Discovering the Hidden Structure of House Prices with a Non-Parametric Latent Manifold Model

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

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

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

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

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

ABSTRACT

In many regression problems, the variable to be predicted depends not only on a sample-specific feature vector, but also on an unknown (latent) manifold that must satisfy known constraints. An example is house prices, which depend on the characteristics of the house, and on the desirability of the neighborhood, which is not directly measurable. The proposed method comprises two trainable components. The first one is a parametric model that predicts the "intrinsic" price of the house from its description. The second one is a smooth, non-parametric model of the latent "desirability" manifold. The predicted price of a house is the product of its intrinsic price and desirability. The two components are trained simultaneously using a deterministic form of the EM algorithm. The model was trained on a large dataset of houses from Los Angeles county. It produces better predictions than pure parametric and non-parametric models. It also produces useful estimates of the desirability surface at each location.

Categories and Subject Descriptors

I.5.1 [**Pattern Recognition**]: Models—*Structural, Statistical, Neural Nets*; I.2.6 [**Artificial Intelligence**]: Learning—*Parameter Learning*

General Terms

Algorithm, Experimentation, Performance

Keywords

Energy-Based Models, Structured Prediction, Latent Manifold Models, Expectation Maximization

1. INTRODUCTION

In a number of real world regression problems, the variables to be predicted not only depend on the features specific to the given sample, but also on a set of other variables that are not known during training. These unknown variables usually have some structural constraints associated with them. One can use these constraints to infer their values from the data. The problem of real estate price prediction falls into such a class of problems. It involves predicting 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, like the number of bedrooms, the number of bathrooms, the living area, etc. They could also include information about the area or the neighbourhood in which the house lies. For example, features could include census tract specific information like the average household income of the neighbourhood, average commute time to work etc. Features could also include school district information.

This problem has a strong underlying spatial structure associated with it, which when exploited can improve the prediction performance of the system. The price of a house is obviously influenced by its individual characteristics. Given a particular locality, a large house with 3 bedrooms and 2 bathrooms will be more expensive compared to a smaller house with 1 bedroom and 1 bathroom. However, in addition to the dependence on its individual features, the price is also influenced by the so called "desirability" of its neighbourhood. The price of a house is primarily determined by the price of similar houses in the vicinity. For example, a house with the same set of features, say 3 bedrooms and 2 bathrooms, will have a higher value if located in an upscale neighbourhood than if it were in a poor neighbourhood. We say that the upscale locality has a higher "desirability" than the poor neighbourhood, and hence houses will generally have higher prices. This desirability has a strong structure associated with it, namely the spatial smoothness. The desirability of a location should change gradually when moving from one neighbourhood to the adjacent one. Hence it can be viewed as a smooth surface in a 3D space where the first two coordinates are GPS coordinates and the third coordinate is the desirability value. However note that this desirability surface is not directly measurable. Its value is only indirectly reflected in the selling prices of similar, nearby houses. While the actual desirability is hidden (latent) and not given during training, the smoothness constraint associated with it can help us infer it from the data.

This paper addresses the problem of predicting the house prices by modeling and learning such a desirability surface. However we note that the model proposed is very general and can be applied to other problems that fall into the class of regression problems described above. The proposed model has two components.

- The first component is a non-parametric model that models such a latent desirability manifold.
- The second component is a parametric model that only considers the individual features of the house and produces an estimate of its "intrinsic price".

Prediction of a sample is given by combining the value of the manifold at its location, along with the description of the sample (output of the parametric model). In addition, the paper also proposes a novel learning algorithm that simultaneously learns both the parameters of the parametric "intrinsic price" model and the desirability manifold.

The first component models the latent desirability surface in a non-parametric manner. The idea is to associate a single desirability coefficient to each training sample. The value of the manifold at any point is obtained by interpolation on the coefficients of the training samples that are local to that point (according to some distance measure). The way this interpolation is done is problem dependent. In fact the interpolation algorithm plays an important role in the performance of the system. There is no restriction imposed on the nature/architecture of the second component. The question remaining is, how to learn the desirability coefficients associated with each training sample.

We propose a novel energy-based learning algorithm, which we call Latent Manifold Estimation (LME), that learns the desirability coefficients of the first component and the parameters of the second component simultaneously. The algorithm consists of iterating alternatively through two phases until convergence. It can be seen as a deterministic form of generalized EM method [7]. In the first phase, the parametric model is kept fixed and the desirability coefficients are learned by minimizing a loss function while at the same time preserving the smoothness. This phase is similar to the expectation phase of the EM algorithm. The second phase fixes the desirability coefficients and learns the parameters of the first component. This is similar to the maximization phase of the EM algorithm. As in the case of generalized EM, in both the phases the loss is not fully minimized but merely decreased up to a certain threshold. The algorithm iterates through the two phases alternatively until convergence. The algorithm is energy-based [15], in the sense that while training we only minimize over the latent variables and not marginalize over their distribution. Moreover our aim is to achieve good prediction accuracy and not to estimate the underlying distribution of the input samples.

1.1 Previous Work

The problem of predicting prices of real estate properties has a long history in the economics literature. Linear parametric methods and their derivatives have been long used by Goodman [11], and Hallvorsen and Pollakowski [12]. An extension of the linear regression is the Box-Cox transformations proposed by Box and Cox [3]. 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 [16] used locally weighted regressions, whereas Clapp in [6] and Anglin and Gencay [1] 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. Can in [4, 5], model house prices using spatial autoregressions. Dubin [9], Pace and Giley [19], and Basu and Thibodeau [2] 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 [18] and Gelfand et al [10].

