University of Massachusetts Dartmouth

Department of Computer and Information Science

Towards Structured Deep Neural Network for Predictive Analytics

A Thesis in

Computer Science

by

Amol S. Gade

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

August 2016

I grant the University of Massachusetts Dartmouth the non-exclusive right to use the work for the purpose of making single copies of the work available to the public on a not-for-profit basis if the University's circulating copy is lost or destroyed.

_____

Amol S. Gade

Date _____

We approve the thesis of Amol S. Gade

Date of Signature

Haiping Xu Associate Professor, Department of Computer and Information Science Thesis Advisor	
Firas Khatib Assistant Professor, Department of Computer and Information Science Thesis Committee	
David Koop Assistant Professor, Department of Computer and Information Science Thesis Committee	
Xiaoqin Zhang Graduate Program Director, Department of Computer and Information Science	
Jan Bergandy Chairperson, Department of Computer and Information Science	
Robert E. Peck Dean, College of Engineering	
Tesfay Meressi Associate Provost for Graduate Studies	

ABSTRACT

Towards Structured Deep Neural Network for Predictive Analytics

by Amol S. Gade

With the rapid growth of global data, predictive analytics has become ever important. Predictive analytics can help people in many different ways including making better decisions, understanding market trends, and improving productivity. Conventional machine learning techniques used for predictive analytics, such as regression techniques, are typically not sufficient to handle complex data associated with structured knowledge. Deep learning, also called deep structured learning, has received great attentions in recent years for modeling high-level abstractions in data with complex structures. In this thesis, we propose a systematic approach to deriving a layered knowledge structure and designing a structured deep neural network based on it. Neurons in a structured deep neural network are structurally connected, which makes the network time and space efficient, and also requires fewer data points for training. Furthermore, the proposed model can significantly reduce chances of overfitting, which has been one of the most common and difficult to solve problems in machine learning. The structured deep neural network 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 for predictive analytics, we use a case study of predicting house selling prices. A deep neural network has been designed to match with a

knowledge structure for house price prediction, with a significantly reduced number of connections comparing to a fully connected neural network. Our experimental results show that a specialized structured deep neural network may outperform conventional multivariate linear regression models, as well as fully connected deep neural networks.

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation and gratitude to my Thesis Advisor, Dr. Haiping Xu for his continuous support and guidance for my Master's thesis. He was always there to help me even during weekends and vacations. Also, he was always quick in responding to my emails. Weekly meetings with Dr. Xu used to be a great motivation for me to put efforts, as well as, the meetings helped me to put efforts in the right direction of completing the thesis. He was there to support and motivate me during tough times. He has helped me not only to be a better student but also to be a better person. I would like to thank him for all his time, efforts, and hard work in regards to completing this thesis.

I would like to thank the other members of my thesis committee, Dr. Firas Khatib and Dr. David Koop for agreeing to be a part of the thesis committee.

To my parents, Shivaji Gade and Rajshree Gade: Thank you for giving me the vision to think beyond my limits along with the strength and confidence to achieve it.

To my fiancée Geeta: We made it! Thank you for your continuous support, motivation, patience, unconditional love and faith in me.

To my uncle Bhaskar Gade and aunt Sunita Gade: I am extremely grateful for your sacrifices for educating and preparing me for my future. You are amongst the most generous and tolerant people I came across in my life. Thank you for tolerating me for 6 years. I will give my best to my life to make you feel proud.

To my elder brother Ganesh Gade and my sister-in-law Renuka Gade: I could come overseas for higher education without any worries only because I knew you both are there to take care of our parents. Also, thank you for being a great support over the years and for your selflessness.

To my future in-laws Ashok Jadhav and Jija Jadhav: Thank you for being a great support during completion of my thesis. Also, I would like to thank you for your faith in me and for considering me qualified enough to take good care of your daughter forever.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 Introduction

Predictive analytics is about using various data mining, statistics, predictive modeling and machine learning techniques to get insights from historical data with an objective of predicting future outcomes[1]–[4]. A large amount of data is continuously being generated by a huge number of digital devices and online activities around you and in the world. In 2011, IBM estimated that 90% of the global data had created in previous two years [5]. Smartly using this data through predictive analytics would help the world to make better decisions, identify trends, improve business performance, better understand psychology, save time and money and much more. However, the data is being generated from different sources in different formats. Also, often complex problems have multidimensional data with a number of nonlinearities in there, which makes it harder for conventional machine learning techniques, such as regression techniques to model a function. As a result, use of most advanced algorithms for predictive analytics is a demand.

