Smart Real Estate Assessments Using Structured Deep Neural Networks

Haiping Xu and Amol Gade
Computer and Information Science Department
University of Massachusetts Dartmouth
Dartmouth, MA, USA
{hxu, agadel}@umassd.edu

***Abstract*—In a smart city, effective and accurate real estate assessments governed by a local government is crucial for determining the property taxes. Such assessments have never been trivial, and inappropriate assessments may result in disputes between property owners and the local government. In this paper, we introduce a deep learning approach to smartly and effectively assessing real estate values. We propose a systematic method to derive a layered knowledge graph and design a structured Deep Neural Network (DNN) based on it. Neurons in a structured DNN are structurally connected, which makes the network time and space efficient; and thus, it requires fewer data points for training. The structured DNN model has been designed to learn from the most recently captured data points; therefore, it allows the model to adapt to the latest market trends. To demonstrate the effectiveness of the proposed approach, we use a case study of assessing real properties in small towns. A structured DNN was designed to match with a layered knowledge graph for property assessments in the real estate domain, which results in a significant reduction of neurons and connections between them. The experimental results show that a structured DNN outperforms conventional multivariate linear regression models, fully-connected neural networks, and prediction methods used by the leading real estate companies.**

Keywords—Smart city, deep learning, real estate assessments, layered knowledge graph, structured deep neural networks

I. INTRODUCTION

As large amount of data being continuously generated by digital devices and online activities in a smart city, intelligently using the data through predictive analytics for product value assessments would help the city to make better decisions, identify trends, and improve city management performance. In the real estate domain, effectively assessing real properties by a local government is crucial for accurate assessments of property taxes; however, such assessments are intrinsically nontrivial, and inappropriate assessments may result in disputes between property owners and the local government. Proper and accurate assessments of real estate is not only important for a local government to effectively determine the tax values of real properties, but also valuable for prospective homeowners, developers, investors, appraisers, and other real estate market participants, such as mortgage lenders and insurers [1, 2], who often rely on real estate agents or certified appraisers to get the estimation of desired properties. However, traditional services are usually expensive, and it has been an insistent demand by the current real estate industry to develop a logical scientific assessment model that is not only easy-to-use, but also reliable and accurate.

Since the collected data for real property assessments are typically multidimensional and nonlinear, it has been very difficult and ineffective to use conventional regression approaches to model the assessment functions. Assessing real properties has long been considered a challenging problem, as the values of real properties depend on many different parametric and non-parametric features. Real estate values are a chronological and time-sensitive sequence with unknown statistical relationships influenced by many factors, which makes it nontrivial to predict property values using predefined functions [3]. In addition, the domain knowledge for real estate may be structured and complex, which makes it hard to use traditional classifiers to learn the functions and to accurately assess real estate values. As a result, the use of most advanced algorithms for predictive analytics is necessary. Data mining and machine learning techniques are becoming more and more important for assessing the values of time-sensitive products based on historical data. Deep learning, as a new branch of machine learning, has gained significant attention from the machine learning and artificial intelligence communities due to its ability to automatically learn multiple level representations from structured data using either supervised or unsupervised learning. There have been many previous efforts on using data mining approaches, such as neural networks, for house prices prediction [3, 4]; however, to the best our knowledge, most of the existing approaches use fully-connected neural networks, where the number of hidden layers and the number of neurons are determined in an ad-hoc way; therefore, they do not necessarily match the knowledge structure of the real estate domain. In this paper, we introduce a structured deep learning approach for real estate assessments. We present a systematic approach to extracting structured knowledge from the real estate domain and designing a structurally connected Deep Neural Network (DNN) to assess real properties. In addition, the structured DNN is designed to learn in real time, which allows it to adapt to new market trends efficiently. To demonstrate the effectiveness of our proposed approach, we collected data from real estate web sites such as Zillow.com, and performed experiments to construct structured DNNs and assess real property values. The experimental results show that our approach is promising, which not only provides a systematic way to construct DNNs, but also improves the accuracy of real property assessments.