However, throughout the economics literature very little emphasis is given on predictability. The focus is more towards estimating the model parameters efficiently and precisely and on index construction. Very little has been done to handle the problem purely 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 (example dataset etc) is not representative of the real world situation. For instance, Do and Grudnitski [8], and Nguyen and Cripps in [17] have used very simple neural networks on a very small dataset. In contrast, the dataset used in the present paper is considerably larger and more diverse and the learning architecture is considerably more flexible. Some work has been done to automatically exploit the locality structure present in the problem. Kauko in [13], used the Self Organizing Map (SOM) technique proposed by Kohonen [14] 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. To the best of our knowledge, the method we propose is the first attempt to automatically learn the influence of the underlying spatial structure inherent in the problem, and use it for prediction. A key characteristic of our learning algorithm is that it learns both the parametric and non parametric models simultaneously.

2. THE LATENT MANIFOLD MODEL

In this section we give the details of the architecture, the training and the inference of the latent manifold model in the light of predicting house prices. Simultaneously we point out that the model is general enough to be used for other problems that have similar characteristics.

2.1 The Architecture

Let $\mathcal{S} = \{(X^1, Y^1), \ldots, (X^n, Y^n)\}$ be the set of labeled training samples. In house price prediction, the input X^i consists of the set of features associated with the house P^i,

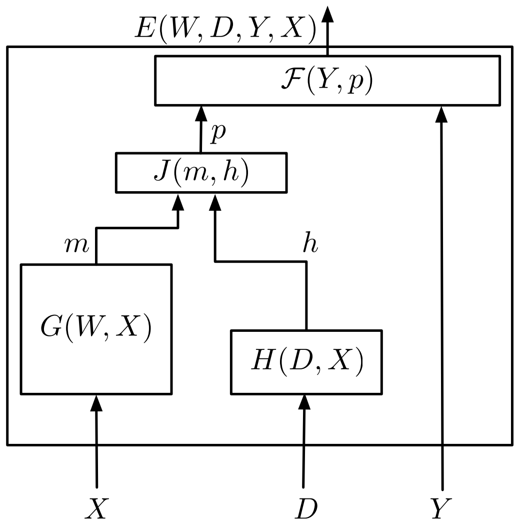


Figure 1: The regression model is composed of two learning modules: the parametric model $G(W, X)$ and the non parametric model $H(D, X)$.

such as the number of bedrooms, number of bathrooms etc. The full list of house specific features that were used in the experiments is discussed in section 3. The architecture of the model is shown in figure 1. It consists of two trainable components. The first component is the parametric function $G(W, X)$, parameterized with W, that takes X as input and produces an output $m = G(W, X)$. This output can be interpreted as the "intrinsic price" of the sample. Other than differentiability with respect to W, no restriction is imposed on the form/architecture of this function. For the house price prediction application, the function $G(W, X)$ was a fully connected neural network with two hidden layers.

The second component $H(D, X)$ models the latent manifold and is non-parametric in nature. In the light of house price prediction, as pointed out in the previous section, the price of the house P with a set of features X, is strongly affected by the "desirability" of its neighbourhood. This "desirability" (which can be viewed as a smooth manifold spanning the concerned geographic region) is modeled by assigning a "desirability coefficient" d^i to each training sample X^i. One can interpret the value of the coefficient d^i as a number specifying how desirable the location of the house that corresponds to X^i is. The higher the value, the more desirable it is. Since the value of these coefficients is not known during training and should be learned, one can view these coefficients as the latent variables associated with the model. For a detailed discussion of latent variable energy-based architectures refer to [15]. Denote by $D = [d^1, \ldots, d^n]$ the vector of all such desirabilities. The hidden desirability manifold is modeled by the desirability vector D and the non-parametric model $H(D, X)$. This function takes as input the sample X (not necessarily a training sample) and the desirability vector D and produces an output $h = H(D, X)$, which gives the estimate of the desirability of the location of sample X as a function D. How this estimate is computed plays a crucial role in the performance of the system. Hence one must take special care while designing $H(D, X)$. In the present paper, two versions were used: one was a simple kernel-based interpolating function and the other was a weighted local linear regression model.

2.1.1 Kernel Based Interpolation

For a sample X, let $\mathcal{N}(X)$ denote the set of indices of K training samples that are closest to X (according to some pre-determined similarity measure), for some K. Then the output h of the function $H(D, X)$ is given by

$$h = H(D, X) = \sum_{j \in \mathcal{N}(X)} Ker(X, X^j) \cdot d^j. \tag{1}$$

The kernel function $Ker(X, X^j)$ is defined as

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

with q a constant.

2.1.2 Weighted Local Linear Regression

Another method used for computing the output of the function $H(D, X)$ involves fitting a weighted local linear regression model in the set of neighbouring training samples to the sample X. Let α be the parameter vector and β be the bias of the local linear model. Then fitting a weighted local linear model in the set of neighbouring training samples, amounts to finding the parameters α^* and the bias β^*, such that

$$(\beta^*, \alpha^*) = \arg\min_{\beta, \alpha} \sum_{j \in \mathcal{N}(X)} (d^j - (\beta + \alpha X^j))^2 Ker(X, X^j). \tag{3}$$

The function $Ker(X, X^j)$ could be any parametric kernel appropriate to the problem. However the one used for the experiments was the same as the one given in equation 2. Then the solution to the system (or the output of the function $H(D, X)$) is given by

$$h = H(D, X) = \beta^* + \alpha^* X. \tag{4}$$

Remark 1. Even in this case the output h of the function $H(D, X)$ for a sample X can be expressed as a linear combination of the desirabilities d^j of the training samples X^j that lie in the neighbourhood of X, such that the linear coefficients do not depend on the desirabilities. That is

$$h = H(D, X) = \sum_{j \in \mathcal{N}(X)} a^j d^j. \tag{5}$$

The outputs m and h are combined using a function J to get the prediction of the input sample X. Particularly in the present paper, instead of predicting the actual prices of the houses, the model predicts the log of the prices. This allows us to combine the intrinsic price m predicted by the parametric model $G(W, X)$ and the desirability coefficient h predicted by the non-parametric model $H(D, X)$ additively, rather than multiplicatively. Thus the function J takes the simple form

$$\begin{aligned} J(m, h) &= J(G(W, X), H(D, X)) & (6) \\ &= G(W, X) + H(D, X). & (7) \end{aligned}$$

Another advantage of predicting the log of the price instead of the actual price is that the absolute prediction error in the log price corresponds to the relative prediction error in the actual price, which is what we care about.

