathbf{Z}'',\mathbf{Y}',\mathbf{X})}}. \tag{4}$$

This marginalization reduces down to using equation 2 with a new free energy function of the following form [13]:

$$E(W, \mathbf{Y}, \mathbf{X}) = -\frac{1}{\beta} \log \int_{\mathbf{Z}' \in \mathcal{Z}} e^{-\beta E(W,\mathbf{Z}',\mathbf{Y},\mathbf{X})} \tag{5}$$

It is easy to show that for large values of β, the above equation can be approximated by $E(W, \mathbf{Y}, \mathbf{X}) = \min_{\mathbf{Z}' \in \mathcal{Z}} E(W, \mathbf{Z}', \mathbf{Y}, \mathbf{X})$. In other words, the marginalization over $\mathbf{Z}$ can be replaced by a simple minimization. This is a practically incontrovertible shortcut when $\mathbf{Z}$ is continuous and very high-dimensional, because a marginalization would be impractical.

This relational factor graph can be trained by maximizing the likelihood of the data or by minimizing the negative log conditional probability of the answer variables with respect to the parameter W. That is accomplished by minimizing the negative log likelihood loss $\mathcal{L}$:

$$\mathcal{L}(W, \mathbf{Y}, \mathbf{X}) = E(W, \mathbf{Y}, \mathbf{X}) + \frac{1}{\beta} \log \int_{\mathbf{Y}' \in \mathcal{Y}} e^{-\beta E(W,\mathbf{Y}',\mathbf{X})}, \tag{6}$$

where the energy $E(W, \mathbf{Y}, \mathbf{X})$ is given by equation 5. The second term is the log partition function. Minmizing this loss with respect to W can be done with a gradient-based method: $W \leftarrow W - \eta \nabla_W \mathcal{L}(W, \mathbf{Y}, \mathbf{X})$, where η is a suitably chosen positive-definite matrix. The gradient of the loss is simply given by

$$\nabla_W \mathcal{L}(W, \mathbf{Y}, \mathbf{X}) = \nabla_W E(W, \mathbf{Y}, \mathbf{X}) - \int_{\mathbf{Y}' \in \mathcal{Y}} P(\mathbf{Y}'|\mathbf{X}, W) \nabla_W E(W, \mathbf{Y}', \mathbf{X}) \tag{7}$$

Other loss functions with different contrastive terms could also be used [13].

Given a test sample feature vector X^0, the process of inference involves predicting the value of the corresponding answer variable Y^0 using the model. One way of doing this is to find the maximum-likelihood value of the answer Y^{0*} and return it. This is performed by minimizing the energy function augmented with the test sample (X^0, Y^0)

$$Y^{0*} = \text{argmin}_{Y^0} \left(\min_{\mathbf{Z}} \{E(W, \mathbf{Z}, Y^0, \ldots, Y^N, X^0, \ldots, X^N)\} \right). \quad (8)$$

To be usable on new test samples without requiring excessive work, the energy function must be carefully constructed is such a way that the addition of a test sample in the arguments will not require retraining the entire system, or re-estimating some high-dimensional hidden variable each time a new test sample comes in. Moreover, the parameterization must be designed in such a way that its estimation on the training sample will actually result in a good prediction on test samples.

We now discuss the efficient training and inference algorithm for the relational factor graphs with the help of an instance of such graphs that was used for the problem of real estate price prediction. Consider the factor graph shown in figure **??**. Each sample i (in this case the i-th house) is associated with two latent variables D^i and Z^i and two factors E^i_{xyz} and E^i_{zz}. The total energy of the factor graph is the sum of all the energies of all the factors. The first factor of each sample $E^i_{xyz}(W_{xyz}, X^i, Y^i, d^i)$ is non-relational, and captures the sample-specific dependencies. Thus it captures the dependence of the price of a house on its individual characteristics, like number of bedrooms, number of bathrooms etc. The parameters W_{xyz} must be shared across all the instances of E^i_{xyz}. That way, when a test sample is added to the model, its E^0_{xyz} factor simply inherits the shared parameter. The second factor $E^i_{zz}(W_{zz}, \mathbf{Z}, d^i)$ is relational in nature, and captures the dependencies between samples via a set of latent (hidden) variables Z^i. These dependencies influence the answer for a sample through the intermediary of hidden variable D^i. Thus in the context of house price prediction, D^i can represent the estimated desirability of place where the house is located. Since the desirability of a house is generally very similar to that of its neighbors, the relational structure may be used advantageously to enforce a local smoothness constraint on the D^i's. One may ask why we chose to have separate D^i and Z^i variables for each sample, when perhaps a single set of variables would have sufficed. The reason is related to efficient inference on test samples, as explained in the next section. The factor E^i_{zz} could either be non-parametric or it could be parametric, in which case W_{zz} must be used in such a way that it does not need to be adjusted when a test sample is added to the model.

The total energy of the system is given by the sum of energies of all the factors.

$$E(W, \mathbf{Z}, \mathbf{Y}, \mathbf{X}) = \sum_{i=1}^{N} E^i_{xyz}(W_{xyz}, X^i, Y^i, D^i) + E^i_{zz}(W_{zz}, D^i, \mathbf{Z}). \quad (9)$$

Although the above model is not capturing the direct dependencies between the answer variables of the samples (that is the direct dependencies between the prices of neighboring houses), one can easily extend the model to incorporate such a feature. This would involve associating a new (third) factor E^i_{yy} with each sample i and creating the necessary links. Note that this factor will be very similar to the factor E^i_{zz}. In this case the energy of the system will be given by

$$E(W, \mathbf{Z}, \mathbf{Y}, \mathbf{X}) = \sum_{i=1}^{N} E^i_{xyz}(W_{xyz}, X^i, Y^i, D^i) + E^i_{zz}(W_{zz}, D^i, \mathbf{Z}) + E^i_{yy}(W_{yy}, Y^i, \mathbf{Y}). \quad (10)$$

2.1 Efficient Inference in Relational Factor Graphs

We will discuss two types of inference in the factor graph described above. The first type of inference is finding the maximum-likelihood value of Y^0 for a test sample X^0. The second type of inference is finding the maximum-likelihood value of the hidden variables $\mathbf{Z}$ during training.

