Factor Graphs for Relational Regression

by

Sumit Prakash Chopra

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

January 2009

Yann LeCun

DEDICATION

To my father *Mr. Lakshmi Naraian Chopra*, and mother *Mrs. Shashi Kanta Chopra*. This thesis would not have been possible without them.

My mother is one person who has sacrificed the most during the past 5 years, being alone in India. She has always selflessly supported me and encouraged me in whatever I intended to endeavor. She always managed to provide me with the best possible resources, even if at times it involved budgeting herself.

My father, my best friend, my mentor, and my inspiration. He was always there to guide me and encourage me at every step of this journey. Whatever I am today is only because of him. He would have been the happiest person on this earth to see the first page of this dissertation. I miss you dad.

ACKNOWLEDGMENTS

It is hard to express in words how thankful I am to my advisor Prof. Yann LeCun, whose guidance and support has made this journey truly magnificent. He has not only been my advisor but also a friend who has taught me so many things beyond research. I will never forget the discussions we had while at Disney Land in Los Angeles, and over countless lunch/dinner meetings. I would also like to especially thank Prof. Foster Provost whose diligent comments and suggestions have helped in improving my dissertation in particular and outlook for research in general. He too has been a genuine source of inspiration and guidance relating to research and beyond.

There are so many people whose support and friendship I have enjoyed during the course of my Ph.D. In particular I would like to sincerely thank Raia Hadsell, and Trivikaraman Thampy. Raia, who was my office mate for 5 years, and co-author of a number of papers has also been a great friend. I have learnt a lot from her during our numerous splendid discussions. "Thampy is Thampy". A true friend who has taught me so many things in the short duration since I've known him. He has always been beside me during the highs and the lows for which I'm very grateful. I would like to acknowledge my friends Shuchi Pandya, Kranthi Gade, Prashant Puniya, and Mohit Gupta for their support, and encouragement, and making this journey immensely pleasurable. I would also like to thank members of CBLL and MRL for their support and friendship.

One person without whom this thesis would not have been a possibility is Dr. Neelima Gupta. She is the person who introduced me to research and has always gone out of the way to support me, guide me, and inspire me to achieve my goals. I cannot thank her enough for all the sacrifices she made, such as ignoring her family, to help me when I needed it.

Finally I would like to thank my extraordinary parents for their relentless support and encouragement in all aspects of life, and making me a person who I am today.

ABSTRACT

Traditional methods for supervised learning treat the input data as a set of independent and identically distributed points in a high-dimensional space. These methods completely ignore the rich underlying relational structure that might be inherent in many important problems. For instance, the data samples may be related to each other in ways such that the unknown variables associated with any sample not only depends on its individual attributes, but also depends on the variables associated with related samples. One regression problem of this nature, whose importance is emphasized by the present economic crises, is understanding real estate prices. The price of a house clearly depends on its individual attributes, such as, the number of bedrooms. However, the price also depends on the neighborhood in which the house lies and on the time period in which it was sold. This effect of neighborhood and time on the price is not directly measurable. It is merely reflected in the prices of other houses in the vicinity that were sold around the same time period. Uncovering and using these spatio-temporal dependencies can certainly help better understand house prices, while at the same time improving prediction accuracy.

The models used to achieve this task fall in the class of *Statistical Relational Learning*. The underlying probabilistic graphical model takes as input a single instance of the entire collection of samples along with their relationship structure. The dependencies among samples is learnt with the help of parameter sharing and collective inference. The drawback of most such models proposed so far is that they cater only to classification problems. To this end, we propose a *relational factor graph* framework for doing regression in relational data. A single factor graph is used to capture, one, dependencies among individual variables of data points, and two, dependencies among variables associated with multiple data points. The proposed models are capable of capturing hidden inter-sample dependencies via latent variables. They also allow for log-likelihood functions that are non-linear in parameter space thereby allowing for considerably more complex architectures. Efficient inference and learning algorithms are proposed.

The models are applied to predicting the prices of real estate properties. A by-product of it is a house price index. The relational aspect of the model accounts for the hidden spatio-temporal influences on the price of every house. The experiments show that one can achieve considerably superior performance by identifying and using the underlying spatio-temporal structure associated with the problem. To the best of our knowledge this is the first work in the direction of relational regression, especially in the frame-based class of statistical relational learning models. Furthermore, this is also the first work in constructing house price indices by simultaneously accounting for the spatio-temporal effects on house prices using large-scale industry standard data set.

TABLE OF CONTENTS

List of Figures

LIST OF TABLES

INTRODUCTION

A majority of supervised learning algorithms assume that the input data is represented by a set of points lying in a high-dimensional space. In addition they assume that these data points are not related to each other in any way. In other words, the assumption that the data points are Independently and Identically Distributed (I.I.D) is inherent in these algorithms. The fundamental principle behind the working of these algorithms is to construct a high-level representation of the attributes of data points (called *features*) and use these to solve the problem using standard tools like neural networks, generalized linear models, or decision trees. These *features* could either be hand-crafted based on the designers prior knowledge of the problem, or they could be learnt. However, the bottom line is that these algorithms focus primarily on the individual attribute-value representation. While such an approach is extremely useful in developing general algorithms and analysis, and solving isolated tasks, such as, detecting a car in an image, this approach completely ignores the rich contextual information and complex underlying structure that could be inherent in the problem. Exploiting this contextual information and structure is essential to moving beyond answering simple "yes/no" questions, pertaining to standard classification or detection, and solving more general and complex problems. For instance, in addition to knowing whether there is a car in an image, a more interesting task would be to know that the car is a ford sedan parked on a tree lined street in a New Jersey suburb: *Scene Analysis*.

Indeed, the importance of contextual information has been well acknowledged in the fields of human and machine vision. When a human observes a scene, or studies a photograph, he/she usually has at its disposal a wealth of information that is not captured by the image alone. Likewise in the domain of machine vision, while detecting a car in an image it is beneficial to use the knowledge that the car is usually parked on the road and sky is always above the ground. Capturing and exploiting these relationships among different objects (the context) not only helps in detecting whether there's a car in the image, it also provides some certainty about other objects present in the image, and hence help solve the problem of scene analysis.

Contextual information is equally essential for reasoning in other domains as well. For example, the box office revenue of a movie, while relying on its unique individual features, will also depend on external factors such as what other movies are currently running in theaters, and how did other movies by the same actor/director combination fared in the past. Likewise, consider the problem of labeling a web-page. Web pages that are linked to each other are more likely to be on the same topic. Thus if we know the label of a particular web-page and its link structure we can say a lot about the labels of the web-pages that link to it and the pages that it links to. In other words, in these problems the data samples interact with one another in highly complex ways to influence the values of unknown variables associated with them. In these scenarios, the notion of context is abstracted through these complex interdependencies among different data samples exhibiting a rich logical structure. Identifying and exploiting this logical structure is essential to solving complex interesting tasks.

More generally, in many problems the values of unknown variables associated with each data point not only depends on the features specific to that data point, but also on the features and variables associated with data points that are related to it. In such problems the standard I.I.D assumption over the samples does not hold. Furthermore, these dependencies could either be given as part of the training data (for example the link structure of the web-pages and its type), or it could be hidden in which case it should be inferred from the data. We say that the data possess an inherent *relational structure* in such problems. Problems of this nature, and the corresponding algorithms that deal with it fall in the domain of *Statistical Relational Learning*.

An important problem that belongs to this class of problems is predicting the price of a real estate properties. In addition to the individual attributes of the house, the spatial context (influence of neighboring houses) and the temporal context (inflation) have a significant effect on the price of the house. This problem and the relational models used to solve it are discussed in far greater detail in the rest of this dissertation.

0.1 Statistical Relational Learning

Unlike in conventional problems with I.I.D assumption, where the uncertainty arises only at the attribute level, in relational problems the uncertainty arises at many levels. Other than the uncertainty in the attributes of the data points, the uncertainty could be in the type of the data points, the number of data points, and the relationships among data points. *Statistical Relational Learning* (SRL) is a research area that tries to learn and account for these multiple levels of uncertainties found in domains with complex relational and rich probabilistic structure. Majority of the SRL systems that have been designed so far can be distinguished along two major dimensions: the first are the rule-based systems that are based on inductive logic, and second are the frame-based or object-oriented systems. Though more recently people have also started to look at systems which are a combination of frame-based and rule-based. In this text, we shall only focus on the frame-based systems.

The underlying probabilistic models in most frame-based SRL systems is based on graphical models, which involve both directed models (e.g., Bayesian networks), which can represent causal relationships and are capable of representing complex generative models, and undirected models (e.g., Markov networks), which can represent complex non-causal relationships. The difference between the so called “flat” models (used in I.I.D setting) and relational models is that a “flat” model takes as input a sequence of I.I.D samples, whereas in a relational model the input to the system is usually just a single, richly connected instance of the entire collection of data. Dependencies among individual attributes of samples, and among different samples is captured by making the underlying graphical models hierarchical. In order to perform any meaningful learning on these models it is imperative to have some form of parameter sharing: the ability to constrain potentially distinct parameters to the same value. When the same set of parameters are used at multiple places of the model, one can hope to extract meaningful statistics from the data. It not only provides us the ability to model rich classes of distributions compactly, but it also enables us to robustly estimate the parameters feasibly. The hierarchical property of the underlying graphical models allows for such parameter sharing. In fact the power of relational models lie in their ability in parameter sharing. Another important and powerful feature of relational models is their ability to do *collective inference*. This involves making simultaneous judgments about the same variables for a set of related data instances.

There are a number of important issues that arise in the SRL framework. One of the most common issue is feature selection and aggregation. Richness of the complex structure along with the need of compact parameterization gives rise to the need of constructing relational features and aggregates which capture the local neighborhood of a random variable. Another common issue is model selection, which involves doing some form of heuristic search over the model space in order to find the best one. Structural uncertainty, which involves uncertainty over relationships, is another issue that the researchers have tried to address recently. Earlier SRL approaches assumed a single relational skeleton, which defined the set of random variables and the probability distribution over them. Now, however researchers have tried to address this issue at different levels, such as uncertainty in the number of related objects, and uncertainty in the identity of neighbors.

0.1.1 Previous Work in Statistical Relational Learning

One of the first attempts at relational learning was by Winston, in (Winston, 1975) where he introduced the arch learning system. It was an online system which was trained using a sequence of positive and negative examples. The system maintained a “current” hypothesis which was used to make a prediction of new example. If the prediction was correct no change was made to the hypothesis. However if the prediction was incorrect then the hypothesis was generalized or specialized based on the nature of the mistake.

Later on people worked along the lines 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. The intuition is that if we know something about one data point, then we can use this knowledge to reach conclusion about other data points that are related to it. For instance, in the web-page classification problem, if we know the label of one web-page then we should be able to propagate this information to the web-pages that it links to and to the web-pages that link to it. These, in turn, will propagate information to other web-pages. After sufficient number of iterations the hope is that the system converges to the correct answer. A number of researchers have proposed algorithms along these lines (Egghe and Rousseau, 1990; Chakrabarti et al., 1998; Kleinberg, 1999; Slattery and Mitchell, 2000; Neville and Jensen, 2000; Cohn and Chang, 2000; Hofmann and Puzicha, 1999; Cohn and Hofmann, 2001).

More recently people have worked along the lines of probabilistic graphical models, both directed and undirected. The first attempt in this direction was the Probabilistic Relational Model (PRM). PRMs were introduced in (Koller and Pfeffer, 1998; Friedman et al., 1999) as an extension of Bayesian networks to relational data. A PRM is a single Bayesian network (a directed graphical model) over the entire data that specifies the joint probability distribution over all the attributes and samples. Learning algorithms for PRMs have originally been generative (Taskar et al., 2001). Furthermore, PRMs being based on directed graphical models, fail to capture cyclic dependencies between data points. Discriminative extensions to PRMs, called Relational Markov Networks (RMN) were later proposed by Taskar et. al., in (Taskar et al., 2002). RMNs are essentially Conditional Random Fields (CRFs) extended to handle relational data. Thus 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. Another line of work is due to Neville and Jensen (Neville and Jensen, 2007) in which they introduce Relational Dependency Networks (RDNs). RDNs extend propositional dependency networks to relational domains. Like dependency networks they have some advantages over directed and undirected graphical networks. Heckerman et al., in (Heckerman et al., 2004) describe a graphical language for probabilistic entity-relationship models (PERs) which also fall in the class of relational models.

Finally a completely different line of work in relational learning involves designing rule-based or logic-based formalism. This can further be categorized into two sub-categories. The first includes models that are purely logic-based, such as Poole's work on probabilistic Horn abduction (Poole, 1993) and independent choice logic (Poole, 1997), Ngo and Haddawy's work on probabilistic knowledge bases (Ngo and Haddaway, 1997), and Sato's work on the PRISM system (Sato, 1995), and Ng and Subrahmanian's work on probabilistic logic programming (Ng and Subrahmanian, 1992). The other class of work involves combining probabilistic graphical models with the logic programs, such as, Bayesian Logic Program (BLPs) by (Kersting et al., 2000), Stochastic Logic Programs (SLPs) (Muggleton, 2000), and Markov Logic Networks (Richardson and Domingos, 2006).

0.2 Relational Regression

Almost all of the above mentioned models, especially in the domain of frame-based or object-oriented systems, are designed to deal with variables that take discrete values. Hence these models are useful only for classification tasks. There are no straight forward generalizations of these models that will enable them to handle continuous variables. However, there are many important real world problem where one is not just restricted to performing classification but is required to do regression. Problems of this nature fall in the class of *Relational Regression*, and their solution requires relational models which can handle continuous variables. In the mid 70's Besag in (Besag, 1974; Besag, 1975; Besag, 1986), proposed models for problems in which discrete and continuous variables exhibited spatial stochastic interactions. However these models either assumed a fixed lattice structure over the interactions or they assumed a Markovian assumption over the interactions.

We propose novel factor graph based relational models which can be used for solving relational regression problems. The proposed model can handle any arbitrary interactions among data points and does not assume any structure over the interactions. In addition to handling continuous variables, the proposed models can account for uncertainties in the types of relationships among data points by being able to handle dependencies among them which are hidden and not given directly as part of the data. Furthermore, the hidden variables that capture these dependencies could themselves be continuous. These models allow for the log-likelihood functions to be non-linear in the parameter space, which leads to non-convex loss functions but are considerably more flexible. These models also eliminates the intractable partition function problem through appropriate design of the relational and non-relational factors of the factor graph. We apply the proposed models to the problem of real estate price prediction which is an important practical problem that falls in the domain of relational regression.

0.3 Real Estate Price Prediction

0.3.1 Background

Even though the problem of understanding real estate prices is of fundamental importance, it is fair to say that till date very little has been understood so far. The evidence of this lack of understanding is illustrated by the recent events in the market, such as the sub-prime crisis, and the use by Congress of tax-payer guarantees on mortgages that remain at risk, and whose value depends on house prices. These recent turn of events have generated massive interest among the people in industry and in academics in trying to understand the economics behind house prices. Answers to a large number of questions are sought as part of this effort. For instance, what is the level of house price risk that the financial sector is currently bearing in terms of likely default on outstanding mortgages? How does ownership of a risky home impact the ability of consumers to smooth life cycle consumption shocks by borrowing against and/or selling the home? How much financial risk is involved in promising use of future taxes to support guarantees being offered by Fannie Mae and Freddie Mac? How predictable are house prices, and what are the key economic fundamentals impacting the evolution of house prices? What is the nature and extent of any mean reversion in house prices? To what extent can individual house price risk be hedged by index-based securities?

In order to answer the above questions and more, there is a need to solve two fundamental problems. First, there is a need for models that are able to accurately predict the prices of houses, not only in the near future but also in the distant future. Furthermore the prediction should be supplemented by a better aggregate understanding of the real estate market in the geographic region. More formally, the problem involves designing a model that predicts the price of a real estate property P, given the set of features X associated with it. These features include attributes that are specific to the individual house, such as, the number of bedrooms, number of bathrooms, the living area etc. They could also include information about the neighborhood in which it lies, such as the census tract specific information: the average household income of the neighborhood, average commute time to work, and school district information. This modeling approach, in which the relevant object is expressed in terms of its constituent characteristics or features and an estimate of the contributory value of each characteristic is obtained, is called *Hedonic Modeling* in economics. The estimate of the prices obtained are called *Hedonic Prices*. However, just using the attributes corresponding to an individual house is not sufficient in predicting its price. The problem has a very strong spatio-temporal structure associated with it. Variables such as the quality of the neighboring houses and the time at which the house is sold have a substantial impact on its price. Note that the values of these variables is not given to us as part of the data. Their values are only indirectly reflected in the selling price of neighboring houses in the space-time domain. Uncovering this underlying structure and using it to refine the estimate for every house, can substantially improve the prediction accuracy, while at the same time giving a high level understanding of the real estate market in the geographic area.

The second fundamental problem that one needs to solve while pursuing answers to the above mentioned questions is to understand the way in which the house prices move over time. For this we need to have a dependable and precise measure that captures these movements. However it is by no means an easy task. The principle difficulty is due to the fact that houses are heterogeneous goods: no two houses are exactly the same. Secondly, houses are sold infrequently: between 3% - 7% of all houses transact annually. Thirdly, as prices are negotiated, particular circumstances for individual buyers and sellers can lead to the situation that even extremely similar houses sell for very different amounts. Given this complex nature of the housing market it is convenient and useful to capture the overall average price trends followed by a group of houses. This is achieved by having a single price index for houses, called the *House Price Index* (HPI). A House

Price Index is simply one of many plausible measures of the central tendency of house price appreciation for a particular group of properties. For almost any non-trivial use of the HPI, the hope is that the index can be used to give a reasonable first approximation to current house values, and therefore be relevant to prices of homes that have not been sold, but may be so in a short period. Thus the underlying procedure for generating a useful HPI involves a model which uses the index while predicting the house prices. The model parameters and the index should be learnt simultaneously so as to minimize the prediction error. Traditionally in economics the index is computed either using only the house specific features (the *Hedonic Price Index*), or using only those houses that have been transacted in a short period of time (the repeat sales of the houses). Index generated using this procedure is called the *Repeat Sales Index*. Both these methods completely ignore the influence of the neighboring houses on the price of the current house and hence are not able to capture the rich spatio-temporal structure that is inherent in the problem. While computing the index if, in addition to taking into account the repeat sales of a particular house, one also considers the sale prices of the neighboring houses in the dimensions of time and space one can expect to generate a better index that predicts the house prices more accurately and is free from various superfluous patterns in the error structure.

To summarize, designing models for accurately predicting the house prices in future, and designing models for constructing aggregate price indices, are the two fundamental problems that one needs to solve on our way to understanding real estate prices. Both these problems have a strong underlying spatio-temporal structure that the model should uncover and exploit, as opposed to dealing with individual houses independently of others. In this thesis we propose novel models that try to answer both these questions, while at the same time uncovering the hidden spatio-temporal structure that is inherent in the problem.

0.3.2 A Relational Regression Problem

We now show explicitly how the problem of house price prediction is actually a relational regression problem. Each house is described by a vector X of features that are specific to that house, such as the number of bedrooms, 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. This is the hedonic component of the house price. In addition, the price of a house is also influenced by the features of the neighborhood in which it lies. Clearly, given two houses of similar characteristics, the house 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 that influence the price are directly measurable, such as the quality of the local schools, median house hold income, average time of commute to work etc. However most of the features, that make one particular neighborhood "desirable" to live in as compared to other neighborhoods, are very difficult to measure directly, and are merely reflected in the prices of houses in that area. These features can be modeled as hidden (latent) variables and their influence on each other must be learnt collectively. In other words, while predicting the prices, houses cannot be treated as independent of each other. Rather they are spatially related in highly complex ways, whereby which they influence each other's sale prices.

In addition to the spatial relationships among houses, there is also a temporal component to it. The price of a house is also a function of the time period in which it is sold. Thus analogous to spatial "desirability" one can associate the notion of "temporal desirability" to any particular time period. Whether a particular time period, in which the current house is to be sold, has high "desirability" or not is a function of the sale price of other houses sold *around* that period. If the other houses are sold at a premium then it will have a positive effect on the sale price of the current house. However if they are sold at a discounted price, then it will have a negative effect.

The exact form in which these neighboring houses (where the neighbors are in time domain) influence the sale price of the current house is again not explicitly given to us and hence should be modeled as hidden variables. Intuitively this effect will be more prominent when the houses sold during the same period are also nearby houses. This indicates towards the fact that the dependencies between houses is not just spatial or temporal, rather it is spatio-temporal.

In summary, the price of a house, in addition to depending on its individual attributes, also depends on the quality of neighborhood in which it lies. Some of these neighborhood features which define its quality are directly measurable, while other are not and are merely reflected in the quality of the houses that comprise that neighborhood. In other words there is a complex relationship structure among houses, namely the spatio-temporal structure, influencing each other's prices. It is this dependency structure that makes this problem a relational regression problem.

0.3.3 Previous Work

The problem of predicting prices of real estate properties has a long history in the economics literature. One line of methods use hedonic price equation to estimate the price of houses. The price of a house is modeled as a function of only its attributes, each of which has an implicit market price. Primarily the effort in this line of work has gone into trying different types of functional forms for the hedonic equation. Linear parametric methods and their derivatives have been long used by Goodman (Goodman, 1978), and Hallvorsen and Pollakowski (Halvorsen and Pollakowski, 1981). An extension of the linear regression is the Box-Cox transformations proposed by Box and Cox (Box and Cox, 1964). All the functional forms studied so far can be seen as special cases of the quadratic Box-Cox transformation. However because these functional forms were too restrictive, they usually resulted in poor performance. Some work has also been done in the domain of non-linear methods. For example, Meese and Wallace in (Meese and Wallace, 1991) used locally weighted regressions, whereas Clapp in (Clapp, 2004) and Anglin and Gencay (Anglin and Gencay, 1996) used semi-parametric methods for the problem.

In line with the widely accepted belief that while predicting the price of a house, the price of its neighbouring houses contain useful information, a number of people have also explored the possibility of using spatio-temporal models. There is now a large hedonic price litrature on the spatio-temporal correlation of prices. Can in (Can, 1990; Can, 1992), model house prices using spatial autoregressions. Dubin (Dubin, 1992), Pace and Giley (Pace and Gilley, 1997), and Basu and Thibodeau (Basu and Thibodeau, 1998) claim that it is hard to capture all spatial and neighborhood effects using available data. Hence they directly model the spatial autocorrelation of the regressions residuals. Finally, there is a class of models that recognizes that vicinity in both space and time will matter. Such Spatio Temporal Autoregressive (STAR) models have been developed by Pace et al (Pace et al., 1998) and Gelfand et al (Gelfand et al., 2004).

The focus of all of the above models is on House Price Index (HPI) construction and towards estimating the model parameters efficiently and precisely. Very little emphasis is given to the issue of accurately predicting the prices of unseen houses in the future (predictability). Not a lot of work has been done in the direction of handling the problem from the machine learning point of view. In the limited attempts at using machine learning methods, either the models are too simplistic or the setting in which they have been applied (in particular the dataset) is not representative of the real world situation. For instance, Do and Grudnitski (Do and Grudnitski, 1992), and Nguyen and Cripps in (Nguyen and Cripps, 2001) have used very simple neural networks on a very small dataset. Some work has been done to automatically exploit the locality structure present in the problem. Kauko in (Kauko, 2002), used the Self Organizing Map (SOM) technique proposed by Kohonen (Kohonen, 1995) to automatically segment the spatial area and learn a separate model for each segment. However, since SOM does not produces a mapping function, it is not possible to predict the price of a new sample that has not been seen before during training.

Furthermore, because of the certain severe weaknesses of hedonic indexes, such as the hetrogeneity among houses and the data tape of housing characteristic not being standardized across municipalities, these indexes are hardly used in practice. Instead, the indexes that have become very popular are based on the repeat sales of houses, which attempt to get around various problems associated with hedonic indexes. The idea is by only looking at houses that have been sold more than once during a period of time, one can get an index that can control for heterogeneities among houses. One of the first works along these lines was due to Bailey et. al., (Bailey et al., 1963). However the underlying assumption behind their model is that houses do not change attributes over time, and there is no selection bias in selecting only households that have been sold at least twice. One of the major drawbacks of their model is that the error term is uncorrelated. What this means is that if a house was sold more than twice, their is no relation of the errors corresponding to its two pairs of sales. Furthermore two possibly very similar houses would have uncorrelated errors even if they sold and resold in exactly the same periods. Case and Shiller in (Case and Shiller, 1989) try to address the limitations associated with this model by arguing against a homoskedastic error term. They claim that the variance of the error term must depend on the time interval between sales. The basic intuition is that changes in attributes, which this model does not explicitly account for, are likely to be more the longer the time between sales, resulting in a larger unexplained variance in prices for these households i.e. there is a drift in house prices over time. Thus these observations are down-weighted while computing the index using a Weighted Repeat Sales (WRS) index. Literature continued to move slowly till 1995 when Goetzmann and Spiegel in (Goetzmann and Spiegel, 1995) showed the importance of non-temporal returns from houses while constructing any price-index using repeat sales. Later on in (Goetzmann and Spiegel, 1997), they incorporated locational returns in their distance weighted repeat sales model, by accounting for the fact that houses in the same neighborhood are likely to exhibit co-movements of returns and houses in different neighborhoods are likely to exhibit very little correlation in returns. However in-spite of using a very small dataset they run into computational issues.

0.4 Contribution and Outline

This thesis proposes novel models for doing regression in relational data using relational factor graphs. In a relational factor graph, single graph models the entire collection of samples by capturing two sets of dependencies. It consists of factors which captures the dependencies among variables associated with individual samples. In addition, it also captures the inter sample relationships using factors that take as input the variables associated with multiple samples. The class of models proposed are different from those discussed in the literature in several ways. Firstly, they pertain to relational regression problems in which the input and answer variables are continuous. Secondly, they allow for arbitrary inter-sample dependencies. Furthermore, these dependencies could be hidden and not given as part of the data. In such situations these are modeled via hidden (latent) variables which themselves could be continuous. Thirdly, they allow the use of log-likelihood functions that are non-linear in the parameter space, which leads to considerably more flexible architectures at the cost of non-convex loss functions. Lastly, they eliminate the intractable partition function problem and provide efficient inference and training algorithms through appropriate design of the relational and non-relational factors. These models are applied to the problem of understanding house prices, where two major questions are answered. First, we propose a model that predicts the price of houses while at the same time capturing the relationships among houses which are spatially close to each other. This is achieved by learning a latent manifold over the geographic area, which also provides an aggregate understanding of the housing market. Second, a model is proposed to construct house prices indices, with the help of an explicitly learnt normalized price surface over space and time. The surface captures the spatio-temporal relationships among transactions. In both cases we conclude that uncovering and exploiting the relational structure associated with this problem significantly improves performance.

The proposed models and their inference and learning framework is discussed in the light of Energy Based Models. Chapter 1 gives a brief introduction to Energy Based Models. Here we talk about the processes of inference and learning in EBMs. Learning in EBMs is accomplished by minimizing a suitably chosen loss function that shapes the energy function in such a way that its minimization will give the desired answer. The choice of energy function and inference algorithm is solely dependent on the problem at hand. The choice of a loss function is independent of these decisions. However there are some guidelines that specify which loss function will work with which energy function. These are discussed in chapter 6. Chapter 2 introduces the factor graph models in the light of energy based models. Here we discuss the algorithms used for doing exact and approximate inference. Chapter 3 extends these factor graph models to address the problem of relational regression. The ideas in this chapter are explained with the help of a relational factor graph used for house price prediction. In spite of their complex link structure, we show how efficient learning and inference can be achieved by clever design of the relational and non-relational factors. In chapter 4 the relational factor graph model is actually used for predicting the prices of houses. The relational component of the model is used to help uncover the underlying hidden spatial structure associated with the problem. This hidden spatial structure is modeled as a learnable spatially smooth latent "desirability" manifold that spans the geographic area. We show that using the information from this learnt manifold (which captures the neighborhood effects of houses), along with the individual features of the house, one can improve upon prediction substantially. In chapter 5, we extend these ideas to build a spatio-temporal model for constructing house price indices. Akin to the "desirability" manifold of chapter 4, we learn a non-parametric "normalized" price surface over space and time which satisfies the spatio-temporal smoothness constraint. The price of a house is modeled as the product of the "normalized" price and a city-wide global index (the House Price Index). Both the index and the normalized prices are simultaneously learnt to minimize the prediction error. We show that our estimated price index is very similar to standard repeat sales index (the Case-Shiller's index) used presently in the market. However, our model convincingly out-performs this index in prediction accuracy, and also is free from the various superfluous patterns in the error structure exhibited by this index. This can again be attributed to exploiting the underlying spatio-temporal structure associated with the problem. To the best of our knowledge, the methods we propose are first attempt to automatically learn the influence of the underlying spatio-temporal structure inherent in the problem, and use it for index construction and prediction. Finally, unlike in the previous works, the dataset used in all the above experiments is industry standard, and substantially larger and very diverse. It has around 1.3 million transactions over the last 24 years spanning the entire Los Angeles county. The dataset is explained in detail in chapter 4.

1 Energy Based Models

The main purpose of statistical modeling and machine learning is to design models that are capable of answering questions relating to *Prediction, Classification, Decision Making, Ranking, Detection*, and *Conditional Density Estimation*. Most approaches involve designing models that encode dependencies between different variables associated with the problem. By capturing these dependencies, a model can then be used to determine the values of unknown variables given the values of known variables, and use this information to answer questions of the above type.

Energy-Based Models (EBMs) capture these dependencies by associating a scalar *energy* to each configuration of the variables. This energy can be viewed as a measure of compatibility between the values of variables. The process of *inference*, i.e., making a prediction or decision, consists in fixing the value of observed variables and finding values of the remaining variables that minimize the energy. *Learning* consists in finding an energy function that associates low energies to the configuration of variables consistent with the training data and high energies to others. A *loss functional*, which measures the quality of an energy function, is minimized during the learning process in order to accomplish the task.

Within this common *inference* and *learning* framework, the wide choice of energy functions and loss functionals allows for the design of many types of statistical models, both probabilistic and non-probabilistic, providing a unified framework for the two approaches to learning.

1.1 Inference in Energy Based Models

Consider a model with two sets of variables, X and Y, as represented in Figure 1.1. Variable X could be a vector containing the pixels from an image of an object. Variable Y could be a discrete variable that represents the possible category of the object. For example, Y could take six possible values: animal, human figure, airplane, truck, car, and “none of the above”. An Energy Based Model can be viewed as a function that measures the “goodness” (or badness) of each possible configuration of X and Y. The output of this function (which we call *energy*) can be interpreted as the degree of *compatibility* between the values of X and Y. We use the convention that small energy values correspond to highly compatible configurations of the variables, while large energy values correspond to highly incompatible configurations of the variables. We will use the term *energy function* and denote it by $E(Y, X)$. This energy function could also be parameterized by a set of parameters W, in which case it is denoted by $E(W, Y, X)$. The *architecture* of an EBM is the internal structure of the parameterized energy function $E(W, Y, X)$. No particular restriction is placed on the nature of the variables X, Y, and the parameters W.

Given a fixed input X – which is observed from the world – the process of inference involves, asking the model to produce a value of the unobserved variable Y that is most compatible with the observed variable X. More precisely, in an energy based model the inference is done by choosing a value Y^*, from the set of all possible values of the unobserved variables $\mathcal{Y}$, for which the energy $E(W, Y, X)$ is the smallest:

$$Y^* = \text{argmin}_{Y \in \mathcal{Y}} E(W, Y, X). \tag{1.1}$$

When the size of the set $\mathcal{Y}$ is small, we can simply compute $E(W, Y, X)$ for all possible values of $Y \in \mathcal{Y}$ and pick the smallest.

In general, however, picking the best Y may not be straight forward, as in many applications an exhaustive search over the set $\mathcal{Y}$ might be impractical. For example, when the model is used to identify the person in a given facial image (*face recognition*), the set $\mathcal{Y}$ is discrete and finite, but its cardinality may be tens of thousands (Chopra et al., 2005). Likewise, consider the situation when the model is used to detect whether there’s a face in a given image or not? Furthermore, if

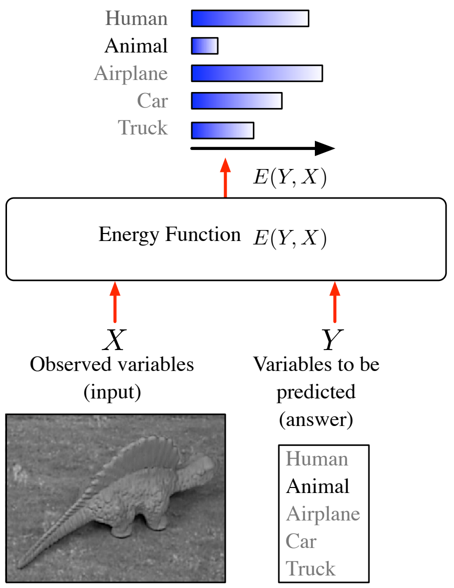


Figure 1.1: *A model measures the compatibility between observed variables X and variables to be predicted Y using an* energy function $E(Y, X)$. *For example, X could be the pixels of an image, and Y a discrete label describing the object in the image. Given X, the model produces the answer Y that minimizes the energy E.*

the face is present, then what is its location and its pose (*face detection*)? In this case the set $\mathcal{Y}$ contains a binary variable for each location indicating whether a face is present at that location, and a set of continuous variables representing the size and orientation of the face (Osadchy et al., 2005). We often think of X as a high-dimensional variable (e.g. an image) and Y as a discrete variable (e.g. a label), but even Y could be a high dimensional variable making things even more complex. For example, when the model is used to restore the image by removing the noise, enhancing the resolution, or removing scratches (*image restoration*), the set $\mathcal{Y}$ contains all possible images (all possible pixel combinations). It is a continuous and high-dimensional set. Or when the model is used to recognize a handwritten sentence in an image, the set $\mathcal{Y}$ contains all possible sentences of the English language, which is a discrete but infinite set of sequences of symbols (LeCun et al., 1998a).

For each of the above situations, a specific *inference procedure*, must be employed to find the Y that minimizes $E(W, Y, X)$. In many real situations, the inference procedure will produce an approximate result, which may or may not be the global minimum of $E(W, Y, X)$ for a given X. In fact, there may be situations where $E(W, Y, X)$ has several equivalent minima. The best inference procedure to use often depends on the internal structure of the model. For example, if $\mathcal{Y}$ is continuous and $E(W, Y, X)$ is smooth and well-behaved with respect to Y, one may use a gradient-based optimization algorithm. If Y is a collection of discrete variables and the energy function can be expressed as a sum of energy functions (called *factors*) that depend on different subsets of variables, the efficient inference procedures for factor graphs, like the *min-sum* algorithm, can be used (Kschischang et al., 2001b; MacKay, 2003). When each element of $\mathcal{Y}$ can be represented as a path in a weighted directed acyclic graph, then the energy for a particular Y is the sum of values on the edges and nodes along a particular path. In this case, the best Y can be found efficiently using dynamic programming (e.g with the Viterbi algorithm or A^*). This situation often occurs in sequence labeling problems such as speech recognition, handwriting recognition, natural language processing, and biological sequence analysis (e.g. gene finding, protein folding prediction, etc). Different situations may call for the use of other optimization procedures, including continuous optimization methods such as linear programming, quadratic programming, non-linear optimization methods, or discrete optimization methods such as simulated annealing, graph cuts, or graph matching. In many cases, exact optimization is impractical, and one must resort to approximate methods, including methods that use surrogate energy functions (such as variational methods).