978-1-5386-0435-9/17/$31.00 ©2017 IEEE

II. Related Work

Previous work on predicting house prices has been based on regression analysis and machine learning techniques. Frew and Jud used hedonic modeling techniques to estimate the house prices in the Greater Portland region [1]. They observed that house values would rise less than proportionally with the size and number of units, and decline with age, but the marginal effect of aging was small. Lowrance used a local linear model and random forest model to predict house sale prices in Los Angeles County [2]. He showed that random forest model may perform better than the local linear model. Hu and Zhong used backpropagation neural networks and Elman neural network to forecast real estate prices [3]. They found that Elman neural network could forecast more accurate and constringe faster than other approaches. Nguyen and Cripps compared the predictive performance of artificial neural networks (ANNs) and multiple regression analysis for single-family housing sales [4]. They found that when enough data points were available for training, ANNs could perform better than multiple linear regressions. Hamzaoui and Perez applied a feed-forward backpropagation neural network with a single hidden layer to predict the selling prices of residential properties in Casablanca, Morocco Kingdom [5]. The experimental results suggested that ANNs could be used as a tool for reliable prediction of house selling prices. Zhang predicted real estate prices using fuzzy neural network to support fuzzy reasoning and learning [6]. His research showed that fuzzy neural network may work better than traditional neural network approaches. Chopra *et al.* used a latent manifold model with two trainable components to evaluate house prices [7]. The first one is a parametric component that predicts the "intrinsic" price of a house using its features, and the second one is a non-parametric component that calculates the desirability of the neighborhood. The study found that the latent manifold model performs better than pure parametric or non-parametric models. Although the above approaches could be used to predict house selling prices, none of them used most advanced deep learning approach to evaluating real properties. Different from the existing approaches, we propose to use a deep learning approach with structured DNNs, which may outperform the conventional ones for real estate assessments.

There is also quite a lot of previous work on using machine learning and deep learning techniques for predictive analytics. Mitchell and Sheppard used simple, non-connectionist dimensionality reduction techniques including conventional Principal Component Analysis (PCA) and deep architecture PCA to generate features for image classification [8]. They concluded that a deep architecture could help to improve the performance of the models. Liao *et al.* presented a structured deep learning architecture for phoneme recognition [9]. When a structured input is given, the approach can learn and find the best-structured object based on the mapping relationships between the structures. Their test results showed that the model outperformed other conventional approaches for phoneme recognition. Ford *et al.* used a neural network model to detect suspicious bidders in online auctions [10]. Their approach focused on training the neural network with newly added structured data, so it can quickly adapt to changing trends in bidding. Although the above approaches used structured input and learning processes, the DNNs are fully connected, whereas in our approach, we use a structurally connected DNN that is significantly different from their approaches.

Additional work on using structured neural networks for predictive analytics is summarized as follows. Dries and Wiering presented a specialized structured neural network approach along with a neural-fitted Temporal Difference (TD) algorithm to improve the learning process on the game of Othello [11]. The neural network focuses on regions on the board, and the experimental results indicate that the structured neural network approach outperforms linear models as well as fully-connected neural network models. Steeg *et al.* extended the above structured neural approach for the game of Tic-Tac-Toe 3D [12]. They added one more hidden layer that is fully connected. They observed that the added hidden layer enables the neural network to integrate patterns learned in the structured hidden layer. The authors compared the performance of three fully-connected neural networks with different hyper-parameters with two structured neural networks. The experimental results show that structured neural network approach works better than fully-connected neural networks. Zhang *et al.* presented a shrinking DNN structure with hidden layers decreasing in size from a lower layer to higher layers for the purpose of reducing the model size and making the model time efficient [13]. They concluded that shrinking structured DNN reduced the model size and the computation time without affecting performance. However, they failed to justify why shrinking DNN would not affect performance, and also there was a lack of systematic approach for the network reduction. Different from the above methods, our approach demonstrates how to derive a structured DNN from a layered knowledge graph for a specific domain; therefore, our approach is not ad-hoc but a systematic one, which may support constructing structured DNN in a more efficient and effective way.

III. Layered Knowledge Graph

Deep learning was shown to be an effective approach in 2006 [14]. In recent years, it has gained significant attention in both the academia and the industry due to its effectiveness. Deep learning supports learning multiple levels of composition [15], and its ability to automatically learn hidden features from data in layered fashion through both supervised and unsupervised training makes it a useful and practical approach. Some popular deep learning approaches such as Convolutional Neural Networks (CNN) and Recurrent Neural Network (RNN) have been successfully applied to the image processing and speech recognition domains, respectively.

A neural network with more than one hidden layer is called a DNN; while a neural network with a single hidden layer is considered a shallow one. With the universal approximation property [16], if enough hidden neurons are given, a shallow neural network can approximate any function with remarkable accuracy. However, a DNN could be much cheaper than a shallow one because a particular function can be simulated using a much less number of hidden neurons comparing to a shallow one. Experimental results show that there is a range of families of functions, for which we may have an exponential advantage in terms of the number of hidden neurons [15]. Note that a DNN may use a less number of hidden units to represent a function because in the network, lower-level features can be reused to produce higher-level or more abstract features.