Let us first discuss the prediction of Y for a test sample. Let $\mathbf{D}$ denote the collection of hidden variables $\{D^i : i = 1 \ldots N\}$. Given the learned parameters W, a perfectly correct way of predicting the value of Y^0 for a test sample X^0 involves creating two new factors E^0_{xyz} and E^0_{zz}, and minimizing the energy with respect to the variables Y^0, $\mathbf{D}$, and $\mathbf{Z}$

$$\min_{y^0, \mathbf{D}, \mathbf{Z}} E(W, \mathbf{Z}, \mathbf{D}, \mathbf{Y}, \mathbf{X}), \quad (11)$$

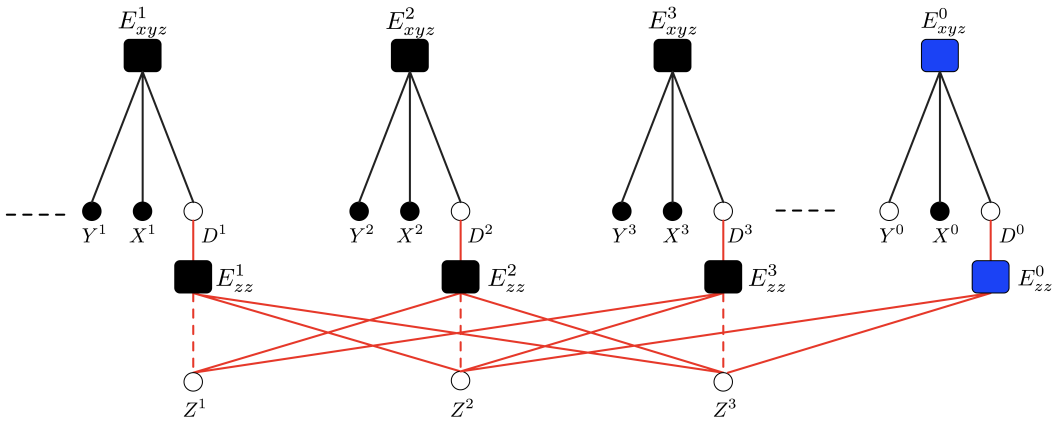

Figure 1: *The factor graph used for the problem of house price prediction, showing the connections between three samples. The factors E^i_{xyz} capture the dependencies between the features of individual samples and their answer variable Y^i, as well as the dependence on local latent variables D^i. The factors E^i_{zz} captures the dependencies between the hidden variables of multiple samples. If the sample i is related to sample j, then there is a link from the variable Z^j to the factor E^i_{zz}.*

where y^o is the output variable of the test sample, whose value we seek. For every test sample, repeating this process of minimization over the values of latent variables of all the training samples, can be extremely expensive. We propose an efficient inference algorithm that does not require such a minimization. This involves a slight modification to the architecture. In addition we also argue that the proposed modification of the architecture makes sense intuitively and is not just an engineering hack.

In figure 1, the factor E^i_{zz} directly depends on the local component of the latent variables Z^i. It is this direct dependence that makes the expensive global optimization with respect to $\mathbf{Z}$ necessary during prediction. This is because once we add the factor and make the necessary connections corresponding to the new test sample X^0, the Z^0 of the test sample, the value of which we do not know, will influence the values Z^i associated with the training samples. These in turn will influence the variable D^0, which influences the answer variable Y^0. Thus adding a new sample results in recomputing the values of hidden variables of the training samples again. To alleviate this problem, we delete the link from Z^0 to the factor E^0_{zz}, shown by dashed lines in figure 1. Hence, when the factor for a test sample is added to the model, it does not have any influence on the values of Z^i associated with the training samples, and hence they remain fixed. Thus the value of Z^0 is no longer required to be estimated. *The sub-graph that is activated for the prediction of Y^0 has no loop now*. Hence the inference can be performed trivially. The inference process comes down to minimizing the following expression with respect to Y^0 and D^0 to compute Y^{o*}

$$\min_{Y^0, D^0} E^0_{xyz}(X^0, Y^0, D^0) + E^0_{zz}(D^0, \mathbf{Z}). \tag{12}$$

In order to be consistent while training we also remove the edges from Z^i to their respective factors E^i_{zz} for every training sample i (shown by red dotted lines).

The above modification to the architecture can be justified with the help of an intuitive explanation. Since we are interested in predicting the prices of the houses in future, clearly the "desirability" Z^0 of a test house at some point in the future cannot influence the past "desirabilities" of the training samples.

2.2 Efficient Training of a Relational Factor Graph

Let $\mathcal{S}$ be the training samples. The task of learning involves finding the parameter W such that the model gives low energies to configurations of (X, Y) pairs from the training set, and higher energies to other (incorrect) values of Y. This is performed by maximizing the likelihood of the data, or equivalently by minimizing the negative log likelihood loss with respect to W, $\mathbf{Z}$ and $\mathbf{D}$.