Finally the discrepancy between the predicted log price p and the actual log price Y is given by the energy function $E(W, D, Y, X)$. The energy function used for the experiment was half of the square of euclidean distance.

$$\begin{aligned} E(W, D, Y, X) &= \frac{1}{2}(Y - (m + h))^2 & (8) \\ &= \frac{1}{2}(Y - (G(W, X) + H(D, X)))^2. & (9) \end{aligned}$$

2.2 The Learning Algorithm

Given the training set $\mathcal{S} = \{(X^1, Y^1), \ldots, (X^n, Y^n)\}$, the objective of the training is to simultaneously find the parameters W and the desirability coefficients D (the latent variables of the system) such that the sum of the energy over the training set is minimized. This is done by minimizing the following loss function over W and D

$$\begin{aligned} \mathcal{L}(W, D) &= \sum_{i=1}^{n} E(W, D, Y^i, X^i) + R(D) & (10) \\ &= \sum_{i=1}^{n} \frac{1}{2}(Y^i - (G(W, X^i) + H(D, X^i)))^2 + R(D) & (11) \end{aligned}$$

$R(D)$ is a regularizer on D that prevents the desirabilities from varying wildly, and helps keep the surface smooth. In the experiments an L_2 regularizer $R(D) = \frac{r}{2}||D||^2$ was used, where r is the regularization coefficient.

The learning algorithm is iterative and can be seen as a deterministic form of an EM (a coordinate descent method) algorithm, where D plays the role of auxiliary variables. The idea is to break the optimization of the loss $\mathcal{L}$ with respect to W and D into two phases. In the first phase the parameters W are kept fixed and the loss is minimized with respect to D (the expectation phase of EM). The second phase involves fixing the parameters D and minimizing the loss with respect to W (maximization phase of EM). The training proceeds by iterating through each of the two phases alternatively until convergence. We now explain the details of the two training phases for the experiments in this paper.

Phase 1 It turns out that the loss function given by equation 10 is quadratic in D and the process of minimizing it reduces to solving a large scale sparse quadratic system. Associate with each training sample X^i a vector U^i of size n (equal to the number of training samples). This vector is very sparse and has only K non zero elements, whose indices are given by the elements of the neighbourhood set $\mathcal{N}(X)$. The value of the j^{th} non-zero element of this vector is equal to the linear coefficient that is multiplied with the desirability d^j while estimating the desirability of X. Thus, when the kernel based interpolation is used (equation 1) then it is equal to $Ker(X, X^j)$, and when local linear regression model is used (equation 5) then it is a^j. The loss function (see Equation 10) for this phase can now be written as the following sparse quadratic program

$$L_1(D) = \frac{r}{2}||D||^2 + \frac{1}{2}\sum_{i=1}^{n}(Y^i - (m^i + D^T U^i))^2. \qquad (12)$$

There are two things in particular that one should be careful about this loss function.

- During the process of training with sample X^i, we must interpolate its desirability h^i using the desirabilities of the training samples whose indices are given by the set $\mathcal{N}(X^i)$. However, special care must be taken to ensure that the index of the sample X^i itself is removed from the set $\mathcal{N}(X^i)$. Not doing so can result in a system that trivially sets the desirability of each training sample equal to the price to be predicted. This would make loss equal to zero, but lead to poor predictions.
- The regularization term plays the crucially important role of ensure that the surface be smooth. Without it, the system would overfit the training data and learn a highly-varying surface, leading to poor predictions.

Another possible modification to the loss given in equation 12, includes an explicit self-consistency term. The idea is to have an explicit constraint that will drive the estimate h^i of the desirability of training sample X^i given by $h^i = D^T U^i$ to its assigned desirability d^i. Note that the estimate h^i does not involve the term d^i. Hence the loss function now becomes

$$L_1(D) =$$
$$\frac{r_1}{2}||D||^2 + \frac{1}{2}\sum_{i=1}^{n}(Y^i - (m^i + D^T U^i))^2 + \frac{r_2}{2}\sum_{i=1}^{n}(d^i - D^T U^i)^2. \qquad (13)$$

Here r_1 and r_2 are some constants. This loss function is still a sparse quadratic program and can be solved in the same way as above.

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

Phase 2 This phase involves updating the parameters W of the function $G(W, X)$ by running a standard stochastic gradient decent algorithm for all the samples (X^i, Y^i) in the training set, keeping D fixed. For a sample X^i the forward propagation was composed of the following steps. Run X^i through the function G to produce the log of the intrinsic price $m^i = G(W, X^i)$. Interpolate the desirability h^i of X^i from its neighbours using the function $h^i = H(D, X^i)$. Add the desirability to the intrinsic price to get the prediction $p^i = m^i + h^i$. Compare the predicted value p^i with the actual value Y^i to get the energy $E(W, D) = \frac{1}{2}(Y^i - p^i)^2$. Finally, the gradient of the energy with respect to W is computed using the back propagation step and the parameters W are updated. Here again, we do not train the system to completion, but rather stop the training after a few epochs.