Although a DNN works more efficient than a shallow one, a typical DNN for solving a practical problem may still require a large number of hidden neurons. Traditionally, a DNN is considered a black-box approach, where the semantics of the hidden neurons are unknown, and there is no clear way to determine a suitable size of the network. As shown in previous work, a structured DNN with a reduced number of links and neurons requires much less computational resources, but would still lead to satisfactory or even better results [13]. This implies that there exists a close relationship between the internal structure of a DNN and its performance. It can be envisioned that useful insights about the internal structure of a DNN could be helpful to reveal more details about the DNN approach under the hood. On the other hand, due to the traditional black-box view of DNNs, designing a structured DNN with a reasonable number of layers and neurons, as well as the needed connections between neurons, has become one of the most challenging tasks. To deal with this issue, in this research, we investigate the semantics of the internal structure of a DNN as well as its relationship with the knowledge structure of a certain domain. In particular, we use real estate assessments as an example of deep learning using structured DNN, and demonstrate how it would be possible to derive a layered knowledge graph for a certain domain, and how a structured DNN can be designed to match with the graph.

We now provide a formal definition of a layered knowledge graph Φ as follows.

Definition A *layered knowledge graph* Φ is a k-partite graph (V, E), where $V = (L_1, L_2, \ldots, L_k)$ is an ordered k-partition of its vertex set V, and $E \subset (L_1 \times L_2) \cup \ldots \cup (L_{k-1} \times L_k)$. Each vertex $v \in V$ is a tagged node with semantic annotation v_{sem}.

To derive a layer knowledge graph for a certain domain, we first collect labeled training and test data points within the domain. Then we design a fully-connected DNN with k layers and n_l neurons in each layer, where $1 \leq l \leq k$, and n_1 and n_k are the number of predefined input features and the number of outputs, respectively. After training the model, we identify the links with weak weights that are below a predefined threshold. We call such links *weak links*, which contribute less to the next layer than a normal link. As such, weak links can be safely deleted from the DNN. Similarly, we identify the neurons with very few links, called *weak neurons*. Such neurons can be either deleted or combined into a stronger one. Finally, we examine the hidden layers with too few neurons, called *weak layers*. In a similar way, a weak layer may be deleted, or two adjacent weak hidden layers can be combined into a single one. The resultant DNN is a structured one with no weak links, no weak neurons, and no weak layers. Such a graph defines the structure of a layered knowledge graph of a given domain. Once the knowledge structure has been built, we identify the semantics of each node based on its inputs, and adjust the graph as needed to make it more meaningful. The complete procedure for deriving a layered knowledge graph using a fully-connected DNN is illustrated as in Algorithm 1.

According to the algorithm, the network is repetitively reduced for its links, neurons, and layers until there is a significant decrease in its accuracy. During each iteration, based on the range and distribution of the weights of the links in the DNN, we define a threshold for each hidden node (i.e., a node belonging to layer l, where $1 < l < k$). A threshold α of a node d is chosen such that only major contributors of node d (i.e., nodes from a previous layer with strong links to node d) could be added into a set of contributing nodes Γ for node d. Let s and d be the source node and destination node of a link ς, respectively, and $w(\varsigma)$ be the connection weight of the link. If the absolute value of $w(\varsigma)$ is greater than or equal to α, then the contributing node s is added into Γ for node d. Consequently, if a source node of a link does not belong to the set of contributing nodes for node d, that link must be a weak link; thus, it shall be deleted from the graph. The weak neurons and weak layers can be processed in a similar way.

Algorithm 1: Construct Structured Knowledge

Input: Training dataset and test dataset in domain D.
Output: Layered knowledge graph Φ for domain D.

1. Initialize DNN Σ with k layers and n_i neurons in each layer.
2. Train Σ using the training dataset.
3. Test Σ using the test dataset and record its accuracy.
4. Repeat the following until there is a significant decrease in accuracy
5. Identify the weak links in Σ, and delete such links.
6. Identify the weak neurons in Σ, and delete or combine them.
7. Identify the weak layers in Σ, and delete or combine them.
8. Train the reduced Σ using the training dataset.
9. Test the reduced Σ using the test dataset and record its accuracy
10. Restore Σ from its previous iteration.
11. Name the hidden nodes and adjust Σ accordingly.
12. Create Φ according to Σ by ignoring all weights.
13. **return** layered knowledge graph Φ.