That is, we need to minimize

$$\mathcal{L}(W, \mathbf{Y}, \mathbf{X}) = E(W, \mathbf{Y}, \mathbf{X}) + \frac{1}{\beta} \log \int_{\mathbf{Y}' \in \mathcal{Y}} e^{-\beta E(W, \mathbf{Y}', \mathbf{X})}, \tag{13}$$

where as discussed before, the energy function $E(W, \mathbf{Y}, \mathbf{X})$ is given by

$$E(W, \mathbf{Y}, \mathbf{X}) = -\frac{1}{\beta} \log \int_{\mathbf{Z}' \in \mathcal{Z}} e^{-\beta E(W, \mathbf{Z}', \mathbf{Y}, \mathbf{X})}. \tag{14}$$

Clearly this task is intractable when $\mathbf{Z}$ and $\mathbf{D}$ are high dimensional and continuous. However there are two important points that one can take advantage of

- When the energy is a quadratic function of $\mathbf{Y}$ with a fixed Hessian, the contrastive term in the negative log likelihood loss (the log partition function, which is the integral in equation 13) is a constant. Hence one can simply ignore the second term altogether and only minimize the first term, which is the average energy term.
- For very large values of β the integral over $\mathbf{Z}$ (in the equation 14 can be approximated by a simple minimization.

The actual loss function that is minimized is therefore give by

$$\mathcal{L}(W, \mathbf{Y}, \mathbf{X}) = \min_{W, \mathbf{Z}' \in \mathcal{Z}} -\frac{1}{\beta} \int_{\mathbf{D}' \in \mathcal{D}} e^{-\beta E(W, \mathbf{Z}', \mathbf{D}', \mathbf{Y}, \mathbf{X})}. \tag{15}$$

This loss is minimized using the following EM-like procedure:

---

repeat
Phase 1
Keeping the parameters W fixed, minimize the loss $\mathcal{L}(W, \mathbf{Y}, \mathbf{X})$ with respect to $\mathbf{Z}$.
Phase 2
Fix the values of $\mathbf{Z}$ and minimize the loss with respect to the parameters W, by updating W in the direction of the negative of the gradient of $\mathcal{L}(W, \mathbf{Y}, \mathbf{X})$.
until $convergence$

Consider the phase involving minimization of the loss with respect to $\mathbf{Z}$. Because of the complex dependency of the energy on $\mathbf{Z}$, we must resort to gradient descent to minimize the loss with respect to $\mathbf{Z}$, after the marginalization of the energy over $\mathbf{D}$ (equation 15). The total energy associated with the sample i is the sum of energies of the two factors E^i_{xyz} and E^i_{zz}. That is

$$E^i(\mathbf{Z}, D^i, Y^i, X^i) = E^i_{xyz}(Y^i, X^i, D^i) + E^i_{zz}(D^i, \mathbf{Z}). \tag{16}$$

To perform gradient descent in $\mathbf{Z}$ we need to compute the sum over i of the followng gradients

$$\nabla_{\mathbf{Z}} E^i(\mathbf{Z}, \mathbf{D}, Y^i, X^i) = \nabla_{\mathbf{Z}} \left(-\frac{1}{\beta} \log \int_{\mathbf{D}' \in \mathcal{D}} e^{-\beta E^i(\mathbf{Z}, \mathbf{D}' Y^i, X^i)} \right) \tag{17}$$

$$= \int_{\mathbf{D}' \in \mathcal{D}} \left(\frac{e^{-\beta E^i(D^i, \mathbf{Z}, Y^i, X^i)}}{\int_{\mathbf{D}' \in \mathcal{D}} e^{-\beta E^i(\mathbf{Z}, \mathbf{D}', Y^i, X^i)}} \right) \cdot \nabla_{\mathbf{Z}} E^i(\mathbf{Z}, \mathbf{D}', Y^i, X^i). \tag{18}$$

The right-most term is like a weighted sum over the values of the set $\mathcal{D}$, of the gradient of the energy (loss) with respect to $\mathbf{Z}$, where the weights are given by

$$\frac{e^{-\beta E^i(D^i, \mathbf{Z}, Y^i, X^i)}}{\int_{\mathbf{D}' \in \mathcal{D}} e^{-\beta E^i(\mathbf{Z}, \mathbf{D}', Y^i, X^i)}}. \tag{19}$$

When β is large, these weights will be equal to zero for all but one value of $\mathbf{D}$, for which it will be equal to one. Hence the outer integral over $\mathbf{D}$ can be approximated by its mode. Thus be have

$$\nabla_{\mathbf{Z}} E^i(\mathbf{Z}, \mathbf{D}, Y^i, X^i) = \nabla_{\mathbf{Z}} E^i(\mathbf{Z}, \mathbf{D}^*, Y^i, X^i), \tag{20}$$

where $\mathbf{D}^*$ is $\operatorname{argmin}_{\mathbf{D}' \in \mathcal{D}} E^i(\mathbf{Z}, \mathbf{D}', Y^i, X^i)$.

We now proceed to show that if E_{xyz} and E_{zz} are both quadratic in D, then the process of their minimization with respect to $\mathbf{D}$ and $\mathbf{Z}$ reduces to simply back propagating the gradient with respect to $\mathbf{Z}$, using the above approximation.