1.1.1 Examples of Energy Based Architectures

To substantiate the above discussion we demonstrate how traditional models like standard regression can be formulated as energy-based models.

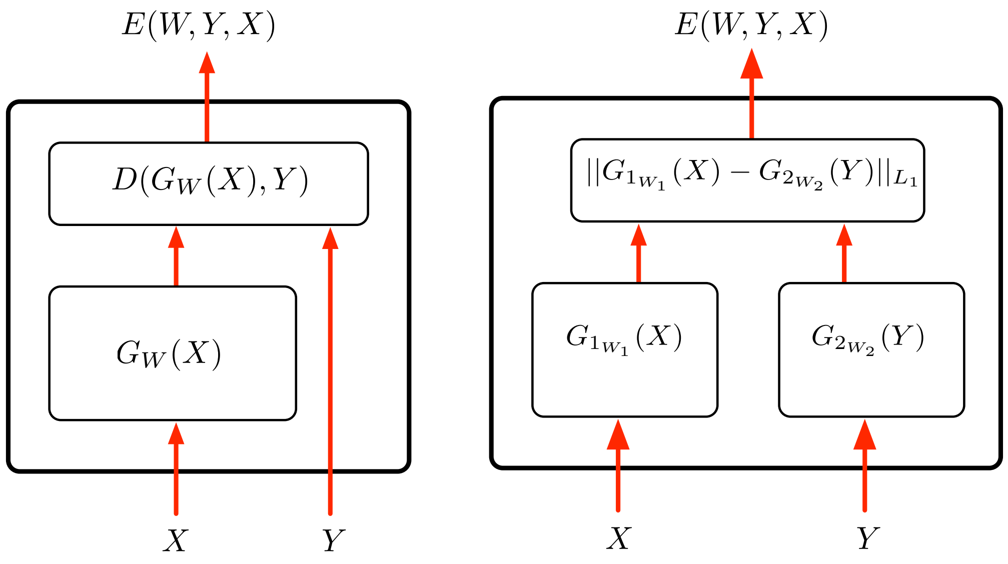


Figure 1.2: *Simple learning models viewed as EBMs:* **(a) a standard regressor:** *The energy is the discrepancy between the output of the regression function $G_W(X)$ and the answer Y. The best inference is simply $Y^* = G_W(X)$;* **(b) an implicit regression architecture:** *X and Y are passed through two functions $G1_{W_1}$ and $G2_{W_2}$. This architecture allows multiple values of Y to have low energies for a given X.*

Regression

Figure 1.2(a) shows a simple architecture for regression or function approximation. The energy function is the squared error between the output of a regression function $G_W(X)$ and the variable to be predicted Y, which may be a scalar or a vector:

$$E(W, Y, X) = \frac{1}{2}||G_W(X) - Y||^2. \tag{1.2}$$

The inference problem is trivial: the value of Y that minimizes E is equal to $G_W(X)$. The minimum energy is always equal to zero.

Implicit Regression

In practice, there are tasks in which for a single value of X multiple answers (Y) are equally good. For example in robot navigation, turning left or right may get the robot around an obstacle equally well, or in a language model in which the sentence segment "the cat ate the" can be followed equally well by "mouse" or "bird". More generally, the dependency between X and Y sometimes cannot be expressed as a function that maps X to Y (e.g., consider the constraint $X^2 + Y^2 = 1$). In this case, which we call *implicit regression*, we model the constraint that X and Y must satisfy and design the energy function such that it measures the violation of the constraint. Both X and Y can be passed through functions, and the energy is a function of their outputs (see Figure 1.2 (b)). A simple example is:

$$E(W, Y, X) = \frac{1}{2}||G_{1_{W_1}}(X) - G_{2_{W_2}}(Y)||^2. \tag{1.3}$$

For some problems, the function $G_{1_{W_1}}$ must be different from the function $G_{2_{W_2}}$. In other cases, $G_{1_{W_1}}$ and $G_{2_{W_2}}$ must be instances of the same function G.

An interesting example is the *Siamese* architecture (Bromley et al., 1993b): variables X_1 and X_2 are passed through two instances of a function G_W. A binary label Y determines the constraint on $G_W(X_1)$ and $G_W(X_2)$: if $Y = 0$, $G_W(X_1)$ and $G_W(X_2)$ should be equal, and if $Y = 1$, $G_W(X_1)$ and $G_W(X_2)$ should be different. In this way, the regression on X_1 and X_2 is implicitly learned through the constraint Y rather than explicitly learned through supervision. Siamese architectures were originally designed for signature verification (Bromley et al., 1993b). More recently they have been used to learn a similarity metric with application to face verification in (Chopra et al., 2005), and for unsupervised learning of manifolds in (Hadsell et al., 2006).

1.2 Learning in Energy Based Models

In the supervised learning framework, the task of training an EBM consists of finding an energy function that produces the best Y for any X. The search for the best energy function is performed within a family of energy functions $\mathcal{E}$ indexed by parameters W:

$$\mathcal{E} = \{E(W, Y, X) : \quad W \in \mathcal{W}\}. \tag{1.4}$$

Since no particular restriction is placed on the nature of X, Y, and W, $\mathcal{E}$ could take any complicated form. For instance when X and Y are real vectors, $\mathcal{E}$ could be a simple linear combination of basis functions (as in the case of kernel methods), or a set of neural net architectures and weight values. When X and Y are variable-size images, sequences of symbols or vectors, or more complex structured objects, $\mathcal{E}$ may represent a considerably richer class of functions. Since little restriction is placed on the nature of $\mathcal{E}$, one can capture highly complex dependencies among variables. This is one of the main advantages of the energy-based approach.

For the purpose of training we are given a set of training samples $\mathcal{S} = \{(X^i, Y^i) : i = 1 \ldots P\}$, where X^i is the input for the i-th training sample, and Y^i is the corresponding desired answer. In order to find the best energy function in the family $\mathcal{E}$, we need a way to assess the quality of any particular energy function, based solely on two elements: the training set, and our prior knowledge about the task. This quality measure is called the *loss functional* (i.e. a function of function) and is denoted by $\mathcal{L}(E, \mathcal{S})$. For simplicity, we often consider it as a function of the parameters W and denote it by $\mathcal{L}(W, \mathcal{S})$ and call it the *loss function*. The learning problem is to find the W that minimizes the loss:

$$W^* = \min_{W \in \mathcal{W}} \mathcal{L}(W, \mathcal{S}). \tag{1.5}$$

When the data is assumed to be independent and identically distributed (i.i.d), the loss functional can be expressed as the average of the *per-sample loss functional* taken over the entire set of training samples. That is,

$$\mathcal{L}(W,\mathcal{S}) = \frac{1}{P}\sum_{i=1}^{P} L(Y^i, E(W,\mathcal{Y},X^i)) + R(W), \tag{1.6}$$

where $L(Y^i, E(W,\mathcal{Y},X^i))$ is the *per-sample loss functional*, which depends on the desired answer Y^i and on the energies obtained by keeping the input sample fixed and varying the answer Y. Thus, for each sample, we evaluate a "slice" of the energy surface. The term $R(W)$ is the *regularizer*, and can be used to embed our prior knowledge about which energy functions in our family are preferable to others (in the absence of training data). With this definition, the loss is invariant under permutations of the training samples and under multiple repetitions of the training set.

For the guarantee of good generalization performance (performance on unseen test set), we can rely on general results from statistical learning theory. These results guarantee that, under simple interchangeability conditions on the samples and general conditions on the family of energy functions (finite VC dimension), the deviation between the value of the loss after minimization on the training set, and the loss on a large, separate set of test samples is bounded by a quantity that converges to zero as the size of training set increases (Vapnik, 1995).

1.2.1 Designing a Loss Functional

Training an EBM consists of choosing an energy function from the family $\mathcal{E}$ of energy functions so that for any given X, the inference algorithm will produce the desired value of Y. This process can be viewed as "carving" the energy function, by adjusting the parameters W so that for any given X, the inference algorithm returns the desired value Y. Since the inference algorithm selects the Y with the lowest energy, the learning procedure must shape the energy surface so that the desired value of Y has lower energy than all other (undesired) values. Figures 1.3 and 1.4 show examples of energy as a function of Y for a given input sample X^i in cases where Y is a discrete variable and a continuous scalar variable. We note three types of answers:

- Y^i: the correct answer
- Y^{*i}: the answer produced by the model, i.e. the answer with the lowest energy.
- $\bar{Y}^i$: the *most offending incorrect answer*, i.e. the answer that has the lowest energy among all the incorrect answers. To define this answer in the continuous case, we can simply view all answers within a distance ϵ of Y^i as correct, and all answers beyond that distance as incorrect.

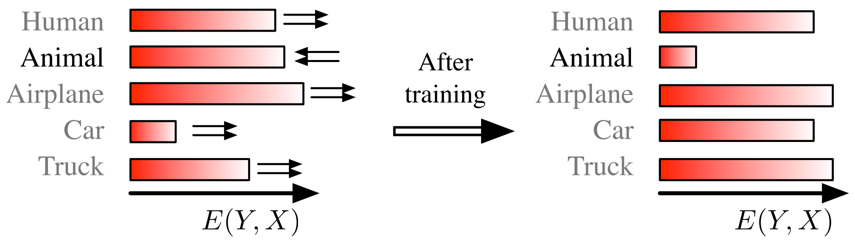


Figure 1.3: *How training affects the energies of the possible answers in the discrete case: the energy of the correct answer is decreased, and the energies of incorrect answers are increased, particularly if they are lower than that of the correct answer.*

As mentioned before, this process of "carving" the energy function to take the desired shape is accomplished by minimizing, with respect to the parameters W, a loss functional, which in turn is a sum of per-sample loss functionals. Intuitively, the per-sample loss functional should be designed in such a way that it assigns a low loss to *well-behaved* energy functions: energy functions that give the lowest energy to the correct configuration of variables (X, Y) and higher energy to all other (incorrect) configurations. Conversely, energy functions that do not assign the lowest energy to the correct answers should have a high loss.

With a properly designed loss function, the learning process should have the effect of "pushing down" on $E(W, Y^i, X^i)$, and "pulling up" on the incorrect energies, particularly on $E(W, \bar{Y}^i, X^i)$.

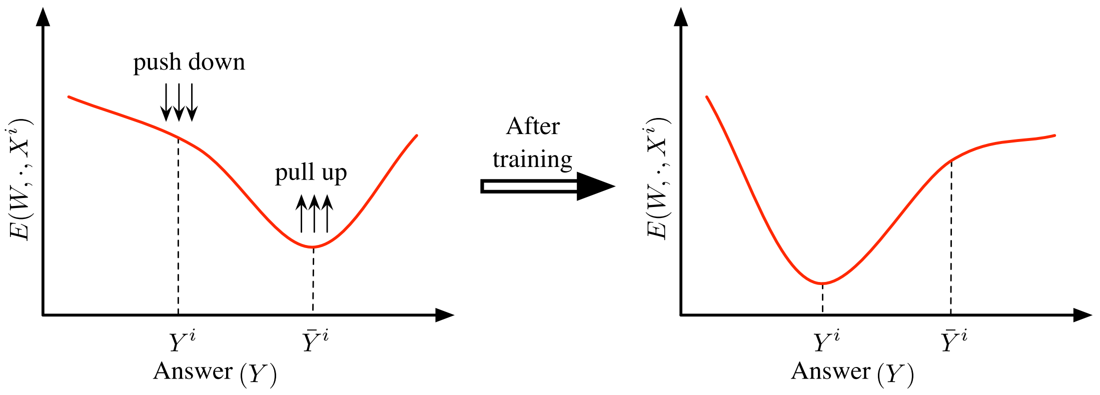


Figure 1.4: *The effect of training on the energy surface as a function of the answer Y in the continuous case. After training, the energy of the correct answer Y^i is lower than that of incorrect answers.*

Different loss functions do this in different ways. Chapter 6 gives sufficient conditions that a loss function must satisfy in order to be guaranteed to shape the energy surface correctly. We show that some widely used loss functions do not satisfy the conditions, while others do.

Properly designing the architecture of the energy function and the loss function is critical. Any prior knowledge we may have about the task at hand is embedded into the architecture and into the loss function (particularly the regularizer). Unfortunately, not all combinations of architectures and loss functions are allowed. With some combinations, minimizing the loss will not make the model produce the correct answers. Choosing the combinations of architecture and loss functions that can learn effectively and efficiently is critical to the energy-based approach.

1.2.2 Examples of Loss Functions

A number of loss functions have been proposed in the literature. Here we discuss a few common loss functions that have been used. A more elaborate list is given in (LeCun et al., 2006). We also comment on whether the loss function is "good" or "bad" depending on whether it is able to "carve" an appropriate energy landscape or not. A more formal treatment to this discussion is given in chapter 6, where we give sufficient conditions that any loss function should satisfy in order to be classified as "good". For the time being, we set aside the regularization term, and concentrate on the data-dependent part of the loss function.

Energy Loss

This is the simplest of all the loss functions and is also very popular for learning regression functions and neural network training. For a training sample (X^i, Y^i), the per-sample loss is defined simply as:

$$L_{energy}(Y^i, E(W, \mathcal{Y}, X^i)) = E(W, Y^i, X^i). \tag{1.7}$$

In spite of its popularity it cannot be used to train most architectures. This is because, while this loss will push down on the energy of the desired answer, it makes no attempt to pull up on the energies of other answers. With some architectures, this can lead to a *collapsed solution* in which the energy is constant and equal to zero. The energy loss will only work with architectures that are designed in such a way that pushing down on $E(W, Y^i, X^i)$ will automatically make the energies of the other answers larger. A simple example of such an architecture is $E(W, Y^i, X^i) = ||Y^i - G(W, X^i)||^2$, which corresponds to regression with mean-squared error with G being the regression function.

Generalized Perceptron Loss

For a training sample (X^i, Y^i), the per-sample loss is defined as

$$L_{perceptron}(Y^i, E(W, \mathcal{Y}, X^i)) = E(W, Y^i, X^i) - \min_{Y \in \mathcal{Y}} E(W, Y, X^i). \tag{1.8}$$

This loss is always positive, since the second term is a lower bound on the first term. Minimizing this loss has the effect of pushing down on $E(W, Y^i, X^i)$, while pulling up on the energy of the answer produced by the model.

While the perceptron loss has been widely used in many settings, including for models with

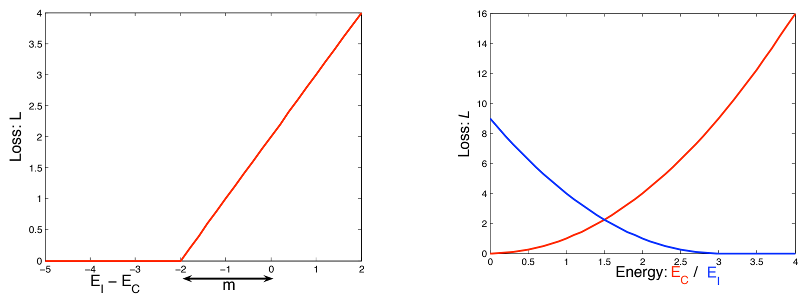


Figure 1.5: *The hinge loss (left) penalize $E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i)$ linearly. The square-square loss (right) separately penalizes large values of $E(W, Y^i, X^i)$ (solid line) and small values of $E(W, \bar{Y}^i, X^i)$ (dashed line) quadratically.*

structured outputs such as handwriting recognition (LeCun et al., 1998a) and parts of speech tagging (Collins, 2002), it has a major deficiency: there is no mechanism for creating an energy gap between the correct answer and the incorrect ones. Hence, as with the energy loss, the perceptron loss may produce flat (or almost flat) energy surfaces if the architecture allows it. Consequently, a meaningful, uncollapsed result is only guaranteed with this loss if a model is used that cannot produce a flat energy surface such as the one in the paper by Collins (Collins, 2002). For other models, one cannot guarantee anything.

Generalized Margin Losses

We now describe a general class of loss functions called *generalized margin loss*, which uses some form of *margin* to create an energy gap between the correct answer and the incorrect answers. The loss functions that belong to this class are the hinge loss, log loss, LVQ2 loss, minimum classification error loss, square-square loss, and square-exponential.

Definition 1.1. Let Y be a discrete variable. Then for a training sample (X^i, Y^i), the **most offending incorrect answer** $\bar{Y}^i$ is the answer that has the lowest energy among all answers that are incorrect:

$$\bar{Y}^i = \text{argmin}_{Y \in \mathcal{Y} and Y \neq Y^i} E(W, Y, X^i). \tag{1.9}$$

If Y is a continuous variable then the definition of the most offending incorrect answer can be defined in a number of ways. The simplest definition is as follows.

Definition 1.2. Let Y be a continuous variable. Then for a training sample (X^i, Y^i), the **most offending incorrect answer** $\bar{Y}^i$ is the answer that has the lowest energy among all answers that are at least ϵ away from the correct answer:

$$\bar{Y}^i = \text{argmin}_{Y \in \mathcal{Y}, \|Y - Y^i\| > \epsilon} E(W, Y, X^i). \tag{1.10}$$

The form of any generalized margin loss is given by:

$$L_{\text{margin}}(W, Y^i, X^i) = Q_m \left(E(W, Y^i, X^i), E(W, \bar{Y}^i, X^i) \right). \tag{1.11}$$

It can be seen as a more robust version of the generalized perceptron loss, since it explicitly uses the energy of the most offending incorrect answer in the contrastive term. Furthermore, it uses a positive parameter m, called the *margin*, to create an energy gap between the correct answer and the most offending incorrect answer. $Q_m(e_1, e_2)$ is a convex function in e_1 and e_2, whose gradient has a positive dot product with the vector $[1, -1]$ in the region where $E(W, Y^i, X^i) + m > E(W, \bar{Y}^i, X^i)$. In other words, the loss surface is slanted toward low values of $E(W, Y^i, X^i)$ and high values of $E(W, \bar{Y}^i, X^i)$ wherever $E(W, Y^i, X^i)$ is not smaller than $E(W, \bar{Y}^i, X^i)$ by at least m.

Hinge Loss: A particularly popular example of generalized margin loss is the *hinge loss*, which is used in combination with linearly parameterized energies and a quadratic regularizer in support vector machines, support vector Markov models (Altun and Hofmann, 2003), and maximum-margin Markov networks (Taskar et al., 2003):

$$L_{\text{hinge}}(W, Y^i, X^i) = \max \left(0, m + E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i) \right), \tag{1.12}$$

where m is the positive margin. The shape of this loss function is given in Figure 1.5. The difference between the energies of the correct answer and the most offending incorrect answer is penalized linearly when larger than $-m$. The hinge loss only depends on energy differences, hence individual energies are not constrained to take any particular value.

Square-Square Loss: Unlike the hinge loss, the square-square loss treats the energy of the correct answer and the most offending answer separately (LeCun and Huang, 2005; Hadsell et al., 2006):

$$L_{\text{sq}-\text{sq}}(W, Y^i, X^i) = E(W, Y^i, X^i)^2 + \left(\max(0, m - E(W, \bar{Y}^i, X^i))\right)^2. \tag{1.13}$$

Large values of $E(W, Y^i, X^i)$ and small values of $E(W, \bar{Y}^i, X^i)$ below the margin m are both penalized quadratically (see Figure 1.5). Unlike the margin loss, the square-square loss "pins down" the correct answer energy at zero and "pins down" the incorrect answer energies above m. Therefore, it is only suitable for energy functions that are bounded below by zero, notably in architectures whose output module measures some sort of distance.

Other loss functions that belong to this class are the log loss, LVQ2 loss, minimum classification error loss, and square-exponential which are discussed in detail in (LeCun et al., 2006).

Negative Log-Likelihood Loss

The negative log-likelihood loss, which is motivated from probabilistic modeling, is defined as:

$$L_{\text{nll}}(W, Y^i, X^i) = E(W, Y^i, X^i) + \mathcal{F}_\beta(W, \mathcal{Y}, X^i). \tag{1.14}$$

Where $\mathcal{F}$ is the *free energy* of the ensemble $\{E(W, y, X^i),\ y \in \mathcal{Y}\}$:

$$\mathcal{F}_\beta(W, \mathcal{Y}, X^i) = \frac{1}{\beta} \log \left(\int_{y \in \mathcal{Y}} \exp\left(-\beta E(W, y, X^i)\right) \right). \tag{1.15}$$

where β is a positive constant akin to an inverse temperature. This loss can only be used if the exponential of the negative energy is integrable over $\mathcal{Y}$, which may not be the case for some choices of energy function or $\mathcal{Y}$.

While many of the previous loss functions involved only $E(W, \bar{Y}^i, X^i)$ in their contrastive term, the negative log-likelihood loss combines all the energies for all values of Y in its contrastive term $\mathcal{F}_\beta(W, \mathcal{Y}, X^i)$. This term can be interpreted as the Helmholtz free energy (log partition function) of the ensemble of systems with energies $E(W, Y, X^i),\ Y \in \mathcal{Y}$. This contrastive term causes the energies of all the answers to be pulled up. The energy of the correct answer is also pulled up, but not as hard as it is pushed down by the first term. This can be seen in the expression of the gradient for a single sample:

$$\frac{\partial L_{\text{nll}}(W, Y^i, X^i)}{\partial W} = \frac{\partial E(W, Y^i, X^i)}{\partial W} - \int_{Y \in \mathcal{Y}} \frac{\partial E(W, Y, X^i)}{\partial W} P(Y|X^i, W), \tag{1.16}$$

where $P(Y|X^i, W)$ is obtained through the Gibbs distribution:

$$P(Y|X^i, W) = \frac{e^{-\beta E(W, Y, X^i)}}{\int_{y \in \mathcal{Y}} e^{-\beta E(W, y, X^i)}}. \tag{1.17}$$

Hence, the contrastive term pulls up on the energy of each answer with a force proportional to the likelihood of that answer under the model.

Unfortunately, for many interesting problems the integral over $\mathcal{Y}$ is intractable. Considerable efforts has been devoted in trying to approximate it with approaches, such as, clever organization of the calculations, Monte-Carlo sampling methods, and Variational methods. While these methods have been devised as approximate ways of minimizing the NLL loss, they can be viewed in the energy-based framework as different strategies for choosing the Y's whose energies will be pulled up. A more elaborate treatment on the relationship between energies and probabilities via the Gibbs distribution, and interpretation of approximate methods is given in Section 1.3.

1.2.3 Example Architectures Revisited

In this section we again look at the example architectures discussed in section 1.1.1, and see how they can be trained using the various loss functions discussed above.

Regression

The architecture for solving regression problems (Figure 1.2(a)) can be trained using any of the above discussed loss functions. In particular its training with with the energy loss, the perceptron loss, and the negative log-likelihood loss are all equivalent, because the contrastive term of the perceptron loss is zero, and that of the NLL loss is constant (it is a Gaussian integral with a constant variance):

$$\mathcal{L}_{\text{energy}}(W, \mathcal{S}) = \frac{1}{P} \sum_{i=1}^{P} E(W, Y^i, X^i) = \frac{1}{2P} \sum_{i=1}^{P} ||G_W(X^i) - Y^i||^2. \tag{1.18}$$

This corresponds to standard regression with mean-squared error.

A popular form of regression occurs when G is a linear function of the parameters:

$$G_W(X) = \sum_{k=1}^{N} w_k \phi_k(X) = W^T \Phi(X). \tag{1.19}$$

The $\phi_k(X)$ are a set of N *features*, and w_k are the components of an N-dimensional parameter vector W. For concision, we use the vector notation $W^T\Phi(X)$, where W^T denotes the transpose of W, and $\Phi(X)$ denotes the vector formed by each $\phi_k(X)$. With this linear parameterization, training with the energy loss reduces to an easily solvable convex least-squares minimization problem.

In simple models, the feature functions are hand-crafted by the designer, or separately trained from unlabeled data. In the dual form of kernel methods, they are defined as $\phi_k(X) = K(X, X^k)$, $k = 1 \ldots P$, where K is the kernel function. In more complex models such as multilayer neural networks and others, the ϕ's may themselves be parameterized and subject to learning, in which case the regression function is no longer a linear function of the parameters and hence the loss function may not be convex in the parameters.

Implicit Regression

Particular instances of the implicit regression architecture, namely the *Siamese architecture* have been trained using the square-exponential loss for learning similarity metric for face verification (Chopra et al., 2005), and using a square-square loss (Eq. 1.13) for unsupervised learning of manifolds (Hadsell et al., 2006). Bengio et al (Bengio et al., 2003) train their energy-based language model using a negative log-likelihood loss. However because of the high cardinality of $\mathcal{Y}$ (equal to the size of the English dictionary), they had to use approximations (importance sampling) and had to train the system on a cluster machine.

Note that even though here only very simple architectures are presented in the light of energy based models, one can design architectures to solve far more complicated problems. For instance energy based models have been effectively used for problems in the domain of structured prediction, which requires generating outputs with complex internal structure (Driancourt et al., 1991a; Driancourt et al., 1991b; Driancourt and Gallinari, 1992b; Driancourt and Gallinari, 1992a; Driancourt, 1994; Bengio et al., 1990; Bourlard and Morgan, 1990; Bottou, 1991; Haffner et al., 1991; Haffner and Waibel, 1991; Bengio et al., 1992; Haffner and Waibel, 1992; Haffner, 1993a; Morgan and Bourlard, 1995; Konig et al., 1996). The present thesis proposes novel energy-based models for the task of Statistical Relational Regression, which requires treating the entire data as a single instance of input to the model and collectively learning the parameters to perform regression in a non I.I.D setting.

1.3 Energy Based Models and Probabilistic Models

Energy Based Models provide a unified framework for many probabilistic and non-probabilistic approaches to learning, particularly for non-probabilistic training of graphical models and other structured models. Most probabilistic models can be viewed as special types of energy-based models in which the energy function satisfies certain normalizability conditions, and in which the loss function, optimized by learning, has a particular form. In this section we explore this connection in detail and show how Energy-based learning can be seen as an alternative to probabilistic estimation for prediction, classification, or decision-making tasks.

Denote the set of given training samples by $\mathcal{S} = \{(X^i, Y^i) : i = 1 \dots P\}$, where X^i is the input for the i-th training sample, and Y^i is the corresponding desired answer. Let us denote by $\mathbf{X} = \{X^1, \dots, X^P\}$ the collection of all inputs and by $\mathbf{Y} = \{Y^1, \dots, Y^P\}$ the collection of all outputs. Let W be the set of trainable parameters of the model. One valid and widely used estimator of the parameters W is the *maximum likelihood estimator* (MLE) or *maximum conditional likelihood estimator*, which involves finding the parameters W that maximizes the conditional likelihood of the output variables given the set of inputs: $P(\mathbf{Y}|\mathbf{X}, W)$. If one goes more along the lines of Bayesian learning then a prior distribution is introduced over the parameters W. Using Bayes theorem, the posterior distribution for W is proportional to the product of the prior distribution and the conditional likelihood function

$$P(W|\mathbf{Y}, \mathbf{X},) \propto P(\mathbf{Y}|\mathbf{X}, W)P(W). \tag{1.20}$$

The parameters W can now be determined by setting them to the value that maximizes this posterior distribution. This technique is also called *maximum posterior* (MAP).

Consider first the maximum likelihood scenario. Assuming that the data samples are independent and identically distributed (i.i.d) we can write the conditional likelihood of the desired answers, given the inputs as

$$P(\mathbf{Y}|\mathbf{X}, W) = P(Y^1, \dots, Y^P|X^1, \dots, X^P, W) = \prod_{i=1}^{P} P(Y^i|X^i, W). \tag{1.21}$$

Applying the maximum likelihood estimation principle, we seek the value of W that maximizes the above product, or the one that minimizes the negative $\log$ of the above product (log being a monotonic function):

$$-\log \prod_{i=1}^{P} P(Y^i|X^i, W) = \sum_{i=1}^{P} -\log P(Y^i|X^i, W). \tag{1.22}$$

We now show that minimizing the above negative log-likelihood is actually equivalent to minimizing the negative log-likelihood loss in the energy based setting, when the probabilities are defined in terms of energies.

One way of converting a collection of arbitrary (un-normalized) energies into a collection of normalized probabilities, is through *Gibbs distribution*. That is, given an energy $E(Y, X)$ associated with some configuration (Y, X), the corresponding probability will be

$$P(Y|X) = \frac{e^{-\beta E(Y,X)}}{\int_{y \in \mathcal{Y}} e^{-\beta E(Y,X)}}, \tag{1.23}$$

where β is an arbitrary positive constant akin to an inverse temperature, and the denominator is called the *partition function*. The choice of Gibbs distribution may seem arbitrary, but other probability distributions can be obtained (or approximated) through a suitable re-definition of the energy function. It should be noted that whether the numbers obtained this way are good probability estimates does not depend on how energies are turned into probabilities, but on how the energy function $E(Y, X)$ is estimated from data. Furthermore, the above transformation of energies into probabilities is only possible if the integral $\int_{y \in \mathcal{Y}} e^{-\beta E(Y,X)}$ converges. Clearly this is somewhat of a restriction on the types of energy functions that can be used and on the domain $\mathcal{Y}$ of the output variables. Whereas if you are working only with energies there is no such requirement for proper normalization. This allows for much more flexibility in designing the architecture of learning machines. In that sense, one can view the energy-based model framework as a superset of probabilistic framework.

Given the above transformation, the negative log of the conditional likelihood can be written as

$$-\log \prod_{i=1}^{P} P(Y^i|X^i, W) = \sum_{i=1}^{P} \beta E(W, Y^i, X^i) + \log \int_{y\in\mathcal{Y}} e^{-\beta E(W,y,X^i)}. \tag{1.24}$$

The final form of the negative log-likelihood loss is obtained by dividing the above expression by P and β (which has no effect on the position of the minimum):

$$\mathcal{L}_{\text{nll}}(W, \mathcal{S}) = \frac{1}{P} \sum_{i=1}^{P} \left(E(W, Y^i, X^i) + \frac{1}{\beta} \log \int_{y\in\mathcal{Y}} e^{-\beta E(W,y,X^i)} \right). \tag{1.25}$$

Thus we conclude that maximizing the conditional likelihood of the data in probabilistic models is the same as minimizing the NLL loss function in energy based models when the probabilities are defined as a function of energies via the Gibbs distribution.

Note that the contrastive term in the NLL loss function (the integral term) tries to pull up on the energies of all the answers. As mentioned earlier, the force with which the loss pulls up on the energies of these answers is proportional to the likelihood of that answer under the model. Unfortunately, there are many interesting models for which computing this integral over $\mathcal{Y}$ is intractable. Considerable effort has been devoted towards designing methods that approximate it. Among various techniques the major ones are:

- Clever organization of the calculations by exploiting the fact that the energy function can be factored into a sum of energy functions.
- Monte-Carlo sampling methods, which tries to approximate the integral in the derivative of the loss function (equation 1.16) by sampling from the conditional distribution.
- Variational methods, which searched for a surrogate distribution which is simpler to work with and is as close as possible to the conditional distribution. A by-product of these methods is a lower bound on the partition function.

The first technique, which exploits the factors of the energy function, forms the central theme of this dissertation and is discussed in great detail in the next and the following chapters. The idea behind variational methods is discussed primarily from the context of inference in factor graphs in the next chapter. Here we shed some light on the Monte-Carlo based sampling methods which directly seek to approximate the above integral.

Monte-Carlo Sampling Methods

In this method the joint distribution, on which it is hard to do inference, is approximated by a set of instantiations (also called *samples*) of its variables. These samples represent a part of the probability mass. The actual use of these methods is for solving problems that involve computing the expectation of some function $f(X)$ relative to the joint distribution $p(X)$

$$\Phi = \sum_{X} p(X) f(X). \tag{1.26}$$

For instance while computing the derivative of the negative log likelihood loss in equation 1.16, when the cardinality of the set $\mathcal{Y}$ is huge, the integral which is given by

$$\Phi = \int_{Y \in \mathcal{Y}} \frac{\partial E(W, Y, X^i)}{\partial W} P(Y|X^i, W), \tag{1.27}$$

is computed by sampling from the conditional distribution $P(Y|X^i, W)$ and approximating it with the sum

$$\Phi \approx \frac{1}{|R|} \sum_{r \in R} \frac{\partial E(W, r, X^i)}{\partial W}, \tag{1.28}$$

where R is the set of samples from $P(Y|X^i, W)$.

One common approach for sampling is called Markov Chain Monte Carlo (MCMC). Since it is very difficult to sample from the joint distribution directly, the idea behind this approach is to construct a process which gradually samples from distributions that are closer and closer to the actual joint distribution. This is accomplished by constructing a Markov Chain which is defined in terms of a set of states and a transition model from one state to another. The chain defined a process that evolves stochastically from state to state. More formally, let $\mathcal{X}$ denote the set of all possible values the set of variables X can take. This is also called the *state space* of X. Let $\mathcal{T}(\mathbf{x} \to \mathbf{x}')$ denote the *transition probability* of going from the state $\mathbf{x}$ to the state $\mathbf{x}'$. Note that both $\mathbf{x}$, and $\mathbf{x}'$ are the elements of the state space $\mathcal{X}$. This is the *transition model* that is associated with the Markov Chain. Let $X^{(t)}$ be the random variable giving the state of the chain at time t. Assume that the initial state $X^{(0)}$ is distributed according to some initial state distribution $q^{(0)}(X^{(0)})$. The distribution over the states at subsequent times can be defined by the chain dynamics

$$q^{(t+1)}(X^{(t+1)} = \mathbf{x}') = \sum_{\mathbf{x} \in \mathcal{X}} q^{(t)}(X^{(t)} = \mathbf{x}) \mathcal{T}(\mathbf{x} \to \mathbf{x}'). \tag{1.29}$$

This means that the probability of being is state $\mathbf{x}'$ at time $t + 1$ is the sum over all possible states $\mathbf{x}$ that the chain could have been at time t, of the probability of being in state $\mathbf{x}$ times the probability that the chain took a transition from $\mathbf{x}$ to state $\mathbf{x}'$. As the process converges we would expect the distribution $q^{(t+1)}$ to be close to $q^{(t)}$. That is

$$q^{(t)}(\mathbf{x}') \approx q^{(t+1)}(\mathbf{x}') = \sum_{\mathbf{x} \in \mathcal{X}} q^{(t)}(X^{(t)} = \mathbf{x}) \mathcal{T}(\mathbf{x} \to \mathbf{x}'). \tag{1.30}$$

Furthermore, at convergence from the resulting distribution $\pi(X)$, we would expect the probability of being in a state is the same as the probability of transitioning into it from a randomly sampled predecessor

$$\pi(X = \mathbf{x}') = \sum_{\mathbf{x} \in \mathcal{X}} \pi(X = \mathbf{x}) \mathcal{T}(\mathbf{x} \to \mathbf{x}'). \tag{1.31}$$

This is called the *stationary distribution*.

The idea behind MCMC sampling is to design the Markov Chain (the transition probability function $\mathcal{T}$) in such a way that its stationary distribution π is the same as the joint distribution p from which the samples are sought. There are a number of other requirements that a Markov Chain should satisfy in order for this to happen. One among them is that it should have a unique stationary distribution. This can be achieved if the chain satisfies the regularity property or the ergodic property.