Deep learning is a new branch of machine learning introduced with the objective of moving machine learning one step closer to the artificial intelligence [6]. The field has gained significant attention from the machine learning and artificial intelligence researchers because of its ability to automatically learn multiple level representations from structured data through both supervised and unsupervised learning. As a result, recently deep learning algorithms are used widely in speech recognition, computer vision, and natural language processing.

In this thesis, we present a systematic approach to design the best-structured knowledge for a problem and to designing a structurally connected deep neural network. Also, a structured deep neural network has been designed to learn in real-time which allows it to easily adapt to new market trends.

To demonstrate the effectiveness of our proposed approach for predictive analytics, we choose house sales price prediction problem as an example as real estate has a huge market globally and it is increasing rapidly day-by-day. All over the world, a large percentage of people are connected to real estate market one or the other way. Accurate predictions of local house prices are important to prospective homeowners, developers, investors, appraisers, tax assessors and other real estate market participants, such as mortgage lenders and insurers [7]. These people typically rely on real estate agents or certified appraisers to get the estimation of the desired property. But, getting property estimation from them is not cheap; it costs estimation seekers a lot of money. Hence, it is an insistent demand by the current real estate industry to develop a logical scientific prediction model that is not only cheap and easy-to-use, but also reliable and accurate [8].

House selling price prediction is a challenging problem; the final selling price of a house depends on many different parametric and non-parametric features. Real estate prices are typically a chronological sequence with unknown statistical relationships influenced by many factors, which makes it difficult to predict house prices using predefined functions

[8]. Also, house price knowledge is structured and complex, which makes it difficult for traditional classifiers to learn the function and to accurately predict selling price of houses. We used data from Redfin.com and zillow.com for the training and testing purpose. Our, experimental results showed that house selling price prediction accuracy is significantly improved using the structured deep learning neural network approach.

2 Related Work

Previous work related to deep learning, machine learning techniques for predictive analytics and house price predictions is summarized in this chapter.

First few paragraphs present overview of work related to deep learning. Mitchell and Sheppard used a non-connectionist data dimensionality reduction approach including conventional Principal Component Analysis (PCA) and deep architectured PCA to generate features for image classification problem, and they concluded that deep architecture helps to improve the performance of the model [9]. Their attempt was to demonstrate the usefulness of deep architectures for non-connectionist models. The approach is not so closely related to our approach but the paper demonstrates that deep architectures can be used for non-connectionist models as well. Liao et al. presented structured learning deep architecture for phoneme recognition which when given a structured input learns to find the best-structured object based on the mapping relationships between the structures [10]. Their test results showed that the model outperforms other conventional approaches proposed for phoneme recognition. In their approach even though they have used structured input and structured learning process, the deep neural network used is fully connected whereas in our research we use a structurally connected deep neural network which is significantly different from their approach.

Van den dries and Wiering presented a specialized structured neural network approach along with a neural fitted temporal difference (TD) algorithm for learning and neural fitted TD-leaf algorithm to improve the learning process [11]. The combined approach is used to improve learning performance on the game of Othello. The neural network is connected in a way to focus on regions on the board rather than connecting every input neuron to every other neuron in a next layer. The performance of the structured neural network approach is compared with fully connected neural network approach and other linear models, their experimental results indicated that the structured neural network approach outperforms linear models and fully connected neural network model. Steeg et al. extended the above structured neural approach for the game of Tic-Tac-Toe 3D [12]. They added one more hidden layer to the approach that is fully connected to first hidden layer (structured) and the output layer. They observed that the added hidden layer enables the neural network to integrate patterns learned in the structured hidden layer. The paper compared the performance of three fully connected neural networks with different hyper parameters and two structured neural networks. Experimental results showed that structured neural network approach works better than fully connected neural network approach. Unlike these methods, in our approach we do not use fully connected layer at all, we present structured knowledge first and then design the structured deep neural network to match with the structured knowledge.