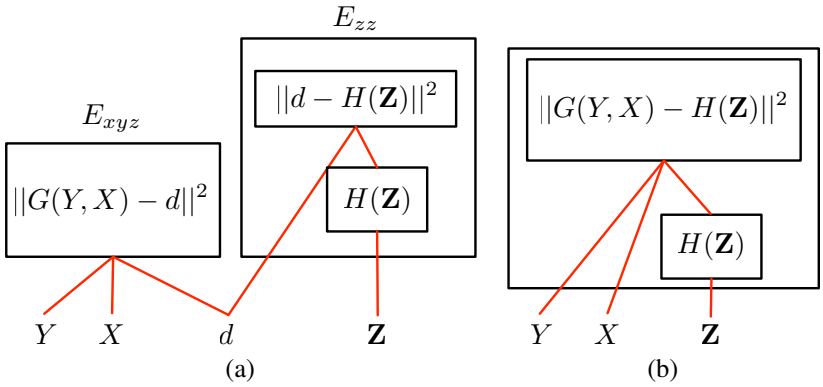

Figure 2: **(a)** *Figure showing the connections between the two factors associated with every sample i. (***(b) Top***), However, when the energy of factors E_{zz} and E_{xyz} is quadratic in D, the factor E_{zz} can be collapsed into the the factor E_{xyz}. In such a situation finding the minimum-energy value of $\mathbf{Z}$ can be performed without explicitly computing the optimal value of D. This is done by simply replacing D by $H(\mathbf{Z})$ in the factor E_{xyz}, and back-propagating the gradients through the function G and H.*

Lemma 2.1 *Let $E_{xyz}(Y, X, D) = (G(Y, X) - D)^2$, and $E_{zz}(D, Z_{\mathcal{N}}) = (D - H(\mathbf{Z}))^2$, where $G(Y, X)$ is some function of Y and X, and $H(\mathbf{Z})$ is some function of $\mathbf{Z}$. Then the gradient of the sum of energies of the two factors $E(\mathbf{Z}) = E_{xyz}(Y, X, D) + E_{zz}(D, \mathbf{Z})$, with respect to $\mathbf{Z}$ at the mode of D is given by $\nabla_{\mathbf{Z}} E(\mathbf{Z}) = (G(Y, X) - H(\mathbf{Z}))\nabla_{\mathbf{Z}} H(\mathbf{Z})$.*

The above lemma essentially means that one can treat the factor E_{zz} like a function and directly feed it into the second factor E_{xyz}. Thus Figure 2(a), now becomes Figure 2(b). This key idea will be used for learning and doing inference efficiently while predicting house prices in the next section.

3 Factors used for Predicting House Prices

We now give the details of the factors that were used in the above factor graph for the problem of house price prediction. The price of a house is modeled as a product of two terms: the "intrinsic" price of the house (a function of house specific variables), and the estimate of the "desirability" of its location. However, rather than predicting the actual price, the model predicts the log of the price. This allows us to additively combine the log of the "intrinsic" price and the log of the "desirability" to get the log price. The input features X^i associated with the i^{th} house can be split into two components $X^i = [X^i_h, X^i_{nb}]$, where X^i_h are the features specific to the house (e.g., number of bedrooms etc) and X^i_{nb} are the features relating to its geographic location (e.g., census tract information, information from the school district etc).

First of the two dependencies that it captures, is between the price of the house and its individual characteristics (including an estimate of its desirability) using the factor E^i_{xyz}. This is done by estimating the "intrinsic" price from the features X^i_h, using a parametric model $G(W_{xyz}, X^i_h)$, and measuring the discrepancy of the the total price - "intrinsic" price + estimated "desirability" (D^i) from Y^i. Thus the factor E^i_{xyz} corresponding to house i takes the form $E^i_{xyz}(Y^i, X^i, d^i) = (Y^i - (G(W_{xyz}, X^i_h) + D^i))^2$.

The second type of dependency, captured by the factor E^i_{zz}, involves the influence of other houses on the price of the i-th house via the hidden variables $\mathbf{Z}$. This is modeled in the following way. Let $\mathcal{N}^i$ be the set of neighboring houses that we think will influence the price of the house i. These are the K houses that have the smallest euclidean distance in the space of the variables X^i_{nb} and GPS coordinates. Each sample i is associated with a latent variable Z^i. The value of this variable can be interpreted as the "desirability" of the location of that house. However in line with the discussion of the previous section, the factor E^i_{zz} does not take as input the variable Z^i. Rather the desirability of the i^{th} house is estimated from the set $Z_{\mathcal{N}^i} = \{Z^j : j \in \mathcal{N}^i\}$ using a non-parametric function using a non-parametric function $H(X_{nb}, Z_{\mathcal{N}^i})$. This estimation could be done using a number of ways. However the bottom line is that one must ensure that the smoothness property is maintained

since desirability in general has a smoothness property associated with it: one sees a gradual change in desirability when moving from one consecutive neighborhood to other.

Kernel Based Interpolation

One way of estimating this is through simple kernel based interpolation.

$$H(X^i_{nb}, Z_{\mathcal{N}^i}) = \sum_{j \in \mathcal{N}^i} Ker(X_{nb}, X^j_{nb}) Z^j, \tag{21}$$

where the kernel function is defined as

$$Ker(X, X^j) = \frac{e^{-q||X - X^j||^2}}{\sum_{k \in \mathcal{N}} e^{-q||X - X^k||^2}}. \tag{22}$$

q is a constant that defines the smoothness of the kernel. The larger its value the smoother is the kernel.

Weighted Local Linear Regression

Another way of estimating this involves fitting a weighted local linear model on the set of samples $\mathcal{N}^i$, with the weights given by the above kernel, and using the learned parameters to get the value of $H(X^i_{nb}, Z_{\mathcal{N}^i})$. Let α be the parameter vector and β be the bias of the local linear model. Then fitting a weighted local linear model on the set of neighbouring training samples $\mathcal{N}^i$, amounts to finding the parameters α^* and the bias β^*, such that

$$(\beta^*, \alpha^*) = \arg\min_{\beta, \alpha} \sum_{j \in \mathcal{N}^i} (Z^j - (\beta + \alpha X^j))^2 Ker(X^i, X^j). \tag{23}$$

The function $Ker(X, X^j)$ could be any appropriate smooth kernel, e.g. the one given in equation 22. The solution to the system (or the output of the function $H(X^i_{nb}, Z_{\mathcal{N}^i})$ is given by

$$H(X^i_{nb}, Z_{\mathcal{N}^i}) = \beta^* + \alpha^* X. \tag{24}$$

Remark 1 *Even in the case of weighted local linear regression the output of the function $H(X^i_{nb}, Z_{\mathcal{N}^i})$ for a sample X^i can be expressed as a linear combination of the desirabilities Z^j of the training samples $X^j \in \mathcal{N}^i$, such that the linear coefficients do not depend on the Z^j's. That is*

$$H(X^i_{nb}, Z_{\mathcal{N}^i}) = \sum_{j \in \mathcal{N}^i} a^j Z^j, \tag{25}$$

where coefficients a^j's are independent of the values of Z^j's.