The algorithm for training the latent manifold model is summarized in algorithm 1.

2.3 The Testing Algorithm

Testing the input sample X involves a single forward propagation step through the system to compute the predicted price $p = m + h = G(W, X) + H(D, X)$, using the learned parameters W and the manifold variables D. This prediction is compared with the desired value Y to get the error on the current sample. The exact measure of error used was the Absolute Relative Forecasting error and is discussed in section 4.

3. EXPERIMENTS

The model proposed in this paper was trained on a very large and diverse dataset. In this section we describe the details of the dataset. In addition, we also discuss the details of the various standard techniques that have been used for the problem, with which the performance of the model was compared. The results of the various techniques are given in the next section.

3.1 The Dataset

The dataset used was obtained from First American. The original dataset has around 750,000 transactions of single-family houses in Los Angeles county. The transactions range from the year 1984 to 2004. The dataset has a very heterogeneous set of homes spread over an area of more than 4000 sq miles, with very different individual characteristics. Each house is described by a total of 125 attribute variables. The attributes specific to the home include, number of bedrooms, bathrooms, the living area of the house, the year built, the type of property (single family residence etc), number of stories, number of parking spaces, presence of a swimming pool, number of fire places, type of heating, type of air conditioning, material used for making the foundations etc. In addition to this, there are financial attributes including the taxable land values. Each house is also labeled with a set of geographic information like its mailing address, the census tract number, and the name of the school district in which the house lies.

In our experiments, we only considered the transactions that took place in the year 2004. For the homes transacting in 2004, the three geographic fields were used to append neighborhood and GPS (latitude and longitude) information to the database. First, the mailing address was used to extract GPS co-ordinates for each home. For neighborhood information, we used the year 2000 census 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.

The dataset is diverse even in terms of the neighbourhood characteristics with transactions spreading across 1754 census tracts and 28 school districts.The smallest census tracts have as few as 1 transaction, while the biggest census tract has 350 transactions. The biggest school district in Los Angeles county is Los Angeles Unified with 25,251 transactions. Other school districts have between 200 and 1500 transactions.

Out of the numerous house specific and neighbourhood-specific attributes associated with each house, we only considered number of bedrooms, number of bathrooms, year of construction, living area, median house hold income, employment accessibility index, proportion of units owner occupied, and the academic performance index (API). All those houses that had missing values for at least one or more of these attributes were filtered from the data. In addition, only single family residences were considered. After the filtering, there were a total of 70,816 labeled houses in the dataset. Out of these, 80% of them (56,652 in total) were randomly selected to be used for training purpose and the remaining 20% (14,164) were used for testing.

Algorithm 1 LME Trainer

Input: training set $S = \{(X^i, Y^i) : i = 1\,to\,N\}$
Initialization: W to random values and D to 0
repeat
  Phase 1
  Fix the value of W
  Solve the quadratic program using conjugate-gradient method with early stopping to get a new value of D
  Phase 2
  Fix the value of D
  for $i = 1$ **to** N **do**
    Make a forward pass through the model to compute the energy
    Make a backward pass to compute the gradients of energy with respect to W
    Update the parameters W
  end for
until *convergence*

3.2 The Latent Manifold Model

The training and testing of the model was done in the way described in the previous section. The function $G(W, X)$ was chosen to be a 2 hidden layer fully connected neural network. The first hidden layer had 80 units, and the second hidden layer had 40 units. The network had 1 output unit that gave the "intrinsic price" of the house. For the function $H(D, X)$, both the modeling options - kernel smoothing, and weighted local linear regression - were tried. In the case of kernel smoothing, the value of K, which gives the size of the neighbourhood set $\mathcal{N}(X)$, was chosen to be 13. The motivation behind such a choice was the fact that the K nearest neighbour algorithm for the problem, gave the best performance with 13 neighbours. When weighted local linear regression model was used, K was set to 20. This is because the local linear regression model, when ran on the dataset directly, performed best when the size of the neighbourhoods was 20. The optimization of the quadratic loss function (see Equation 12) was done using the conjugate gradient method with early stopping. The idea is to stop the optimization once the residual of the linear system reaches a pre-determined threshold. The value of the threshold is decreased gradually as a function of the number of iterations of the training algorithm. A number of experiments were performed using different values of the regularization coefficient r and the factor q in the kernel function. A variation of the quadratic loss that involved an additional explicit smoothing term (Equation 13) was also tried. Experiments were done with a number of different values of the coefficients r_1 and r_2. The results are reported for the best combination of values of these coefficients.

3.3 Other Models Used for Comparison

The performance of the proposed model was compared to a number of standard techniques that have been in use to solve this problem. We now briefly give a description of these techniques.

3.3.1 K - Nearest Neighbour

In this technique, the process of predicting the price of the sample X involves finding the K nearest training samples (using some similarity measure) and computing the average price of these neighbours. Two different similarity measures were tried. One measure was a euclidean distance in the input space, where the input consisted of only house specific features and no neighbourhood information like GPS, school district information, and census tract information. The other measure was also a euclidean distance but with the input having both the house specific and neighbourhood specific information. Experiments were done with different values of K and results for the best value are reported.