While these approximate methods have been designed as ways of minimizing the NLL loss, another way of looking at them is in the light of energy-based framework. Essentially, all these methods (particularly the sampling method) are trying to do is, come up with intelligent ways of choosing the incorrect answers Y's whose energies needs to be pulled up.

1.4 Latent Variable Architectures

In the usual scenario for an energy based model, the energy is minimized with respect to the variables to be predicted Y, given the observed variables X. During training, the correct value of Y is given for each training sample. However there are numerous applications where it is convenient to use energy functions that depend on a set of hidden variables Z whose correct value is never (or rarely) given to us, even during training. For example, we could imagine training a face detection system with data for which the scale and pose information of the faces is not available. For these architectures, the inference process for a given set of variables X and Y involves minimizing over these unseen variables Z:

$$E(Y, X) = \min_{z \in \mathcal{Z}} E(z, Y, X). \tag{1.32}$$

Such hidden variables are called *latent variables*, by analogy with a similar concept in probabilistic modeling. The fact that the evaluation of $E(Y, X)$ involves a minimization over Z does not significantly impact the approach described so far, but the use of latent variables is so ubiquitous that it deserves special treatment. In particular, some insight can be gained by viewing the inference process in the presence of latent variables as a simultaneous minimization over Y and Z:

$$Y^* = \text{argmin}_{y \in \mathcal{Y}, z \in \mathcal{Z}} E(z, y, X). \tag{1.33}$$

Latent variables can be viewed as intermediate results on the way to finding the best output Y. At this point, one could argue that there is no conceptual difference between the Z and Y variables: Z could simply be folded into Y. The distinction arises during training: we are given the correct value of Y for a number of training samples, but we are never given the correct value of Z.

Latent variables are very useful in situations where a hidden characteristic of the process being modeled can be inferred from observations, but cannot be predicted directly. One such example is in recognition problems. For example, in face recognition the gender of a person or the orientation of the face could be a latent variable. Knowing these values would make the recognition task much easier. Likewise in invariant object recognition the pose parameters of the object (location, orientation, scale) or the illumination could be latent variables. They play a crucial role in problems where segmentation of the sequential data must be performed simultaneously with the recognition task. A good example is speech recognition, in which the segmentation of sentences into words and words into phonemes must take place simultaneously with recognition, yet the correct segmentation into phonemes is rarely available during training. Similarly, in handwriting recognition, the segmentation of words into characters should take place simultaneously with the recognition.

When the best value of the latent variable for a given X and Y is ambiguous, one may consider combining the contributions of the various possible values by marginalizing over the latent variables as opposed just minimizing over them. In particular, in the presence of latent variables the joint conditional probability distribution over Y and Z given X is given by the Gibbs distribution

$$P(Z, Y|X) = \frac{e^{-\beta E(Z,Y,X)}}{\int_{y\in\mathcal{Y}, z\in\mathcal{Z}} e^{-\beta E(z,y,X)}}. \tag{1.34}$$

The marginalizing over the latent variables Z gives

$$P(Y|X) = \frac{\int_{z\in\mathcal{Z}} e^{-\beta E(z,Y,X)}}{\int_{y\in\mathcal{Y}, z\in\mathcal{Z}} e^{-\beta E(z,y,X)}}. \tag{1.35}$$

Then finding the best Y after this marginalization over Z reduces to computing

$$Y^* = \text{argmin}_{y \in \mathcal{Y}} - \frac{1}{\beta} \log \int_{z \in \mathcal{Z}} e^{-\beta E(z,y,X)}. \tag{1.36}$$

This is actually a conventional energy-based inference in which the energy function has merely been redefined from $E(Z, Y, X)$ to $\mathcal{F}(\mathcal{Z}) = -\frac{1}{\beta} \log \int_{z \in \mathcal{Z}} e^{-\beta E(z,Y,X)}$, which is the *free energy* of the ensemble $\{E(z, Y, X),\ z \in \mathcal{Z}\}$. The above inference formula by marginalization reduces to the previous inference formula by minimization when $\beta \to \infty$ (zero temperature).

1.4.1 An Example of Latent Variable Architecture

To illustrate the concept of latent variables, we consider the task of face detection: determine whether a face is present in a small image or not. Imagine that we are provided with a face detecting function $G_{\text{face}}(X)$ which takes a small image window as input and produces a scalar output. It outputs a small value when a human face fills the input image, and a large value if no face is present (or if only a piece of a face or a tiny face is present). An energy-based face detector built around this function is shown in Figure 1.6(a). The variable Y controls the position of a binary switch ($1 =$ "face", $0 =$ "non-face"). The output energy is equal to $G_{\text{face}}(X)$ when $Y = 1$, and to a fixed threshold value T when $Y = 0$:

$$E(Y, X) = Y G_{\text{face}}(X) + (1 - Y)T.$$

The value of Y that minimizes this energy function is 1 (face) if $G_{\text{face}}(X) < T$ and 0 (non-face) otherwise.

Let us now consider the more complex task of *detecting and locating* a single face in a large image. We can apply our $G_{\text{face}}(X)$ function to multiple windows in the large image, compute which window produces the lowest value of $G_{\text{face}}(X)$, and detect a face at that location if the value is lower than T. This process is implemented by the energy-based architecture shown

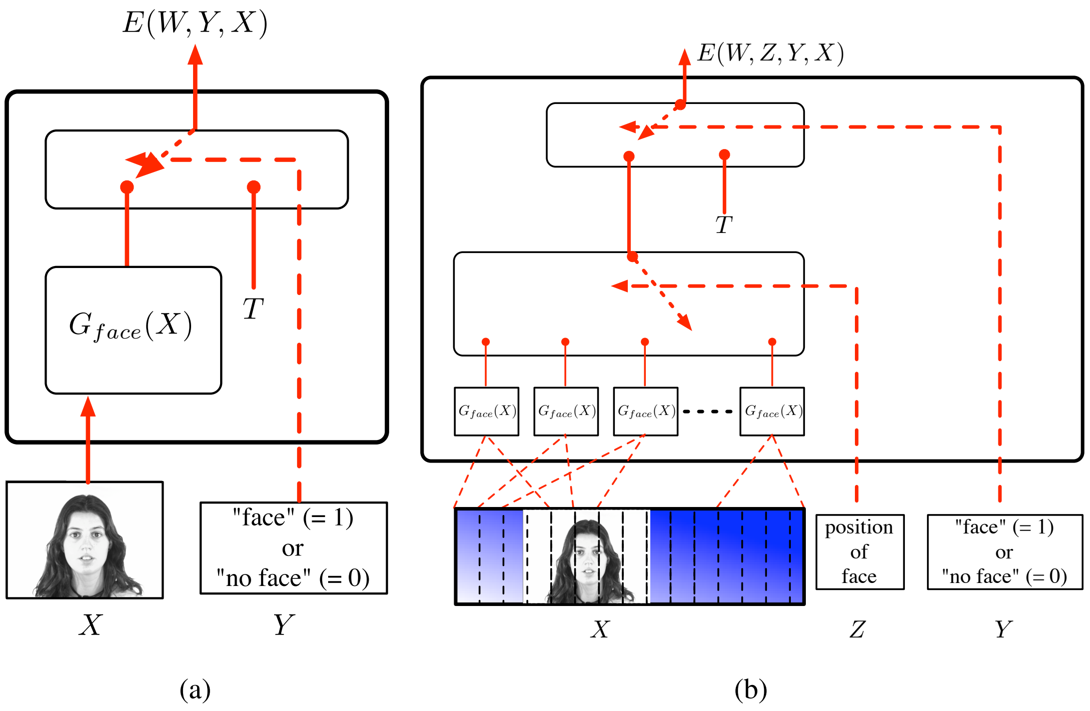


Figure 1.6: (a): *Architecture of an energy-based face detector. Given an image, it outputs a small value when the image is filled with a human face, and a high value equal to the threshold T when there is no face in the image.* (b): *Architecture of an energy-based face detector that simultaneously locates and detects a face in an input image by using the location of the face as a latent variable.*

in Figure 1.6(b). The latent "location" variable Z selects which of the K copies of the G_{face} function is routed to the output energy. The energy function can be written as

$$E(Z, Y, X) = Y \left[\sum_{k=1}^{K} \delta(Z - k) G_{\text{face}}(X_k) \right] + (1 - Y)T, \qquad (1.37)$$

where the X_k's are the image windows. Locating the best-scoring location in the image consists in minimizing the energy with respect to Y and Z. The resulting value of Y will indicate whether a face was found, and the resulting value of Z will indicate the location.

1.5 Summary

EBMs provide a unified framework for inference and learning. They capture dependencies among variables by assigning a scalar energy to each configuration of the variables. Given a training set $\mathcal{S}$, building and training an energy-based model involves specifying four components:

1. First is the *architecture* of the model. This is essentially the internal structure of the energy function $E(W, Y, X)$. Its design is governed by the prior knowledge that one might have about the task to be solved. No restriction is placed on the functional form of E.

2. The second component is the *inference algorithm*. This involves specifying the method for finding a value of Y that minimizes the energy function $E(W, Y, X)$ for any given X. If there are latent variables involved in the model, then one is free to either marginalize or minimize over them in this step. It again depends on the problem at hand and what is sought. The only restriction on the inference algorithm is that it should seek to find a Y that minimizes the energy function E.

3. The third component is the *loss function*. The loss function $\mathcal{L}(W, \mathcal{S})$ measures the quality of an energy function using the training set. The design of the loss function should be such that its minimization should find an energy function E such that the value(s) of Y for which E attains a minimum corresponds to the correct answer(s). Note that the design of the loss function is independent of what inference algorithm one uses. Whether one is marginalizing over the latent variables, or minimizing over them does not contribute in its design.

4. The last component that must be specified while designing an energy based learning machine is the *learning algorithm*. This is the method for finding the set of parameters W

that minimizes the loss functional over the family of energy functions $\mathcal{E}$, given the training set. Again, how one minimizes over the parameters W does not affect the design of the energy function. Though the shape of the loss function in the parameter space W might be a factor in selecting a learning algorithm. For instance, for convex loss functions one might use second order methods, while for non-convex functions gradient descent type methods might work best.

2 Factor Graphs for Energy Based Models

In the previous chapter we discussed how energy-based models provide more flexibility in the design of the architecture of learning machines over probabilistic models because of lack of normalization. We discussed how learning can be made more efficient when instead of pulling up on the energies of every answer (as in the NLL loss function in probabilistic models), one only pulls up on the energies of the most offending incorrect answer. To this end we introduced a number of "more general" loss functions, besides the NLL loss, whose contrastive term does not involve integrating over the potentially huge space of answer variables. However, there exists many situations where the energy is a function of very large number of variables. In such scenarios the tasks such as marginalization or minimization over the set of hidden variables, or even finding the most offending incorrect answer might be intractable. But when the full energy function (which we call *global energy function*) can be expressed as a sum of a number of *local energy functions*, then one can exploit this structure and design efficient inference algorithms. *Factor Graphs* are a useful tool that can compactly represent these relationships among energy functions. The framework also defines a way of doing inference using an algorithm based on the "message passing" scheme which exploits this structure of the energy function. Factor graphs were first introduced by Kschischang et al. in (Kschischang et al., 2001b), primarily in the light of probabilistic learning. However they can be studied outside the context of probabilistic models and we show that energy-based framework can be applied to them.

We first give a brief introduction to factor graphs for energy-based models in general, and then discuss a "message passing" based algorithm which is used to do efficient inference using them. This is called the *logsumexp-sum* algorithm. We then discuss the various algebras in which algorithms similar to the logsumexp-sum algorithm can be applied. This leads to a number of variants of this algorithm called the *sum-product* algorithm, *min-sum* algorithm, the *max-product* algorithm. We then discuss a general approximate inference algorithm in factor graphs in situations where the above message passing algorithms become infeasible. Finally we shed some light on training in an energy-based factor graph.

2.1 Factor Graphs

More often than not we have to deal with complicated functions of a large number of variables to perform tasks, such as, marginalization over one or a set of variables, or computing the configuration of variables that minimizes or maximizes the value of the function. One approach is to design algorithms which directly deal with the "global" energy function. However, this is likely to be infeasible in most cases, especially when the number of variables is large and the cardinality of each variable is also large or the variables are continuous. Another approach is to exploit the manner in which the "global" energy function can be written as a sum of "local" functions each of which depends only on a subset of variables. In this case the algorithm is split into a number of parts each of which deals with a particular "local" energy function, and hence with only a subset of variables, making the computations efficient. Such factorization can be visualized using *factor graphs*, which is a bipartite graph that expresses which variables are arguments to which local functions. In addition there is a generic algorithm, called the *logsumexp-sum* algorithm, that operates on the factor graph and computes the various marginal functions associated with the global function. This algorithm is based on the message passing regime (Kschischang et al., 2001b).

Consider, for example, the energy function of five variables $E(x_1, x_2, x_3, x_4, x_5)$. Suppose

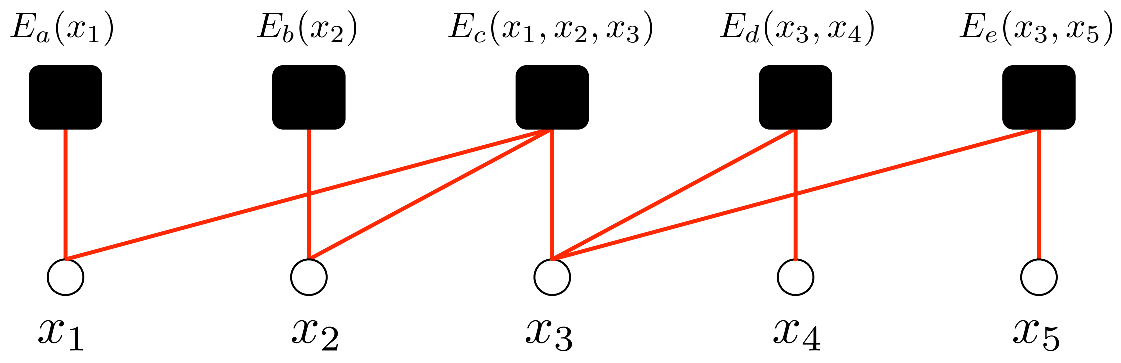


Figure 2.1: *Factor graph representing the factorization* $E(x_1, x_2, x_3, x_4, x_5) = E_a(x_1) + E_b(x_2) + E_c(x_1, x_2, x_3) + E_d(x_3, x_4) + E_e(x_3, x_5)$.

that this function can be written as a sum of five other energy functions, such that

$$E(x_1, x_2, x_3, x_4, x_5) = E_a(x_1) + E_b(x_2) + E_c(x_1, x_2, x_3) + E_d(x_3, x_4) + E_e(x_3, x_5). \quad (2.1)$$

The function E is called the *global energy function* and the functions E_a, E_b, E_c, E_d, E_e are called *local energy functions*. This factorization can be expressed using a factor graph shown in Figure 2.1. This graph consists of two types of nodes. Each node in the shape of a hollow circle corresponds to one variable, and is called the *variable node*. Each node in the shape of a filled square corresponds to one local energy function (a *factor*), and is called *factor node* (also called the *subset node*). An edge in this graph connects the variable node x_i to a factor node E_j if and only if the variable x_i is an argument to the local function E_j.

More formally, let $X = \{x_i : i = 1 \ldots n\}$ be the set of variables associated with the global function. Let $N = \{1, \ldots, n\}$ be the set of indexes of these variables. For each $i \in N$, let x_i take on the value from the set A_i. The set A_i could either be discrete and finite for every i, or it could be continuous. For the moment, in the interest of simplicity, we assume that it is discrete and finite. We later deal with the continuous case. Let S be a non-empty subset of the index set N, and let us denote by X_S the subset of X whose indexes are in S. A particular assignment of values to variables in the set X is referred to as the *configuration* of the variables. The configurations of variables can be viewed as the elements of the cartesian product $T = \prod_{i \in N} A_i$, called the *configuration space*.

Let E be a function that takes the elements of the set X as its arguments. Thus the domain of E is T, and let R be the co-domain of E. We refer to this function $E : T \rightarrow R$, as the *global energy function*. The co-domain R could be any semiring, but for the moment we assume it to be the set of real numbers. Let Q be the collection of the subsets of N, not including the empty set. Suppose the function E can be written as a sum of a number of functions whose arguments are indexed by the elements of Q. That is

$$E(x_1, \ldots, x_n) = \sum_{S \in Q} E_S(X_S). \tag{2.2}$$

For every $S \in Q$, the function E_S is called a *local energy function*. The factor graph corresponding to the equation 2.2 is a bipartite graph with the vertex set $N \cup Q$, and the edge set $\{\{i, S\} : i \in N, S \in Q, i \in S\}$. The nodes which are elements of N (represented by hollow circles) are called *variable nodes*, and the nodes which are elements of Q (represented by shaded squares) are called *factor nodes* or *subset nodes*. There is an edge connecting a variable node i to a factor node E_S if and only if, the index i belongs to the set S. Hence a factor graph can be viewed as a graphical representation of the relation "element of" in NxS.

2.1.1 Factor Graph Examples

We now give some practical examples where the factor graphs are a very useful tool. However the examples we discuss are primarily from probabilistic models, and hence their is a slight difference between the factorization of the global function discussed above and here. In particular, in these models the global function factorizes into a product of local functions rather than the sum of functions. Thus the summation operator in equation 2.2 is replaced by a product operation. Factor graphs has a close relationship with a number of directed and undirected graphical models, such as, Markov Chains, Hidden Markov Models (HMMs), Markov Random Fields, and Bayesian Networks. The key feature of all these models is that they imply a non-trivial factorization of the joint probability mass function. This factorization can be represented in terms of factor graphs and the generic message passing algorithm and its variants can be applied to them, which boils down to applying the forward/backward algorithm, the Viterbi algorithm, the iterative "turbo" decoding algorithm and the belief propagation algorithm.

Markov Random Fields

Consider first the case of Markov random fields, which is an undirected graphical model. Let $V = \{v_1, \ldots, v_n\}$ be the set of vertices of the undirected graph, each of which correspond to a random variable. We know that under very general conditions, such as, positivity of the joint probability density, the joint probability mass function of an MRF can be expressed in terms of a collection of Gibbs potential functions, defined on the set Q of maximal cliques in the MRF

$$p(v_1, \ldots, v_n) = Z^{-1} \prod_{E \in Q} g_E(V_E). \tag{2.3}$$

This equation has the precise structure needed to represent it in terms of a factor graph, where there is one factor corresponding to each maximal clique potential. In fact, in this case the factor graph representation is preferred over the straight forward undirected graph representation, because two distinct factorization can yield the same undirected graph, while it will result in distinct factor graphs.

Bayesian Networks

Bayesian networks are directed graphical models that captures dependencies between a collection of random variables. Each vertex v in a Bayesian network is associated with a random variable. Let $a(v)$ be the set of parent vertices of the vertex v (the set of vertices from which there's an edge to v). Then the joint probability distribution represented by the Bayesian network takes the form

$$p(v_1, \dots, v_n) = \prod_{i=1}^{n} p(v_i | a(v_i)). \tag{2.4}$$

If $a(v_i) = \varnothing$, (v_i has no parents), then we take $p(v_i|\varnothing) = p(v_i)$. Again this factorization can be represented by a factor graph. Furthermore, when the sum-product algorithm is applied to a factor graph corresponding to the Bayesian network the result is the Pearl's "belief propagation algorithm" (Kschischang et al., 2001b).

2.2 Efficient Inference in Factor Graphs

We now informally show how this factorization property of energy functions can be exploited to achieve computationally efficient algorithms for inference. Consider the global energy function $E(x_1, x_2, x_3, x_4, x_5)$. Let all the five variables $x_1, \dots, x_5$ be unknown variables, possibly latent variables. For notational simplicity we do not write the known variables, because their values remain fixed during inference. As discussed in chapter 1, section 1.4, inference would involve marginalization over these variables, which involves computing

$$-\frac{1}{\beta} \log \sum_{x_1,x_2,x_3,x_4,x_5} e^{-\beta E(x_1,x_2,x_3,x_4,x_5)}. \tag{2.5}$$

For small values of β (the inverse temperature) the above marginalization reduces to a minimization operation over the energy function

$$\min_{x_1,x_2,x_3,x_4,x_5} E(x_1, x_2, x_3, x_4, x_5). \tag{2.6}$$

Since β is a constant, for notational simplicity we remove it from the expression. At this point let us define an operator called the *logsumexp* operator and denote it by "$\oplus$". Let $x_1, \dots, x_N$ be a set of variables, and $f(x_1, \dots, x_N)$ be some function of these variables, then the *logsumexp* of the function f with respect to these these set of variables, denoted by $\oplus_{x_1,\ldots,x_N} f$ is defined as

$$\oplus_{x_1,\ldots,x_N} f = -\log \sum_{x_1,\ldots,x_N} e^{-f(x_1,\ldots,x_N)}. \tag{2.7}$$

Consider again the global function of five variables $E(x_1, x_2, x_3, x_4, x_5)$ given by equation 2.1, which is factorized into five local functions. Let each x_i take on values from the discrete and finite set A_i. Thus the domain of E is the set $T = A_1 \times A_2 \times A_3 \times A_4 \times A_5$. Let R be the co-domain of E, which could be, for example, a set of real numbers.The inference procedure for this system involves computing

$$\begin{aligned} F &= -\log \sum_{x_1,x_2,x_3,x_4,x_5} e^{-E(x_1,x_2,x_3,x_4,x_5)} & (2.8) \\ &= \oplus_{x_1,\ldots,x_5} E(x_1, x_2, x_3, x_4, x_5) & (2.9) \end{aligned}$$

Let us assume that each A_i has a cardinality of 10. One way to compute the above marginalization is to directly sum over the exponentials of the global energy function E over the variables x_1, x_2, x_3, x_4, x_5. The number of summations required to achieve this is 10^4. However, if we use the factorization given by equation 2.1 and exploit the distributive law, then the summation over the variables can be broken down in the following way:

$$\begin{aligned} &F \\ &= -\log \sum_{x_1,x_2,x_3,x_4,x_5} e^{-E(x_1,x_2,x_3,x_4,x_5)} \\ &= -\log \sum_{x_1,x_2,x_3,x_4,x_5} e^{-[E_a(x_1)+E_b(x_2)+E_c(x_1,x_2,x_3)+E_d(x_3,x_4)+E_e(x_3,x_5)]} \\ &= \oplus_{x_1,\ldots,x_5} \left(E_a(x_1) + E_b(x_2) + E_c(x_1, x_2, x_3) + E_d(x_3, x_4) + E_e(x_3, x_5)\right) \\ &= -\log \sum_{x_1} e^{-E_a(x_1)} \cdot \left(\sum_{x_2} e^{-E_b(x_2)} \cdot \left(\sum_{x_3} e^{-E_c(x_1,x_2,x_3)} \cdot \left(\sum_{x_4} e^{-E_d(x_3,x_4)} \right) \left(\sum_{x_5} e^{-E_e(x_3,x_5)} \right) \right) \right) \end{aligned} \tag{2.10}$$

We now show how the *logsumexp* operator distributes in a similar way. Let us denote by t the term

$$t = e^{-E_b(x_2)} \cdot \left(\sum_{x_3} e^{-E_c(x_1,x_2,x_3)} \cdot \left(\sum_{x_4} e^{-E_d(x_3,x_4)} \right) \left(\sum_{x_5} e^{-E_e(x_3,x_5)} \right) \right) \tag{2.11}$$

Then the expression in equation 2.10 can be written as

$$\begin{aligned}
F &= -\log \sum_{x_1} e^{-E_a(x_1)} \cdot \sum_{x_2} t && (2.12)\\
&= -\log \sum_{x_1} e^{\log\left(e^{-E_a(x_1)} \cdot \sum_{x_2} t\right)} && (2.13)\\
&= -\log \sum_{x_1} e^{\log e^{-E_a(x_1)} + \log \sum_{x_2} t} && (2.14)\\
&= -\log \sum_{x_1} e^{\log e^{-E_a(x_1)} + \oplus_{x_2} t} && (2.15)\\
&= \oplus_{x_1} \left(E_a(x_1) + \oplus_{x_2} t \right) && (2.16)
\end{aligned}$$

Thus the *logsumexp* operator distributes over sum. Expanding t above and proceeding in the same way, F in equation 2.10 can be written as

$$\begin{aligned}
&F\\
&= -\log \sum_{x_1,x_2,x_3,x_4,x_5} e^{-[E_a(x_1)+E_b(x_2)+E_c(x_1,x_2,x_3)+E_d(x_3,x_4)+E_e(x_3,x_5)]}\\
&= \oplus_{x_1,\ldots,x_5} \left(E_a(x_1) + E_b(x_2) + E_c(x_1,x_2,x_3) + E_d(x_3,x_4) + E_e(x_3,x_5) \right)\\
&= \oplus_{x_1} \left(E_a(x_1) + \oplus_{x_2} \left(E_b(x_2) + \oplus_{x_3} \left(E_c(x_1,x_2,x_3) + \oplus_{x_4} E_d(x_3,x_4) + \oplus_{x_5} E_e(x_3,x_5) \right)\right)\right)
\end{aligned} \tag{2.17}$$

Thus the number of summations in this case is $10 \times 10(10 + 10) = 2000$. These ideas can be generalized to any tree shaped factor graph, and the result is what we call the *logsumexp-sum* algorithm, for efficiently computing the marginals with respect to all the variables. It is based on passing "messages" between the nodes of the graphs along its edges (Kschischang et al., 2001b).

A “message” sent between two nodes can be seen as an appropriate description of the result of the log-sum operation applied to a part of the energy function.

2.2.1 The *logsumexp-sum* Algorithm

The *logsumexp-sum* algorithm works by passing messages between nodes in a distributed manner to compute the log-sum of the global energy function with respect to all the unknown varaibles. To get an intuition behind message passing, imagine that each node of the factor graph has a processor associated with it and the edges are the channels by which these processors communicate with each other. These processors send some appropriate description of the result of the log-sum operator applied to a part of the energy function (the messages) to each other. We first explain a message passing algorithm that computes the log-sum of the energy function E with respect to its unknown variables when the underlying factor graph is does not have cycles. The case where the factor graph has cycles is handled separately.

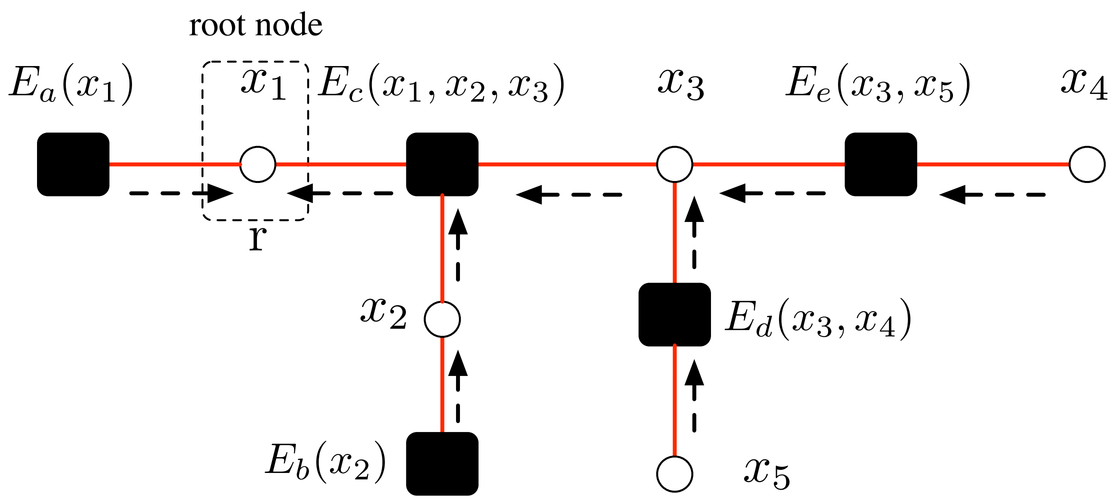


Figure 2.2: *The factor graph of figure 2.1 drawn in the form of a tree with arrows indicating the flow of messages. The node corresponding to the variable x_1 is chosen as the arbitrary root node.*

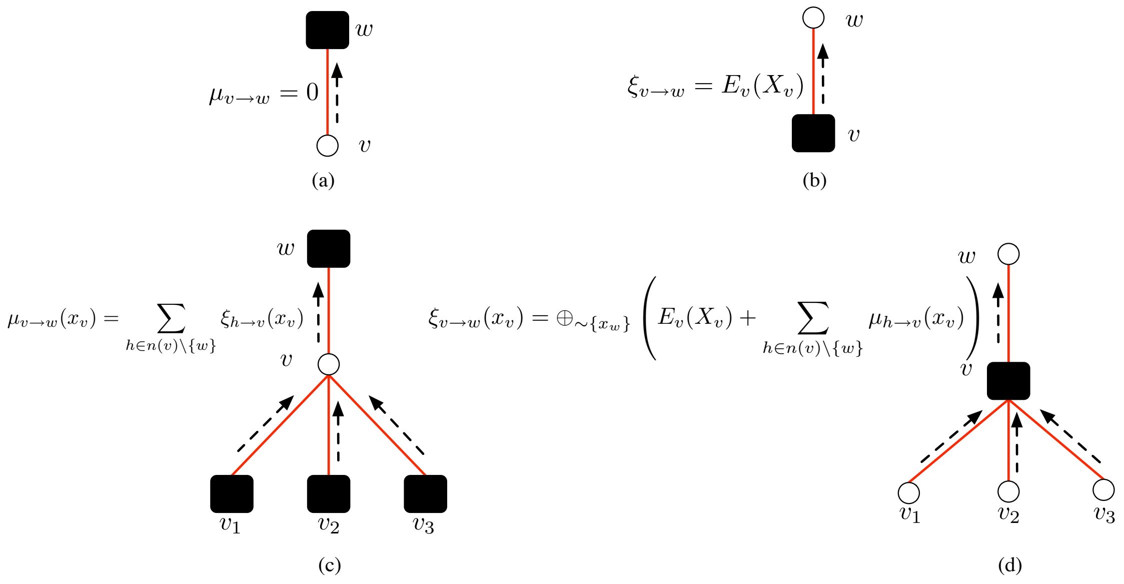
Let us denote by v the variable node corresponding to the variable x_v. Let the cycle-free factor graph be rooted at the node r. In the Figure 2.2 it is the node corresponding the variable x_1. The computations begins at the leaf nodes and the messages are sent upwards from the leaf nodes to the root node. Let the message sent from the variable node to the parent factor node be denoted by μ, and the message sent from the factor node to the variable node be denoted by ξ. If the leaf node v is a variable node then it sends a trivial message, an identity function (additive identity), to its parent w. If the leaf v is a factor node then it sends a description of itself to its parent w. See figure 2.3. Each non leaf node v waits for the messages from all of its children before sending a message to its parent w. If the node v is a variable node, and it has received messages from all of its children, the message sent to its parent w is the sum of messages received from all its children. That is

$$\mu_{v\to w}(x_v) = \sum_{h\in n(v)\setminus\{w\}} \xi_{h\to v}(x_v). \tag{2.18}$$

If the node v is a factor node corresponding to the local energy function E_j, the message sent to its parent w is computed using a two step process. In the first step the function E_j is added to the sum of messages received from its children. The second step involves applying the *logsumexp* operator to the result with respect to all the variables except the variable associated with the parent node w. That is

$$\xi_{v\to w}(x_v) = \oplus_{\sim\{x_w\}} \left(E_v(X_v) + \sum_{h\in n(v)\setminus\{w\}} \mu_{h\to v}(x_v) \right), \tag{2.19}$$

where X_v is the collection of variables associated with the local energy function E_v, and the operator $\oplus_{\sim x_w}$ means that you take a logsumexp with respect to all the variables except x_w. We call this the *logsumexp summary* operator. This is similar to the *summary* operator introduced by Kschischang et al. in (Kschischang et al., 2001b). See figures 2.3 (c) and (d). The computation terminates at the root node r associated with the variable x_r where the final result of the log-sum



w
$\mu_{v\to w} = 0$
v
(a)
w
$\xi_{v\to w} = E_v(X_v)$
v
(b)
w
$\mu_{v\to w}(x_v) = \sum_{h\in n(v)\setminus\{w\}} \xi_{h\to v}(x_v)$
v
v_1
v_2
v_3
(c)
w
$\xi_{v\to w}(x_v) = \oplus_{\sim\{x_w\}}\left(E_v(X_v) + \sum_{h\in n(v)\setminus\{w\}} \mu_{h\to v}(x_v)\right)$
v
v_1
v_2
v_3
(d)

Figure 2.3: *Figure showing the messages passed between various nodes in the factor graph during the logsumexp-sum algorithm.*

operation is obtained by taking the sum of all the messages received at x_r and performing the final logsumexp to the result with respect to x_r.

Note that a message passed on an edge between the variable node x_k and the factor node E_j, either from the variable node to factor node, or from the factor node to the variable node, is a single argument function of x_k, the variable associated with the edge. In fact one can show (Kschischang et al., 2001b) that this message is simply a *logsumexp summary* for x_k of the sum of the local functions descending from the vertex that originates the message. Since the algorithm operates by computing a number of logsumexps and sums, it is called the *logsumexp-sum* algorithm.

2.2.2 Other Algebras

The *logsumexp-sum* algorithm of the previous section allows us to efficiently compute the marginals over the unknown variables of the global energy function expressed as a factor graph. However instead of computing the marginals with respect to the unobserved variables one could also seek to minimize the energy function with respect to them. This constitutes an approximate inference procedure. In addition, one might also want to know the configuration of variables that achieve this value. Variants of the *logsumexp-sum* algorithm can be used to answer these questions.

Min-Sum **and** ***Max-Sum*** **Algorithms**

In the *logsumexp-sum* algorithm we made use of the following distributivity property of the "+" operator over the logsumexp operator $\oplus$

$$a + (b \oplus c) = (a + b) \oplus (a + c) \tag{2.20}$$

In equation 2.5, when $\beta \to -\infty$ the inference formula reduces to a minimization process. That is

$$\begin{aligned} &F \\ &= \min_{x_1,x_2,x_3,x_4,x_5} E(x_1,x_2,x_3,x_4,x_5) \\ &= \min_{x_1,x_2,x_3,x_4,x_5} E_a(x_1) + E_b(x_2) + E_c(x_1,x_2,x_3) + E_d(x_3,x_4) + E_e(x_3,x_5) \end{aligned} \tag{2.21}$$

In order to compute this, one can similarly exploit the distributivity of the "+" operator over $\min$ operator

$$x + \min\{y, z\} = \min\{x + y, x + z\}. \tag{2.22}$$

Thus the computation of F in equation 2.5 becomes

$$\begin{aligned}
&F \\
&= \min_{x_1,x_2,x_3,x_4,x_5} E(x_1,x_2,x_3,x_4,x_5) \\
&= \min_{x_1}\left(E_a(x_1) + \min_{x_2}\left(E_b(x_2) + \min_{x_3}\left(E_c(x_1,x_2,x_3) + \min_{x_4} E_d(x_3,x_4) + \min_{x_5} E_e(x_3,x_5)\right)\right)\right)
\end{aligned} \tag{2.23}$$

In general the messages from the variable node to factor node are

$$\mu_{v\to w}(x_v) = \sum_{h\in n(v)\backslash w} \xi_{h\to v}(x_v). \tag{2.24}$$

Messages from the factor node to the variable node are of the form

$$\xi_{v\to w}(x_v) = \min_{\sim\{x_w\}}\left(E_v(X_v) + \sum_{h\in n(v)\backslash w} \mu_{h\to v}(x_v)\right), \tag{2.25}$$

where the operator $\min_{\sim\{x_w\}}$ involves minimization over all variables except the variable x_w. If the leaf node is a variable node, the message sent to its parent is the additive identity function 0. If the leaf node is a factor node it sends the description of the local energy function associated with it. These modifications lead to what we call the *min-sum* algorithm. When working with the negative energies, the minimization operation is replaced by a maximization operation above and we have the *max-sum* algorithm.