The factor E^i_{zz} takes the form $E^i_{zz}(D^i, H(X^i_{nb}, Z_{\mathcal{N}^i})) = (D^i - H(X^i_{nb}, Z_{\mathcal{N}^i}))^2$. Thus the energy corresponding to sample i, which is given by the sum of energies of the two factors, is given by

$$E^i_{xyz}(Y^i, X^i, D^i) + E^i_{zz}(D^i, H(X^i_{nb}, Z_{\mathcal{N}^i}) = (Y^i - (G(W_{xyz}, X^i_h) + D^i))^2 + (D^i - H(X^i_{nb}, Z_{\mathcal{N}^i}))^2. \tag{26}$$

Clearly one can see that both the energy terms on the right hand side are quadratic functions of the variable D^i. Hence, following the discussions of the previous section and the lemma 2.1, the second factor can be treated as a function and can be included in the first factor directly. Thus in effect we have a single factor E^i associated with each sample i.

The exact architecture of the factor E^i is given in Figure 3. Each factor consists of two trainable components. The first component is the parametric function $G(W_{xyz}, X^i_h)$, that estimates the "intrinsic price" of the sample. Other than differentiability with respect to W_{xyz}, no restriction is imposed on the form/architecture of this function. In the experiments, this function was a fully connected neural network with two hidden layers. The second component $H(X^i_{nb}, Z_{\mathcal{N}^i})$ takes the non-parametric form discussed above. This function can be viewed as modeling the smooth "desirability" manifold over the geographic region spanned by the samples. As mentioned above, the output of this function can be interpreted as an estimate of the "desirability" of the location of the i^{th} sample. The higher the value, the more desirable is the location. The outputs of the two components are added to get the final price of the house in log domain. Finally, the function $\mathcal{F}$ measures the descrepancy between Y^i and the combined outputs $G(W_{xyz}, X^i_h) + H(X^i_{nb}, Z_{\mathcal{N}^i})$.

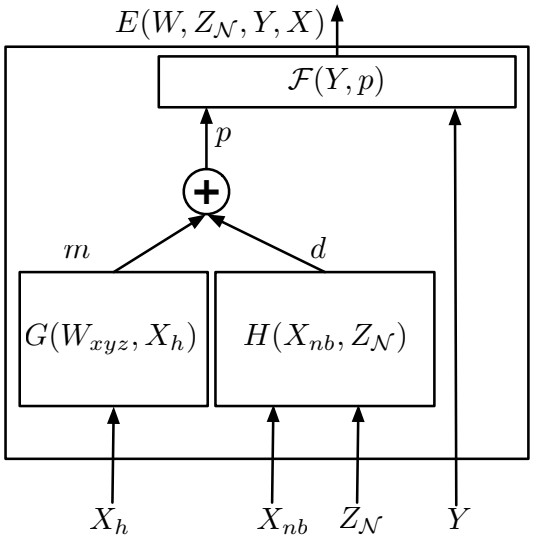

Figure 3: *The architecture of the factor associated with each sample. It consists of two trainable components: the parametric function G_W and the non-parametric function H.*

3.1 Learning

Training the machine involves maximizing the likelihood of the training data. This is achieved by marginalizing the negative log likelihood function over the hidden variables $\mathbf{Z}$, and then minimizing it with respect to the parameters W associated with the system. However, in line with the discussion of section **??**, the process of marginalization over $\mathbf{Z}$ can be approximated with a minimization operation. Further more, when the energy is a quadratic function of $\mathbf{Y}$, the contrastive integral term over $\mathbf{Y}$ is a constant. Hence the entire process reduces to minimizing the energy loss simultaneously with respect to the parameters W and the hidden variables $\mathbf{Z}$.

$$\mathcal{L}(W, \mathbf{Z}) = \sum_{i=1}^{n} \frac{1}{2}(Y^i - (G(W_{xyz}, X^i) + H(Z_{\mathcal{N}^i}, X^i)))^2 + R(\mathbf{Z}), \tag{27}$$

where $R(\mathbf{Z})$ is a regularizer on $\mathbf{Z}$, which was an L_2 regularizer in the experiments. This regularization term plays a crucial role to ensure the smoothness of the hidden manifold. Without it, the system would over-fit the training data and learn a highly-varying surface, leading to poor generalization.

Minimization of this loss simultaneously with respect to W and $\mathbf{Z}$ can be achieved by applying a type of deterministic generalized EM algorithm. It consists of iterating through the following two phases until convergence.