3.3.2 Regularized Linear Regression

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

$$R(W) = \frac{r}{2}||W||^2 + \frac{1}{2n}\sum_{i=1}^{N}(Y^i - W^T X^i)^2. \tag{14}$$

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

3.3.3 Box-Cox Transformation

An extenstion of the linear regression is the Box-Cox transform of the linear regression [3], which while maintaining the basic structure of the linear regression, allows for some nonlinearities. The quadratic Box-Cox transform of the hedonic equation is given by

$$P^{(\theta)} = \alpha_0 + \sum_{i=1}^{k}\alpha_i Z_i^{(\lambda_i)} + \frac{1}{2}\sum_{i=1}^{k}\sum_{j=1}^{k}\gamma_{ij} Z_i^{(\lambda_i)} Z_j^{(\lambda_j)}. \tag{15}$$

P is the price, Z_i are the attributes, and $P^{(\theta)}$ and $Z_i^{(\lambda_i)}$ are the Box-Cox transforms.

$$P^{(\theta)} = \frac{P^\theta - 1}{\theta}, \qquad \theta \neq 0 \tag{16}$$
$$= ln(P), \qquad \theta = 0 \tag{17}$$
$$Z_i^{(\lambda_i)} = \frac{Z_i^{\lambda_i} - 1}{\lambda_i}, \qquad \lambda_i \neq 0 \tag{18}$$
$$= ln(Z_i), \qquad \lambda_i = 0 \tag{19}$$

All popularly used functional forms in the literature from linear ($\theta = 1$), semi-log ($\theta = 0$), log linear ($\theta = 0$, $\lambda = 0$, $\gamma_{ij} = 0$), and translog ($\theta = 0$, $\lambda = 0$) etc. are all special cases of the above equation. The above system is first solved for the optimal parameters using a combination of maximum likelihood estimation and grid search on the training data.

3.3.4 Weighted Local Linear Regression

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

$$\min_{\beta(X)} \sum_{i\in\mathcal{N}(X)} K_\lambda(X, X^i)[Y^i - \beta(X)f(X^i)]^2. \tag{20}$$

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

$$P = \beta(X) \cdot f(X). \tag{21}$$

A variation of this model, called the *Varying Coefficient Model* (VCM) provides the flexibility of choosing the attributes from the input space that are to be used for regression. The idea is to pick two subsets of attributes of the input sample X. The first subset X_1 is used to make a prediction, while the second subset X_2 is used to determine the neigbors. The following loss function is minimized:

$$\min_{\beta(X_2)} \sum_{i\in\mathcal{N}(X)} K_\lambda(X_1, X_1^i)[Y^i - \beta(X_2)f(X_1^i)]^2. \tag{22}$$

We used this model to study the variation of prediction errors as a function of attributes, by trying a number of different combinations. In particular, the model was tested using only house specific attributes in X_1, using different neighbourhood attributes in X_2, like GPS coordinates.

3.3.5 Fully Connected Neural Network

A fully connected neural network also falls into the class of architectures that do not explicitly make use of the location information, which characterizes this type of data. However the motivation behind using such an architecture is to capture some non linearities that are hidden in the regression function. A number of architectures were tried, and the one that achieved the best performance was a 2-hidden layer network with 80 units in the first layer, 40 units in the second, and 1 unit in the output layer.

LME combines the best of both worlds: since there is no restriction on the function $G(W, X)$, it can be a highly complicated non linear function capturing non linearities of regression function, and at the same time D and $H(D, X)$ model the latent manifold which captures the location information associated with the data.

4. RESULTS AND DISCUSSION

The performance of the systems was measured in terms of the Absolute Relative Forecasting error (fe) [8]. Let A_i be the actual price of the house P_i, and let Pr_i be its predicted price. Then the Absolute Relative Forecasting error (fe_i) is defined as

$$fe_i = \frac{|A_i - Pr_i|}{A_i}. \tag{23}$$

Two performance quantities on the test set are reported; percentage of houses with a forecasting error of less than 5%, 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.

A comparison of the performance of various algorithms is given in table 1. The second and third columns give the performance of the algorithms when the location dependent

Table 1: Prediction accuracies of various algorithms on the test set. The second and third columns ("Without Loc") gives the results when no location dependent information was used as part of the input. The fourth and fifth column ("With Loc") give the results when the location dependent information (GPS coordinates, census tract fields and school district fields) is used in the inputs. The various versions of LME algorithms reported are: (a) LME - *kernel*: when kernel smoothing is used in $H(D, X)$. (b) LME - *llr*: when local linear regression is used in $H(D, X)$. (c) S-LME - *llr*: when local linear regression is used, and in addition to it, an explicit smoothing constraint in the quadratic loss is used.

Algorithm	Without Loc		With Loc	
	$< 5\%$	$< 15\%$	$< 5\%$	$< 15\%$
Nearest Neighbor	25.29	62.81	27.51	68.31
Linear Regression	14.04	40.75	19.99	55.20
Box-Cox	19.00	51.00	-	-
Local Regression	24.87	62.89	29.20	70.43
Neural Network	24.60	64.54	27.43	70.10
LME - *kernel*	-	-	**29.39**	**71.70**
LME - *llr*	-	-	**29.67**	**72.06**
S-LME - *llr*	-	-	**29.69**	**72.15**

information, like GPS, census tract information, and school district information is not used as part of the input to the algorithm. The fourth and fifth columns give the performance when the location information is used as part of the input. Various versions of the LME algorithm were trained and tested. Three such versions are reported. "LME - *kernel*" denotes the LME algorithm when kernel smoothing is used to model the function $H(D, X)$ (Equation 1), "LME - *llr*" means when local linear regression is used (Equation 5), and "S-LME -*llr*" means that in addition to using local linear regression, an explicit smoothing constraint in the loss function is used (Equation 13).

One can clearly see that the LME algorithm outperforms all the other algorithms. The best performing version is the "S-LME-*llr*", which predicts 29.69% of houses within an error margin of less than 5% and 72.15% of houses within an error margin of less than 15%.

4.1 Importance of Location Dependent Information

From the results given in the table, one can conclude that information dependent on the geographic location of the house, like its GPS coordinates, fields from census tract data, and fields from the school district in which the house lies, play a crucial role in predicting its price. All the algorithms perform significantly better when used with these variables than when used without them.