The *min-sum* algorithm can also be seen as computing the shortest path in a trellis graph. For example consider an energy function $E(y_1, y_2, z_1, z_2, z)$ which can be expressed as a sum of four energy functions

$$E(y_1,y_2,z_1,z_2,z) = E_a(x,z_1) + E_b(x,z_1,z_2) + E_c(z_2,y_1) + E_d(y_1,y_2), \tag{2.26}$$

where $y = [y_1, y_2]$ are the output variables and $z = [z_1, z_2]$ are the latent variables.

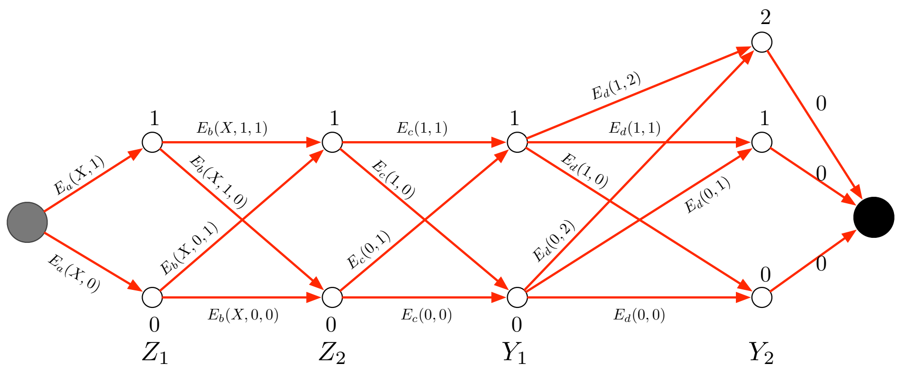


Figure 2.4: *Each possible configuration of z and y can be represented by a path in a trellis. Here z_1, z_2, y_1 are binary variables and y_2 is a ternary variable.*

Assume that the variables z_1, z_2, and y_1 can take two values and y_2 can take three values. The idea is to represent the set of possible configurations in the form of a trellis graph as shown in the figure 2.4. The graph has a single start node and a single end node. The nodes in each of the other column of the graph represents the possible values that the corresponding variable can take. Each edge is weighted by the energy of the factor for the corresponding values of its input variables. A path from the start node to the end node represent one possible configuration of the variables. The sum of the weights along a path is equal to the total energy of that configuration. Hence the inference problem reduces to finding the shortest path, which will be the path with the lowest energy. This can be obtained by dynamic programming methods such as the Viterbi algorithm, or the A* algorithm. The complexity of this algorithm is equal to the number of edges in the graph, which in this case is 16. This is exponentially smaller than the number of paths in general (corresponding to the total number of possible configuration), which in this case is $2 \times 2 \times 2 \times 3 = 24$. Furthermore, if we are only interested in finding the values of z for a fixed y and x, $E(y, x) = \min_{z' \in \mathcal{Z}} E(y, z', x)$, we simply restrict the shortest path algorithm to the edges that are compatible with the given values of y and x.

In many situations we not only seek the minimum value of the energy function but also the configuration of variables which result in such a minimum. This can be achieved by storing the values of variables at each node which results in the minimum value of the associated local energy function. The actual configuration can then be obtained by backtracking from the root to the leaves, collecting the stored values of the variables.

Sum-Product **and** ***Max-Product*** **Algorithm**

From the above discussion and the example, the *logsumexp-sum* algorithm is used to compute the value of F which is of the form

$$\begin{aligned} F &= -\log \sum_{x_1,x_2,x_3,x_4,x_5} e^{-E(x_1,x_2,x_3,x_4,x_5)} && (2.27) \\ &= -\log \sum_{x_1,x_2,x_3,x_4,x_5} e^{-[E_a(x_1)+E_b(x_2)+E_c(x_1,x_2,x_3)+E_d(x_3,x_4)+E_e(x_3,x_5)]}. && (2.28) \end{aligned}$$

Taking the negative exponential of F we get

$$\begin{aligned} & G \\ &= e^{-F}, \\ &= \sum_{x_1,x_2,x_3,x_4,x_5} e^{-[E_a(x_1)+E_b(x_2)+E_c(x_1,x_2,x_3)+E_d(x_3,x_4)+E_e(x_3,x_5)]}, \\ &= \sum_{x_1,x_2,x_3,x_4,x_5} e^{-E_a(x_1)} e^{-E_b(x_2)} e^{-E_c(x_1,x_2,x_3)} e^{-E_d(x_3,x_4)} e^{-E_e(x_3,x_5)}, \\ &= \sum_{x_1,x_2,x_3,x_4,x_5} G_a(x_1)G_b(x_2)G_c(x_1,x_2,x_3)G_d(x_3,x_4)G_e(x_3,x_5), \\ &= \sum_{x_1} G_a(x_1) \sum_{x_2} G_b(x_2) \sum_{x_3} G_c(x_1,x_2,x_3)(\sum_{x_4} G_d(x_3,x_4))(\sum_{x_5} G_e(x_3,x_5)). && (2.29) \end{aligned}$$

where $G_a(x_1) = e^{-E_a(x_1)}$ and so on. This is exactly the *sum-product* algorithm discussed in (Kschischang et al., 2001b). In the general case over a tree based factor graph, the messages passed from the variable node v to its sole parent node which is a factor node w is given by

$$\mu_{v\to w}(x_v) = \prod_{h\in n(v)} \xi_{h\to v}(x_v). \tag{2.30}$$

If the node v is a factor node then the message sent to its solve parent node w, which is a variable node is given by

$$\xi_{v\to w}(x_v) = \sum_{\sim\{x_w\}} \left(G_v(X_v) \prod_{h\in n(v)} \xi_{h\to v}(v) \right), \tag{2.31}$$

where $\sum_{\sim\{x_w\}}$ is the *summary* operator defined in (Kschischang et al., 2001b), and it involves taking the sum over all the variables except x_w. Let R be the co-domain of the function G. The *sum-product* algorithm makes use of the distributivity of the multiplication operator "." over the sum operator "+" on the co-domain R of G. That is

$$a \cdot (b + c) = a \cdot b + a \cdot c \tag{2.32}$$

where $a, b, c \in R$.

The function G, usually in the context of *sum-product* algorithm, represents a joint probability distribution, either normalized or un-normalized. In such a case there are many situation where one seeks the configuration of variables that maximizes this joint probability. The result is the *max-product* algorithm which is identical to the *sum-product* algorithm with the summation operator replaced by the max operator. A slight variant of this algorithm can be used to also return the configuration of variables that maximizes the joint probability. The basis of the *max-product* algorithm is the application of distributive law of the multiplication operator over the $\max$ operator under very mild conditions, on the co-domain R of the function G. That is

$$\max\{ab, ac\} = a \cdot \max\{b, c\}, \tag{2.33}$$

when $a \geq 0$.

This idea is readily generalized to arbitrary tree structured factor graphs. The message passing scheme works in exactly the same was as in the sum-product algorithm. The difference though is in the messages passed. If v is a variable node the message that it sends to its parent w is of the form

$$\mu_{v \to w}(x_v) = \prod_{h \in n(v)} \xi_{h \to v}(x_v), \tag{2.34}$$

where $n(v)$ are the neighboring nodes of v. If however v is a factor node then the message sent to its parent w is

$$\xi_{v \to w}(x_v) = \max_{\sim\{x_w\}} \left(G_v(X_v) \prod_{h \in n(v)} \mu_{h \to v}(v) \right), \tag{2.35}$$

where the operator $\max_{\sim\{w\}}$ is akin to the "summary" operator except it is for the max operator: take the max of the argument over all the variables except w. The final maximization is performed over the product of all the messages arriving at the root node. This gives the maximum value of the global function G. Note that the choice of the root node is arbitrary and the final result does not depend on it.

As shown in (Kschischang et al., 2001b), all the above modifications to the *sum-product* algorithm are actually nothing but the *sum-product* algorithm applied to different semirings. In general, if the co-domain of the global function is any semiring with two operations "+" and "." that satisfies the distributive law

$$a \cdot (b + c) = a \cdot b + a \cdot c, \tag{2.36}$$

$\forall a, b, c \in R$. Then in any such semirings, a product of local functions is well defined, as is the notion of summation of values of g. In addition the "summary" operator is also well defined, and hence the *sum-product* algorithm can be applied to it.

2.3 General Inference in Factor Graph Models

If during the process of inference we are seeking to minimize over the latent variables, we can exploit the knowledge of the structure of the energy function to perform efficient inference using the *logsumexp-sum* type algorithm over the corresponding factor graph. However there may be problem instances where we are required to marginalize over the latent variables. As discussed in section 1.4 of chapter 1 the process of inference (finding the best Y that minimizes the energy) becomes

$$Y^* = \text{argmin}_{y \in \mathcal{Y}} - \frac{1}{\beta} \log \int_{z \in \mathcal{Z}} e^{-\beta E(z,y,X)}, \tag{2.37}$$

which is a conventional energy-based inference in which the energy function has been redefined to

$$\mathcal{F} = -\frac{1}{\beta} \log \int_{z \in \mathcal{Z}} e^{-\beta E(z,Y,X)}. \tag{2.38}$$

This is the *free energy* of the ensemble $\{E(z, Y, X),\ z \in \mathcal{Z}\}$.

For a number of interesting architectures of the energy function E, it is difficult to compute the *free energy* $\mathcal{F}$ because the integral on the left hand side is intractable to compute. Hence one must resort to computing an approximation of it. *Variational Free Energy Minimization* is a method that tries to compute an approximation of the above *free energy*. The idea is to define a class of "easy" surrogate distribution $Q(Z; \theta)$ over the latent variables Z, parameterized with parameters θ, and optimize the parameters θ to search for the instance of the distribution $Q(Z, \theta^*)$ which helps compute an upper bound of the *free energy* function. The class of distributions is "easy" in the sense that operations such as marginalization and expectation over them can be performed easily.

The details of the method are as follows. The *free energy* equation can be written as

$$\mathcal{F} = -\frac{1}{\beta} \log \int_{z \in \mathcal{Z}} Q(Z; \theta) \frac{e^{-\beta E(z,Y,X)}}{Q(Z; \theta)}. \tag{2.39}$$

From Jensen's inequality we know that for a concave function G, the following holds

$$G\left[\int_X Q(X)F(X)\right] \geq \int_X Q(X)G(F(X)), \tag{2.40}$$

when

$$0 \leq Q(X) \leq 1 \quad \forall X \qquad and \quad \int_X Q(X) = 1. \tag{2.41}$$

Thus we have

$$-\log\left[\int_X Q(X)F(X)\right] \leq -\int_X Q(X)\log F(X). \tag{2.42}$$

Combining equations 2.39 and 2.42, we get

$$\begin{aligned}
\mathcal{F} &= -\frac{1}{\beta}\log\int_{z\in\mathcal{Z}} Q(z;\theta)\frac{e^{-\beta E(z,Y,X)}}{Q(z;\theta)} && (2.43)\\
&\leq -\frac{1}{\beta}\int_{z\in\mathcal{Z}} Q(z;\theta)\log\frac{e^{-\beta E(z,Y,X)}}{Q(z;\theta)} && (2.44)\\
&= \int_{z\in\mathcal{Z}} Q(z;\theta)E(z,Y,X) + \frac{1}{\beta}\int_{z\in\mathcal{Z}} Q(z;\theta)\log Q(z;\theta). && (2.45)
\end{aligned}$$

The first term $\int_{z\in\mathcal{Z}} Q(z;\theta)E(z,Y,X)$ in equation 2.45 denoted by $< E(Z,Y,X) >_{Q(Z;\theta)}$ is also called average energy. The second term $\int_{z\in\mathcal{Z}} Q(z;\theta)\log Q(z;\theta)$ denoted by $-TS(Q(Z;\theta))$ is the entropy of the distribution and is also called the inverse temperature.

Thus the upper bound on the *free energy* is given by

$$\mathcal{F} \leq \int_{z\in\mathcal{Z}} Q(z;\theta)E(z,Y,X) + \frac{1}{\beta}\int_{z\in\mathcal{Z}} Q(z;\theta)\log Q(z;\theta). \tag{2.46}$$

Finding an approximate value of the *free energy* then boils down to minimizing the upper bound with respect to the parametric distribution $Q(Z;\theta)$.

2.4 Learning in Factor Graphs

Learning in energy based factor graphs follows the same underlying learning methodology as in a standard energy based model discussed in chapter 1. The first step is to choose the architecture of the energy function E. In this case it could be as complex as any of the factor graphs discussed above that capture the dependencies among different variables. In fact, as we show in the next chapter, the architecture could be complex enough to even capture the dependencies among variables associated with multiple samples. Once chosen, the process of learning involves choosing an appropriate loss function whose minimization will lead to adjusting the parameters of energy function in such a way that inference over it gives the correct answer. Any of the loss functions discussed in chapter 1 and (LeCun et al., 2006) could be used. The decision of choosing a loss function is independent of the choice of architecture of the factor graph (energy function). However there are some guidelines which specify which combinations of loss and energy functions do not work. These are discussed in chapter 6. Choosing a negative log likelihood loss function, for example, will require computing the normalization constant (the partition function), implying some restriction on the choice of the architecture of the energy function. However using contrastive loss functions other than NLL loss does not require computing the normalization constant. Hence one is free to design any complex architecture of the energy function capable of capturing all the dependencies among variables.

In contrast probabilistic models, particularly the undirected models such as Markov Networks, always require the computation of a global normalization constant. Even though the likelihood function in these models can be expressed as a product of factors, the global normalization constraint couples the parameter estimation among factors, ruling out the closed-form solution. For this reason one has to settle for simple functional forms for the potentials, such as, log-linear in parameter space. With log-linear potential functions, the likelihood function becomes concave and can be maximized using gradient ascent based procedures. However a key step in computing the gradient of the likelihood require running the entire inference procedure making the learning inefficient. This is one of the prices one has to pay for the flexibility of global normalization in Markov networks. It is not the case with directed models such as Bayesian networks, where the factors (the conditional distributions) are locally normalized. However the flip side of these networks is that they only capture causal dependencies among variables.

3 FACTOR GRAPHS FOR RELATIONAL REGRESSION

In chapter 2 we showed how one can design efficient inference and learning algorithms when an energy function can be expressed as a sum of local energy functions each taking a subset of variables. We discussed the framework of factor graphs which is appropriate in expressing these relationships among subsets of variables and local energy functions. In this chapter we extend the factor graph framework to deal with problems in statistical relational learning. In particular we propose a factor graph model for the problem of relational regression.

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 $\mathbf{X} = \{X^i,\ i = 1 \ldots N\}$, $\mathbf{Y} = \{Y^i,\ i = 1 \ldots N\}$ to denote the entire collection of input and answer variables. Since we are dealing with regression, each Y^i is a single continuous variable, and each X^i could be a high dimensional vector of continuous variables. A model for relational learning will take as a single input all the training samples along with the relationship structure among them and try to learn the uncertainties present in the data by adjusting the parameters of the model. In energy based framework this translates to having an energy function that takes all the training samples as its input and tries to adjust the parameters so as to produce correct answers for new unseen samples whose relationships to the training samples is known. More formally, when there are no latent variables in the system we have a single global energy function of the form

$$E(W, \mathbf{Y}, \mathbf{X}) = E(W, Y^1, \ldots, Y^N, X^1, \ldots, X^N). \tag{3.1}$$

W is the set of parameters to be estimated by learning. As before, this energy function can be used to derive a conditional distribution over the answer variables using a Gibbs distribution

$$P(\mathbf{Y}|\mathbf{X}, W) = \frac{e^{-\beta E(W,\mathbf{Y},\mathbf{X})}}{\int_{\mathbf{Y}' \in \mathcal{Y}} e^{-\beta E(W,\mathbf{Y}',\mathbf{X})}}, \tag{3.2}$$

where β is a suitably chosen positive constant, and the set $\mathcal{Y}$ is the set of all possible combinations of answer variables $\mathbf{Y}$. Note that this is a huge space of dimension N.

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

$$E(W, \mathbf{Z}, \mathbf{Y}, \mathbf{X}) = E(W, \mathbf{Z}, Y^1, \ldots, Y^N, X^1, \ldots, X^N). \tag{3.3}$$

Again, $\mathbf{Z}$ could possibly be a very high dimensional continuous vector. The conditional distribution in the presence of latent variables can be obtained by marginalizing over them to give

$$P(\mathbf{Y}|\mathbf{X}, W) = \int_{\mathbf{Z}' \in \mathcal{Z}} P(\mathbf{Y}, \mathbf{Z}'|\mathbf{X}, W) = \int_{\mathbf{Z}' \in \mathcal{Z}} \frac{e^{-\beta E(W,\mathbf{Z}',\mathbf{Y},\mathbf{X})}}{\int_{\mathbf{Y}' \in \mathcal{Y}, \mathbf{Z}'' \in \mathcal{Z}} e^{-\beta E(W,\mathbf{Z}'',\mathbf{Y}',\mathbf{X})}}. \tag{3.4}$$

Where $\mathcal{Z}$ is the set of all possible values the latent variables $\mathbf{Z}$ can take. This marginalization reduces down to using equation 3.2 with a new free energy function of the following form (LeCun et al., 2006):

$$E(W, \mathbf{Y}, \mathbf{X}) = -\frac{1}{\beta} \log \int_{\mathbf{Z}' \in \mathcal{Z}} e^{-\beta E(W,\mathbf{Z}',\mathbf{Y},\mathbf{X})}. \tag{3.5}$$

It is easy to show that for large values of β, the above equation can be approximated by

$$E(W, \mathbf{Y}, \mathbf{X}) = \min_{\mathbf{Z}' \in \mathcal{Z}} E(W, \mathbf{Z}', \mathbf{Y}, \mathbf{X}). \tag{3.6}$$

In other words, the marginalization over $\mathbf{Z}$ can be replaced by a simple minimization. This is a practically incontrovertible assumption that energy based models make. This is because when $\mathbf{Z}$ is continuous and very high-dimensional, a marginalizing over it will be impractical. However, even such an assumption might not be sufficient to perform tractable inference because when $\mathbf{Z}$

is high dimensional and continuous, searching for the minimum is not a trivial task. We show later how this problem can be potentially avoided by exploiting the structure associated with the global energy function, if there is any.

Training of this model can be achieved by maximizing the likelihood of the data or by minimizing the negative log conditional probability of the answer variables with respect to the parameters W. This is accomplished by minimizing the negative log likelihood loss $\mathcal{L}$:

$$\mathcal{L}(W, \mathbf{Y}, \mathbf{X}) = E(W, \mathbf{Y}, \mathbf{X}) + \frac{1}{\beta} \log \int_{\mathbf{Y}' \in \mathcal{Y}} e^{-\beta E(W, \mathbf{Y}', \mathbf{X})}, \tag{3.7}$$

where the energy $E(W, \mathbf{Y}, \mathbf{X})$ is given by equation 3.5. 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 given by

$$\nabla_W \mathcal{L}(W, \mathbf{Y}, \mathbf{X}) = \nabla_W E(W, \mathbf{Y}, \mathbf{X}) - \int_{\mathbf{Y}' \in \mathcal{Y}} P(\mathbf{Y}' | \mathbf{X}, W) \nabla_W E(W, \mathbf{Y}', \mathbf{X}) \tag{3.8}$$

Computing the integral in the contrastive term is intractable in practically all interesting problems. A lot of effort is spent in trying to approximate it. Sampling based methods discussed in earlier chapters is one such way. However, energy based models allows the use of more general loss functions, such as the ones discussed in chapter 1 and in (LeCun et al., 2006), which have a variety of contrastive terms. The contrastive terms need not just be the integral over all possible values of answer varaibles.

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 learned 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). When the latent variables are marginalized inference involves minimizing

$$Y^{0*} = \text{argmin}_{Y^0} \left(-\frac{1}{\beta} \log \int_{\mathbf{Z}} e^{-\beta E(W,\mathbf{Z},Y^0,\ldots,Y^N,X^0,\ldots,X^N)} \right). \tag{3.9}$$

For large values of β the integral over Z reduces to a minimization operation. Thus the inference reduces to

$$Y^{0*} = \text{argmin}_{Y^0} \left(\min_{\mathbf{Z}} \{E(W, \mathbf{Z}, Y^0, \ldots, Y^N, X^0, \ldots, X^N)\} \right). \tag{3.10}$$

However there are a number of problems with this system in its present form. First, simply approximating the marginal over $\mathbf{Z}$ with a minimization procedure in equation 3.6 is not sufficient. This is because $\mathbf{Z}$ could be a very high dimensional continuous vector, and hence even minimizing the global energy function with respect to it could be difficult. If the global energy function has some structure associated with it, for instance if it could be expressed as a sum of local energy functions, this problem can be circumvented by taking advantage of this structure, and reducing the potential search space of $\mathbf{Z}$.

Second, when the global energy function is augmented with an additional input X^0, equation 3.10 requires that we search for the value of the latent variables $\mathbf{Z}$ that minimize the augmented energy. Clearly repeating this process for every new test sample is not feasible, since $\mathbf{Z}$ could be a very high dimensional and continuous vector. To avoid this problem the energy function must be carefully constructed in such a way that the addition of a test sample in the arguments will not require retraining the entire system, or re-estimating the high-dimensional hidden variable each time a new test sample comes in.

Third, the parameterization of the energy function must be designed in such a way that its estimation on the training sample will actually result in a good prediction on test samples.

In this chapter we present the framework of *Factor Graphs for Relational Regression* that addresses all of the above problems while solving a relational regression problem. We propose efficient inference and learning algorithm for the model. In addition, by allowing log-likelihood functions that are non-linear in parameter space our framework is flexible enough to capture complex dependencies among sample points.

3.1 Relational Factor Graphs for Regression

From the previous discussion it is clear that the fundamental problem behind both inference and learning is the minimization of the energy function with respect to the hidden variables and other variables. However since the global energy function takes as input the entire set of training samples as its input and the variable $\mathbf{Z}$ could be very high dimensional directly minimizing the energy function might not be practical. However if we can express the energy function as a sum of a number of local energy functions, we can hope to do better. In particular, let us denote by I the index set $I = \{1, \dots, N\}$, which indexes the training samples $\mathbf{X}$ and $\mathbf{Y}$. If S is a non-empty subset of I, we denote by X_S, the set of input vectors of all the training samples whose index is in S, and by Y_S the corresponding outputs of the same training samples. Let Q be a collection of subsets of I (i.e., a subset of the power set of I) and not including the empty set. Suppose that the global energy function can be written as a sum of Q local energy functions, such that

$$E(W, \mathbf{Z}, \mathbf{Y}, \mathbf{X}) = \sum_{S \in Q} E_S(W_S, Z_S, Y_S, X_S), \tag{3.11}$$

where Z_S are the subset of hidden variables ($Z_S \subset \mathbf{Z}$) that are associated with the local energy function E_S, and W_S are the subset of parameters associated with E_S. In general each of the local energy function E_S could take the same set of all the parameters W as its arguments: this is called *parameter sharing* and is a key to learning a relational system. However, with regards to other variables computational advantage is achieved only when the local energy functions take only a subset of these variables as input. With the help of equation 3.11, we can express the model as an energy-based factor graph. When the graph does not have loops we can make use of *min-sum* type algorithm to efficiently minimize the energy function with respect to $\mathbf{Z}$. However when the graph has loops, and dependencies among the variables is complex, one cannot use the *min-sum* algorithm for there is no guarantee that it will converge. Instead one can resort to gradient descent type optimization. Note that the difference between the factor graph discussed this chapter and in chapter 2 is that here we have a single instance of a huge factor graph capturing the dependencies among different samples. The cardinality of the set Q could be as large; as large as the number of training samples or even more. Also, since the variables corresponding to all the samples are fed into the energy function at once and they could interact with each other in complex ways, we need to make inference on all of them simultaneously. This process is called *collective inference* and is essential to relational learning problems.

We now elaborate on these ideas by working with a concrete example. We discuss the relational factor graph used to solve the problem of house price prediction and show how one can make use of the factorization property, and design energy functions in a way that leads to efficient inference and learning algorithms. We emphasize that this framework is general enough and can be applied to any relational regression problems.

3.2 Relational Factor Graph for House Price Prediction

Consider the factor graph shown in figure 3.1. Each sample i (in this case the i-th house) is associated with a set of features X^i, and a price Y^i. Since these variables are observed (given to us as part of the training data) they are represented by shaded circles. We also associate two latent variables, denoted by D^i and Z^i to the i-th house. Since they are un-observed variables they are represented by hollow circles. Let the collection of these variables be denoted by $\mathbf{D} = \{D^1, \ldots, D^N\}$ and $\mathbf{Z} = \{Z^1, \ldots, Z^N\}$. Furthermore, each house i is associated with two local energy function (factors) E^i_{xyz} and E^i_{zz}, represented by shaded squares. The factor

$E^i_{xyz}(W_{xyz}, X^i, Y^i, D^i)$ captures the dependencies among the house-specific variables. Thus it captures the dependence of the price of a house on its individual characteristics, like number of bedrooms, number of bathrooms etc. We say that it is a non-relational factor because the energy function associated with it is not a function of variables associated with other houses. This is a parametric factor with parameters W_{xyz}. These parameters are shared across all the instances of the factors E^j_{xyz}, thus providing parameter sharing in the model, which is essential to robust parameter estimation and meaningful statistical inference in relational learning. Furthermore, when a test sample is added to the model, its non-relational factor E^0_{xyz} simply inherits the shared parameter.

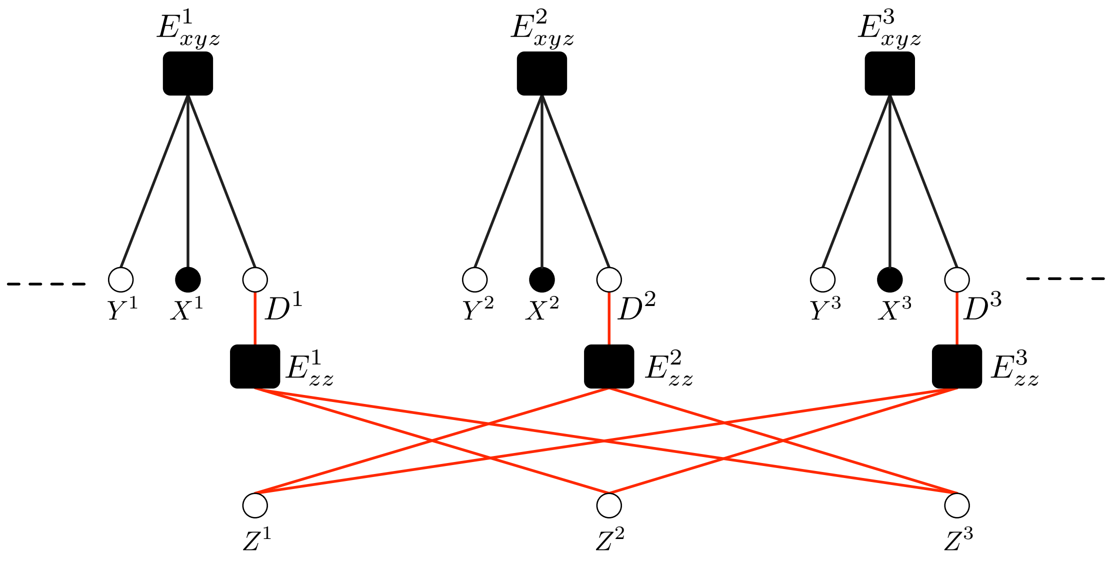


Figure 3.1: *The factor graph used for the problem of house price prediction, showing the connections between three samples. The factors E^i_{xyz} capture the dependencies between the features of individual samples and their answer variable Y^i, as well as the dependence on local latent variables D^i. 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}.*

Let us denote by $\mathcal{N}_i$, the set of indexes of the samples to which the sample i is related. For example, in the case of house price prediction this set will include all the spatio-temporal neighbors of the house i. We call this the neighborhood set of sample i. The second factor $E^i_{zz}(W_{zz}, Z_{\mathcal{N}_i}, D^i)$ captures the dependencies between related samples via a set of latent variables $Z_{\mathcal{N}_i}$. These dependencies influence the answer Y^i for a sample through the intermediary of hidden variable D^i. In the context of house price prediction, D^i can represent a smooth estimate of the desirability of the location of house i, where this estimate is obtained from the desirabilities $Z_{\mathcal{N}_i}$ of the location of the neighboring houses. Since the desirability of a location is generally very similar to nearby locations, the relational structure may be used advantageously to enforce a local smoothness constraint on the Z^i's. Note that there is no direct link between Z^i and the factor E^i_{zz}. This is because using Z^i directly in E^i_{zz} can result in a system that trivially sets the value of each Z^i to the value of output variable Y^i and thus minimize the energy. 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. Again, one way of achieving this is by parameter sharing across all the factors E^j_{zz}. The factor E^i_{zz} is called a relational factor, since it captures the dependencies among different samples.

There are a total of N samples (houses) and each sample has two factors associated with it. Thus the total number of factors is $2N$ and the energy of the system is given by the sum of energies of all the factors

$$E(W, \mathbf{Z}, \mathbf{Y}, \mathbf{X}) = \sum_{i=1}^{N} E^i_{xyz}(W_{xyz}, X^i, Y^i, D^i) + E^i_{zz}(W_{zz}, D^i, Z_{\mathcal{N}^i}). \qquad (3.12)$$

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 third relational factor E^i_{yy} with each sample i and creating the necessary links. The energy of the system will now be given by

$$E(W, \mathbf{Z}, \mathbf{Y}, \mathbf{X}) = \sum_{i=1}^{N} E^i_{xyz}(W_{xyz}, X^i, Y^i, D^i) + E^i_{zz}(W_{zz}, D^i, Z_{\mathcal{N}^i}) + E^i_{yy}(W_{yy}, Y^i, Y_{\mathcal{N}^i}), \tag{3.13}$$

where $Y_{\mathcal{N}^i}$ is the set of answer variables associated with the samples related to the sample i (the spatio-temporal neighbors of the house i).

3.3 Efficient Inference in Relational Factor Graphs

We now discuss two types of inference procedures in the relational factor graph described above. The first type of inference involves finding the maximum-likelihood value of the answer variable Y^0 for a test sample X^0. The second type of inference is the process of finding the values of latent variables $\mathbf{D}$ and $\mathbf{Z}$ that minimize the global energy function $E(W, \mathbf{Z}, \mathbf{Y}, \mathbf{X})$.

Let us first discuss the prediction of the answer variable Y^0 for a test sample X^0. Given the learned parameters W, a perfectly correct way of predicting the value of Y^0 involves creating two new factors E^0_{xyz} and E^0_{zz}, linking the factor E^0_{zz} with the latent variables corresponding to the neighboring training samples, and minimizing the energy with respect to the variables Y^0, $\mathbf{D}$, and $\mathbf{Z}$

$$\min_{y^0, \mathbf{D}, \mathbf{Z}} E(W, \mathbf{Z}, \mathbf{D}, \mathbf{Y}, \mathbf{X}) = \min_{y^0, \mathbf{D}, \mathbf{Z}} \sum_{i=1}^{N} E^i_{xyz}(W_{xyz}, X^i, Y^i, D^i) + E^i_{zz}(W_{zz}, D^i, Z_{\mathcal{N}^i}) \tag{3.14}$$

where y^o is the output variable of the test sample, whose value we seek. Even though the global energy function is expressed as a sum of local energy functions, repeating this process of minimization over the values of latent variables of all the training samples for every test sample 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 to the architecture makes sense intuitively and is not just an engineering hack.

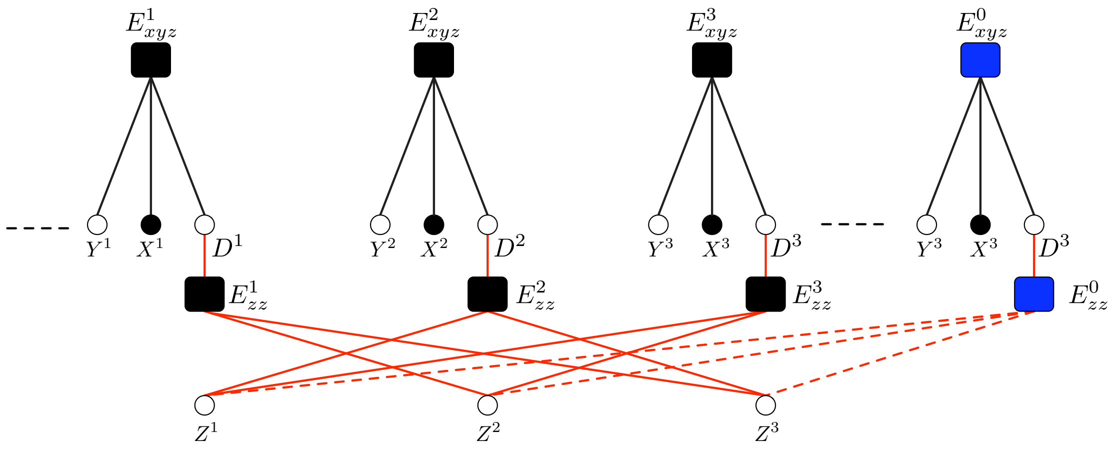


Figure 3.2: *The augmented factor graph with factors corresponding to the test sample X^0 used for inference of Y^0.*

Once we add the factors corresponding to the test sample X^0 (shaded blue in figure 3.2) and make the necessary connections, the variable D^0 whose value we do not know, influences the values of Z^j's belonging to the set $Z_{\mathcal{N}^0}$. This in turn influences the values of the rest of the Z^i's and D^i's. It is this dependence between variables that leads to the expensive optimization with respect to **D** and **Z** during prediction. To alleviate this problem, we make an approximation by fixing the values of $Z_{\mathcal{N}^0}$ while making a prediction. The inference process comes down to minimizing the following expression with respect to Y^0 and D^0 to compute Y^{o*}

$$\min_{Y^0, D^0} E^0_{xyz}(X^0, Y^0, D^0) + E^0_{zz}(D^0, Z_{\mathcal{N}^0}). \tag{3.15}$$

This approximation is not just a simple engineering hack but has an intuitive justification, especially from the context of the house price prediction problem. In the case of house price prediction the latent variable Z^i is interpreted as the actual "desirability" of the location of the i-th house and the variable D^i is interpreted as a smooth estimate of the desirability of the house i from its neighbors. Since we are interested in predicting the prices of any house in future, clearly the D^0 of a test house at some point in the future cannot influence the past "desirabilities" of the location of training samples.