After the DNN has been successfully reduced by removing or combining weak links, weak nodes, and weak layers, we can look into the nodes and identify their semantics. The purpose of this step is to adjust the layered knowledge graph and make it more meaningful. We process one layer at a time from the lowest hidden layer to the highest one. The semantic of a destination node d in layer l is determined based on d's list of contributing nodes, which contains source nodes from layer l-1. For example, in the real estate domain, if the following input features "Number of Beds", "Number of Baths", "Square Footage", and "Lot Size" are the impacting contributors to hidden node d in the second layer, then we may name node d as "Property Size" because all input features are closely related to the size of a property. On the other hand, if some nodes exist due to possible noise, we may remove it to make the layered knowledge graph more meaningful. For example, if nodes named "Location", "Amenities", and "Rooms" contribute to hidden node d, we could name d as "Neighborhood Features," and remove its connection with the node named "Rooms." This is because the feature 'Rooms" is clearly not related to the other two features; thus it shall not be grouped with them in the list of contributing nodes. Similarly, if any clearly related nodes linking to node d are identified, they could be added into the list of contributing nodes of node d as needed. Once we have named each hidden node in the first hidden layer, the first hidden layer is considered the input layer of the second hidden layer, and thus, the above procedure can be repeated until all hidden nodes in the remaining hidden layers are properly named. After the naming process completes, the layered knowledge graph Φ is returned for further processing.

IV. A Framework for Real-Time Training of a DNN

Once a layered knowledge graph Φ is captured for a given domain, it can be used to derive a structured DNN Ψ for training and prediction. In this section, we describe the procedure of creating a structured DNN from a layered knowledge graph, and provide a framework that supports training a structured DNN model in real time.

A. Contructing Structued DNN

Choosing the number of hidden layers and the number of hidden neurons in each layer is one of the most difficult and time-consuming tasks while constructing a DNN model. In our approach, we have made this task easy by designing a structured DNN to match with a layered knowledge graph derived from a given domain. The structured DNN contains the same number of layers and the same number nodes in each layer as in the layered knowledge graph. Once the number of layers and the number of nodes in each layer are defined, the next task is to structurally connect the neurons in the DNN. The procedure for making the connections between neurons is summarized as in Algorithm 2.

Algorithm 2: Construct a Structured DNN

Input: Layered knowledge graph Φ for domain D.
Output: Structured DNN Ψ for domain D.

1. Create Ψ with k layers and n_i neurons in each layer as in Φ.
2. Let $\pi(n)$ be the mapping of node n in Φ to a neuron in Ψ
2. **for** layer l from 1 to k-1 in Φ
3. **for each** node n in layer l
4. **if** node n has an edge to node x in layer l+1 of Φ
   Connect $\pi(n)$ and $\pi(x)$ in Ψ
5. **for** layer l from 2 to k in Ψ
6. **for each** neuron n in layer l
7. add a bias node and connect it to neuron n
8. **return** structured DNN Ψ

According to Algorithm 2, the returned structured DNN reflects the structured knowledge of the given domain due to its one-to-one mapping of the nodes and links from the layered knowledge graph. Note that naming process in Algorithm 1 is used for the sole purpose of adjusting the layered knowledge graph, and make it semantically sound. As the semantic annotations of the nodes in the layered knowledge graph are no longer needed in the training process of the structured DNN, they are simply ignored while constructing the structured DNN from the knowledge graph.

B. A Framework for Training a DNN in Real Time

With a structured DNN in the real estate domain, we can use it to assess real property values using predictive analytics. The framework for real-time training of a structured DNN is illustrated in Fig. 1. The first step is to collect the training and test data points. As the collected data are raw data that usually contain a lot of unnecessary information, we need to pre-process them and retrieve the needed fields in a desired format. In addition, data points with missing information or wrong information could negatively affect the training results of a neural network, such data points are considered outliers, and thus they are removed from the training and test datasets.

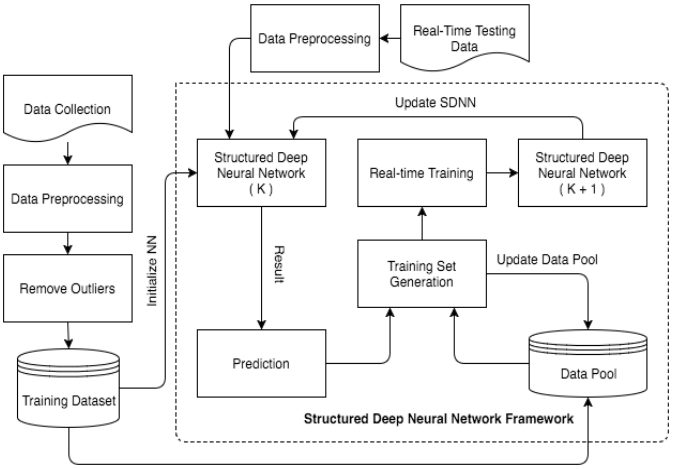

Fig. 1. A framework for training a DNN in real time

Before starting the training process, the structured DNN is initialized with random weights. As an important goal of our approach, we allow a structured DNN to be trained with newly acquired labeled data points, so it can automatically adapt to recent market trends and support real-time training and prediction. Inspired by a real-time classifier for shill detection introduced in previous work [10], we define a window size for the data pool as shown in Fig. 2. When new data points become available, they are added into the data pool once they are properly labeled; meanwhile, the old data points, which are now shifted out of the window, must be removed from the data pool. Each time when new labeled data points are added into the data pool, the structured DNN is incrementally trained and learns from the new data points in real time. Since the structured DNN is always trained using the most recent data points within the window, it is able to follow new market trends, and would produce more accurate and reliable assessments for time-sensitive products, such as real properties.