Phase 1

In Phase 1, the parameters W are kept fixed and the loss is minimized respect to the hidden variables $\mathbf{Z}$. It turns out that the above loss is quadratic in $\mathbf{Z}$ and its minimization reduced to solving a large scale sparse quadratic system. In particular, associate a vector U^i of size N (equal to the number of training samples), with each training sample X^i. This vector is very sparse and has only K non-zero elements, at the indices given by the elements of the neighbourhood set $\mathcal{N}^i$. The value of the j^{th} non-zero element is equal to the linear coefficient that is multiplied with the Z^j, while estimating D^i. For example, when the kernel based interpolation is used, the coefficients are given by equation 22, and when local linear regression model is used, the coefficients are a^j's in equation 25. Thus the function $H(Z_{\mathcal{N}^i}, X^i)$ can now be written as

$$H(Z_{\mathcal{N}^i}, X^i) = \mathbf{Z} \cdot U^i. \tag{28}$$

The loss now becomes the following sparse quadratic program

$$\mathcal{L}(\mathbf{Z}) = \frac{r}{2}||\mathbf{Z}||^2 + \frac{1}{2}\sum_{i=1}^{n}(Y^i - (G(W_{xyz}, X^i) + \mathbf{Z} \cdot U^i))^2. \tag{29}$$

Another possible modification to the loss is to includes an explicit self-consistency term. The idea is to have an explicit constraint that will drive the estimate $D^i = \mathbf{Z} \cdot U^i$ of the desirability of training sample X^i assigned desirability Z^i. Note that the estimate D^i does not involve the term Z^i. Hence the loss function now becomes

$$\mathcal{L}(\mathbf{Z}) =$$
$$\frac{r_1}{2}||\mathbf{Z}||^2 \; + \; \frac{1}{2}\sum_{i=1}^{n}(Y^i \; - \; (G(W_{xyz}, X^i) \; + \; \mathbf{Z} \; \cdot \; U^i))^2 \; + \; \frac{r_2}{2}\sum_{i=1}^{n}(Z^i \; - \; \mathbf{Z} \; \cdot \; U^i)^2. \quad (30)$$

Here r_1 and r_2 are some constants. This loss function is still a sparse quadratic program.

The above systems can be solved using any sparse system solvers. However, instead of using a direct method we resorted to iterative methods. The motivation was that at each iteration of the algorithm, we were only interested in the approximate solution of the system. We used the conjugate gradient method with early stopping (also called partial least squares). The conjugate gradient was started with a pre-determined tolerance which was gradually lowered until convergence.

Phase 2

This phase involves keeping the hidden variable $\mathbf{Z}$ fixed and updating the parameters W of the system, which boils down to updating W_{xyz} of the function $G(W_{xyz}, X)$ by running a standard stochastic gradient decent algorithm for all the samples (X^i, Y^i) in the training set. For a sample X^i the forward propagation was composed of the following steps. Run X^i through the function G to produce the log of the intrinsic price $M^i = G(W_{xyz}, X^i)$. Estimate the desirability D^i of the location of sample i by interpolating from its neighbours in the training set, using the function $D^i = H(Z_{\mathcal{N}^i}, X^i)$. Add the desirability to the intrinsic price to get the prediction $Pr^i = M^i + D^i$. Compare the predicted value Pr^i with the actual value Y^i to get the energy $E(W, \mathbf{Z}) = \frac{1}{2}(Y^i - Pr^i)^2$. Finally, the gradient of the energy with respect to W_{xyz} is computed using the back propagation step and the parameters W_{xyz} are updated. Here again, we do not train the system to completion, but rather stop the training after a few epochs. Note that even though the factor graph is relational we can still use stochastic gradient to learn the weights W_{xyz} because they are all shared among the factors of all the samples.

3.2 Inference

The process of inference on a new sample X^o is trivial. It involves computing its neighboring training samples, and using the learnt values of their hidden variables $Z_{\mathcal{N}^o}$ to get an estimate of its "desirability"; passing the house specific features X_h^o through the learnt function $G(W_{xyz}, X_h)$ to get its "intrinsic" price; and combining the two to get its predicted price in log domain.

4 Experiments and Results

The proposed model was trained and tested on a very diverse dataset that included 750,000 sale transactions of houses in Los Angeles county in the year 2004. Each house was described by a total of 125 attribute variables. For the experiments, a subset of 18 attributes were used to describe house specific features X_h. This included living area, year build, number of bedrooms, number of bathrooms, pool, prior sale price, parking spaces, parking type, lot acerage land value, improvement value, percentage improvement, new construction, foundation, roof material type, heat type, site influence, and gps coordinates. In addition, for each house, the census tract number and the school district in which it lies was also given. This information was used to construct the neighborhood specific variables X_{nb}. It included attributes like average house hold income of that area, average time of commute for people not working from home, percentage of owner occupied homes, academic performance index of the corresponding school. The gps coordinates of the house were also used as part of X_{nb}.

All the numerical variables containing any form of price information, area information, and income information were mapped into log space. The non-numeric discrete variables like "pool", "parking type" etc, were coded using a 1-of-N code. In addition, all the variables were normalized so as to

have a 0 mean and a standard deviation between -1 and 1. After the above preprocessing there were a total of 65 fields in X_h variables and 14 fields in X_{nb} variables. All those houses that had a missing value for one or more of its attributes were discarded. In addition, we used only those houses that belonged to the class of "single family residences". In the end we were left with a total of 42025 labeled samples. In order to do actual prediction, the samples were sorted according to their sale dates. After sorting, the first 90% of them (37822) were used as a training set and the remaining 10% (4203) were used as a test set. The dataset is particularly challenging because it is a real world dataset encompassing very large area with diverse individual and neighborhood characteristics: it spans 1754 census tracts and 28 school districts.

4.1 Non-Relational Models

The performance of the proposed model was compared with a number of standard non-relational techniques that have been in use for this problem. In particular, we compared the model with k-nearest neighbors, regularized linear regression, weighted local linear regression and a fully connected neural network.

K - Nearest Neighbour

In this technique, the process of predicting the price of the sample X^o involves finding the K nearest training samples (using some similarity measure) and computing the average price of these neighbours. Three different similarity measures were tried. First was a euclidean distance in the entire input space $[X_h, X_{nb}]$, the second was the euclidean distance in the neighborhood space X_{nb} plus the GPS coordinates, and the third was the euclidean distance only in the GPS space. Best performance was observed when the first measure was used. Experiments were done with different values of K and results for the best value are reported.