The second kind of inference involves minimizing the global energy function $E(W, \mathbf{D}, \mathbf{Z}, \mathbf{Y}, \mathbf{X})$ with respect to the variables $\mathbf{D}$ and $\mathbf{Z}$. Because of the complex dependency structure of the energy on $\mathbf{Z}$, the factor graph could have loops. Hence algorithms, such as, the *min-sum* algorithm does not guarantee convergence. While using this framework for house price prediction, the problem reduces to solving a linear system. One can show that when the optimization problem with respect to the hidden variables reduces to solving a linear system, solving the system is equivalent to applying belief-propagation algorithm over the factor graph (Wang and Guo, 2006). We resort to gradient descent algorithm to minimize the energy function with respect to both $\mathbf{Z}$ and $\mathbf{D}$, and hence solve the system. 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

$$E^i(D^i, Z_{\mathcal{N}^i}, Y^i, X^i) = E^i_{xyz}(Y^i, X^i, D^i) + E^i_{zz}(D^i, Z_{\mathcal{N}^i}). \tag{3.16}$$

Now consider the minimization of this local energy function with respect to the variables D^i and $Z_{\mathcal{N}^i}$. One approximation that we make here is that instead of minimizing with respect to D^i and $Z_{\mathcal{N}^i}$ simultaneously we first minimize with respect to D^i and then with respect to $Z_{\mathcal{N}^i}$. It is equivalent to doing a coordinate descent first in the direction of D^i and then in the direction of $Z_{\mathcal{N}^i}$. That is we have

$$\begin{aligned}
\min_{D^i, Z_{\mathcal{N}^i}} E^i(D^i, Z_{\mathcal{N}^i}, Y^i, X^i) &= \min_{D^i, Z_{\mathcal{N}^i}} E^i_{xyz}(Y^i, X^i, D^i) + E^i_{zz}(D^i, Z_{\mathcal{N}^i}), && (3.17)\\
&= \min_{Z_{\mathcal{N}^i}} \left(\min_{D^i} E^i_{xyz}(Y^i, X^i, D^i) + E^i_{zz}(D^i, Z_{\mathcal{N}^i}) \right). && (3.18)
\end{aligned}$$

We now proceed to show that if E^i_{xyz} and E^i_{zz} are both quadratic in D^i, then the above process of minimization of their sum with respect to D^i followed by $Z_{\mathcal{N}^i}$ reduces to simply back propagating the gradient with respect to $Z_{\mathcal{N}^i}$, using the above approximation. In particular, we prove the following lemma

Lemma 3.1. *Let* $E^i_{xyz}(Y^i, X^i, D^i) = (G(Y^i, X^i) - D^i)^2$*, and* $E^i_{zz}(D^i, Z_{\mathcal{N}^i}) = (D - H(Z_{\mathcal{N}^i}))^2$*, where* $G(Y^i, X^i)$ *is some function of* Y^i *and* X^i*, and* $H(Z_{\mathcal{N}^i})$ *is some function of* $Z_{\mathcal{N}^i}$*. Then the gradient of the sum of energies of the two factors* $E^i_{xyz}(Y^i, X^i, D^i) + E^i_{zz}(D^i, Z_{\mathcal{N}^i})$*, with respect to* $Z_{\mathcal{N}^i}$ *at the mode of* D^i *is given by* $\nabla_{Z_{\mathcal{N}^i}} E(Z_{\mathcal{N}^i}) = (G(Y^i, X^i) - H(Z_{\mathcal{N}^i}))\nabla_{Z_{\mathcal{N}^i}} H(Z_{\mathcal{N}^i})$*.*

Proof: The mode of D^i is computed by taking the gradient of the sum of the energy functions with respect to D^i and equating it to zero. It is given by

$$D^{i*} = \frac{1}{2}(G(Y^i, X^i) + H(Z_{\mathcal{N}^i})). \tag{3.19}$$

The gradient of the sum of energy functions with respect to $Z_{\mathcal{N}^i}$ at this mode is given by

$$\nabla_{Z_{\mathcal{N}^i}} E(Z_{\mathcal{N}^i}) = -2(D^{i*} - H(Z_{\mathcal{N}^i}))\nabla_{Z_{\mathcal{N}^i}} H(Z_{\mathcal{N}^i}). \tag{3.20}$$

Substituting the value of D^{i*} in the above equation we get the desired result.

$$\nabla_{Z_{\mathcal{N}^i}} E(Z_{\mathcal{N}^i}) = (G(Y^i, X^i) - H(Z_{\mathcal{N}^i}))\nabla_{Z_{\mathcal{N}^i}} H(Z_{\mathcal{N}^i}). \quad \square \tag{3.21}$$

The above lemma essentially means that one can treat the factor E^i_{zz} like a function and directly feed it into the second factor E^i_{xyz}. Thus Figure 3.3(a), now becomes Figure 3.3(b). Since the lemma is true for any sample i, it holds for all the samples. Thus the inference process which involves minimizing the global energy function with respect to **Z** can be accomplished by first treating the factor E^i_{zz} as a function of E^i_{xyz} and then computing the gradient of each of these factors with respect to appropriate Z variables and finally updating all the variables **Z**. Note that each factor influences multiple Z^i's. Furthermore each variable Z^i is an input to factors corresponding to multiple samples, the update will take place only when it has received gradients from all the factors it is an input of. Thus the latent variables are collectively updated. This key idea will be used even for doing efficient learning discussed below and is used while predicting house prices in the next section.

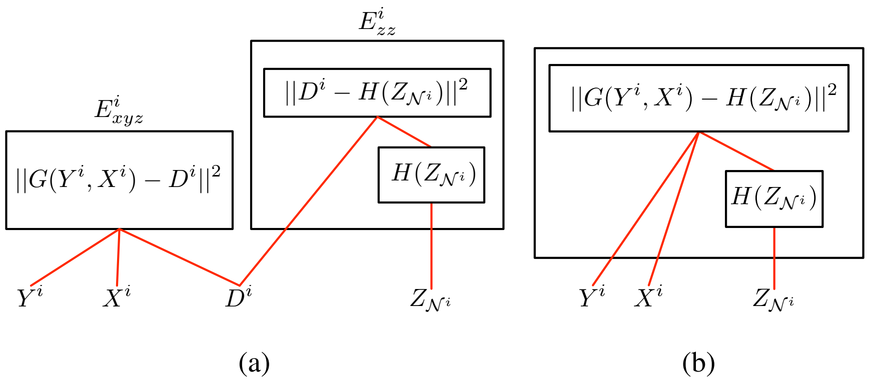


Figure 3.3: **(a)** *Figure showing the connections between the two factors associated with every sample i. (**(b) Top**), However, when the energy of factors E^i_{zz} and E^i_{xyz} is quadratic in D^i, the factor E^i_{zz} can be collapsed into the the factor E^i_{xyz}. In such a situation finding the minimum-energy value of $Z_{\mathcal{N}^i}$ can be performed without explicitly computing the optimal value of D^i. This is done by simply replacing D^i by $H(Z_{\mathcal{N}^i})$ in the factor E^i_{xyz}, and back-propagating the gradients through the function G and H.*

3.4 Efficient Training of a Relational Factor Graph

Let $\mathcal{S}$ be the training samples. Learning in an energy based framework 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. That is, we need to minimize

$$\mathcal{L}(W, \mathbf{Y}, \mathbf{X}) = E(W, \mathbf{Y}, \mathbf{X}) + \frac{1}{\beta} \log \int_{\mathbf{Y}' \in \mathcal{Y}} e^{-\beta E(W, \mathbf{Y}', \mathbf{X})}, \tag{3.22}$$

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

$$E(W, \mathbf{Y}, \mathbf{X}) = -\frac{1}{\beta} \log \int_{\mathbf{D}' \in \mathcal{D}, \mathbf{Z}' \in \mathcal{Z}} e^{-\beta E(W, \mathbf{D}', \mathbf{Z}', \mathbf{Y}, \mathbf{X})}. \tag{3.23}$$

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 3.22) 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 $\mathbf{D}$ and $\mathbf{Z}$ (in the equation 3.23) can be approximated by a simple minimization.

The actual loss function that is minimized is the energy loss and is therefore give by

$$\begin{aligned}
\mathcal{L}(W, \mathbf{Y}, \mathbf{X}) &= \min_{W} - \log \left(\min_{\mathbf{Z}' \in \mathcal{Z}, \mathbf{D}' \in \mathcal{D}} e^{-\beta E(W, \mathbf{Z}', \mathbf{D}', \mathbf{Y}, \mathbf{X})} \right) & (3.24) \\
&= \min_{W, \mathbf{Z}' \in \mathcal{Z} \mathbf{D}' \in \mathcal{D}} - \log \left(e^{-\beta E(W, \mathbf{Z}', \mathbf{D}', \mathbf{Y}, \mathbf{X})} \right) & (3.25) \\
&= \min_{W, \mathbf{Z}' \in \mathcal{Z} \mathbf{D}' \in \mathcal{D}} E(W, \mathbf{Z}', \mathbf{D}', \mathbf{Y}, \mathbf{X}) & (3.26)
\end{aligned}$$

This loss is minimized using the EM-like procedure given in Algorithm 1.