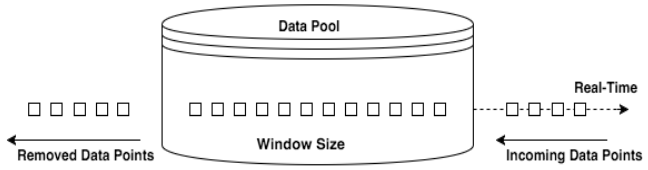

Fig. 2. Storage of recent data points using a predefined window size

It is worth noting that as a major advantage of using a structured DNN, it helps to mitigate the overfitting problem. When a neural network has a large number of hidden neurons, it can simulate a complex function that fits perfectly (overfit) with the majority of the training data points. On the other hand, a structured DNN contains only the needed connections among neurons. With a much less number of connections compared to a fully-connected neural network, a structured DNN will not become unnecessarily over-powerful, and thus, it will have less chance of overfitting. In addition, since over-training a model may also lead to the overfitting problem, early stopping is critical during the training process, where training must be stopped before the error rate starts to increase significantly.

V. CASE STUDY

In this section, we use real property assessments as a case study to demonstrate the usefulness of a structured DNN for predictive analytics. To demonstrate the effectiveness of our approach, we conduct various experiments and compare the performance of our approach with that of the existing ones.

A. Data Collection and Preprocessing

To establish the data pool, we collected real estate data from a leading real estate listings website Zillow.com. This website maintains all recent and past house listings data including house features, market features, public records of houses, neighborhood features, and so on. A total number of 15 features are predefined and their associated values are collected. The predefined features include number of beds, number of baths, square footage, lot size, built year, yearly tax, similar houses average sold price, nearby schools average ratings, fireplace, waterfront, number of stories, heating, cooling, patio, and park. In our approach, we first remove the following outliers: 1) data points with insufficient information, e.g., missing square footage; 2) data points with unreasonable values, e.g., a sold price of $1 (as a gift). We further calculate the average selling price-per-square-feet of houses sold in last 6 months. Houses with too high or too low price-per-square-feet are considered outliers, and thus they are excluded from the training and test datasets.

B. Design of Layered Knowledge Graph

We performed experiments on fully-connected DNNs with different numbers of hidden layers and different numbers of neurons in each hidden layer. We assume that a DNN, which can produce satisfactory outputs with less weak connections between the neurons, would be closer to a representation of a layered knowledge graph in a given domain. Based on the experimental results, we select a fully-connected DNN with 2 hidden layers and 12 neurons in each hidden layer for our case study. In this selected neural network, the input layer contains 15 neurons, representing the predefined input features for house value assessments, and a single output neuron representing the assessed house value or predicted selling price. Using Algorithm 1 given in Section III, we derived a layered knowledge graph for the real estate domain as illustrated in Fig. 3. From the figure, we can see that the first layer contains 15 input nodes indicating that all selected features have significant impacts on house value assessments. The number of hidden neurons in the second and third layer has been reduced from 12 to 10 and 7, respectively. This is due to the weak links and weak neurons being removed from the network. Finally, a single neuron in the last layer represents the assessed house value or predicted selling price.

Figure 3 also shows our attempts in understanding the semantics of the hidden nodes. For example, the number of bedrooms, the number of bathrooms, the number of stories, and the patio all contribute to the second node in the second layer. As these input features are related to the structure of a house, we label this node as "House Structure". Similarly, the nodes named as "External Features", "House Structure", "Interior Features", "House Condition", and "Property Size" all contribute to the second node in the third layer; consequently, we name that node as "House Features".