Another thing that is clear from the table is that it is difficult to fit a single parametric model on the entire dataset. Rather one should try to look for models in the non-parametric domain that change according to the neighbourhod. This is reflected from the fact that methods like linear regression perform very badly. Whereas a simple method like the K nearest neighbour, which is a highly local, non smooth, and a non-parametric method does a reasonable job. Adding non-linearities to the linear model does not help either, as evident from the marginally better performance of the Box-Cox method over its linear counterpart. The fully connected neural network, though not a non-parametric method, still gives good performance because of its highly non linear nature. But the fact that a simple local linear regression model on the entire input space performs better than this neural network further strengthens our belief that part of the model should be non-parametric that should take into account the locality dependent information.

Moreover, how intelligently the locality dependent information is used is also crucial in making predictions. For instance local linear regression method performs better than the non linear neural network. Again, this is so because this method fits a separate linear model on the neighbouring samples for the sample X. It is these samples that are very likely to have a huge influence on the price of X. Here note that the term "neighbouring" does not necessarily mean physical proximity. One could define a neighbourhood space that includes physical proximity (GPS coordinates), and area information (census fields and school fields). Among all the algorithms, the LME uses the location dependent information in the most sophisticated manner. As mentioned before, the price of a house not only depends on its individual characteristics but also on the so called "desirability" of the neighbourhood, which can be modeled as a smooth manifold. LME models this desirability manifold in a non-parametric manner using the function $H(D, X)$ and the desirability coefficients D assigned to each training sample, and learns them from the data. Prediction is done by combining the local desirability value obtained from the learned manifold with the description of the house (the "intrinsic price" obtained from the parametric model). This process is very intuitive and highly reflective of the real world situation. From the table one can see that all the versions of LME perform better than the rest of the algorithms. In particular "LME - *llr*" performs better than "LME - *kernel*. This indicates that the way the locality dependent information is used is very crucial to the performance of the system. The best performance of "S-LME - *llr*" speaks in favor of smooth desirability manifolds as opposed to non-smooth ones.

4.2 The Desirability and Sensitivity Maps

In this section we give some discussion that provides insights into the working of LME 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 2 gives the color coded prediction error map on the test samples. Each point in the figure corresponds to a house in the test sample and is superimposed on top of the satellite image of Los Angeles county using its GPS coordinates. The points are colored according to the error in prediction made by LME. The blue color corresponds to lower prediction error and the red color corresponds to higher error. In order to make the picture discernible, the color of each point is smoothed out by assigning it the average color of its 15 nearest neighbours. As one can see, the errors made by LME are not random and seem to have a pattern. In particular, the algorithm does a fairly good job on the out skirts of the county. However it is in and around the central part (near the downtown area), that it makes a lot of mistakes. This could be attributed to the fact that their is a very high variability in the data around the central part, and the handful of attributes used in the experiments might not be enough to capture such a variability.

We also show the desirability map learned by the LME algorithm (Figure 3(a)). 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 D of the training samples and the function $H(D, X)$, 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.

Finally, we provide some sensitivity analysis; which involves measuring the change in the predicted price for a sample when the value of one of its attribute is perturbed by a small amount. The motivation behind such an analysis is to check whether the learning algorithm is able to capture in a meaningful way, the non-linearities that are hidden in the prediction function with respect to the particular attribute. For a sample X, the "sensitivity value" Sv_X associated with it with respect to some attribute is computed as follows. First the original price is predicted using the actual values of the attributes of X. This is denoted by Pr_{orig}. Next, the value of the attribute with respect to which the sensitivity is sought, is increased by one unit. For example, if the attribute is the number bedrooms, then its value is incremented by 1. Next the price of the sample X is predicted again using the same machine parameters but with a perturbed attribute value. This is denoted by Pr_{purt}. Finally the "sensitivity value" Sv_X is computed, which is given by

$$Sv_X = \frac{Pr_{purt} - Pr_{orig}}{Pr_{orig}}. \tag{24}$$

The value of Sv_X can be interpreted as the expected gain in the price of the house when its corresponding attribute is changed by one unit. This information is very important in solving a seller's dilemma - whether making certain modification to his/her house before selling would increase its value or not.

The experiments were done using the number of bedrooms as the concerned attribute. For every house in the test set, its number of bedrooms were increased by 1 and the "sensitivity value" computed. The results are shown in the form of a color coded map in figure 3(b). The blue color implies lower values of Sv_X; which means that the price of the house will not change by much even when an additional bedroom is added to it. The red color implies higher values of Sv_X and indicates that the price of the house will change substantially when a new bedroom is added. From the map, one can see that the prices in suburban areas of the county are not as sensitive to an increase in the number of bedrooms as the central (more congested) parts. Thus we conclude that LME indeed is able to capture the correct non-linear relationship between the number of bedrooms and the price of the house.

5. CONCLUSIONS

In this paper, we proposed a new approach to regression for a class of problems in which the variables to be predicted, in addition to depending on features specific to the sample itself, also depend on an underlying hidden manifold. Our approach, called LME, combines a trainable parametric model and a non-parametric manifold model to make a prediction. We give a novel learning algorithm that learns both models simultaneously. The algorithm was applied to the problem of real estate price prediction, which falls into such a class of problems. The performance of LME was compared with a number of standard parametric and non-parametric methods. The advantages of the LME approach was demonstrated through the desirability map and sensitivity map infered by the model. These show that the algorithm is indeed doing something that is a reflection of a real world situation.

Finally, we emphasize that the model proposed here is quite general and can be applied to any regression problem which can be modeled as depending upon an underlying non-parametric manifold. The real estate prediction model can easily extended to include temporal dependencies, so as to learn a spatio-temporal latent manifold in the GPS+Time space. Such a manifold would be able to capture the influence on the individual price of a house of neighbourhood factors, and also of temporal factors such as local market fluctuations.