Algorithm 1 RFG-Learn

```
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 phase 1 which involves minimization of the loss with respect to $\mathbf{D}$ and $\mathbf{Z}$. Because of the complex dependency of the energy on $\mathbf{D}$ and $\mathbf{Z}$, minimization is done using the gradient descent method. This is exactly the second type of inference procedure discussed in the previous section on inference hence the same analysis holds here as well. The process boils down to merging the two factors to form a single factor and simply backpropagating the gradients with respect to $\mathbf{Z}$.

In phase 2, minimization of the energy function with respect to W is achieved by gradient descent method. This phase is trivial because the parameters W are shared among all the factors, hence one can sequentially run through all the training samples and update W based on the gradients generated.

4 Spatial Models for House Price Prediction

In this chapter we make the discussions of the previous chapter concrete by instantiating the factor graph for the problem of house price prediction. We specify the architecture of both the relational and non-relational factors in the factor graph, and also give the details of the learning and inference algorithm. This chapter focusses on the problem of house price prediction. The problem of constructing the house price indexes (HPI) is the topic of next chapter. Hence the model discussed here is purely spatial without any temporal component to it. To this end we only use a subset of the full data set available to us. Before discussing the model, we first discuss in detail the industry level dataset that we used to learn our models.

4.1 Dataset

Heterogeneity of homes and the neighborhoods they occupy implies that uncovering the spatial and spatio-temporal patterns in home prices requires a large panel dataset over a large heterogeneous housing area. To this end the dataset was provided to us by First America Corelogic (http://www.facorelogic.com). The dataset has a very heterogeneous set of homes spread over the entire Los Angeles county spanning around 4000 sq miles. In particular there are three distinct tapes that were provided to us. These were the Deeds tape, the 2007 Tax-roll tape and the Arms tape. The Arms tape which contains information on mortgage history and also comes from the counties' registry of deeds was not used in any of our experiments. The Deeds tape is a transaction tape and comes from various counties' registry of deeds. It contains all recorded transactions in LA county from Jan 1984 to Apr 2008. The Tax-roll tape consists of the compiled information obtained from the tax assessor's office and contains detailed information on all homes in LA county irrespective of their transaction history. This information is collected on periodic basis, usually after every few years. All observations in the three tapes have associated with them a 10 digit APN number which uniquely identifies a house. It is therefore possible to combine the three tapes and obtain a rich history of mortgage and transaction information, and home and neighborhood specific attributes for all houses in LA county. We now discuss the Deeds and Tax-roll tape in some detail. For the spatial model of the present chapter only a subset of the dataset was used which we will describe later.

4.1.1 The Full Deeds Tape

The deeds tape records all the transactions that took place between the periods of Jan 1984 and April 2008. Note that there's a difference between a transaction and a house. A house could be transacted multiple number of times during a particular period. The deeds tape records the information relevant to these transactions. Each record (transaction) has a total of 116 fields associated with it, having information relevant to the identity of the property associated with the transaction, the address of the property, and the sale information of the transaction, such as, sale amount, mortgage amount, lender information etc. There are a total of $13,527,413$ transactions with valid APNs in the tape. Out of the 116 fields, those that were relevant are

- Sale Price: the price at which the property was sold in the present transaction,
- Recording date: the date on which the transaction was recorded,
- Sale date: the actual sale date of the property,
- Transaction Category Code: variable indicating whether it was a full market transaction (*arms length transaction*), or less than full market transaction (*non-arms length purchase*), or transfer of ownership rights without the sale (*non-arms length non-purchase*),

- Transaction Type Code: variable indicating whether the transaction was a resale, refinance, or nominal,

- Document Type Code: variable recording the type of the deed. It could either be *grant deed* (the most common deed type used to transfer property from seller to buyer), *deed of trust* (stand alone mortgage) or *quite claim* (sale of less than full value).

In our experiments (especially in the next chapter) we only consider arms length transactions that are resales or new constructions with either a grant deed or foreclosure document type.

4.1.2 The Tax-Roll Tape

A Tax-Roll tape gives a detailed description of a house independent of how many times it has been transacted. This information is collected periodically by the tax assessor's office. First American Corelogic provided us with two tax-roll tapes, one from the year 2004 and one for 2007. Among the various property types in LA county, single family residences (SFR), condominiums, duplexes, apartments, and planned unit development (PUD), only single family residences were considered in both the experiments. The tape provides a detailed description of the characteristics and the location information of every property using 198 fields.

In addition to the basic home level data, such as, the number of bedrooms, number of bathroom, land area, living area, and the year built, the tape provides information about a wide range of other characteristics. Among others, these include effective year built (the last year in which the property had the same physical characteristics as it has in the year corresponding to the tax-roll tape), condition (the state of the property: good, poor, fair etc), quality (refers to the luxuries and amenities available in the home), style (conventional, contemporary, spanish etc), number of fire places, garage type (carports, detached garage etc), parking spaces, whether there's a pool or not, type of heating, type of air conditioning, and number of stories among others.

The tax-roll tape also provides extensive locational information associated with every property. The mailing addresses provided for all properties were used to extract the GPS coordinates for every house using the software purchased from www.geolitics.com. The GPS coordinates are used in identifying the spatial neighbors for every house. Other locational information provided by the tax-roll tape include census tract (census tract number in which the property lies), school district (school district number in which the property lies), view (whether the property has a city view, or lake view etc), location influence (whether the house located at the corner of a street, or cul-de-sac etc) etc.

4.1.3 Other Neighborhood Fields: Census Tracts and School District

Two fields in the tax-roll tape, namely the *census tract number* and the *school district number* are used to further enhance the locational (neighborhood) information associated with every property. This process involves taking the census tract number and the school district number for every house, extracting the local information on demographics, and education from census databases and school district databases respectively and appending these features to the existing features associated with that house.

A census database is the single-most exhaustive source of local information on demographics, income, employment and education all of which affect quality of a neighborhood and hence house prices. This information is collected every ten years at the census tract level. A "Census tract" is a small, relatively permanent statistical subdivisions of a county or statistically equivalent entity delineated by local participants as part of the U.S. Census Bureau's Participant Statistical Areas Program. The primary purpose of census tracts is to provide a stable set of geographic units for the presentation of decennial census data. When first delineated, census tracts were designed to be relatively homogeneous with respect to population characteristics, economic status and living conditions. The spatial size of census tracts varies widely depending on the density of settlement. We used data from the 2000 census at which point LA county had 2054 census tracts with a median population of 4484. In particular we used primarily four sets of variables from the census data. The first variable captures the average commute time to work by people living in the neighborhood. The underlying assumption is that if the average travel time from a neighborhood to work is high, then on average the neighborhood is relatively far away from work places and hence the demand of houses over there is low. The variable that captures this information is "average time to commute". The second variable that we extract is the "median household income" of the census tract. Clearly there is a strong correlation between the prices of houses in an area and the average income of people in that area. Third variable is the "proportion of units occupied by owner". The assumption is that an area where majority of homes are occupied by the owner is a family oriented neighborhood and hence is more desirable. Finally the census tape also provides detailed information on educational attainment of the resident population in that census tract.

However the only information relating to eduction that we use comes from the school district tape. The field that we use is called the "academic performance index" which is a single number that summarizes the performance of the schools in that area. Unfortunately the effort towards collecting the information on the performance of school district was only started after the "No Child Left Behind" program. Hence we do not have this information for early years. As a result this information is used only in the experiments in this chapter which only uses the transactions from the year 2004. It is not used while constructing the house price indexes in the next chapter which uses the transactions from all the years.

4.1.4 Data Subset Used for Spatial Model

This chapter discusses the spatial model to predict the prices of real estate properties. There is no temporal component to this model. Hence as mentioned before we use only a subset of the data that was provided to us. In particular we only use all the single family transactions in the deeds tape that took place in the year 2004. The tax-roll tape used is also of the year 2004. Even this small subset of the data is fairly heterogeneous and spans evenly over the entire LA county area.

Of the numerous house specific attributes 18 of them were used in the experiments. These were the living area, year built, number of bedrooms, number of bathrooms, pool or no pool, prior sale price, parking spaces, parking types, lot acerage, land value, improvement value, percentage improvement value, new construction, foundation, roof type, heat type, and site influence. The address of the property was used to get its GPS coordinates which were also appended to the house specific attributes. This set of house specific attributes for the i-th house are denoted by X_h^i in the rest of the chapter. In order to collect the neighborhood specific attributes, we used the 2000 census tape and school district tape. For each census tract in our database, we used data on median household income, proportion of units that are owner-occupied, and information on the average commuting time to work. Finally, we used the school district field for each home to add an academic performance index (API) to the database. These variables along with the GPS coordinates of the location of the house are collectively denoted by X_{nb}^i in the rest of the chapter. Also let us denote by X_{gps}^i as just the GPS coordinates of the house i.

Minimal pre-processing and data cleaning was done to this dataset. All the records which had one or more fields missing were removed. All the fields corresponding to the price/area/income variables were mapped into log space. All the non-numeric discrete variables, such as pool, parking type etc, were coded using 1-of-K coding. In addition all the variables were normalized so as to have a 0 mean and a standard deviation between -1 and 1. After the pre-processing there were a total of 6 fields in the X_h^i variables and 14 fields in X_{nb}^i variables. In the end we were left with a total of 42025 transactions, spanning 1754 census tracts and 28 school districts. Since we are interested in predicting the prices of houses in the future, the transactions were sorted according to their sale dates. The first 90% of them (37822) were treated as training samples, out of which the later 10% were used as a validation set. The remaining 10% (4203) were treated as testing samples.

4.2 Factor Graphs for House Price Prediction

We now give details of the factor graph that was used for the problem of house price prediction, by first giving the architecture of the relational and non-relational factors and giving the details of the learning and inference algorithms.

We model the price of a house as a product of two components. The first component, which we call its "intrinsic price", is a function of only house specific variables X_h^i. The second component of the price is the estimate of the "desirability" of the neighborhood in which the house lies. This estimate of the "desirability" depends on the features of the neighborhood. Some of these neighborhood features are measurable and are given by X_{nb}^i. 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 and the "desirability" of the houses that constitute the neighborhood. Since this "desirability" of a location is not directly measurable it is modeled as a latent variable. In the factor graph, each house (sample) i is associated with one latent variable Z^i, the value of which is interpreted as the "desirability" of the location of the house. The variable D^i can be interpreted as a smooth estimate of the desirability of that house, which depends on the Z^j's associated with its spatial neighbors. In order to avoid a bias towards expensive houses, 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.

Each house is assigned two factors, one of which is non-relational E_{xyz}^i and the other one is relational E^i_{zz}. The non-relational factor E^i_{xyz} captures first of the two dependencies, which is between the log price of the house and its individual characteristics (including an estimate of its desirability). This is done by first estimating the log of "intrinsic" price from the house specific features X^i_h, using a parametric model $G(W_{xyz}, X^i_h)$, and then measuring the discrepancy of the the total estimated log price – log "intrinsic" price + the log of estimated "desirability" D^i – from the log of actual price Y^i. Thus the factor E^i_{xyz} corresponding to house i takes the form $E^i_{xyz}(Y^i, X^i, d^i) = (Y^i - (G(W_{xyz}, X^i_h) + D^i))^2$. There is no restriction on the architecture of the parametric function $G(W_{xyz}, X^i_h)$. It could either be a linear function or a non-linear function. In the present experiments it was a two hidden layer fully connected 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 second factor associated with every house is the relational factor E^i_{zz} which captures the influence of other "related" houses on the price of the i-th house. Here the relationship between two houses is governed by their spatial proximity. The related houses influence the price of a house through the latent variables Z^j's (which are the "desirabilities"of their location) associated with them. Thus this factor is able to capture the hidden dependencies among variables. This is modeled in the following way. Let $\mathcal{N}^i$ be the set of houses that we think are related to the i-th house and will influence its price. These are the K houses that have the smallest euclidean distance in the GPS space. The value of K is chosen via the process of validation. The factor E^i_{zz} takes as input the latent variables Z^j's associated with these neighboring houses, denoted by $Z_{\mathcal{N}^i}$. In addition it also takes as input the variable D^i associated with the house i which is interpreted as a smooth estimate of the desirability of the i-th house from the desirabilities of the neighboring houses $Z_{\mathcal{N}^i}$. Note that in line with the discussion of the previous chapter, 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 $H(X^i_{nb}, Z_{\mathcal{N}^i})$.

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 the other.

Kernel Based Interpolation

One way of estimating the desirability is by simply taking the weighted average of the Z^j's of the neighbors, where the weights are given by some smooth kernel. In this case the function H takes the form

$$H(X^i_{nb}, Z_{\mathcal{N}^i}) = \sum_{j\in\mathcal{N}^i} Ker(X^i_{nb}, X^j_{nb})Z^j. \tag{4.1}$$

The kernel function could be any smooth kernel, such as, a cubic kernel or an exponential kernel. In the present experiments it is the exponential kernel

$$Ker(X^i, X^j) = \frac{e^{-q||X^i-X^j||^2}}{\sum_{k\in\mathcal{N}^i} e^{-q||X^i-X^k||^2}}. \tag{4.2}$$

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

Weighted Local Linear Regression

Another way of estimating the desirability involves fitting a weighted local linear model on the set of samples $\mathcal{N}^i$, with the weights given by a kernel similar to the one discussed above, 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

$$(\beta^*, \alpha^*) = \arg\min_{\beta,\alpha} \sum_{j\in\mathcal{N}^i} (Z^j - (\beta + \alpha X^j_{nb}))^2 Ker(X^i_{gps}, X^j_{gps}). \tag{4.3}$$

The function $Ker(X, X^j)$ could be any appropriate smooth kernel, e.g. the one given in equation 4.2. The solution to the system (or the output of the function $H(X^i_{nb}, Z_{\mathcal{N}^i})$) is given by

$$H(X^i_{nb}, Z_{\mathcal{N}^i}) = \beta^* + \alpha^* X^i_{nb}. \tag{4.4}$$

Remark 4.1. *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*

$$H(X^i_{nb}, Z_{\mathcal{N}^i}) = \sum_{j \in \mathcal{N}^i} a^j Z^j, \tag{4.5}$$

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

The relational 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 the sum of energies of the two factors, is given by

$$\begin{aligned} E^i(W, Z^i, Y^i, X^i) &= E^i_{xyz}(Y^i, X^i, D^i) + E^i_{zz}(D^i, H(X^i_{nb}, Z_{\mathcal{N}^i}) & (4.6) \\ &= (Y^i - (G(W_{xyz}, X^i_h) + D^i))^2 + (D^i - H(X^i_{nb}, Z_{\mathcal{N}^i}))^2. & (4.7) \end{aligned}$$

Since both the energy terms on the right hand side are quadratic functions of the variable D^i, following the discussions of the previous section and the lemma 3.1, the second factor can be treated as a function of the first and can be included in the first factor directly. Thus in effect we have a single factor E^i associated with each house i.

The exact architecture of each local energy function (the factor E^i) is given in Figure 4.1. The 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 house. Again we emphasize that other than differentiability with respect to W_{xyz}, no restriction is imposed on the form/architecture of this function. The second component $H(X^i_{nb}, Z_{\mathcal{N}^i})$ takes the non-parametric form discussed

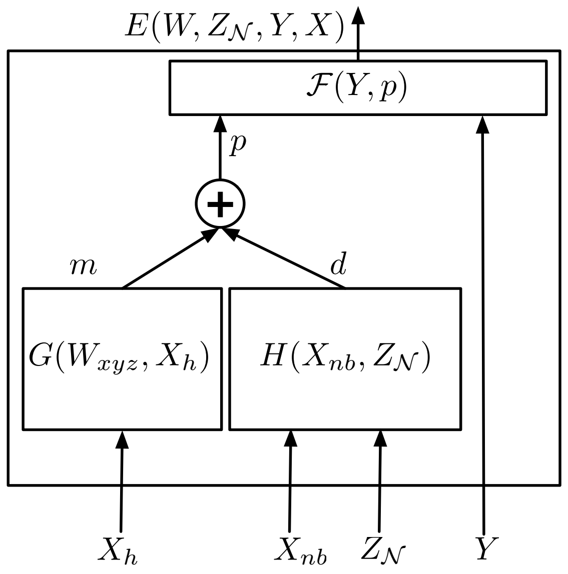


Figure 4.1: *The architecture of the factor associated with each sample. It consists of two trainable components: the parametric function G_W and the non-parametric function H.*

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 house. 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 discrepancy between log of the actual price Y^i and the combined outputs $G(W_{xyz}, X_h^i) + H(X_{nb}^i, Z_{\mathcal{N}^i})$. The function $\mathcal{F}$, which is the local energy associated with each factor, is given by

$$E^i(W, Z^i, Y^i, X^i) = \mathcal{F} = (Y^i - (G(W_{xyz}, X_h^i) + H(X_{nb}^i, Z_{\mathcal{N}^i})))^2. \tag{4.8}$$

The global energy function of the system is given by

$$E(W, \mathbf{Z}, \mathbf{Y}, \mathbf{X}) = \sum_{i=1}^{N} E^i(W, Z^i, Y^i, X^i) \tag{4.9}$$

$$= \sum_{i=1}^{N} (Y^i - (G(W_{xyz}, X_h^i) + H(X_{nb}^i, Z_{\mathcal{N}^i})))^2. \tag{4.10}$$

4.2.1 Learning

The system can be trained by 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 chapter 2, the process of marginalization over the hidden variables can be approximated with a minimization operation. Further more, when the energy is a quadratic function of $\mathbf{Y}$, the contrastive integral term of the loss function 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 $\mathbf{Z}$, where the energy of the system is given by equation 4.10. Thus the loss function is

$$\mathcal{L}(W, \mathbf{Z}) = \sum_{i=1}^{n} \frac{1}{2} (Y^i - (G(W_{xyz}, X_h^i) + H(Z_{\mathcal{N}^i}, X_{nb}^i)))^2 + R(\mathbf{Z}), \tag{4.11}$$

where $R(\mathbf{Z})$ is a regularizer on $\mathbf{Z}$. In the experiments it was an L_2 regularizer. 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. When thinking in terms of probabilities, the regularizer can be viewed as a prior over the hidden variables. Then minimization of the above loss function is equivalent to seeking a *maximum aposteriori estimate* (MAP) of $\mathbf{Z}$.

Minimization of the 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 with respect to the hidden variables $\mathbf{Z}$. It turns out that the above loss is quadratic in $\mathbf{Z}$ and its minimization reduces 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 Z^j, while computing the value of function H. For example, when the kernel based interpolation is used, the coefficients is the value of the kernel (equation 4.2), and when local linear regression model is used, the coefficients are a^j's in equation 4.5. Thus the function $H(Z_{\mathcal{N}^i}, X^i)$ can now be written as

$$H(Z_{\mathcal{N}^i}, X^i_{nb}) = \mathbf{Z} \cdot U^i. \tag{4.12}$$

The loss now becomes the following sparse quadratic program

$$\mathcal{L}(\mathbf{Z}) = \frac{1}{2}\sum_{i=1}^{N}(Y^i - (G(W_{xyz}, X^i_h) + \mathbf{Z} \cdot U^i))^2 + \frac{r}{2}||\mathbf{Z}||^2. \tag{4.13}$$

Note that the value of the function $G(W_{xyz}, X^i_h)$ is fixed in this phase. Clearly this loss is quadratic in $\mathbf{Z}$.

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 $\mathbf{Z} \cdot U^i$ of the desirability of training sample X^i to the learnt desirability Z^i. Note that the estimate $\mathbf{Z} \cdot U^i$ for the house i does not involve the term Z^i. Rather it is estimated from Z^j's of its neighbors. Hence the loss function now becomes

$$\mathcal{L}(\mathbf{Z}) =$$
$$\frac{1}{2}\sum_{i=1}^{N}(Y^i - (G(W_{xyz}, X_h^i) + \mathbf{Z} \cdot U^i))^2 + \frac{r_1}{2}||\mathbf{Z}||^2 + \frac{r_2}{2}\sum_{i=1}^{N}(Z^i - \mathbf{Z} \cdot U^i)^2. \quad (4.14)$$

Here r_1 and r_2 are some constants governing how much weight needs to be given to each term. This loss function is still a sparse quadratic program.

This system 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 $G(W_{xyz}, X)$. This is achieved 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 the house specific attributes X_h^i through the function G to produce the log of the intrinsic price $M^i = G(W_{xyz}, X_h^i)$. Estimate the log of the desirability of the location of sample i by interpolating from its neighbors in the training set, using the function $H(Z_{\mathcal{N}^i}, X_{nb}^i)$. Add the estimated log desirability to the log of the intrinsic price to get the prediction $Pr^i = G(W_{xyz}, X_h^i) + H(Z_{\mathcal{N}^i}, X_{nb}^i)$. Compare the predicted value Pr^i with the actual value Y^i to get the energy

$$E^i(W, Z_{\mathcal{N}^i}, Y^i, X^i) = \frac{1}{2}(Y^i - Pr^i)^2 \tag{4.15}$$

$$= \frac{1}{2}(Y^i - (G(W_{xyz}, X_h^i) + H(Z_{\mathcal{N}^i}, X_{nb}^i)))^2 \tag{4.16}$$

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.

4.2.2 Inference

The process of inference on a new house X^0 is trivial. It involves first computing its neighboring houses in the training set, and using the learnt values of their hidden variables $Z_{\mathcal{N}^0}$ to get an estimate of its "desirability" through the non-parametric function H passing the house specific features X_h^0 through the learnt function $G(W_{xyz}, X_h)$ to get its "intrinsic" price, and adding the two to get its predicted price in log domain.

4.3 Experiments

An extensive experimental study was performed, using the subset of the dataset described above, in which we compared the relational factor graph model with a number of other models. Unfortunately all these models are non-relational in nature. Since we are not aware of any work on relational regression till date, we did not have any benchmark statistical relational model to compare with. Nevertheless the models that we compare against are those that have been traditionally used in literature for the problem of house price prediction.

4.3.1 Non-Relational Models

The non-relational models that we compared with were regularized linear regression, a fully connected neural network, k-nearest neighbors, and weighted local linear regression.

Regularized Linear Regression

We first tried two models that completely ignores any neighborhood information associated with the problem and only look at the features associated with every sample. The first of those was the regularized linear regression, in which one tries 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

$$R(W) = \frac{r}{2}||W||^2 + \frac{1}{2N}\sum_{i=1}^{N}(Y^i - W^T X^i)^2. \tag{4.17}$$

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

Fully Connected Neural Network

The second 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 which could not be captured by the linear regression model. 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.

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 X_{gps}. Best performance was observed when the first measure was used. Experiments were done with different values of K and the one for which it worked best was 90. The results reported use this value.

Weighted Local Linear Regression

Like the nearest neighbour method another method that somewhat takes into account the neighbors of a house while making a prediction in the weighted local linear regression method. The motivation behind using this model is to exploit the local structure and improve upon prediction. In weighted local linear regression models, in order to make a prediction for a sample X^0, a separate weighted linear regression model is fitted using only those training samples that are neighbors to the sample X^0. The neighbors 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^0, if $\mathcal{N}^0$ gives the indices of the neighbouring training samples, then the loss function that is minimized is

$$\min_{\beta(X^0)} \sum_{i \in \mathcal{N}^0} K_\lambda(X^0_{nb}, X^i_{nb})[Y^i - \beta(X^0) f(X^i)]^2. \tag{4.18}$$

In the above loss $\beta(X^0)$ are the regression parameters that needs to be learned, $f(X^i)$ is some polynomial function of X^i, and $K_\lambda(X^0, X^i)$ is an appropriately chosen kernel with parameter λ. In the experiments it was the exponential kernel given in equation 4.2. Once minimized, the prediction Pr^0 of the sample X^0 is given by

$$Pr^0 = \beta(X^0) \cdot f(X^0). \tag{4.19}$$

Different number of neighbors were tried in the experiment. Best result obtained was using 70 nearest neighbors.

4.3.2 Relational Factor Graph

The relational factor graph model can be seen as a hybrid version of the above two classes of models (those that only take into account the individual features, and those that only take into account the neighborhood data). One difference between relational factor graph model and those discussed above which take the neighborhood information into account (k-nearest neighbors and local linear regression), is that these models do not perform any form of collective learning. While make a prediction of a sample they assume that features of neighbors are fixed. Whereas in relational factor graph model, the "desirabilities" of houses, which influence each other, are learnt collectively.

The model is a combination of two components one parametric $G(W, X_h^i)$ and the other non-parametric $H(X_{nb}^i, Z_{\mathcal{N}^i}^i)$. The parametric model only takes into account the individual features of the house. While the non-parametric takes into account the neighborhood information, including the features of the neighboring houses. This component can be seen as learning a latent manifold of "desirabilities" Z^j's over the geographic area. The value of the manifold at any point is obtained by interpolation on the coefficients Z^j's associated with the neighboring training samples that are local to that point according to some distance measure. Thus clearly the relational factor graph model combines the benefits of both the classes of models and comes up with a prediction accuracy which is better than either of the individual models. This is evident from the results table 4.1.

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. The best values of the number of neighbors, regularization coefficients and other hyper-parameters were obtained via the process of validation. The results are reported for the best combination of the values.

4.4 Results and Discussion

The performance of the systems was measured in terms of the Absolute Relative Forecasting error (fe) (Do and Grudnitski, 1992). 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

$$fe_i = \frac{|A_i - Pr_i|}{A_i}. \tag{4.20}$$

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. Furthermore, minimizing the square difference between the actual price and the predicted price in log space is equivalent to minimizing the Absolute Relative Forecasting error in original space.

4.4.1 The Desirability Maps

The discussion in this section 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 4.2, shows the color 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

Table 4.1: *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**

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.

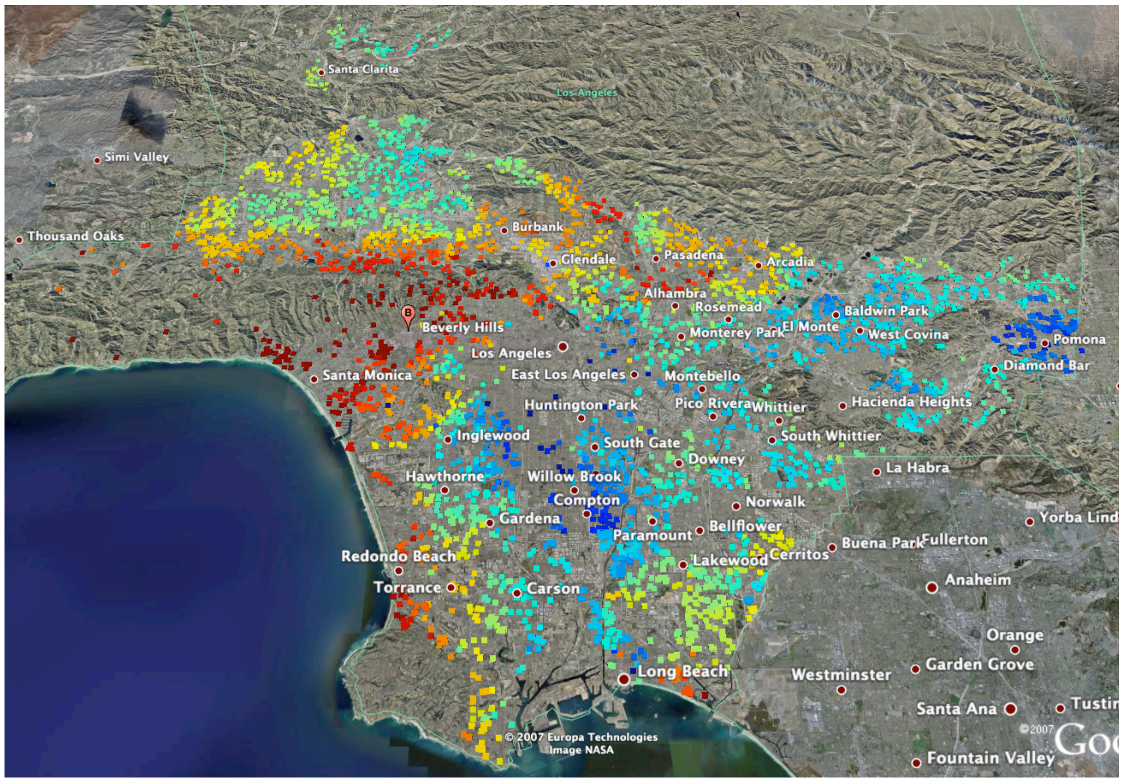


Figure 4.2: *The learnt hidden "desirability" manifold over the Los Angeles county area. Each point in the map correspond to a unique test sample and is color coded according to the "desirability" of its location. The red color correspond to high desirability and the blue color correspond to low desirability.*

5 Spatio-Temporal Models for House Price Index Construction

Studying the way by which house prices move over time is imperative towards understanding the dynamics of the real estate market. One needs a precise and dependable measure that can capture these movements. Because of various factors, such as, heterogeneity of houses, infrequency of sale, and certain unexplained circumstances influencing individual buyers and sellers, one cannot expect to capture the trend on individual properties. Rather one resorts to capturing an aggregate trend of a group of "similar" houses. A *House Price Index* (HPI) is one such way of encoding this trend. This aggregation could be at multiple levels. For instance, we could have a city wide index, or a zip-code level index or even an Metropolitan Statistical Areas (MSA) level index.

There are two primary ways in which people have tried to compute an HPI in economics. When one uses only the features associated with the individual houses, the result is the so called *Hedonic Price Index*. The hedonic price index, though popular among academics, is not widely used in industry because of various reasons. One reason is the lack of theoretical justification behind any functional form used. Another involves lack of standardization of the housing characteristic data across municipalities. Another related issue is the difficulty in gathering data on all the relevant characteristics of a home and its neighborhood. This data may either not be recorded/available or just too costly to obtain. Finally, depending on how the index is computed, hedonic models either assume constant marginal prices over time or run pair-wise regression for consecutive time periods to compute an index. The later approach selectively ignores large chunks of information.

Another type of indices which are by far the most widely used are the *Repeat Sales Index* and are based on the repeat (previous) sale prices of houses. A repeat sales index obviates the need to deal with many issues associated with the hetrogeneous nature of houses by assuming that the underlying unit has not gone through any physical changes. In particular the most widely used repeat sales index is the S&P Case-Shiller index, due to Case and Shiller (Case and Shiller, 1989). However there are two major problems with this way of index construction. First, it completely ignores the rich spatio-tempral structure that is associated with the problem. In particular these methods treat each transaction independently of others and do not consider the influence of neighboring transactions in space and time on the current transaction. The importance of the use of such spatio-temporal information while computing the index is further emphasized by the various striking and superfluous error patterns that show up with this method, particularly the Case-Shiller repeat sales method. For instance, prediction errors are very high on homes that have very low or very high initial prices. Highly segmented nature of housing markets and the fact that homes with high and low prices are typically clustered in certain areas, suggests that while constructing the index if, in addition to taking into account the repeat sales of a particular house, one also considers the sale prices of the neighboring transactions both in the dimensions of space and time, one can expect to generate a better index that is free from such artificial patterns. Second, most methods proposed so far focus towards estimating the parameters efficiently and precisely. Little effort is spent on building models that simultaneously generates a price index and also accurately predicts the prices of new unseen houses using this index. This is important because for almost any non-trivial use of a housing index, the underlying hope is that the index can be used to give a reasonable first approximation to the current house values, and therefore be relevant to predict prices of homes that have not yet been sold but may be so in a short period of time.

In this chapter we extend the relational factor graph framework discussed in chapter 3 and 4 to construct house price indices that captures the spatio-temporal structure associated with the problem. The result is a model with superior prediction accuracy. In addition to the repeat sales of houses, the model also takes into account the single sales, and the hedonics associated with every house. Rather than treating each transaction independently of others, it tries to capture the influence of the neighboring transactions in space and time on the prices of every transaction. This is achieved by learning a latent "normalized price" manifold over space and time, akin to the "desirability" manifold of the previous chapter. The underlying characteristic of this manifold is its spatio-temporal smoothness. The model is applied to a highly diverse industry standard dataset, where it simultaneously constructs the house price indices and learns the "normalized price" manifold.

To the best of our knowledge, this is the first attempt to simultaneously construct the house price index and build a prediction model by making use of the associated spatio-temporal structure. The results show that the price index estimated is very similar to the Case-Shiller's index. Though the indices are the same, the proposed model has a far superior prediction accuracy. Furthermore, unlike with Case-Shiller's model, there are no artificial patterns in the error structure. Rather these patterns are absorbed by the spatio-temporal latent surface. Since the two indices behave in nearly the same manner, the stark difference between the performances of the two models points towards the importance of understanding and using the strong relational structure which is inherent in the problem.

5.1 Dataset

Heterogeneity of houses and the neighborhood in which they lie necessitates the need for a highly diverse dataset spanning a large heterogeneous area, while constructing a good house price index which also makes use of the spatio-temporal dependencies among transactions. To this end we used all the valid transactions that were available to us in the deeds tape (see chapter 4) starting from January 1984 to April 2008. There were a total of $14,713,346$ recorded transactions in the LA county over this period, out of which $13,527,413$ transactions had a valid APN (a 10 digit number identifying the underlying property). Note that a transaction is different from an actual physical property. Each transaction is associated with some physical property, and is a record of the sale/re-sale/transfer of that property at some period of time. Thus multiple transactions could correspond to the same underlying property.

Among the various variables in the deeds tape, for every transaction we extracted information about its *Sale Price, Recording Date, Sale Date, Transaction Category Code, Transaction Type Code, and Document Type Code*. See Chapter 4 for a complete description of these variables. From the tax-roll tape the house specific variables selected were the APN number, the Land square footage, Living square feet, and the mailing address. The mailing address was used to extract the GPS coordinates for every house. The list of variables chosen is deliberately small for two reasons. As informed by our data source, other parts of the US do not have as rich a tax-roll tape as LA. Thus any model that has to be generalized for other parts of the country should not depend too much on the detailed house characteristics. Secondly, the purpose of this exercise was to show that explicitly modeling the relational structure inherent in the problem can lead to better understanding of the house prices, Hence one must be able to show significant performance gain without overly depending on other house specific variables.

Since our model is compared to the Case-Shiller repeat sales index, the data cleaning process was designed to be as close as possible to the cleaning process used by S&P Case Shiller index. Following are the steps that were taken in this direction.

- Using the fields selected from the deeds tape, only those transactions were kept for which Transaction Category Code was equal to arms length transactions, Transaction Type Code was either re-sales or new constructions, and Document Type Code was either grant deed or foreclosure document type.

- All the transactions which had one or more attributes missing were removed. This rule was also applied to the tax-roll fields. In particular, if a house had a missing GPS coordinate or Land square footage, or Living square feet, it was removed from the dataset.

- Transactions on the same house with same sale date and different sale price were removed. Transactions on the same house with same sale date and same sale price were all but one removed.

- Repeat transactions for houses that happen too quickly (within less than 7 months of each other) are removed. The underlying assumption is that these houses undergo a structural change and hence is not a valid repeat sales transactions. We call them the *dirty repeat sales transaction*. Furthermore, transactions on a home before and after this dirty transaction are de-coupled. One easy way to visualize the entire dataset is as a collection of sale prices (transactions). Each transaction is either a stand-alone transaction (single sales) or it is a repeat sales transaction in which case it is explicitly tied to some previous transaction (sales price).

- If for a repeat sales transaction pairs, the value of *effective year built* of the underlying property is between the two sale dates, the pair is removed. Repeat sales transaction pairs for which either of the sale prices is less than $\$5000$ and more than $\$100,000,000$ are removed. Repeat sales transaction pairs that have annualized returns less than -50% or greater than 100% are removed. The repeat sales transaction pairs that our perform or under perform the median house price index by more or less than 25% respectively on an annualized basis are also removed.

- Finally among all the single sales transactions, those transactions for which sale price is greater than the maximum transaction price in any repeat sales is removed. All the single sale transactions whose sale price is less than the minimum transaction price in any repeat sales are also removed.

After the above cleaning process we were left with $591,239$ repeat sales transactions and $367,973$ single sales transactions, making a total of $1,550,451$ transactions. Again, this dataset is highly diverse consisting of transactions from around 24 years spanning the entire Los Angeles County with 2054 census tracts and 28 school districts.

5.2 Spatio-Temporal Latent Manifold Index Model

A *House Price Index* can be viewed as an aggregate measure over a collection of houses, that can capture the movements of real estate properties. For instance, one could have a city wide price index which encodes the common trend exhibited by all the houses in that city at that particular time. The primary input to any model designed to compute the index, is the information about the transactions associated with various houses over a period of time. Each transaction has an associated time, a price, and an underlying real estate property with it. A property could be transacted multiple number of times over a particular period, during which it might or might not have undergone any structural change. In the rest of the chapter we shall talk in terms of transactions (which are identified by the underlying property and time of sale), as opposed to talking just in terms of properties, which was the case in previous chapter.