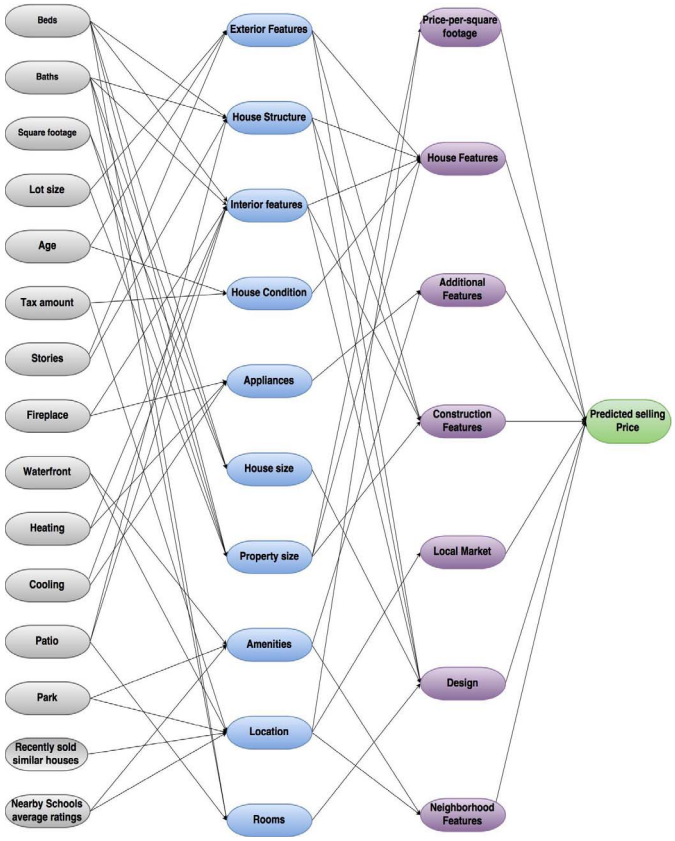

Fig. 3. An example of layered knowledge graph for the real estate domain

As we can see from the figure, in the layered knowledge graph for the real estate domain, multiple features in a given layer may contribute to a "hidden" feature in the next layer. On the other hand, each feature in a given layer may also contribute to more than one "hidden" feature in the next layer. Due to the complexity of such connections in the graph, the semantics of some nodes might not be very clear (e.g., the feature "Patio" from the first layer contributes to the node "Rooms" in the second layer). However, as we can see it, the layout the knowledge graph is generally meaningful and satisfactory, and thus, it would be sufficient to be used to create structured DNN for house value assessments.

C. Design of Structured DNN

The structured DNN is designed to match with the layered knowledge graph derived in the previous section. According to Algorithm 2 presented in Section IV.A, the structured DNN has four layers: an input layer, two hidden layers, and an output layer. We set up suitable hyper-parameters for the structured DNN, and trained it using standard feedforward back-propagation algorithm with problem-specific real-time training and fitting techniques. Note that the first layer of the network contains 15 input neurons, which always produce outputs, as there are no biases (thresholds) connected to the input layer neurons. Although smaller initialized weights make a neural network learn slower, experimental results show that, with enough available data points, initializing a neural network with smaller weights helps to get better generalization, and hence to achieve better performance. Therefore, in this case study, we

initialize all connection weights with relatively small random values within the range of [-0.5, 0.5].

Learning rate impacts the learning speed of connections during the training process. Learning rate for the first hidden layer is set higher than the other layers in the model because after a certain amount of training time, the first hidden layer will learn comparatively slower than the second hidden layer [17]. Therefore, we set the learning rate to 0.9 for the first hidden layer; whereas for all other layers, the learning rate is set to 0.1. Choosing a small learning rate makes small changes in weights, and thus it takes longer time to train the model; however, it makes the learning curve much smoother. On the other hand, although choosing a large learning rate value may speed up the learning process, it may lead to an unstable model. To balance the training speed and the stability of the network, a momentum is used to diminish the fluctuations in weight changes. Experimental results show that choosing a large value of momentum for lower level layers helps to get a stable model and hence to achieve better generalization [18].

D. Experimental Results and Analysis

Based on our experiments, we noticed that the most recent 6-month data would well represent the real estate market; therefore, we selected real estate data of a small town (zip code: 02127) in the past two years (2014-2016) and used a window size of 6 months for training and testing. We compared the accuracy of our approach with existing approaches including the naive price-per-square-foot approach, the conventional multivariate linear approach, and advanced fully-connected DNN approaches. The same training dataset and test datasets are used for all experiments.

In addition, we also compared our approach with house price prediction methods adopted by leading real estate companies, namely Zillow and Redfin. Fig. 4 shows the median and mean percent errors for house price predictions using different methods (mean percent error is not available for predictions from Zillow and Redfin). Results show that structured DNN gives the best house value assessments as it has the lowest median percent error and the lowest mean percent error among all methods.