6. ADDITIONAL AUTHORS

7. REFERENCES

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

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

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

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

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

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

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

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

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

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

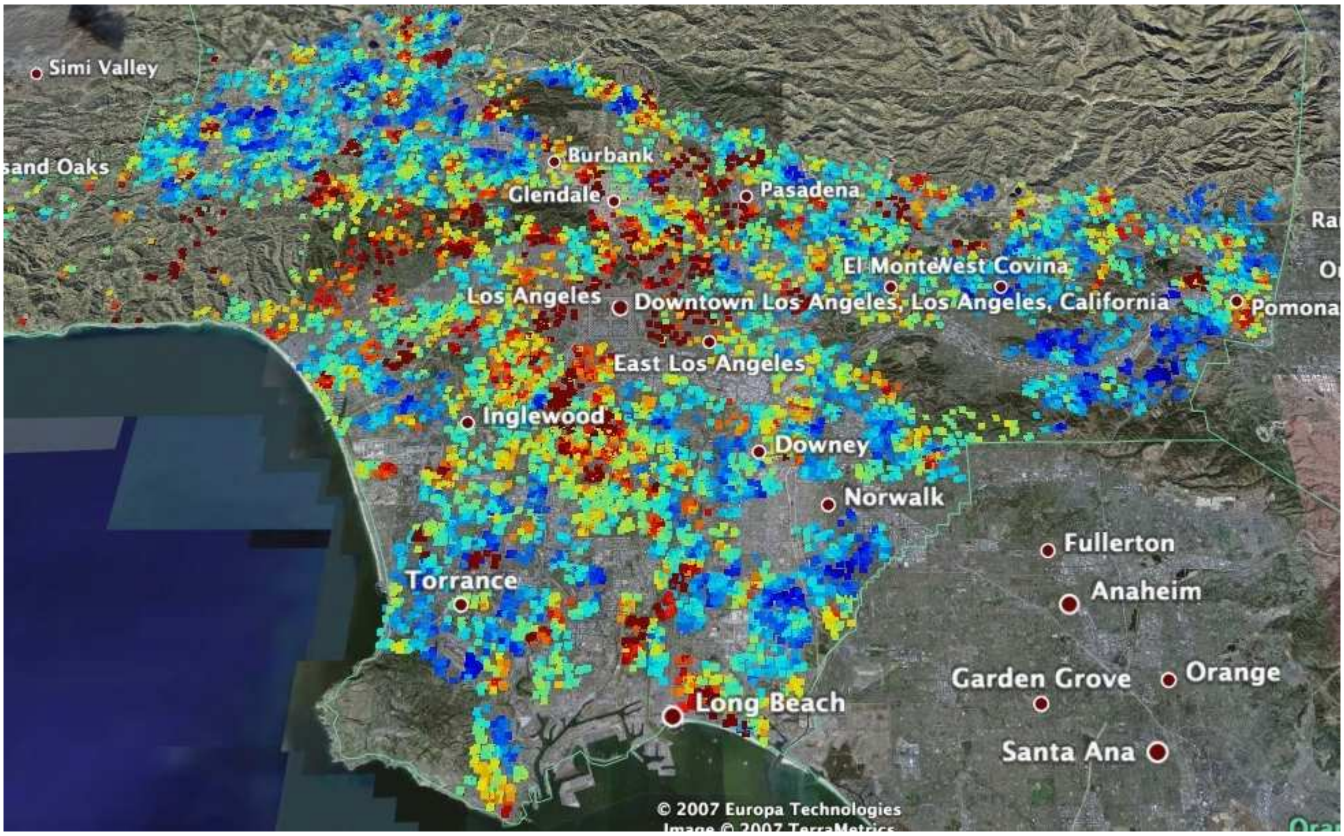


Figure 2: The color coded map of prediction error on test samples. Each point corresponds to a house in the test set and is colored according to its prediction error. Blue color corresponds to smaller error and red color corresponds to larger error.

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

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

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

[14] T. Kohonen. Self organizing maps. Springer Verlag, Germany, 1995.