We model the price associated with a transaction as a combination of two quantities. The first is a city wide index and the second is a "normalized price" associated with the underlying property at the time of the transaction. The index is modeled as a scalar coefficient and can be viewed as a parameter shared among all the transactions that took place at that particular time. The time resolution we work with is one month. Thus there is one city wide index for every month.

The “normalized price” of the underlying property of a transaction at a particular time can also be viewed as a house specific deviation from the global index at that time. There are primarily three aspects to it. First, it will depend on the features that are specific to the underlying property. Second, the “normalized price” is also depends on whether the neighborhood in which the house lies is desirable to live in or not. Some of the features that define the desirability of a neighborhood, such as the median household income, or the performance of the schools in that area can be directly measured. However there are a number of features which cannot be measured directly and are only reflected in the prices of neighboring houses. The sale price of a house will also depend on the time of sale. A house sold during a boom time will have a premium on it as opposed to during slump time. The influence of time on the “normalized price” also cannot be directly measured and is merely reflected in the prices of nearby houses sold around that period. Thus the “normalized prices” of houses which are spatially and temporally close are related and influence each other. Since these “normalized prices” themselves are not given to us as part of the data they are modeled as latent variables. We have one latent variable associated with every transaction whose value can be interpreted as the “normalized price”. These latent variables have a strong underlying structure associated with them, namely the spatio-temporal smoothness. Discounting the features specific to a house, the “normalized price” will have a gradual change when moving from one location to a nearby location. Furthermore, this change is also gradual when going from one time period to the next. Thus one can view the “normalized prices” of various houses over time as a 4-dimensional latent manifold over space and time. The smoothness constraint over these prices is exploited to learn the shape of this latent manifold. Lastly, the “normalized price” of a house at a particular time is also influenced by the previous “normalized price” of that house, if that house had a previous transaction. This is in line with the repeat sales methodology.

We extend the relational factor graph framework of the previous chapter to capture the relationships among the "normalized prices" of transactions that are spatio-temporal neighbors of each other. In addition the factor graph also captures the dependence of the price of every transaction on the global city wide monthly index. Since the dependencies among variables is highly complex, leading to loops in the factor graph we resort to gradient based algorithms for learning. The learning algorithm proposed efficiently learns both the global index and the 4-dimensional latent "normalized price" surface simultaneously, while improving the prediction accuracy. Inference for a new transaction sometime in the future consists of two steps. In the first step we estimate the global index for that time period using an Autoregressive model. We use an AR(1) process. The second step involves finding the transactions in the training set that are spatio-temporal neighbors to it and interpolating them to get an estimate of the "normalized price" of the new transaction. Finally the two estimates are combined to get the price of the new transaction. All these ideas are made formal in the next and the following sections.

5.2.1 The Spatio-Temporal Factor Graph

Let there be a set of M transactions spanning a total of P time periods. Let the collection of these transactions be denoted by $\mathcal{Q} = \{Q^1, \ldots, Q^M\}$, and $\mathbf{T} = [T^i, \ldots, T^P]'$ denote the vector consisting of time periods. Since the unit of time in the present case is one month, each T^i corresponds to a month. At the same time let there be a set of N underlying properties, with X^i denoting the features associated with property i. Let the collection of these properties in vector form be denoted by $\mathbf{X} = [X^1, \ldots, X^N]'$. Each transaction $Q^i[j,k]$ is indexed by two numbers $[j,k]$, associating property X^j and time period T^k to transaction Q^i. Let the collection of all such indexes be denoted by the set $\mathcal{I}$. Furthermore, each transaction Q^i is also associated with an indicator variable $R^{(j,k)}$ which equals 0 if there is no past transaction of the underlying property X^j, and 1 if there is a past transaction of X^j. Let $Z^{(j,k)}$ denote the "normalized" price of the property X^j at time period T^k associated with the transaction $Q^i[j,k]$. For the purpose of notational simplicity we will denote the transaction $Q^i[j,k]$ by $Q^{(j,k)}$ where ever required in the rest of the chapter. The feature vector X^i associated with the i-th house is decomposed into three components $X^i = [X^i_h, X^i_{nb}, X^i_{gps}]$. X^i_h denotes the house specific features, such as, living area, number of bedrooms etc, X^i_{nb} denotes the observed neighborhood features, such as, median household income etc, and X^i_{gps} denotes the GPS coordinates of the house. In the experiments discussed below we only used the living area of the property as the house specific feature X^i_h and no neighborhood features was used. Finally, with each time period T^k we associate a learnable coefficient C^k which is interpreted as the global house price index. The collection of these indices in vector form is denoted by $\mathbf{C} = [C^1, \ldots, C^P]'$.

The price $p^{(i,j)}$ of the transaction $Q^{(i,j)}$ is modeled as a product of two terms. The first term is the global city wide index C^j, and the second term is the "normalized" price $Z^{(i,j)}$ of the underlying property at that time. However, as before, we work in the log domain and thus the log of the price is the sum of the log of the global index and the log of the "normalized price" of the house. With a slight abuse of notation however, we shall denote by C^j the log of the index, and $Z^{(i,j)}$ the log of the "normalized price", and $p^{(i,j)}$ as the log of the price. In fact in the rest of the chapter we implicitly assume that everything is in log domain, unless otherwise stated. Thus we have

$$p^{(i,j)} = C^j + Z^{(i,j)}. \tag{5.1}$$

One can interpret the "normalized price" as the house specific deviation from the global trend exhibited by all the houses in the geographic area (which in the experiments below is the entire city). This "normalized price" is dependent on three quantities. First, it depends on the house specific features, second it depends on prices and hence the "normalized prices" of the transactions that are spatially and temporally close to it, and third it depends on the price of the previous transaction of the underlying property, if there exists one. When the underlying property X^j of transaction $Q^{(j,k)}$ has previously been sold, we say that the transaction $Q^{(j,k)}$ is a *repeat sales* *transaction*.

The Non-Relational Factor

We extend the factor graph framework of chapters 3 and 4 to capture these relationships among samples. Let us denote by $\mathbf{Y} = [Y^1, \ldots, Y^M]'$ the vector of actual prices associated with the transactions. Thus for a transaction $Q^{(i,j)}$ the corresponding price is denoted by $Y^{(i,j)}$. The structure of the factor graph is shown in figure 5.1. We assign two factors $E_{xyz}^{(i,j)}$, and $E_{zz}^{(i,j)}$ to every transaction $Q^{(i,j)}$. Note that the assignments of factors is according to the transaction and not to the individual houses. The first factor $E_{xyz}^{(i,j)}$ captures the dependencies between the individual price of a transaction, the estimated "normalized price" of the underlying property at that time $D^{(i,j)}$, and the global index at that time period. As before $D^{(i,j)}$ is the intermediary hidden variable via which the "normalized prices" of the spatio-temporal neighbors influence the price of the current transaction. It represents a smoothed estimate of the "normalized price" of the current transaction, where the estimate is computed from the "normalized prices" of these neighbors. Thus the factor $E_{xyz}^{(i,j)}$ takes the form

$$E_{xyz}^{(i,j)} = [Y^{(i,j)} - (C^j + D^{(i,j)})]^2. \tag{5.2}$$

The Relational Factor

The second factor $E_{zz}^{(i,j)}$ is relational in nature and captures the influence of the "related" transactions on the price of the current transactions. In the proposed model relationships among transactions are defined in a specific way. The transaction $Q^{(k,l)}$ belongs to the relationship set of $Q^{(i,j)}$ if their underlying houses X^k and X^i are geographically close to each other, and the time T^l of the transaction $Q^{(k,l)}$ is close to the time T^j of the transaction $Q^{(i,j)}$, and $T^l \leq T^j$. We say that $Q^{(k,l)}$ is a spatio-temporal neighbor of $Q^{(i,j)}$. However, note that while considering the transaction $Q^{(k,l)}$ the transaction $Q^{(i,j)}$ will not be considered as its spatio-temporal

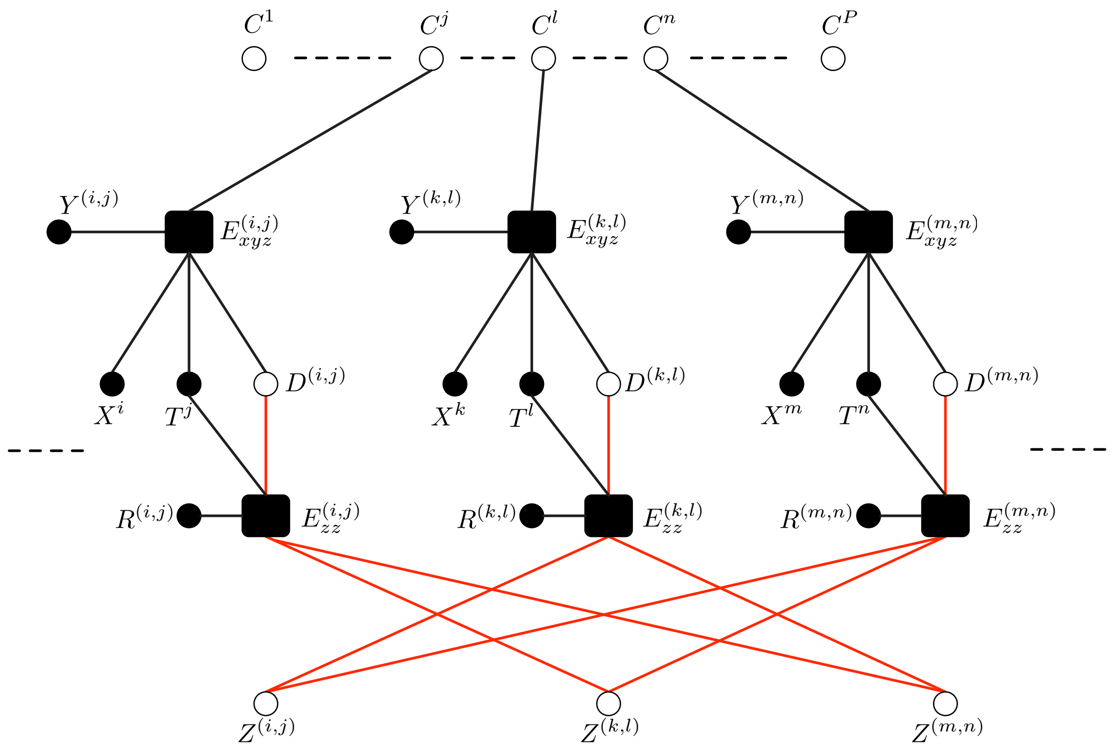


Figure 5.1: *The relational factor graph used for constructing house price indices using a spatio-temporal latent "normalized" price surface.*

neighbor because $T^j \geq T^l$. In other words, we only look in the past to get the temporal neighbors of a transaction. The spatial and temporal closeness between two transactions is given by two hyper-parameters which are learnt through validation. The related transactions influence the price of the current transaction through their "normalized prices" $Z^{(i,j)}$, which themselves are latent variables. In particular, let the set $\mathcal{N}^{(i,j)}$ be a collection of two quantities. First, it contains the set of indices corresponding to transactions that are spatio-temporal neighbors of the transaction $Q^{(i,j)}$. Second, it contains the index (i, j') of the previous transaction of the house X^i (the repeat sales), if there exists one. A previous transaction is only considered if it has taken place within some threshold time in the past. The threshold is learnt during the process of validation.

Let $Z_{\mathcal{N}^{(i,j)}} = \{Z^{(k,l)} : (k,l) \in \mathcal{N}^{(i,j)}\}$ denote the collection of the "normalized prices" of the corresponding transactions (neighbors plus the repeat sales if there is one). The factor $E_{zz}^{(i,j)}$ takes as input the latent variables $Z^{(k,l)}$ (the "normalized prices") associated with these neighboring transactions. It also takes as input the local latent variable $D^{(i,j)}$, via which the $Z^{(k,l)}$'s of the neighboring houses influence the price of the current transaction. This factor also takes as input the observed variable $R^{(i,j)}$, which helps identify whether the underlying property of the transaction has a previous sale or not. In line with the discussion of chapter 3 the factor $E_{zz}^{(i,j)}$ does not takes as input the variable $Z^{(i,j)}$, in order to avoid a trivial solution. Instead the normalized price of the transaction $Q^{(i,j)}$ is estimated from the set of neighboring "normalized prices" $Z_{\mathcal{N}^{(i,j)}}$ using the non-parametric function $H(X^i, T^j, R^{(i,j)}, Z_{\mathcal{N}^{(i,j)}})$. One defining characteristic of the function H is that it ensures spatio-temporal smoothness over the "normalized prices". It performs a smooth interpolation over the "normalized prices" of the neighboring transactions to get an estimate of the "normalized price" of the current transaction. This process can be seen as constructing a smooth latent manifold in space and time. We now discuss in detail the form of the function H.

Depending on whether the transaction is a repeat sales or not, the function H can take two forms. When it is not a repeat sales, the set $Z_{\mathcal{N}^{(i,j)}}$ only contains the spatio-temporal neighbors and $R^{(i,j)} = 0$. The estimate of the "normalized price" is obtained by fitting a weighted local linear model on the set of transactions in $Z_{\mathcal{N}^{(i,j)}}$, with the weights given by some appropriate kernel in space and time. The learnt parameters from the local linear model, along with the house specific characteristics are used to get the value of H. More formally, let α and β respectively be the parameters and the bias of the model. We find the value of α^* and β^* by fitting a linear model over the neighbors using the house specific characteristics

$$[\beta^*, \alpha^*] = \text{argmin}_{\alpha,\beta} \sum_{(k,l)\in\mathcal{N}^{(i,j)}} (Z^{(k,l)} - (\beta + \alpha X_h^k))^2 \left(\frac{\mu(X^i_{gps}, T^j, X^k_{gps}, T^l)}{\sum_{(k,l)\in\mathcal{N}^{(i,j)}} \mu(X^i_{gps}, T^j, X^k_{gps}, T^l)} \right). \tag{5.3}$$

$\mu(X^i_{gps}, T^j, X^k_{gps}, T^l)$ is the smoothing kernel over the GPS coordinates of the underlying houses and the time periods of the transactions. Note that the model is fitted over the house specific characteristic of the neighbors. Once the parameters are computed, the value of the function H is given by

$$H(X^i, T^j, R^{(i,j)}, Z_{\mathcal{N}^{(i,j)}}) = \beta^* + \alpha^* X^i_h. \tag{5.4}$$

From remark 4.1, the function H can be expressed as a linear combination of the "normalized prices" $Z^{(k,l)}$ of the spatio-temporal neighbors

$$H(X^i, T^j, R^{(i,j)}, Z_{\mathcal{N}^{(i,j)}}) = \sum_{(k,l)\in\mathcal{N}^{(i,j)}} a^{(k,l)} Z^{(k,l)}, \tag{5.5}$$

where the coefficients $a^{(k,l)}$ are independent of the values of $Z^{(k,l)}$

When the current transaction is a repeat sale, in addition to containing the spatio-temporal neighbors the set $Z_{\mathcal{N}^{(i,j)}}$ also contains the previous transaction of the underlying property X^i, which is given by $Q^{(i,j')}$. $T^{j'}$ denotes the time period of the previous sale of the property. Here $R^{(i,j)} = 1$. In this case the function H is a linear combination of two terms. The first term is a contribution from the spatio-temporal neighbors of the transaction. The second term is a contribution from the previous sale of the underlying property. Let $\lambda(T^1, T^2)$ be a kernel function defined over time periods. The general property of λ is that its a decreasing function in $|T^1 - T^2|$. Then the function H is given by

$$H(X^i, T^j, R^{(i,j)}, Z_{\mathcal{N}^{(i,j)}}) = (1 - \lambda(T^j, T^{j'})) \left(\sum_{(k,l)\in\mathcal{N}^{(i,j)}} a^{(k,l)} Z^{(k,l)} \right) + \lambda(T^j, T^{j'}) Z^{i,j'}. \tag{5.6}$$

Combining the two cases given by equations 5.5 and 5.6, the function H is given by

$$H = \begin{cases} \sum_{(k,l)\in\mathcal{N}^{(i,j)}} a^{(k,l)} Z^{(k,l)} & \text{if} \quad R^{(i,j)} = 0 \\ (1-\lambda(T^j, T^{j'})) \left(\sum_{(k,l)\in\mathcal{N}^{(i,j)}} a^{(k,l)} Z^{(k,l)}\right) + \lambda(T^j, T^{j'}) Z^{i,j'} & \text{if} \quad R^{(i,j)} = 1 \end{cases}$$

The only components of the function H remaining to be specified are the spatio-temporal kernel $\mu(X^i_{gps}, T^j, X^k_{gps}, T^l)$, and the temporal kernel $\lambda(T^j, T^{j'})$. Let $d(X^i_{gps}, X^j_{gps})$ denote the spatial distances (in miles) between the two properties X^i and X^j, and $r(T^i, T^j) = |T^i - T^j|$ denote the temporal distance between the two time periods T^i and T^j. Then the spatio-temporal kernel μ is given by

$$\mu(X^i_{gps}, T^j, X^k_{gps}, T^l) = \max\left[0, 1 - \frac{d(X^i_{gps}, X^k_{gps})}{\bar{d}} - \frac{r(T^j, T^l)}{\bar{r}}\right]. \tag{5.7}$$

Here $\bar{d}$ and $\bar{r}$ are the parameters of the kernel and are fixed exogenously. They are computed by the process of validation. One can interpret $\bar{d}$ as the maximum distance in miles one is willing to go to look for the spatial neighboring houses. Likewise $\bar{r}$ can be interpreted as the maximum distance in time we are willing to go back in the search of temporal neighbors. Thus the kernel gives more weight to the transactions whose underling properties are spatially close and whose time of sale is temporally close. It is a linear kernel and assumes that the spatio-temporal contours are straight lines. The simplicity of the kernel is in line with the aim of this chapter, which is to show the importance of just taking the relationships among samples into account during learning and inference in relational problems. One is free to chose any non-linear kernel in the general case.

The temporal kernel λ is given by

$$\lambda(T^i, T^j) = \max\left[0, 1 - \frac{|T^i - T^j|}{\bar{T}}\right], \tag{5.8}$$

where $\bar{T}$ is another fixed parameter whose value is obtained from the process of validation. It can be interpreted as the maximum possible distance that one is willing to look back in time in search of a previous transaction of the underlying property. The kernel gives more weight to the previous sale which was closer to the current sale in time as opposed to the sale which was further back in time. Again, its simplicity (linearity) is motivated by the aim of the chapter. However one is free to choose any non-linear kernel.

The energy associated with the relational factor $E_{zz}^{(i,j)}$ is given by

$$E_{zz}^{(i,j)} = (D^{(i,j)} - H(X^i, T^j, R^{(i,j)}, Z_{\mathcal{N}(i,j)}))^2. \tag{5.9}$$

The total energy corresponding to every transaction is

$$\begin{aligned} E^{(i,j)} &= E_{xyz}^{(i,j)} + E_{zz}^{(i,j)} && (5.10) \\ &= (Y^{(i,j)} - (C^j + D^{(i,j)}))^2 + (D^{(i,j)} - H(X^i, T^j, R^{(i,j)}, Z_{\mathcal{N}(i,j)}))^2. && (5.11) \end{aligned}$$

Since the local energies of the two factors are quadratic functions of $D^{(i,j)}$, using lemma 3.1 we can merge the two factors and treat the second factor as a function of the first. Thus the energy corresponding to the single factor associated with every transaction is given by

$$E^{(i,j)} = (Y^{(i,j)} - (C^j + H(X^i, T^j, R^{(i,j)}, Z_{\mathcal{N}(i,j)})))^2. \tag{5.12}$$

Figure 5.2 shows the reduced factor graph and figure 5.3 shows the architecture of the factors in this factor graph. Each factor consists of two trainable components. The first component is a switch module which, depending on the value of the variable T^j, makes a connection between the variable C^j and the upper layers. The C^j's encode the global index and one can view them as learnable parameters. The second component is the non-paramtric function H over space and time, that estimates the "normalized" price of the current transaction from the "normalized" prices of its spatio-temporal neighbors. This component models the smooth latent price manifold over the given geographic area and the specified time interval. The outputs of the two components are added (since they are in the log domain) to get an estimate of the total price of the transaction. Finally, the discrepancy between the actual price $Y^{(i,j)}$ and the predicted price $(C^j + H(X^i, T^j, R^{(i,j)}, Z_{\mathcal{N}(i,j)}))$ is measured to get the energy associated with the factor.

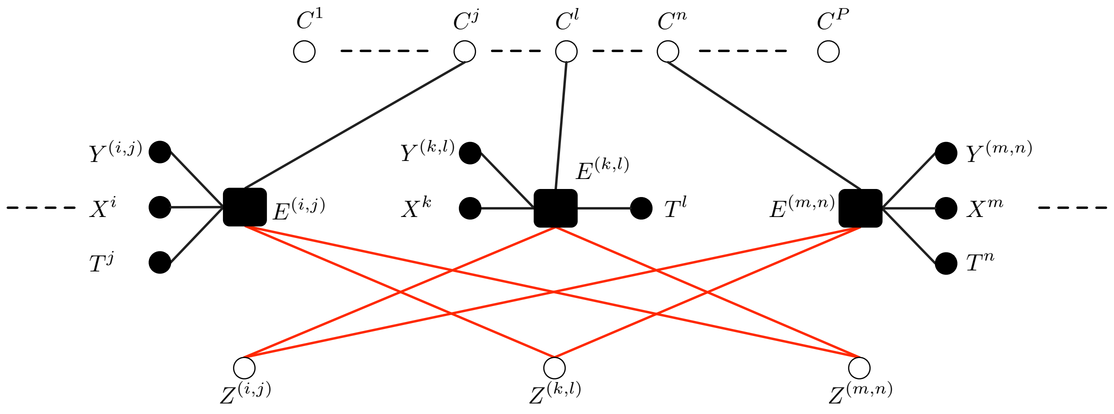


Figure 5.2: *The reduced factor graph used for constructing house price indices using a latent "normalized" price manifold over space and time.*

The global energy function of the system is the sum of energies of all the local energy functions and is given by

$$\begin{aligned} E(\mathbf{C}, \mathbf{Z}, \mathbf{Y}, \mathbf{X}, \mathbf{T}) &= \sum_{(i,j)\in\mathcal{I}} E^{(i,j)}(C^j, Z^{(i,j)}, Y^{(i,j)}, X^i, T^j), && (5.13) \\ &= \sum_{(i,j)\in\mathcal{I}} (Y^{(i,j)} - (C^j + H(X^i, T^j, R^{(i,j)}, Z_{\mathcal{N}(i,j)})))^2. && (5.14) \end{aligned}$$

5.2.2 The Learning Algorithm

Since the factors associated with every transaction $Q^{(i,j)}$ are quadratic in $Y^{(i,j)}$, exactly the same analysis to an efficient learning algorithm of chapter 3 goes through here as well. In particular, the system is trained by maximizing the likelihood of the training data. This is achieved by marginalizing the negative log likelihood loss with respect to the hidden variables $\mathbf{Z}$ and minimizing it with respect to the parameters $\mathbf{C}$. Note that the set of indices $\mathbf{C}$ can be viewed as

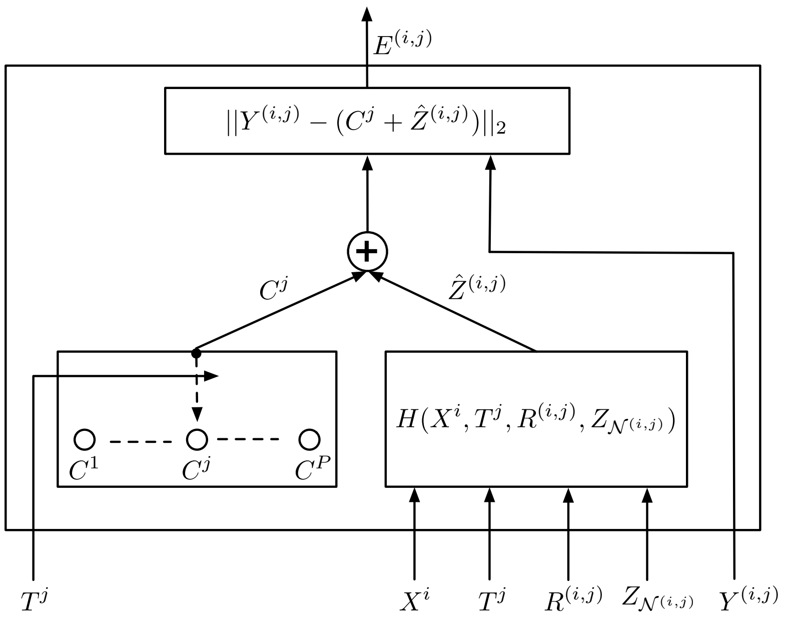


Figure 5.3: *The architecture of each factor in the factor graph of figure 5.2. It consists of two trainable components: the set of parameters* $\mathbf{C}$ *and the non-parametric function* H.

parameters to the model since the number of these indices are independent of the number of training samples. As discussed in chapter 3, the marginalization over $\mathbf{Z}$ can be approximated by a minimization operation, and when the energy function is a quadratic function of $\mathbf{Y}$, the contrastive term in the negative log-likelihood loss function vanishes. Hence the entire process reduces to minimizing the energy loss simultaneously with respect to $\mathbf{C}$ and $\mathbf{Z}$. That is

$$\mathcal{L}(\mathbf{C}, \mathbf{Z}) = \min_{\mathbf{C},\mathbf{Z}} \sum_{(i,j)\in\mathcal{I}} (Y^{(i,j)} - (C^j + H(X^i, T^j, R^{(i,j)}, Z_{\mathcal{N}(i,j)})))^2. \tag{5.15}$$

Since maximizing the likelihood of the data in general can lead to over fitting, simply minimizing the energy is not sufficient. In order to avoid over fitting and achieve the spatio-temporal smoothness of the price surface we add two additional regularization terms to the loss function.

Regularization

The first is the L_2 regularizer over the latent variables $\mathbf{Z}$, and is given by

$$\Phi_1 = \sum_{(i,j)\in\mathcal{I}} (Z^{(i,j)})^2. \tag{5.16}$$

In order to interpret this regularization term we can re-write the sum of squares as

$$\Phi_1 = (M\bar{Z}^2 + \sum_{(i,j)\in\mathcal{I}} (Z^{(i,j)} - \bar{Z})^2, \tag{5.17}$$

where $\bar{Z}$ is the mean of $Z^{(i,j)}$, and M is the total number of transactions. Minimizing Φ_1 results in driving the mean $\bar{Z}$ to zero and minimizing the variance of $Z^{(i,j)}$'s around the mean $\bar{Z}$. This process ensures that any common component of house price appreciation is absorbed by the index and not by the "normalized price" surface. This gives some "stiffness" to the price surface.

The second regularization term added to the loss function is what we call the *self consistency term*, similar to the one discussed in chapter 4. Note that while estimating the "normalized price" of the current transaction the function H does not take the $Z^{(i,j)}$ of the current transaction as its argument. Rather it only looks at the "normalized prices" of the neighboring transaction. Hence there is no way to ensure that the estimate (output of the function H) will asymptotically converge to the learnt $Z^{(i,j)}$ of the current sample. This can be made sure by explicitly adding the self consistence term in the loss function

$$\Phi_2 = \sum_{(i,j)\in\mathcal{I}} (Z^{(i,j)} - H(X^i, T^j, R^{(i,j)}, Z_{\mathcal{N}(i,j)}))^2. \tag{5.18}$$

This also helps ensure smoothness in the price surface. In words what it means is that when there is no past sale, the "normalized price" of any transaction must be close to the weighted sum of the normalized prices of its spatio-temporal neighbors. When there is a past sale then the weights are distributed between the past "normalized price" and the weighted sum of "normalized prices" of spatio-temporal neighbors, depending on the time between the current and past sale.

Thus the complete loss function is given by

$$\begin{aligned}
\mathcal{L}(\mathbf{C}, \mathbf{Z}) &= \sum_{(i,j) \in \mathcal{I}} (Y^{(i,j)} - (C^j + H(X^i, T^j, R^{(i,j)}, Z_{\mathcal{N}(i,j)})))^2 && (5.19) \\
&+ \delta_1 \sum_{(i,j) \in \mathcal{I}} (Z^{(i,j)})^2 && (5.20) \\
&+ \delta_2 \sum_{(i,j) \in \mathcal{I}} (Z^{(i,j)} - H(X^i, T^j, R^{(i,j)}, Z_{\mathcal{N}(i,j)}))^2, && (5.21)
\end{aligned}$$

where δ_1 and δ_2 are coefficients that govern the importance of the two regularization terms. Their values were determined via the process of validation.

The Optimization Problem

Since the above loss function is quadratic in both sets of parameters $\mathbf{C}$ and $\mathbf{Z}$, one can express the optimization problem as a solution to the linear system of the form $Ax = b$. We know that the output of the function H can be written as a linear combination of the latent variables associated with the spatio-temporal neighbors (and the previous sales if present). Thus we can express the function H as

$$H^i = \mathbf{Z}' \cdot U^i, \tag{5.22}$$

where H^i denotes the value of the function H for the transaction Q^i, $\mathbf{Z}'$ denotes the transpose of the vector $\mathbf{Z}$, and U^i is a sparse vector of size M (equal to the number of transactions in the training set). The number of non-zero elements in U^i is either equal to the number of spatio-temporal neighbors of the i-th transaction if the transaction is not a repeat sales, or is equal to one plus the number of spatio-temporal neighbors when the transaction is a repeat sales. The j-the non-zero element is equal to the coefficient corresponding to the j-th spatio-temporal neighbor. Let $\mathbf{U}$ be an NxN matrix whose i-th row is the vector $U^{i'}$. Then the L_2 regularization term can be written as

$$\Phi_1 = \mathbf{Z}' \cdot \mathbf{Z}', \tag{5.23}$$

and the explicit self consistency term can be written as

$$\Phi_2 = (\mathbf{Z} - \mathbf{U} \cdot \mathbf{Z})'(\mathbf{Z} - \mathbf{U} \cdot \mathbf{Z}). \tag{5.24}$$

Let $\mathbf{V}$ be a MxT matrix (where T is the total number of time periods), such that $\mathbf{V}_{ij} = 1$ if the i-th transaction Q^i is transacted in time period T^j and 0 otherwise. Then the loss function in the matrix form can be written as

$$\begin{aligned} \mathcal{L}(\mathbf{C}, \mathbf{Z}) &= [\mathbf{Y} - \mathbf{V} \cdot \mathbf{C} - \mathbf{U} \cdot \mathbf{Z}]'[\mathbf{Y} - \mathbf{V} \cdot \mathbf{C} - \mathbf{U} \cdot \mathbf{Z}] && (5.25) \\ &+ \delta_1 \mathbf{Z}' \cdot \mathbf{Z} + \delta_2 (\mathbf{Z} - \mathbf{U} \cdot \mathbf{Z})'(\mathbf{Z} - \mathbf{U} \cdot \mathbf{Z}). && (5.26) \end{aligned}$$

The configuration of variables $\mathbf{Z}$ and $\mathbf{C}$ that minimizes this loss function is the same as the solution to the following linear system.

$$\begin{bmatrix} \mathbf{V}' \cdot \mathbf{V} & \mathbf{V}' \cdot \mathbf{U} \\ \mathbf{U}' \cdot \mathbf{V} & (1 + \delta_2)\mathbf{U}' \cdot \mathbf{U} + (\delta_1 + \delta_2)I - \delta_2(\mathbf{U} + \mathbf{U}') \end{bmatrix} \begin{bmatrix} \mathbf{C} \\ \mathbf{Z} \end{bmatrix} = \begin{bmatrix} \mathbf{V}' \cdot \mathbf{Y} \\ \mathbf{U}' \cdot \mathbf{Y} \end{bmatrix}, \tag{5.27}$$

I is a $M \times M$ identity matrix. The above system of equations is of the form $Ax = b$ where $x = [\mathbf{C} \quad \mathbf{Z}]'$, $b = [\mathbf{V}' \cdot \mathbf{Y} \quad \mathbf{U}' \cdot \mathbf{Y}]'$ and

$$A = \begin{bmatrix} \mathbf{V}' \cdot \mathbf{V} & \mathbf{V}' \cdot \mathbf{U} \\ \mathbf{U}' \cdot \mathbf{V} & (1 + \delta_2)\mathbf{U}' \cdot \mathbf{U} + (\delta_1 + \delta_2)I - \delta_2(\mathbf{U} + \mathbf{U}') \end{bmatrix}. \tag{5.28}$$

It is computationally infeasible to directly solve this system since the size of the vector x is of the order of 1.3 million. Thus we resort to iterative methods, in particular conjugate gradient, to solve for this system and train the model. Note that we have shown here that learning the parameters of the model collectively reduces to solving a linear system. Thus our learning algorithm is effectively doing a belief propagation over the relational factor graph (Wang and Guo, 2006).

5.2.3 The Testing Algorithm

If the model is trained using all the transactions that took place within the time periods T^1 to T^k, for some k, then its performance is measured based on the accuracy with which it predicts the prices of the transactions in time period T^{k+1}. However note that the price is modeled as a combination of a global index and the "normalized price" of the transaction. Since the prediction is required for a transaction in some future time period, we also need to estimate the global index for this time period. Hence the inference procedure for a new future transaction Q^0 consists of two steps. First, it estimates the global price index for that future period. Second, it computes the "normalized price" of the transaction from the relational factor graph.

An autoregressive model is used to estimate the price index for the test period. Let us denote by nR^t the nominal monthly returns for time period T^t. Then nR^t is given by

$$nR^t = \frac{C^t}{C^{t-1}} - 1. \tag{5.29}$$

We assume that nR^t follows a simple AR(1) process. That is

$$nR^{t+1} = \rho(nR^t) + \theta^{t+1}, \tag{5.30}$$

where θ^{t+1} is assumed to be a gaussian with zero mean, and ρ is a parameter to be estimated. Using this equation we estimate the value of ρ and denote it by $\hat{\rho}$. Once we have computed $\hat{\rho}$, the estimated value of the index in the future time period T^{t+1} is given by

$$C^{t+1} = \hat{\rho}(nR^t)C^t. \tag{5.31}$$

The process of estimating the "normalized price" of a future unseen transaction $Q^0[k, l]$ is the same as estimating the "desirability" of a house discussed in the previous chapter. This involves creating a new factor corresponding to the new transaction and augmenting it with the trained factor graph by making the connections from the new factor to the latent variables $Z^{(i,j)}$'s associated with the spatio-temporal neighbors of the new transaction. The nonparametric function

H is then used to compute the estimate of the "normalized price" $Z^{(k,l)}$ from the "normalized prices" of its neighbors.

Finally the two estimates, one of the index C^l and the other of the "normalized price" $Z^{(k,l)}$ are added to get the log of the price of the new transaction.

5.3 Experiments

Experiments were performed using the dataset discussed in section 5.1. The reason for choosing so few variables was to show the effect of spatio-temporal relationships among prices of transactions, without resorting to complex individual architectures for individual transactions.

The model is tested on eighteen different test periods which were constructed on the basis of the month of transaction. The test periods were March 2000 to August 2000, January 2004 to Jun 2004, and July 2007 to December 2007[1]. The samples are specifically chosen to represent three very different periods in Los Angeles house price history. The first set of test periods March to August 2000 saw average growth in house prices, the second set of test periods January to June 2004 were right in the middle of the housing boom and the last set of test periods July to December 2007 saw a significant decline in house prices. The training samples corresponding to each test period included all the transactions from January 1984 to one month before the test period. For example, when the test set contained transactions from the period March 2004, the training data included all transactions from the period January 1984 to February 2004. The period of February 2000 was used for validation.

The model was compared with the S&P Case-Shiller model for index construction. It is a repeat sales model and has been an industry standard for quite sometime. The goodness of an index is measured by its accuracy in predicting the prices of unseen transactions that took

[1]It was not possible to test on the entire period $2000 - 08$ because the hard disk space required to store all the relevant neighborhood matrices exceeds 250 GBs for just these 18 months

place in the future. As evident from the results discussed below, accounting for spatio-temporal relationships among samples significantly boosts the performance of the model.

5.3.1 Repeat Sales indices

Extreme heterogeneity in the attributes of houses makes the construction of house price indices very difficult. The constructed index must therefore control for these heterogeneities. One way to solve this problem is to only look at the repeat sales of houses. If one assumes that the attributes of houses do not change over time, and there is no selection bias in selecting only households that have been sold at least twice, then this approach can construct a house price index with the desired property.

In particular the price equation of a basic repeat sales model is given by

$$\frac{Pr^{(i,t)}}{Pr^{(i,t')}} = \frac{C^{t'}}{C^{t}} \epsilon^{(i,t,t')}, \tag{5.32}$$

where $Pr^{(i,t)}$ is the price of the house i at time period t, C^t is the price index at time period t, and $\epsilon^{(i,t,t')}$ is the error term associated with the house i sold at time periods t and t' and it is I.I.D and distributed according to $N(0, \sigma^2)$. In the log domain the above price equation is

$$Pr^{(i,t)} - Pr^{(i,t')} = C^{t'} - C^{t} + \epsilon^{(i,t,t')}. \tag{5.33}$$

If for every house i we associate P variables $B^j : j \in [1, \ldots, P]$, such that $B^t = 1$ if the house was resold in time period t, and $B^t = -1$ if the house was first sold in period t, then the above price equation can be written as

$$D^{(i,t,t')} = \sum_{j=1}^{P} C^j B^j + \epsilon^{(i,t,t')}, \tag{5.34}$$

where $D^{(i,t,t')} = Pr^{(i,t)} - Pr^{(i,t')}$. In matrix form this equation becomes

$$D = B \cdot C + \epsilon, \tag{5.35}$$

which can be solved for the log of the indices C.

One major limitation of the above model is that the error term is homoskedastic: the error terms are independently and identically distributed with the same variance. What this means is that if a particular house is sold more than twice its error terms will be uncorrelated. Furthermore two possibly similar houses sold and resold in exactly the same periods will also have independent error terms. Case and Shiller argue against such independence of error terms. The intuition is that changes in attributes of houses are likely to be more, the longer the time between sales, resulting in a larger unexplained variance in prices for these houses. That is, there is a drift in house prices over time. In the above model houses resold after a longer time have a greater impact on the index (because of higher unexplained variance as argued by Case and Shiller). Hence, Case and Shiller formulate a weighted repeat sales index model that down weights the repeat sales transactions which have large time difference. The basic methodology is as follows.

The price of a house i is given by

$$Pr^{(i,t)} = C^t + \alpha^{(i,t)} + \beta^{(i,t)}, \tag{5.36}$$

where C^t denoted the area wide price index, $\alpha^{(i,t)}$ is the drift term and is a gaussian random walk (i.e., $\alpha^{(i,t)} - \alpha^{(i,(t-1))}$ is I.I.D normal), and $\beta^{(i,t)}$ is the sale specific error term. The price difference equation can be written as

$$\begin{aligned} Pr^{(i,t)} - Pr^{(i,t')} &= C^t - C^{t'} + (\alpha^{(i,t)} - \alpha^{(i,t')}) + (\beta^{(i,t)} - \beta^{(i,t')}), && (5.37) \\ &= \sum_{j=1}^{T} B^{(i,j)} C^j + \epsilon^{(i,t,t')}, && (5.38) \end{aligned}$$

where $B^{(i,j)}$'s are time dummy variables with $B^{(i,j)} = 1$ if $j = t$, and $B^{(i,j)} = -1$ if $j = t'$ and 0 otherwise. $\epsilon^{(i,t,t')}$ is the random error term for every repeat sales transaction distributed from a normal distribution with 0 mean and $(\sigma_\alpha^2(t - t') + 2\sigma_\beta^2)$ variance.

The indices are estimated using a 3 stage least squares method. First the log of the prices are regressed on the time dummy variables B using the price difference equation 5.38, using standard least squares. Then the residuals are regressed on a constant and $(t - t')$ to get the estimates of σ_α^2, and σ_β^2. Finally the estimate of variance of ϵ are used to weight the transactions and obtain a weighted least squares estimates of the indices C^t.

The above model works with geometric averages by using the log of the prices instead of the actual prices. However in practice, in particular in the S&P Case-Shiller model, actual prices are used and arithmetic averages are computed. This is because the arithmetic averages are less sensitive to outliers which any housing tape is likely to have. This is the model which we implemented in the present chapter.

5.3.2 Spatio-Temporal Model

The relational factor graph model proposed in this chapter tries to address all the issues which a general repeat sales index methodology, such as Case-Shiller lacks. It not only makes use of the transactions that are repeat sales, but it also uses the single sales transactions for index construction. In addition it also captures the influence of the spatio-temporal neighbors on the price of any transaction. This is achieved with the help of the non-parametric function H. While estimating the "normalized" price of the current transaction this function uses the "normalized prices" of the neighboring transactions. However since the "normalized prices" are not given to us as part of the data, they are modeled as latent variables. Furthermore, since the "normalized prices" of every transaction influences the "normalized prices" of others, the model tries to learn them collectively. Thus one can view the model as trying to learn a latent normalized price manifold over space and time, in addition to constructing the index simultaneously. The final price of any transaction is given by the combination of the index of the time period in which the transaction took place, and the value of this manifold at the point of the transaction (in space and time).

The set of learnable indices $\mathbf{C}$ were modeled as scalar coefficients, and the function H was a non-parametric function which involved fitting a weighted local linear regression model on the spatio-temporal neighbors of the transaction. The spatio-temporal kernel μ, and the repeat sales kernel λ used were the ones discussed in section 5.2.1. All the hyper-parameters of the model, namely $\bar{d}$, $\bar{r}$, $\bar{T}$, δ_1, and δ_2 were chosen using the process of validation. In particular the model was trained using all the transactions from the period January 1984, to January 2000, and validated on the set of transactions in the period February 2000. The hyper-parameters were adjusted to get the best performance on this validation set. Once found, their values were kept fixed for testing the model in other test periods. The validated values of the hyper-parameters were, $\bar{d} = 0.5$ (miles), $\bar{r} = 36$ (months), $\bar{T} = 120$ (months), $\delta_1 = 0.01$, and $\delta_2 = 1$. The results were not very sensitive to small variations of any of the hyper-parameters around these values.

5.4 Results and Discussion

The performance of the model, and hence the goodness of the index is measured by looking at how well it predicts the price of new transactions in the test periods. As before, we measure the Absolute Relative Forecasting Error for every transaction, which is given by

$$fe^{(i,j)} = \frac{|Y^{(i,j)} - Pr^{(i,j)}|}{Y^{(i,j)}}. \tag{5.39}$$

Computing the root mean square error is not useful because of its sensitivity of outliers. Here $Y^{(i,j)}$ is the actual price of the transaction $Q^{(i,j)}$, and $Pr^{(i,j)}$ is the predicted price of the transaction by the model. Two performance quantities on the test sets are reported. The first is the percentage of houses with less than 5% forecasting error, and the second is the percentage of houses with less than 15% forecasting error. The larger these numbers the better the model.

Overall Index Performance

As mentioned earlier, we compare our index model with the S&P Case-Shiller index model which has been an industry standard. The figure 5.4 shows the pattern of the two indices. The indices are normalized so that the index value in the period January 2000 is 100. Barring the first 5 years, from 1984 - 1989, there is very little difference between the two indices. However the difference between the two models lie in trying to predict the prices of new transactions and also in the error patterns of the two models.

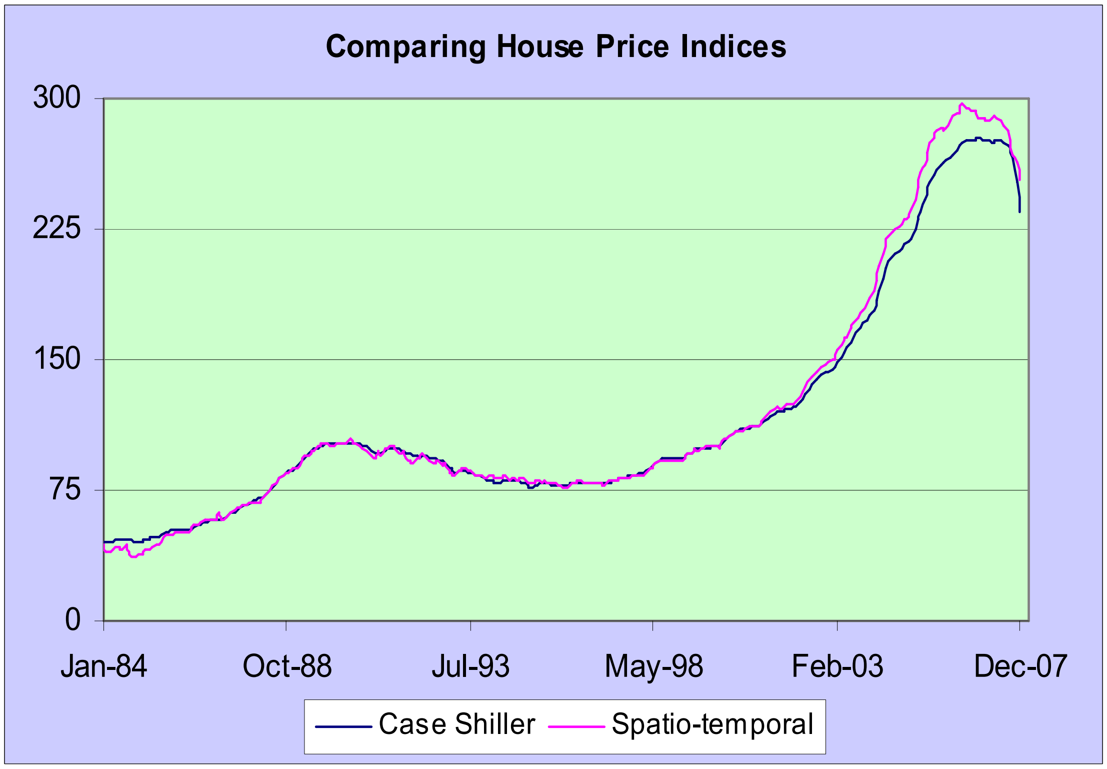


Figure 5.4: *Plot showing the pattern of the Case-Shiller index and our Spatio-Temporal index. Other than in the first* 5 *years, there is very little difference between the two indices.*

Table 5.1 gives the overall prediction performance of the two indices for all the repeat sales

Table 5.1: *The overall prediction errors for the Relational Factor graph and Case-Shiller model. The errors are segmented according to the time periods.* ARF Err*: Absolute Relative Forecasting percentage error,* Med Err (Ab)*: Absolute Median percentage error, and* Med Err(Ac)*: Actual Median percentage error.*

	RELATIONAL FG				**CASE SHILLER**			
	ARF ERR %		MED ERR %		ARF ERR %		MED ERR %	
TIME PERIOD	$\leq 5\%$	$\leq 15\%$	AB	AC	$\leq 5\%$	$\leq 15\%$	AB	AC
MAR. – AUG. 2000	**29.04**	**69.30**	**9.43**	**-2.47**	24.79	60.91	11.29	-3.50
JAN. – JUN. 2004	**32.95**	**76.42**	**8.00**	**-3.17**	25.95	65.56	10.31	3.61
JUL. – DEC. 2007	**29.03**	**71.05**	**9.24**	**0.50**	17.73	50.24	14.90	-8.06

transactions that took place in the given periods in Los Angeles county. Clearly the relational factor graph model comprehensively outperforms the Case-Shiller index model in all the testing periods. The difference is particularly stark in the period July – December 2007, which saw a sharp decline in the house prices. The Case-Shiller index was found to be over-predicting the prices in this period by an average of about 8% median error.

After accounting for hetroscedasticity, the errors in the Case-Shiller model should be I.I.D normal. However the experiments show a lot of structure associated with the error patterns. In particular we find patterns with respect to initial prices, with respect to time between transactions, and with respect to geography. These patterns are present because the Case-Shiller's model treats all houses as similar and differentiates them only on the basis of the turnover time. However homes are heterogeneous in quality and in terms of the location they inhabit. Though controlling for quality is generally hard because of lack of data, the present set of results show that accounting for relationships among transactions both in space and time can eliminate some of these systematic patterns in the error structure. We now discuss in detail the various patterns that were observed in the Case-Shiller's error structure and discuss how they were taken care of