Regularized Linear Regression

In the process of regularized linear regression we try to fit a single linear model on the entire data set without considering the inherent local structure that is associated with it. This is done by minimizing the following objective function

$$R(W) = \frac{r}{2}||W||^2 + \frac{1}{2n}\sum_{i=1}^{N}(Y^i - W^T X^i)^2. \tag{31}$$

In this equation W are the parameters to be learned and r is the regularization coefficient.

Weighted Local Linear Regression

Apart from the nearest neighbour method, the above methods ignore the local structure that is inherent to the problem of house price prediction. The motivation behind using local regression models is to exploit such a local structure and improve upon prediction. In weighted local linear regression models, in order to make a prediction for a sample X^o, a separate weighted linear regression model is fitted using only those training samples that are neighbours to the sample X^o. The neighbours were computed using the euclidean distance in the space of the variables X_{nb} and GPS coordinates. The weights are obtained from an appropriately chosen kernel function. For a sample X^o, if $\mathcal{N}^o$ gives the indices of the neighbouring training samples, then the loss function that is minimized is

$$\min_{\beta(X^o)} \sum_{i \in \mathcal{N}^o} K_\lambda(X^o, X^i)[Y^i - \beta(X^o) f(X^i)]^2. \tag{32}$$

In the above loss $\beta(X^o)$ are the regression parameters that needs to be learned, $f(X^i)$ is some polynomial function of X^i, and $K_\lambda(X^o, X^i)$ is an appropriately chosen kernel width parameter λ. In the experiments it was the exponential kernel given in equation 22. Once minimized, the prediction Pr^o of the sample X^o is given by

$$Pr^o = \beta(X^o) \cdot f(X^o). \tag{33}$$

Table 1: *Prediction accuracies of various algorithms on the test set.*

MODEL CLASS	MODEL	$< 5\%$	$< 10\%$	$< 15\%$
NON-PARAMETRIC	NEAREST NEIGHBOR	25.41	47.44	64.72
NON-PARAMETRIC	LOCALLY WEIGHTED REGRESSION	32.98	58.46	75.21
PARAMETRIC	LINEAR REGRESSION	26.58	48.11	65.12
PARAMETRIC	FULLY CONNECTED NEURAL NETWORK	33.79	60.55	76.47
HYBRID	**RELATIONAL FACTOR GRAPH**	**39.47**	**65.76**	**81.04**

Fully Connected Neural Network

The last non-relational technique tried was a fully connected neural network. The motivation behind this was that using such an architecture one might be able to capture some non linearities that are hidden in the regression function. A number of architectures were tried, and the one that achieved the best performance was a 2-hidden layer network with 250 units in the first layer, 80 units in the second, and 1 unit in the output layer.

4.2 Relational Factor Graph

The parametric model $G(W, X_h)$ was a fully connected 2-hidden layer neural network with 250 units in the first hidden layer, 80 units in the second hidden layer and 1 unit in the output layer. The input to this network were the set of variables X_h. For the non-parametric function $H(X_{nb}, Z_{\mathcal{N}})$, both the modeling options - kernel smoothing and weighted local linear regression - were tried. The size of the neighborhood set $\mathcal{N}$ used was 13 and 70 for the kernel smoothing and weighted local linear regression respectively. Different values of the regularization coefficients and other parameters were tried. The results are reported for the best combination of the values in table 1.

5 Results and Discussion

The performance of the systems was measured in terms of the Absolute Relative Forecasting error (fe) [5]. Let A_i be the actual price of the house i, and let Pr_i be its predicted price. Then the Absolute Relative Forecasting error (fe_i) is defined as

$$fe_i = \frac{|A_i - Pr_i|}{A_i}. \tag{34}$$

Three performance quantities on the test set are reported; percentage of houses with a forecasting error of less than 5%, houses with a forecasting error of less than 10%, and percentage of houses with a forecasting error of less than 15%. The greater these numbers the better the system. Simply using the root mean square error in this setting is not very informative, because it is overly influenced by outliers.

5.1 The Desirability Maps

In this section we give some discussion that provides insights into the working of the algorithm and argue that it is representative of the real world scenario. This claim is supported by providing a number of energy maps of the test samples which we shall discuss.

Figure 4, shows the colored coded desirability map learned by the model. The map shows the desirability estimates of the location of all the houses in the test set. For each test sample, this estimate is computed from the learned desirabilities Z^i of the training samples and the function $H(Z_{\mathcal{N}}, X)$, as described earlier. The points are colored according to the value of their desirability estimates. Blue color implies less desirable and red color implies more desirable. One can conclude that the value of the desirabilities estimated by the algorithm does encode some meaningful information which is a reflection of the real world situation. This is evident from the fact that the areas around the coastline are generally labeled more desirable. Likewise, the areas of Pasadena and near Beverly Hills are also classified as highly desirable. Areas around the downtown area of the county, particularly in the south eastern and immediate east direction, are marked with low desirability.