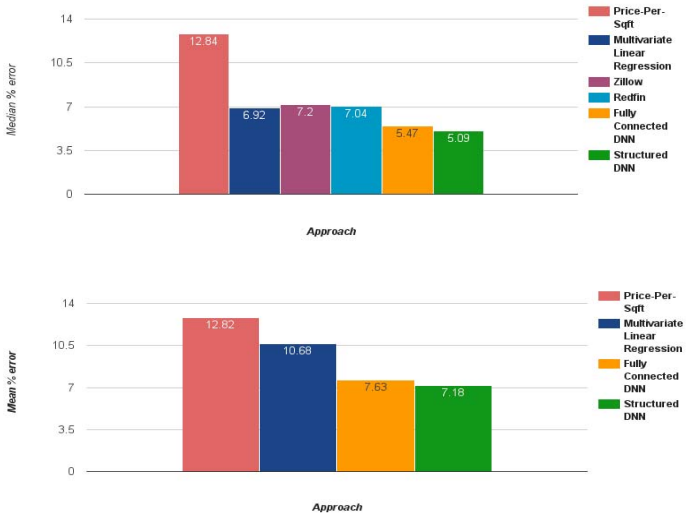

Fig. 4. Comparison of median and mean percent errors of different methods

Figure 5 shows the comparison of accuracy for different error ranges using different methods. Three error ranges are used, which are "within 5%", "within 10%", and "within 20%". For example, if a model predicts the selling price of a house as 108K and if the house is sold for 100K, then the percent error for the prediction using that model is 8%. In this case, the prediction percent error is within 10% error range. Results from Fig. 5 show that the structured DNN performs better than other methods, including those used by the leading real estate companies.

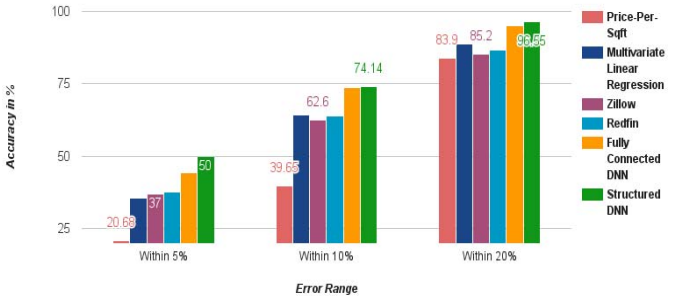

Fig. 5. Comparison of accuracy for different error ranges

To further verify the usefulness of our approach, we collected additional real estate data within previous 5 years (2011-2016) for different zip codes (02125, 02127 and 02128). We compare our previous results with experimental results using the new data. Fig. 6 shows the median percent errors and the mean percent errors for different zip codes. Note that the previous results are identified by "`Zip 02127*`" in Fig. 6.

Fig. 6. Median percent error and mean percent error for various zip codes

From the figure, we can see that with more data points from zip code 02127 used for training and testing, the performance of the structured DNN has slightly gone down. This is because the real estate market in recent years has been quite unstable and emerged dramatic fluctuations. Although unstable market values are hard to predict, the prediction accuracy has only slightly decreased for the 5-year data thanks to the real-time training. In addition, we can see that results for zip code 02128 are not as accurate as compared to the others. This is due to the insufficient data points available from that area to train the model, which results in a decrease of the model performance. Finally, we compare the results using structured DNNs and fully-connected DNNs. As shown in Fig. 6, in all cases, structured DNNs perform better than fully-connected DNNs.

VI. Conclusuions and Future Work

Predictive analytics has become more and more important in the era of big data and machine learning. Many big data analytical techniques and deep learning algorithms have been proposed to assess product values. However, due to the large size of a deep learning architecture, deep learning typically requires a large amount of data to train the model, which makes the training process very time and space inefficient. In addition, a large deep architecture may also prone to have the overfitting issue. In this paper, we proposed a structured deep learning approach for predictive analytics. We presented a systematic approach to deriving a layered knowledge graph and designing structured DNN based on it. The structured DNN contains only the needed connections between neurons; therefore, it is time and space efficient, and can be trained using fewer data points compared to a fully-connected DNN. Furthermore, as the model contains fewer connections, the use of the model may significantly reduce the chances of overfitting. To demonstrate the feasibility of our approach, we used the case study of smart real estate assessments for house values. Our experimental results show that the proposed approach outperforms other conventional methods and leading real estate companies such as Zillow and Redfin, with significantly improved accuracy for house-price prediction.

For future research, we will study how to automate the process of extracting layered knowledge graphs from the real estate domain based on historical data, and design structured DNNs using the graphs. We will allow a DNN model to automatically change its network structure along the time, so it can be more scalable and better adapt to new market changes. Furthermore, we plan to implement our approach using mobile cloud computing [19] that supports assessments of real estate via mobile devices with computation functions deployed in the clouds. Finally, we will try to apply our approach to predictive analytics problems from other domains [20, 21], such as stock market, healthcare, transportation, marketing, e-commerce, security, business, and many more.

References

[1] J. Frew and G. Jud, "Estimating the Value of Apartment Buildings," *Journal of Real Estate Research*, Vol. 25, No. 1, 2003, pp. 77-86.