Table 5.2: *Error patterns for Case-Shiller and Relational Factor Graph model segmented with respect to time between sales.* ARF Err*: Absolute Relative Forecasting percentage error,* Med Err (Ab)*: Absolute Median percentage error, and* Med Err(Ac)*: Actual Median percentage error.*

	RELATIONAL FG				**CASE SHILLER**			
	ARF ERR %		MED ERR %		ARF ERR %		MED ERR %	
TIME PERIOD	$\leq 5\%$	$\leq 15\%$	AB	AC	$\leq 5\%$	$\leq 15\%$	AB	AC
$<$ 1 YEAR	32.47	75.39	7.79	**-1.48**	**41.78**	**90.18**	**6.36**	2.61
1-2 YEARS	29.71	72.17	9.21	-1.98	**31.04**	**72.12**	**8.65**	**0.60**
2-3 YEARS	**30.71**	**73.48**	**8.71**	-3.15	28.68	71.08	9.22	**2.41**
3-5 YEARS	**30.21**	**71.93**	**8.80**	-2.81	23.23	62.65	11.18	**1.13**
5-7 YEARS	**29.12**	**72.90**	**8.78**	-1.94	21.53	57.70	12.67	**-1.20**
7-10 YEARS	**31.43**	**72.77**	**8.53**	-2.65	23.98	61.26	11.50	**-2.54**
10-12 YEARS	**30.89**	**73.44**	**8.51**	**-2.57**	21.16	55.96	12.87	-5.51
12-15 YEARS	**31.20**	**71.63**	**8.78**	**-1.82**	19.66	50.49	14.74	-5.46
15-20 YEARS	**32.51**	**72.20**	**8.4**	**-1.76**	16.04	43.24	18.04	-5.31
$>$ 20 YEARS	**28.37**	**71.81**	**9.53**	**0.72**	9.78	30.52	24.13	-10.51

by the relational factor graph model.

Index Performance Segmented by Time Between Sales

Table 5.2 shows the errors for both the Case-Shiller model and the factor graph model segmented according to the time between successive sales of the houses – also called the *turnover time*. From the table one can see that the Case-Shiller model does a good job of predicting prices on quick trades, particularly those that are within one year of time, where it beats the factor graph model. This makes intuitive sense, since in the short run any change in prices is unlikely to be because of the changes in neighborhood, or the structure of the house. It is likely driven by the citywide supply and demand, interest rates and similar factors.

The performance of the Case-Shiller model however falls dramatically and monotonically with increasing turnover time. Furthermore the median error is found to be increasingly negative as the turnover time increases, implying that the homes that transact after a long time are over-predicted. In contrast, the relational factor graph model does not show any such pattern. The prediction accuracy of the model roughly stays constant throughout the time period, with around and above 70% of houses being predicted with less than 15% error. One possible explanation behind this scenario could be that in the long run, the houses as well as the neighborhood undergo dramatic changes. Hence, if the previous sale of a house took place long time back, the price of a transaction will not be influenced as much by its previous sale, as by the current status of the neighborhood in which the house lies, and by its present characteristics. Though it is difficult to capture the house specific changes, one can certainly capture the changes in the neighborhoods and use the findings to influence the price of the transaction. This is exactly what the proposed factor graph model does. It tries to learn the changes in neighborhoods over time by learning the latent "normalized price" manifold over space and time, and uses it to predict the price of a transaction at any point in time.

Index Performance Segmented by Initial Price

Table 5.3 show the errors made by the two models, segmented according to the initial price of the transactions. The Case-Shiller model does a very poor job in predicting the prices of houses that either have very low initial price, or have a very high initial price. In particular for houses with price less than $50,000$, the Case-Shiller model predicts only 3.88% of houses with less than 15% error. The accuracy steadily increases until it peaks at 71.11% for homes with the prices in the range of $200,000 - 250,000$. It again gradually decreases as the initial prices increase and drops down to 54.53% for houses with a price of more than a million. In addition to performing badly on the cheap and expensive houses, the error structure of the Case-Shiller's model exhibits

Table 5.3: *Error patterns segmented according to the initial price of houses.* ARF Err*: Absolute Relative Forecasting percentage error,* Med Err (Ab)*: Absolute Median percentage error, and* Med Err(Ac)*: Actual Median percentage error.*

	Relational FG				**Case Shiller**			
	ARF err %		Med err %		ARF err %		Med err %	
Initial Price ($)	$\leq 5\%$	$\leq 15\%$	Ab	Ac	$\leq 5\%$	$\leq 15\%$	Ab	Ac
$\leq$ 50,000	**27.40**	**62.59**	**10.46**	**0.91**	0.92	3.88	69.25	67.51
50-75,000	**25.49**	**62.10**	**10.71**	**1.22**	12.28	30.64	26.17	15.72
75-100,000	**30.37**	**69.47**	**9.28**	**0.07**	15.88	44.47	17.26	-0.77
100-125,000	**30.18**	**73.31**	**8.83**	-0.43	18.12	50.95	14.70	**-0.20**
125-150,000	**32.28**	**75.84**	**8.11**	**-0.95**	21.24	57.48	12.64	-1.32
150-200,000	**34.22**	**77.97**	**7.74**	-2.39	25.24	65.58	10.35	**0.83**
200-250,000	**33.22**	**75.84**	**8.00**	-3.37	28.91	71.11	9.26	**1.45**
250-300,000	**30.73**	**73.62**	**8.51**	-3.24	29.73	69.81	9.26	**0.48**
300-350,000	**29.18**	**72.16**	**9.13**	-3.38	27.38	67.02	9.71	**-1.20**
350-400,000	**29.79**	**70.39**	**9.15**	**-2.56**	28.65	65.89	9.85	-2.84
400-450,000	**28.20**	**68.41**	**9.70**	**-2.64**	25.78	64.00	10.96	-3.83
450-500,000	**27.38**	**68.99**	**9.63**	**-1.59**	23.61	59.83	11.70	-6.30
500-600,000	**27.41**	**66.34**	**10.32**	**-1.09**	22.83	57.52	12.54	-8.50
600-750,000	**25.19**	**65.64**	**10.51**	**-2.46**	19.36	54.57	13.45	-8.87
750,000-1 million	**23.23**	**57.07**	**12.75**	**-5.00**	21.31	53.23	13.66	-7.66
> 1 million	17.03	44.73	16.59	-13.94	**21.44**	**54.53**	**13.41**	**-6.53**

another interesting pattern. It systematically under predicts the prices of homes which have very low initial price, and over predicts the prices of homes with very high initial price. This is evident from the fact that for houses with less than $\$50,000$ initial price the actual median error is positive and is as high as 67.51%, and for houses with more than 1 million initial price the median error is negative and is -6.53%.

The table also shows that such patterns are not so much apparent with the relational factor graph model. There is no under-prediction of homes with low initial prices ($\leq 50,000$), since the actual median error for these homes is around 1%. There is also a dramatic improvement in prediction accuracy in the factor graph model over the Case-Shiller model at low initial prices. Since the indices are the same, this implies that location or geography has a huge impact. Indeed the homes are clustered according to prices, and the relational factor graph framework is able to capture this relationship among nearby homes to get better prediction and remove the superfluous patterns in the error.

Note that the factor graph model does poorly on homes with very high initial prices by systematically over predicting them (actual median error is -13.94%). Also the prediction accuracy for these homes is very poor. The factor graph model smooths the "normalized price" surface by ensuring that the "normalized price" of nearby homes (in the spatio-temporal sense) isn't too different. The model therefore faces obvious limitations in neighborhoods that are sparsely transacting and/or not homogeneous. However, though the higher magnitude of errors could be explained by the sparseness of transactions and/or by a lack of homogeneity of homes in high price neighborhoods, this would not explain over-prediction.

Index Performance Segmented by Geography

Finally, table 5.4 gives the breakdown of the errors for the two models according to smaller geographic areas. The relational factor graph framework clearly out performs the Case-Shiller model even for individual neighborhood, other than Pasadena.

Table 5.4: *Error patterns segmented with respect to geography.* ARF Err*: Absolute Relative Forecasting percentage error,* Med Err (Ab)*: Absolute Median percentage error, and* Med Err(Ac)*: Actual Median percentage error.*

	RELATIONAL FG				**CASE SHILLER**			
	ARF ERR %		MED ERR %		ARF ERR %		MED ERR %	
NEIGHBORHOOD	≤ 5%	≤ 15%	AB	AC	≤ 5%	≤ 15%	AB	AC
PALMDALE	**31.20**	**75.31**	**8.40**	**-0.46**	19.78	52.6	13.96	-5.35
PASADENA	21.85	57.63	12.43	-6.03	**24.42**	**61.29**	**11.12**	**2.87**
TORRANCE ETC.	**30.92**	**74.84**	**8.48**	**-1.44**	24.78	64.01	10.16	-1.79
VAN NUYS	**26.44**	**72.65**	**9.38**	-5.03	18.61	54.3	13.72	**2.19**
LONG BEACH	**29.21**	**74.25**	**9.19**	-4.30	25.14	64.82	10.95	**2.26**
INDUSTRY ETC.	**38.18**	**81.90**	**6.91**	**0.27**	25.79	63.55	10.53	-4.35

6
MORE RESULTS ON ENERGY BASED MODELS

This chapter elaborates on a set of results discussed informally in chapter 1. In particular we discussed earlier that not all combinations of loss functions and architectures of energy functions are compatible, and learning in them can lead to "flat" energy surfaces leading to trivial solutions. For example, in (LeCun et al., 2006) LeCun et al., give examples of combination of energy function and loss function which work and which don't. They show that when a straight forward regression type architecture is trained with simple energy loss, it leads to successful training. However when this loss is used with a more complicated architecture, akin to the one discussed in implicit regression in chapter 1, it leads to a flat energy surface. Furthermore is the implicit regression architecture is trained with a loss function with an explicit contrastive term, such as *square-square* loss of NLL loss, the machine is successfully trained and we get the right shape of the energy function. Here we make these statements formal by giving a set of sufficient conditions that any loss function must satisfy so that its minimization leads to a successful training in an energy based setting: an energy surface whose minimum corresponds to the correct answer.

6.1 Sufficient Conditions for Good Loss Functions

We first state a set of sufficient conditions that that the energy function and the loss function must satisfy in order to be guaranteed to work in an energy-based setting. We then explain what these conditions mean by discussing them in the light of few example loss function.

6.1.1 Conditions on the Energy

Inference method in energy-based learning involves choosing the answer with minimum energy. Thus the condition for the correct inference on a sample (X^i, Y^i) is:

Condition 6.1. For sample (X^i, Y^i), the machine will give the correct answer for X^i if

$$E(W, Y^i, X^i) < E(X, Y, X^i),\ \forall Y \in \mathcal{Y}\ and\ Y \neq Y^i. \tag{6.1}$$

In words this means that the inference algorithm will give the correct answer if the energy of the desired answer Y^i is less than the energies of all the other answers Y.

To ensure that the correct answer is robustly stable, we may choose to impose that its energy be lower than energies of incorrect answers by a positive margin m. If $\bar{Y}^i$ denotes the most offending incorrect answer, then the condition for the answer to be correct by a margin m is:

Condition 6.2. For a variable Y and sample (X^i, Y^i) and positive margin m, the inference algorithm will give the correct answer for X^i if

$$E(W, Y^i, X^i) < E(W, \bar{Y}^i, X^i) - m. \tag{6.2}$$

6.1.2 Sufficient Conditions on the Loss Functional

If the system is to produce the correct answers, the loss functional should be designed in such a way that minimizing it will cause $E(W, Y^i, X^i)$ to be lower than $E(W, \bar{Y}^i, X^i)$ by some margin m. Since only the relative values of those two energies matter, we only need to consider the shape of a slice of the loss functional in the 2D space of those two energies. For example, in the case where $\mathcal{Y}$ is the set of integers from 1 to k, the loss functional can be written as:

$$L(W, Y^i, X^i) = L(Y^i, E(W, 1, X^i), \ldots, E(W, k, X^i)). \tag{6.3}$$

The projection of this loss in the space of $E(W, Y^i, X^i)$ and $E(W, \bar{Y}^i, X^i)$ can be viewed as a function Q parameterized by the other $k - 2$ energies:

$$L(W, Y^i, X^i) = Q_{[E_y]}(E(W, Y^i, X^i), E(W, \bar{Y}^i, X^i)), \tag{6.4}$$

where the parameter $[E_y]$ contains the vector of energies for all values of Y except Y^i and $\bar{Y}^i$.

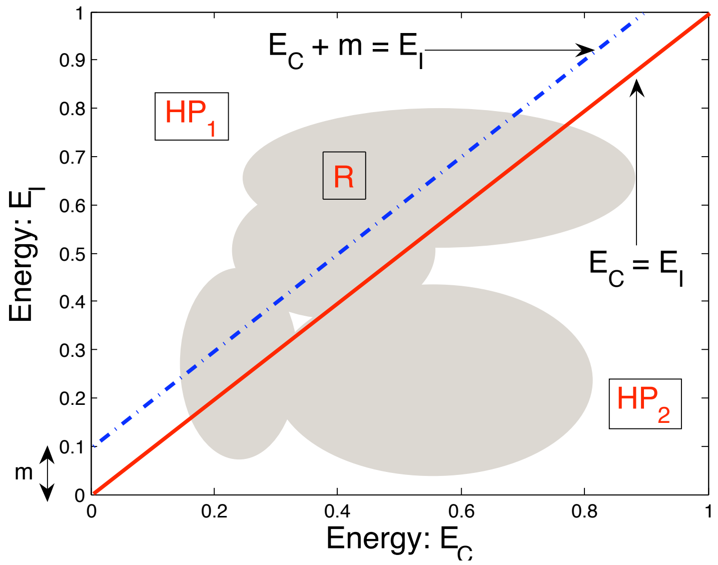


Figure 6.1: *Figure showing the various regions in the plane of the two energies E_C and E_I. E_C are the (correct answer) energies associated with (X^i, Y^i), and E_I are the (incorrect answer) energies associated with $(X^i, \bar{Y}^i)$.*

We assume the existence of at least one set of parameters W for which condition 6.2 is satisfied for a single training sample (X^i, Y^i). Clearly, if such a W does not exist, there cannot exist any loss function whose minimization would lead to condition 6.2. For the purpose of notational simplicity let us denote the energy $E(W, Y^i, X^i)$ associated with the training sample (X^i, Y^i) by E_C (as in "correct energy") and $E(W, \bar{Y}^i, X^i)$ by E_I (as in "incorrect energy"). Consider the plane formed by E_C and E_I. As an illustration, Figure 6.4 shows a 3-dimensional plot of the *square-square* loss function in which the abscissa is E_C and the ordinate is E_I. The third axis gives the value of the loss for the corresponding values of E_C and E_I. In general, the loss function is a family of 2D surfaces in this 3D space, where each surface corresponds to one particular configuration of all the energies except E_C and E_I. The solid red line in the figure corresponds to the points in the 2D plane for which $E_C = E_I$. The dashed blue line correspond to the margin line $E_C + m = E_I$. Let the two half planes $E_C + m < E_I$ and $E_C + m \geq E_I$ be denoted by HP_1 and HP_2 respectively.

Let R be the *feasible region*, defined as the set of values (E_C, E_I) corresponding to all possible values of $W \in \mathcal{W}$. This region may be non-convex, discontinuous, open, or one-dimensional and could lie anywhere in the plane. It is shown shaded in Figure 6.1. As a consequence of our assumption that a solution exists which satisfies conditions 6.2, R must intersect the half plane HP_1.

Let two points (e_1, e_2) and (e'_1, e'_2) belong to the feasible region R, such that $(e_1, e_2) \in HP_1$ (that is, $e_1 + m < e_2$) and $(e'_1, e'_2) \in HP_2$ (that is, $e'_1 + m \geq e'_2$). We are now ready to present the sufficient conditions on the loss function.

Condition 6.3. Let (X^i, Y^i) be the i^{th} training example and m be a positive margin. Minimizing the loss function L will satisfy conditions 6.1 or 6.2 if there exists at least one point (e_1, e_2) with $e_1 + m < e_2$ such that for all points (e'_1, e'_2) with $e'_1 + m \geq e'_2$, we have

$$Q_{[E_y]}(e_1, e_2) < Q_{[E_y]}(e'_1, e'_2), \tag{6.5}$$

Table 6.1: *A list of loss functions, together with the margin which allows them to satisfy condition 6.3. A margin* > 0 *indicates that the loss satisfies the condition for any strictly positive margin, and "none" indicates that the loss does not satisfy the condition.*

Loss (equation #)	**Formula**	**Margin**
energy loss (1.7)	$E(W, Y^i, X^i)$	none
perceptron (1.8)	$E(W, Y^i, X^i) - \min_{Y \in \mathcal{Y}} E(W, Y, X^i)$	0
hinge (1.12)	$\max\big(0, m + E(W, Y^i, X^i) - E(W, \bar{Y}^i, X^i)\big)$	m
square-square (1.13)	$E(W, Y^i, X^i)^2 - \big(\max(0, m - E(W, \bar{Y}^i, X^i))\big)^2$	m
NLL (3.22)	$E(W, Y^i, X^i) + \frac{1}{\beta} \log \int_{y \in \mathcal{Y}} e^{-\beta E(W, y, X^i)}$	> 0

where $Q_{[E_y]}$ is given by

$$L(W, Y^i, X^i) = Q_{[E_y]}(E(W, Y^i, X^i), E(W, \bar{Y}^i, X^i)). \tag{6.6}$$

In other words, the surface of the loss function in the space of E_C and E_I should be such that there exists at least one point in the part of the feasible region R intersecting the half plane HP_1 such that the value of the loss function at this point is less than its value at all other points in the part of R intersecting the half plane HP_2.

Note that this is only a sufficient condition and not a necessary condition. There may be loss functions that do not satisfy this condition but whose minimization still satisfies condition 6.2.

6.1.3 Which Loss Functions are "Good" or "Bad"

We say a loss function is "good" if it satisfies condition 6.3, and hence whose minimization will lead to a successfully training of the energy based models. Likewise, a loss function is "bad" if it does not satisfy condition 6.3. Table 6.1 lists discussed in chapter 1, together with the value of the margin with which they satisfy the sufficiency condition. The energy loss is marked “none” because it does not satisfy condition 6.3 for a general architecture. The perceptron loss satisfies it with a margin of zero. The hinge loss, the square-square loss and the NLL loss satisfies the condition with a positive margin. A more complete list of loss functions along with their margins is given in (LeCun et al., 2006). We now discuss in detail the loss functions discussed in chapter 1.

Energy Loss

The energy loss is classified as a “bad” loss function in general, but there are certain forms of energies for which it is a good loss function. For example consider an energy function of the form

$$E(W, Y^i, X^i) = \sum_{k=1}^{K} \delta(Y^i - k)||U^k - G_W(X^i)||^2. \tag{6.7}$$

This energy passes the output of the function G_W through K radial basis functions (one corresponding to each class) whose centers are the vectors U^k. If the centers U^k are fixed and distinct then the energy loss satisfies condition 6.3 and hence is a good loss function.

To see this, consider the two-class classification case (the reasoning for $K > 2$ follows along the same lines). The architecture of the system is shown in Figure 6.2. Let $d = ||U^1 - U^2||^2$, $d_1 = ||U^1 - G_W(X^i)||^2$, and $d_2 = ||U^2 - G_W(X^i)||^2$. Since U^1 and U^2 are fixed and distinct, there is a strictly positive lower bound on $d_1 + d_2$ for all G_W. Being only a two-class problem, E_C and E_I correspond directly to the energies of the two classes. In the (E_C, E_I) plane no part of the loss function exists in where $E_C + E_I \leq d$. The region where the loss function is defined is shaded in Figure 6.3(a). The exact shape of the loss function is shown in Figure 6.3(b). One can see from the figure that as long as $d \geq m$, the loss function satisfies condition 6.3. We conclude that this is a good loss function.

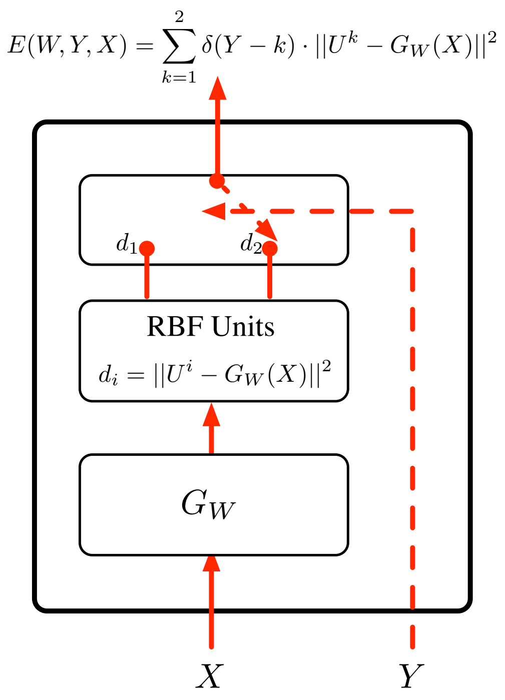
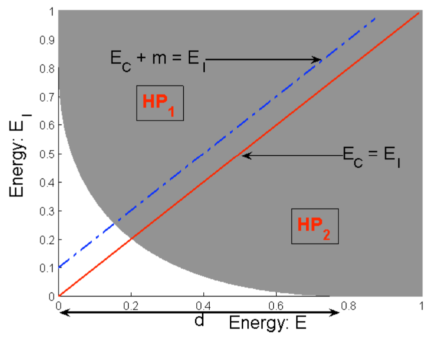

$E(W,Y,X) = \sum_{k=1}^{2} \delta(Y-k) \cdot ||U^k - G_W(X)||^2$
d_1
d_2
RBF Units
$d_i = ||U^i - G_W(X)||^2$
G_W
X
Y

Figure 6.2: *The architecture of a system where two RBF units with centers U^1 and U^2 are placed on top of the machine G_W, to produce distances d_1 and d_2.*

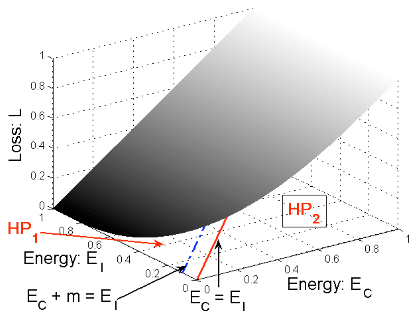
However, when the RBF centers U^1 and U^2 are not fixed and are allowed to be learned, then there is no guarantee that $d_1 + d_2 \geq d$. Then the RBF centers could become equal and the energy could become zero for all inputs, resulting in a collapsed energy surface. Such a situation can be avoided by having a contrastive term in the loss function.

Generalized Perceptron Loss

The generalized perceptron loss has a margin of zero. Therefore, it could lead to a collapsed energy surface and is not generally suitable for training energy-based models. However, the absence of a margin is not always fatal (LeCun et al., 1998a; Collins, 2002). First, the set of collapsed solutions is a small piece of the parameter space. Second, although nothing prevents



(a)



(b)

Figure 6.3: **(a)**: *When using the RBF architecture with fixed and distinct RBF centers, only the shaded region of the (E_C, E_I) plane is allowed. The non-shaded region is unattainable because the energies of the two outputs cannot be small at the same time. The minimum of the energy loss is at the intersection of the shaded region and vertical axis.* **(b)**: *The 3-dimensional plot of the energy loss when using the RBF architecture with fixed and distinct centers. Lighter shades indicate higher loss values and darker shades indicate lower values.*

the system from reaching the collapsed solutions, nothing drives the system toward them either. Thus the probability of hitting a collapsed solution is quite small.

Generalized Margin Loss

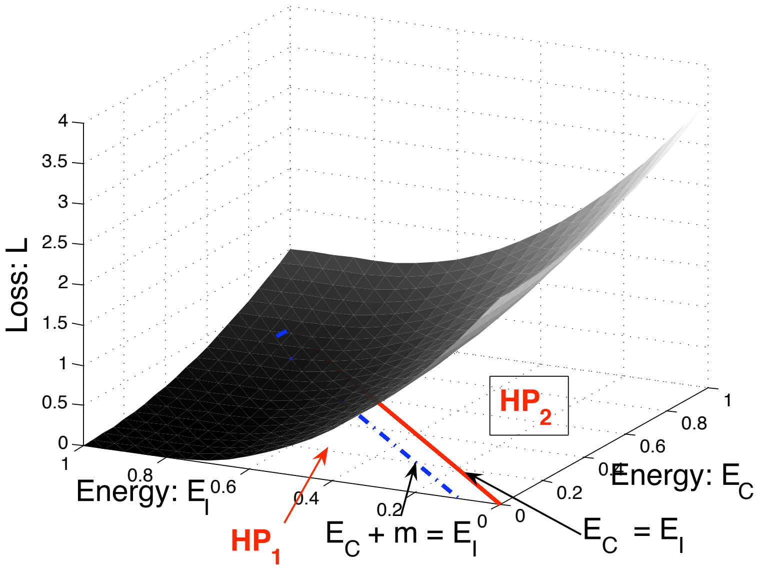


Figure 6.4: *The* square-square *loss in the space of energies E_C and E_I). The value of the loss monotonically decreases as we move from HP_2 into HP_1, indicating that it satisfies condition 6.3.*

Consider the *square-square* loss. For the two-class case, the shape of the surface of the loss function in the space of E_C and E_I is shown in Figure 6.4. One can clearly see that there exists at least one point (e_1, e_2) in HP_1 such that

$$Q_{[E_y]}(e_1, e_2) < Q_{[E_y]}(e'_1, e'_2), \tag{6.8}$$

for all points (e'_1, e'_2) in HP_2. These loss functions satisfy condition 6.3.

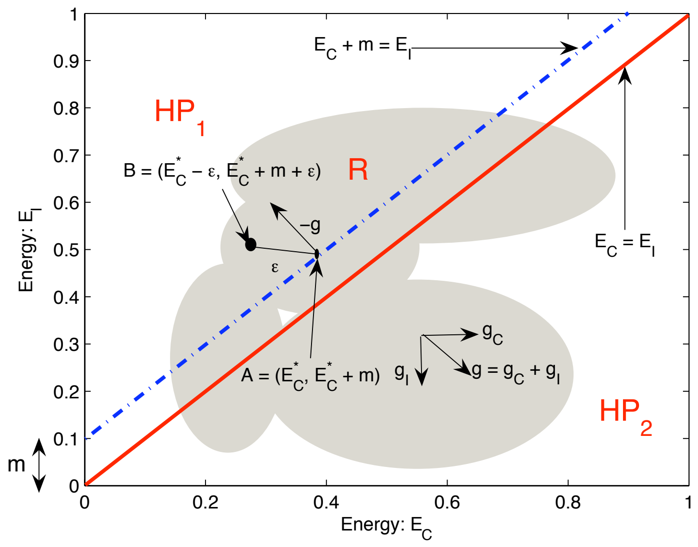


Figure 6.5: *Figure showing the direction of gradient of the negative log-likelihood loss in the feasible region R in the space defined by the two energies E_C and E_I.*

Negative Log-Likelihood Loss

It is not obvious that the negative log-likelihood loss satisfies condition 6.3, hence we formally prove that it does. For any fixed parameter W and a sample (X^i, Y^i) consider the gradient of the loss with respect to the energy E_C of the correct answer Y^i and the energy E_I of the most offending incorrect answer $\bar{Y}^i$. We have

$$g_C = \frac{\partial L(W, Y^i, X^i)}{\partial E_C} = 1 - \frac{e^{-E(W,Y^i,X^i)}}{\sum_{Y\in\mathcal{Y}} e^{-E(W,Y,X^i)}}, \tag{6.9}$$

and

$$g_I = \frac{\partial L(W, Y^i, X^i)}{\partial E_I} = -\frac{e^{-E(W,\bar{Y}^i,X^i)}}{\sum_{Y\in\mathcal{Y}} e^{-E(W,Y,X^i)}}. \tag{6.10}$$

Clearly, for any value of the energies, $g_C > 0$ and $g_I < 0$. The overall direction of the gradient at any point in the space of E_C and E_I is shown in Figure 6.5. One can conclude that when going from HP_2 to HP_1, the loss decreases monotonically.

Now we need to show that there exists at least one point in HP_1 at which the loss is less than at all the points in HP_2. Let $A = (E_C^*, E_C^* + m)$ be a point on the margin line for which the loss is minimum. E_C^* is the value of the correct energy at this point. That is,

$$E_C^* = \text{argmin}\{Q_{[E_y]}(E_C, E_C + m)\}. \tag{6.11}$$

Since from the above discussion, the negative of the gradient of the loss $Q_{[E_y]}$ at all points (and in particular on the margin line) is in the direction which is inside HP_1, by monotonicity of the loss we can conclude that

$$Q_{[E_y]}(E_C^*, E_C^* + m) \leq Q_{[E_y]}(E_C, E_I), \tag{6.12}$$

where $E_C + m > E_I$.

Consider a point B at a distance ϵ away from the point $(E_C^*, E_C^* + m)$, and inside HP_1 (see Figure 6.5). That is the point

$$(E_C^* - \epsilon, E_C^* + m + \epsilon). \tag{6.13}$$

Using the first order Taylor's expansion on the value of the loss at this point, we get

$$\begin{aligned}
&Q_{[E_y]}(E_C^* - \epsilon, E_C^* + m + \epsilon) \\
&\quad = Q_{[E_y]}(E_C^*, E_C^* + m) - \epsilon \frac{\partial Q_{[E_y]}}{\partial E_C} + \epsilon \frac{\partial Q_{[E_y]}}{\partial E_I} + O(\epsilon^2) \\
&\quad = Q_{[E_y]}(E_C^*, E_C^* + m) + \epsilon \left[\frac{\partial Q_{[E_y]}}{\partial E_C} + \frac{\partial Q_{[E_y]}}{\partial E_I} \right] \begin{bmatrix} -1 \\ 1 \end{bmatrix} + O(\epsilon^2).
\end{aligned} \tag{6.14}$$

From the previous discussion the second term on the right hand side is negative. So for sufficiently small ϵ we have

$$Q_{[E_y]}(E_C^* - \epsilon, E_C^* + m + \epsilon) < Q_{[E_y]}(E_C^*, E_C^* + m). \tag{6.15}$$

Thus we conclude that there exists at least one point in HP_1 at which the loss is less than at all points in HP_2.

Note that the energy of the most offending incorrect answer E_I is bounded above by the value of the energy of the next most offending incorrect answer. Thus we only need to consider a finite range of E_I's and the point B cannot be at infinity.

CONCLUSION

We have described a novel factor graph based framework for the problem of *relational regression*. In addition to capturing the dependencies among data point specific variables, the model also captures arbitrary observable and hidden dependencies among variables associated with different data points. When the dependencies are hidden they are inferred collectively from the data. The framework was applied to the problem of understanding house prices, where we tried to answer two major questions. First, we used the model to predict the price of a house, while at the same time capturing the relationships among houses which are spatially close to each other. Second, the model is used to construct house price indices by explicitly learning a normalized price surface over space and time, which captures the spatio-temporal relationships among transactions. In both cases we conclude that uncovering and exploiting the relational structure associated with this problem significantly improves performance.

The framework is general enough to be applied to other relational regression problems. Besides addressing the issue of doing regression in relational setting, the advantages of the model are three fold over previous work. First, it allows for inter-sample dependencies among data points through potentially continuous hidden variables. Second, it allows for log-likelihood functions that are non-linear in parameter space. Lastly, it eliminates the intractable partition function problem and provides efficient inference and learning algorithm through appropriate design of relational and non-relational factors.

BIBLIOGRAPHY

Altun, Y. and Hofmann, T. (2003). Large margin methods for label sequence learning. In *Proc. of 8th European Conference on Speech Communication and Technology (EuroSpeech)*.

Altun, Y., Johnson, M., and Hofmann, T. (2003). Loss functions and optimization methods for discriminative learning of label sequences. In *Proc. EMNLP*.

Anglin, P. M. and Gencay, R. (1996). Semiparametric estimation of a hedonic price function. *Journal of Applied Econometrics*, 11:633–648.

Bahl, L., Brown, P., de Souza, P., and Mercer, R. (1986). Maximum mutual information estimation of hidden markov model parameters for speech recognition. In *Proceedings of Acoustics, Speech, and Signal Processing Conference*, pages 49–52.

Bailey, M. J., Muth, R. F., and Nourse, H. O. (1963). A regression model for real estate price index construction. *Journal of American Statistical Association*, 58(304):933 – 942.

Basu, S. and Thibodeau, T. G. (1998). Analysis of spatial autocorrelation in home prices. *Journal of Real Estate Finance and Economics*, 16(1):61 – 85.

Bengio, Y. (1996). *Neural Networks for Speech and Sequence Recognition*. International Thompson Computer Press, London, UK.

Bengio, Y., Cardin, R., De Mori, R., and Normandin, Y. (1990). A hybrid coder for hidden markov models using a recurrent network. In *Proceeding of ICASSP*, pages 537–540.

Bengio, Y., De Mori, R., Flammia, G., and Kompe, R. (1992). Global optimization of a neural network-hidden Markov model hybrid. *IEEE Transaction on Neural Networks*, 3(2):252–259.

Bengio, Y., DeMori, R., Flammia, G., and Kompe, R. (1991). Global optimization of a neural network - hidden markov model hybrid. In *Proceedings of EuroSpeech'91*.

Bengio, Y., Ducharme, R., Vincent, P., and Jauvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155.

Bengio, Y. and Frasconi, P. (1996a). An input/output HMM architecture. In Tesauro, G., Touretzky, D., and Leen, T., editors, *Advances in Neural Information Processing Systems*, volume 7, pages 427–434. MIT Press, Cambridge, MA.

Bengio, Y. and Frasconi, P. (1996b). Input/Output HMMs for sequence processing. *IEEE Transactions on Neural Networks*, 7(5):1231–1249.

Besag, J. (1974). Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society, Series B (Methodological)*, 36(2):192–236.

Besag, J. (1975). Statistical analysis of non-lattice data. *The Statistician*, 24(3):179–195.

Besag, J. (1986). On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society, Series B (Methodological)*, 48(3):259–302.

Bottou, L. (1991). *Une Approche théorique de l'Apprentissage Connexionniste: Applications à la Reconnaissance de la Parole*. PhD thesis, Université de Paris XI, 91405 Orsay cedex, France.

Bottou, L. (2004). Stochastic learning. In Bousquet, O. and von Luxburg, U., editors, *Advanced Lectures on Machine Learning*, number LNAI 3176 in Lecture Notes in Artificial Intelligence, pages 146–168. Springer Verlag, Berlin.

Bottou, L. and Gallinari, P. (1991). A framework for the cooperation of learning algorithms.

In Touretzky, D. and Lippmann, R., editors, *Advances in Neural Information Processing Systems*, volume 3, Denver. Morgan Kaufmann.

Bourlard, H. (1990). How connectionist models could improve markov models for speech recognition. In Eckmiller, R., editor, *Proccedings of the International Symposium on Neural Networks for Sensory and Motor Systems*.

Bourlard, H. and Morgan, N. (1990). A continuous speech recognition system embedding mlp into hmm. In Touretzky, D., editor, *Advances in Neural Information Processing Systems 2*, pages 186–193. Morgan Kaufmann.

Bourlard, H. and Wellekens, C. (1988). Links between markov models and multi-layer perceptrons. In *Advances in Neural Information Processing Systems (NIPS 1)*, pages 502–510. Morgan Kauffman.

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

Bromley, J., Bentz, J. W., Bottou, L., Guyon, I., LeCun, Y., Moore, C., Sackinger, E., and Shah, R. (1993a). Signature verification using a siamese time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(4).

Bromley, J., Guyon, I., LeCun, Y., Sackinger, E., and Shah, R. (1993b). Signature verification using a siamese time delay neural network. In Cowan, J. and Tesauro, G., editors, *Advances in Neural Information Processing Systems*, volume 6. Morgan Kaufmann.

Can, A. (1990). The measurement of neighborhood dynamics in urban house prices. *Economic Geography*, 66(3):254 – 272.

Can, A. (1992). Specification and estimation of hedonic housing price models. *Regional Science and Urban Economics*, 22:453 – 474.

Case, K. E. and Shiller, R. J. (1989). The efficiency of the market for single family homes. *American Economic Review*, 32(1):125 – 137.

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

Chopra, S., Hadsell, R., and LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. In *Proc. of Computer Vision and Pattern Recognition Conference*. IEEE Press.

Clapp, J. M. (2004). A semiparametric method for estimating local house price indices. *Real Estate Economics*, 32:127 – 160.

Cohn, D. and Chang, H. (2000). Probabilistically identifying authoritative documents. *In Proc. SIGIR 2000*.

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

Collins, M. (2000). Discriminative reranking for natural language parsing. In *Proceedings of ICML 2000*.

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

Collobert, R., Weston, J., and Bottou, L. (2006). Trading convexity for scalability. In *Proceedings of the Twenty-third International Conference on Machine Learning (ICML 2006)*. IMLS/ICML. ACM Digital Library.

Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood estimation from incomplete data via the em algorithm. *Journal of Royal Statistical Society*, B39:1 – 38.

Denker, J. S. and Burges, C. J. (1995). Image segmentation and recognition. In *The Mathematics of Induction*. Addison Wesley.

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

Driancourt, X. (1994). *Optimisation par descente de gradient stochastique de systèmes modulaires combinant réseaux de neurones et programmation dynamique. Application à la reconnaissance de la parole. (optimization through stochastic gradient of modular systems that combine neural networks and dynamic programming, with applications to speech recognition)*. PhD thesis, Université de Paris XI, 91405 Orsay cedex, France.

Driancourt, X., Bottou, L., and Gallinari, P. (1991a). MLP, LVQ and DP: Comparison & cooperation. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2, pages 815–819, Seattle.

Driancourt, X., Bottou, L., and P., G. (1991b). Comparison and cooperation of several classifiers. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*.

Driancourt, X. and Gallinari, P. (1992a). Empirical risk optimisation: neural networks and dynamic programming. In *Proceedings of Neural Networks for Signal Processing (NNSP)*.

Driancourt, X. and Gallinari, P. (1992b). A speech recognizer optimaly combining learning vector quantization, dynamic programming and multi-layer perceptron. In *Proceedings of ICASSP*.

Dubin, R. A. (1992). Spatial autocorrelation and neighborhood quality. *Regional Science and Urban Economics*, 22:432 – 452.

Egghe, L. and Rousseau, R. (1990). *Introduction to Informetrics*. Elsevier.

Fisher, D. H. (1989). Noise-tolerant conceptual clustering. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 825–830, San Francisco. Morgan Kaufmann.

Franzini, M., Lee, K. F., and Waibel, A. (1990). Connectionnist viterbi training: A new hybrid method for continuous speech recognition. In *Proceedings of ICASSP*, page 245.

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

Gelfand, A. E., Ecker, M. D., Knight, J. R., and Sirmans, C. F. (2004). The dynamics of location in home prices. *Journal of Real Estate Finance and Economics*, 29(2):149 – 166.

Goetzmann, W. N. and Spiegel, M. (1995). Non-temporal components of residential real estate price appreciation. *The Review of Economics and Statistics*, 77(1):199 – 206.

Goetzmann, W. N. and Spiegel, M. (1997). A spatial model of housing returns and neighborhood substitutability. *Journal of Real Estate Finance and Economics*, 14:11 – 31.

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

Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *Proc. Computer Vision and Pattern Recognition Conference (CVPR'06)*. IEEE Press.

Haffner, P. (1993a). Connectionist speech recognition with a global MMI algorithm. Berlin.

Haffner, P. (1993b). Connectionist speech recognition with a global MMI algorithm. In *Eurospeech'93*, Berlin.

Haffner, P., Franzini, M., and Waibel, A. H. (1991). Integrating time-alignment and neural networks for high performance continuous speech recognition. In *Proceeding of ICASSP*, pages 105–108. IEEE.

Haffner, P. and Waibel, A. H. (1991). Time-delay neural networks embedding time alignment: a performance analysis. Genova, Italy.

Haffner, P. and Waibel, A. H. (1992). Multi-state time-delay neural networks for continuous speech recognition. volume 4, pages 579–588. Morgan Kaufmann, San Mateo.

Halvorsen, R. and Pollakowski, H. O. (1981). Choice of functional form for hedonic price equations. *Journal of Urban Economics*, 10:37–49.

Heckerman, D., Meet, C., and Koller, D. (2004). Probabilistic models for relational data. Technical report, MSR-TR-2004-30.

Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14:1771–1800.

Hofmann, T. and Puzicha, J. (1999). Latent class models for collaborative filtering. *In Proc. IJCAI99*.

Jaakkola, T., Diekhans, M., and Haussler, D. (2000). A discriminative framework for detecting remote protein homologies. *Journal of Computational Biology*, 7((1,2)):95–114.

Juang, B.-H., Chou, W., and Lee, C.-H. (1997). Minimum classification error rate methods for speech recognition. *IEEE Transactions on Speech and Audio Processing*, 5(3):257–265.

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

Kersting, K., Raedt, L. D., and Kramer, S. (2000). Interpreting bayesian logic programs. *In Proc. AAAI-2000 Workshop on Learning Statistical Models from Relational Data*.

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

Kohonen, T. (1995). Self organizing maps. Springer Verlag, Germany.

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

Konig, Y., Bourlard, H., and Morgan, N. (1996). REMAP: Recursive estimation and maximization of A posteriori probabilities — application to transition-based connectionist speech recognition. In Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E., editors, *Advances in Neural Information Processing Systems*, volume 8, pages 388–394. The MIT Press.

Kschischang, F., Frey, B., and Loeliger, H.-A. (2001a). Factor graphs and the sum-product algorithm. *IEEE Trans. Information Theory*, 47(2):498–519.

Kschischang, F. R., Frey, B., and Loeliger, H. A. (2001b). Factor graphs and sum product algorithm. *IEEE Transactions on Information Theory*, 47:498 – 519.

Kumar, S. and Hebert, M. (2004). Discriminative fields for modeling spatial dependencies in natural images. In Thrun, S., Saul, L., and Schölkopf, B., editors, *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA.

Lafferty, J., McCallum, A., and Pereira, F. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. International Conference on Machine Learning (ICML)*.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998a). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

LeCun, Y., Bottou, L., Haffner, P., and Howard, P. (1998b). Djvu: a compression method for distributing scanned documents in color over the internet. In *Color 6*. IST.

LeCun, Y., Bottou, L., Orr, G., and Muller, K. (1998c). Efficient backprop. In Orr, G. and K., M., editors, *Neural Networks: Tricks of the trade*. Springer.

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

LeCun, Y. and Huang, F. (2005). Loss functions for discriminative training of energy-based models. In *Proc. of the 10-th International Workshop on Artificial Intelligence and Statistics (AIStats'05)*.

Ljolje, A., Ephraim, Y., and Rabiner, L. R. (1990). Estimation of hidden markov model parameters by minimizing empirical error rate. In *Proc. of International Conference on Acoustics, Speech, and Signal Processing*, pages 709–712.

MacKay, D. J. C. (2003). *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press. Available from `http://www.inference.phy.cam.ac.uk/mackay/itila/`.

McCallum, A., Freitag, D., and Pereira, F. (2000). Maximum entropy markov models for information extraction and segmetnation. In *Proc. International Conference on Machine Learning (ICML)*, pages 591–598.

McDermott, E. (1997). *Discriminative Training for Speech Recognition*. PhD thesis, Waseda University.

McDermott, E. and Katagiri, S. (1992). Prototype-based discriminative training for various speech units. In *Proceedings of ICASSP-92, San Francisco, CA, USA*, pages 417–420.

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

Mohri, M. (1997). Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2):269–311.

Morgan, N. and Bourlard, H. (1995). Continuous speech recognition: An introduction to the hybrid hmm/connectionist approach. *IEEE Signal Processing Magazine*, 12(3):25–42.

Muggleton, S. (2000). Learning stochastic logic programs. *In Proc. AAAI-2000 Workshop on Learning Statistical Models from Relational Data*.

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

Neville, J. and Jensen, D. (2007). Relational dependency networks. *Journal of Machine Learning Research*, 8:653 – 692.

Ng, R. and Subrahmanian, V. (1992). Probabilistic logic programming. *Information and Computation*, 101(2):150 – 201.

Ngo, L. and Haddaway, P. (1997). Answering queries from context-sensitive probabilistic knowledge bases. *Theoretical Computer Science*, 171:147 – 171.

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

Osadchy, R., Miller, M., and LeCun, Y. (2005). Synergistic face detection and pose estimation with energy-based model. In *Advances in Neural Information Processing Systems (NIPS 2004)*. MIT Press.

Pace, K. R., Barry, R., Clapp, J. M., and Rodriquez, M. (1998). Spatio-temporal autoregressive models of neighbourhood effects. *Journal of Real Estate Finance and Economics*, 17(1):15 – 33.

Pace, K. R. and Gilley, O. (1997). Using the spatial configuration of the data to improve estimation. *Journal of Real Estate Finance and Economics*, 14(3):333 – 340.

Poole, D. (1993). Probabilistic horn abduction and bayesian networks. *Artificial Intelligence*, 64(1):81 – 129.

Poole, D. (1997). The independent choice logic for modelling multiple agents under uncertainty. *Artificial Intelligence*, 94(1-2):5 – 56.

Richardson, M. and Domingos, P. (2006). Markov logic networks. *Machine Learning*, 62:107 – 136.

Sakoe, H., Isotani, R., Yoshida, K., Iso, K., and Watanabe, T. (1988). Speaker-independant word recognition using dynamic programming neural networks. In *Proceedings of ICASSP-88, New York*, pages 107–110.

Sato, T. (1995). A statistical learning methods for logic programs with distribution semantics. *In Proc. of the International Conference on Inductive Logic Programming*.

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

Solla, S., Levin, E., and Fleisher, M. (1988). Accelerated learning in layered neural networks. *Complex Systems*, 2(6):625–639.

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

Taskar, B., Guestrin, C., and Koller, D. (2003). Max-margin markov networks. In *Proc. NIPS*.

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

Teh, Y. W., Welling, M., Osindero, S., and E., H. G. (2003). Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 4:1235–1260.

Thibodeau, T. G. (2003). Marking single-family property values to market. *Real Estate Economics*, 31(1):1 – 22.

Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer Verlag.

Vapnik, V. N. and Bottou, L. (1993). Local algorithms for pattern recognition and dependencies estimation. *Neural Computation*, 5(6):893 – 909.

Vishwanathan, S. V. N., Schraudolph, N. N., Schmidt, M. W., and Murphy, K. P. (2006). Accelerated training of conditional random fields with stochastic gradient methods. In *Proceedings of the Twenty-third International Conference on Machine Learning (ICML 2006)*. IMLS/ICML.

Wang, C.-C. and Guo, D. (2006). Belief propagation is asymptotically equivalent to map estimation for sparse linear systems. *Forthy-Fourth Annual Allerton Conference, Allerton House, UIUC, Illinois, USA*.

Winston, P. (1975). Learning structural description from examples. *The Psychology of Computer Vision*, pages 157 – 209.

Woodland, P. and Povey, D. (2000). Large scale discriminative training for speech recognition. In *Proc. ASR*.

Yedidia, J., Freeman, W., and Weiss, Y. (2005). Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312.

Zheng, X. P. (1991). Metropolitan spatial structure and its determinants: A case study of tokyo. *Urban Studies*, 28:87 – 104.