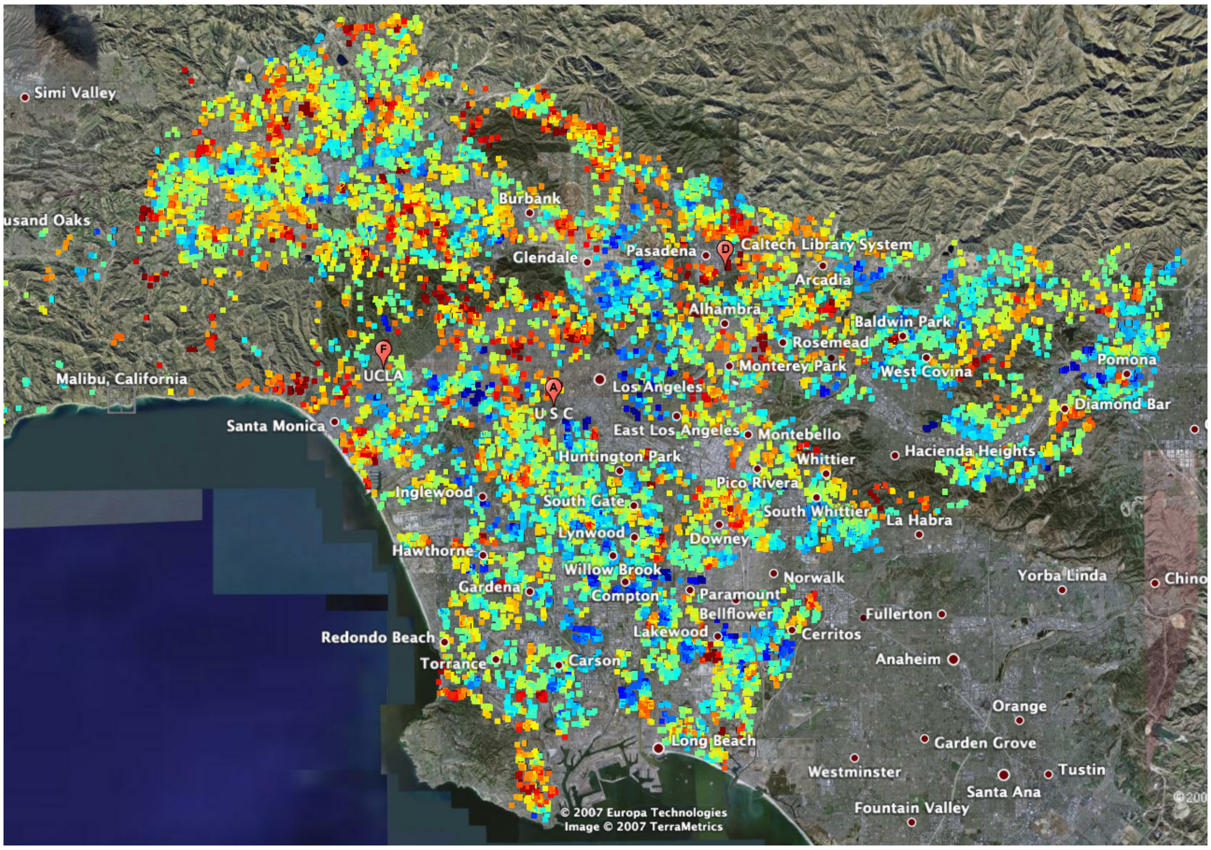
[15] Y. LeCun, S. Chopra, R. Hadsell, F. J. Huang, and M. Ranzato. A tutorial on energy-based learning. In B. et al, editor, *Predicting Structured Outputs*. MIT Press, 2006.

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

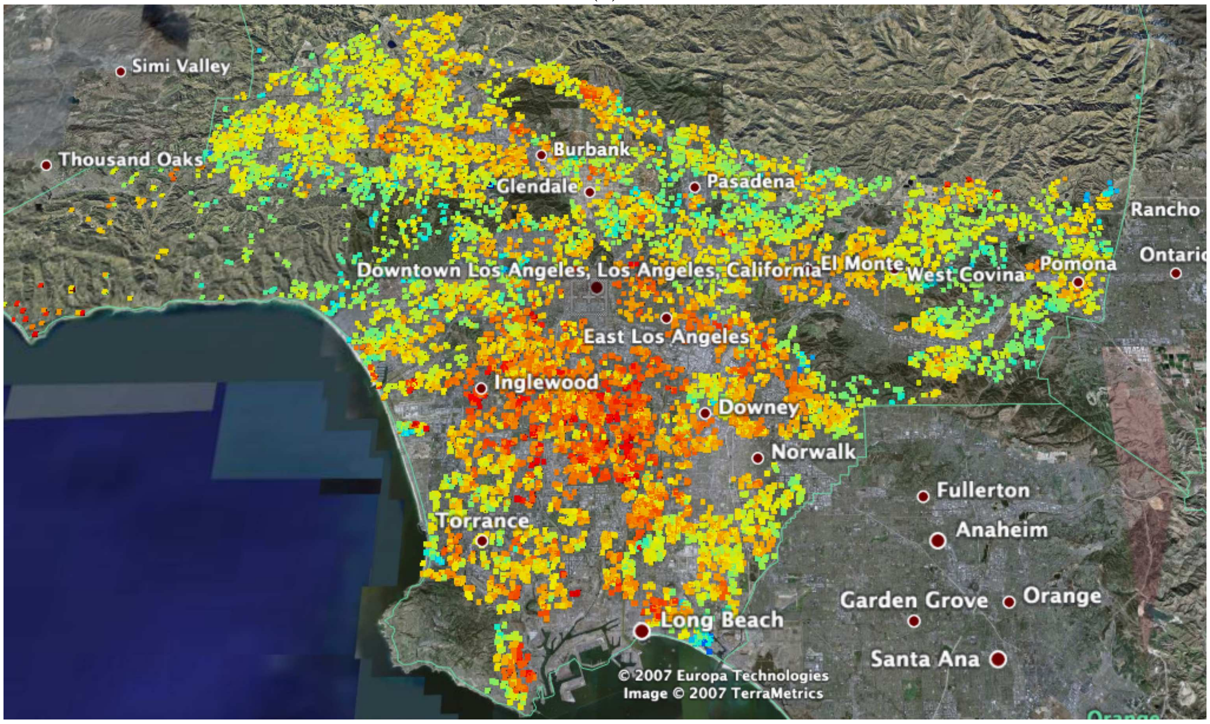
[17] N. Nguyen and A. Cripps. Predicting housing value: A comparison of multiple regression analysis and artificial neural networks. *The Journal of Real Estate Research*, 22(3):313 – 336, 2001.

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

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



(a)



(b)

Figure 3: (a). The color coded values of the desirability surface at the location of the test samples. For every test sample X, the estimate of its desirability is computed using the function $H(D, X)$ and the learned vector D and is color coded according to its value. Blue color implies low desirability and red color implies high desirability. (b) The color coded map showing the sensitivity of the prices of houses in the test set with respect to the number of bedrooms. Blue color implies the price of the house is less sensitive to the change (increase) in the number of bedrooms and red color implies that the price of the house is more sensitive to increase in the number of bedrooms.