[2] R. E. Lowrance, "Predicting the Market Value of Single-Family Residential Real Estate," *Technical Report*, Department of Computer Science, New York University, 2015.

[3] X. Hu and M. Zhong, "Applied Research on Real Estate Price Prediction by the Neural Network," In *Proceedings of the 2nd International Conference on Environmental Science and Information Application Technology (ESIAT 2010)*, July 17-18, 2010, pp. 384-386.

[4] N. Nguyen and A. Cripps, "Predicting Housing Value: A Comparison of Multiple Regression Analysis and Artificial Neural Networks," *Journal of Real Estate Research*, vol. 22, no. 3, pp. 313-336, 2001.

[5] Y. E. Hamzaoui and J. A. H. Perez, "Application of Artificial Neural Networks to Predict the Selling Price in the Real Estate Valuation Process," In *Proceedings of the 10th Mexican International Conference on Artificial Intelligence (MICAI)*, Nov. 26 - Dec. 4, 2011, pp. 175-181.

[6] X. Zhang, "Using Fuzzy Neural Network in Real Estate Prices Prediction," In *Proceedings of the 26th Chinese Control Conference*, July 26-31, 2007, Zhangjiajie, Hunan, China, 2006, pp. 399-402.

[7] S. Chopra, T. Thampy, J. Leahy, A. Caplin, and Y. LeCun, "Discovering the Hidden Structure of House Prices with a Non-Parametric Latent Manifold Model," In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'07)*, August 12-15, 2007, San Jose, CA, USA, pp. 173-182.

[8] B. Mitchell and J. Sheppard, "Deep Structure Learning: Beyond Connectionist Approaches," In *Procedings of the 11th International Conference on Machine Learning and Applications (ICMLA)*, Dec. 12-15, 2012, , pp. 162-167.

[9] Y.-H. Liao, H.-Y. Lee, and L.-S. Lee, "Towards Structured Deep Neural Network for Automatic Speech Recognition," In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, Scottsdale, AZ, USA, Dec. 13-17, 2015, pp. 137-144.

[10] B. J. Ford, H. Xu, and I. Valova, "A Real-Time Self-Adaptive Classifier for Identifying Suspicious Bidders in Online Auctions," *The Computer Journal (COMPJ)*, Vol. 56, No. 5, 2013, pp. 646-663.

[11] S. Dries and M. Wiering, "Neural-Fitted TD-Leaf Learning for Playing Othello with Structured Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 23, No. 11, Nov. 2012, pp. 1701-1713.

[12] M. Steeg, M. Drugan, and M. Wiering, "Temporal Difference Learning for the Game Tic-Tac-Toe 3D: Applying Structure to Neural Networks," In *Proceedings of the IEEE Symposium Series on Computational Intelligence*, Dec. 7-10, 2015, pp. 564-570.

[13] S. Zhang, Y. Bao, P. Zhou, H. Jiang, and L. Dai, "Improving Deep Neural Networks for LVCSR Using Dropout and Shrinking Structure," In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Florence, Italy, May 4-9, 2014, pp. 6849-6853.

[14] G. E. Hinton., "Learning Multiple Layers of Representation," *Trends in Cognitive Sciences*, Vol. 11, No. 10, 2007, pp. 428-434.

[15] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016, Accessed on Jan. 1, 2017 from http://www.deeplearningbook.org, pp. 13.

[16] M. Stinchcombe and H. White, "Universal Approximation Using Feedforward Networks with Non-Sigmoid Hidden Layer Activation Functions," In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 1989, pp. 613-617.

[17] M. Nielsen, *Neural Networks and Deep Learning*, Jan. 2017, Retrived on Jan, 31, 2017 from http://neuralnetworksanddeeplearning.com/

[18] G. Batres-Estrada, "Deep Learning for Multivariate Financial Time Series," *Technical Report*, Stockholm, May 2015, Retrieved on March 15, 2017 from: https://www.math.kth.se/matstat/seminarier/reports/M-exjobb15/150612a.pdf

[19] L. Zhou and H. Xu, "An Efficient Double Auction Mechanism for On-Demand Transport Services in Cloud-Based Mobile Commerce," In *Proceedings of the 5th IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (IEEE Mobile Cloud 2017)*, San Francisco, CA, USA, April 6-8, 2017, pp. 25-30.

[20] S. Earley, "Big Data and Predictive Analytics: What's New?" IT Professional, Vol. 16, No. 1, Jan.-Feb. 2014, pp. 13-15.

[21] M. Nural , M. Cotterell and J. Miller, "Using Semantics in Predictive Big Data Analytics," In *Proceedings of the IEEE International Congress on Big Data (BigData Congress)*, June 27 - July 2, 2015, New York, NY, USA, pp. 254-261.