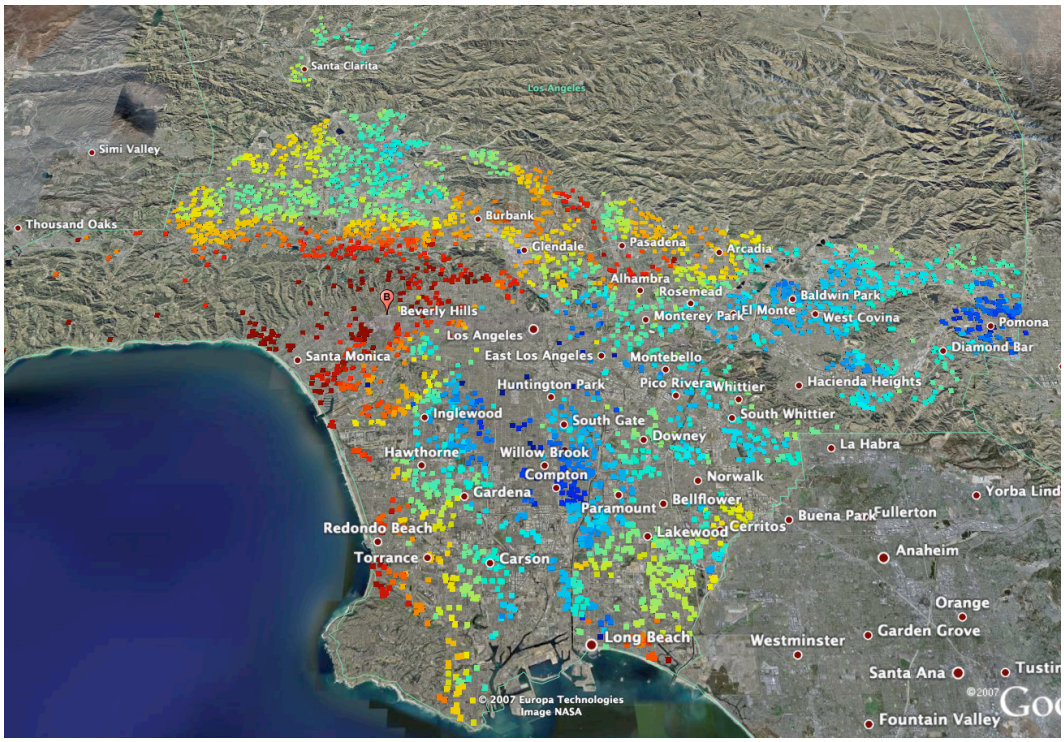

Figure 4: *The learnt hidden "desirability" manifold over the Los Angeles county area. Each point in the map correspond to a unique test sample and is color coded according to the "desirability" of its location. The red color correspond to high desirability and the blue color correspond to low desirability.*

6 Conclusion

We proposed a general non-probabilistic framework for doing prediction in relational data. It extends the framework of energy-based factor graphs onto relational data. Relational factor graphs associate a single factor graph to the entire data and the relationships between samples are captured by the factors that connect variables from multiple samples. The main idea of the paper is an efficient learning and inference algorithm for a new test sample, that does not require minimization over the latent variables each time. There is no restriction on the architecture of the factors, which could be any non-linear function capable of capturing the complicated interactions between the related variables. Minimization of the loss with respect to the parameters and the hidden variables was achieved by using a deterministic form of generalized EM algorithm. Results on the problem of predicting house prices were reported, which showed a significant performance gain by the proposed model.

References

[1] G. E. P. Box and D. R. Cox. An analysis of transformations. *Journal of Royal Statistical Society*, B26:211 – 243, 1964.

[2] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. *In. Proc. of ACM SIGMOD98*, pages 307 – 318, 1998.

[3] D. Cohn and H. Chang. Probabilistically identifying authoritative documents. *In Proc. SIGIR 2000*, 2000.

[4] D. Cohn and T. Hofmann. The missing link: A probabilistic model of document content and hypertext connectivity. *In Proc. NIPS 2001*, 2001.

[5] A. Q. Do and G. Grudnitski. A neural network approach to residential property appraisal. *Real Estate Appraiser*, 58(3):38 – 45, 1992.

[6] L. Egghe and R. Rousseau. *Introduction to Informetrics*. Elsevier, 1990.

[7] N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. *In Proc. IJCAI99*, pages 1300 – 1309, 1999.

[8] A. C. Goodman. Hedonic prices, price indices and housing markets. *Journal of Urban Economics*, 5:471 – 484, 1978.

[9] T. Hofmann and J. Puzicha. Latent class models for collaborative filtering. *In Proc. IJCAI99*, 1999.

[10] T. Kauko. *Modeling Locational Determinants of House Prices: Neural Network and Value Tree Approaches*. PhD thesis, Utrecht University, 2002.

[11] J. M. Klienberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 46(5):604 – 632, 1999.

[12] D. Koller and A. Pfeffer. Probabilistic frame-based systems. *In Proc. AAAI98*, pages 580 – 587, 1998.

[13] Y. LeCun, S. Chopra, R Hadsell, F. J. Huang, and M. Ranzato. A tutorial on energy-based learning. In Bakir et al, editor, *Predicting Structured Outputs*. MIT Press, 2006.

[14] R. Meese and N. Wallace. Nonparametric estimation of dynamic hedonic price models and the construction of residential housing price indices. *AREUEA Journal*, 19(3):308 – 331, 1991.

[15] J. Neville and D. Jensen. Iterative classification in relational data. *In Proc. AAAI00 Workshop on Learning Statistical Models From Relational Data*, pages 13 – 20, 2000.

[16] N. Nguyen and A. Cripps. Predicting housing value: A comparison of multiple regression analysis and artificial neural networks. *The Journal of Real Estate Research*, 22(3):313 – 336, 2001.

[17] S. Slattery and T. Mitchell. Discovering test set regularities in relational domain. *In. Proc. ICML00*, pages 895 – 902, 2000.

[18] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. *Eighteenth Conference on Uncertainty on Machine Intelligence (UAI02)*, 2002.

[19] B. Taskar, E. Segal, and D. Koller. Probabilistic classification and clustering in relational data. *In Proc. IJCAI01*, pages 870 – 876, 2001.