Zhang et al. presented a shrinking deep neural network 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 deep neural network reduced the model size and computation time without affecting performance. The paper does not justify reasons behind using shrinking deep neural network. Also, the paper does answer questions like why to decrease size from lower layers to higher layers? Why not increase the size instead? Or even if decreasing size is used, what is the systematic approach to decrease the layer size? As one can not just decrease the size by some random factor. In this approach, we present a systematic approach to getting the structured knowledge and the structured deep neural network.

Next few paragraphs talk about usefulness of neural networks for predictive analytics and in particular for house price predictions.

Frew and Donald Jud used hedonic modeling techniques to estimate the house prices in the greater Portland and Oregon region [7]. They observed that values of the house rise less than proportionally with the size and number of units and decline with age, but the marginal effect of aging is small. Nghiep and Cripps compared the predictive performance of artificial neural networks and multiple regression analysis for single-family housing sales [14]. They found that when enough data points are used for training, artificial neural networks performs better than multiple linear regression. Hamzaoui and Perez applied single hidden layer feed-forward backpropagation neural network to predict the sales price of residential properties in Casablanca, Morocco kingdom [15]. Their

experimental results suggested that artificial neural networks could be used as a tool for reliable prediction of house sales prices.

Hu Xiaolong et al. used backpropagation neural networks and Elman neural network to evaluate house sales price [8]. They found that both the approaches have reasonable accuracy for evaluating the prices. Zhang Xiao-li predicted real estate prices using fuzzy neural network having the ability of fuzzy reasoning and learning which is constructed with a combination of fuzzy reasoning technique with neural networks [16]. His research concluded that fuzzy neural network works better than traditional neural network approach. Chopra et al. used a latent manifold model with two trainable components to evaluate house sales price [17]. The first one is a parametric component that predicts the 'intrinsic' price of the house using its features and the second one is a non-parametric component which calculates desirability of the neighborhood. The study found that the latent manifold model performs better than pure parametric or non-parametric models. Lowrance used his own designed local linear model and random forest model to predict house sale prices in Los Angeles County [18]. In which random forest model he tested performed better than his own designed local linear model. Although above approaches predict house selling prices, none of them use advance deep learning approach to solving the problem. In this thesis, we use specialized structured deep neural network approach which outperforms other conventional approaches for house price predictions.

Ford et al. used a neural network model to detect suspicious bidders in online auctions [19]. Instead, of training the neural network on complete training dataset, they focused more on training with newly added data. They demonstrate that the incremental learning approach can be useful for quickly adapting to changing trends in bidding. However, this approach has not yet been applied to house sales price predictions. In this research, we demonstrate that incremental learning is a useful approach for house selling price predictions. Also, as the model learns in iterations, saving old training data is not a necessity anymore, which would save a lot of memory in predictive big data analytics.

3 Background Knowledge

In this section, we discuss some fundamental concepts which are necessary for understanding our approach.

3.1 Neural networks

Artificial neural network (ANN) commonly referred as ‘neural network’, is inspired by the human brain. Neural network structure and the learning process is an attempt to simulate the brain. The brain has much more complicated structure and it consists of millions of neurons and billions of connections are there between these neurons. We can think of a human brain as a much more powerful computing machine than highly powerful supercomputers available these days. A human brain can make highly complex information processing decisions like image recognition, speech recognition, pattern recognition, more efficiently and faster than any supercomputer available.

3.1.1 Biological neuron

Biological neurons are connected to each other through synapses. Neuron has a structure with dendrites, nucleus, axon, synapses etcetera, which are connected to each other and

all these subparts have specific functionalities. A neuron perceives information through the dendrites connected to the nucleus and the nucleus does all the information processing and computations inside. If the computation reaches a certain value, neuron produces output also called as neuron fires. Axon then transmits the outcome as electric signals to other neurons connected through synapses. This is then received by next connected neurons through dendrites, so on and so forth.

3.1.2 Perceptrons

Perceptrons are designed by Frank Rosenblatt in the 1950s and 1960s, inspired by earlier work by Warren McCulloch and Walter Pitts [20]. More advanced neural network types are being used these days, however, perceptrons are still the building blocks of the neural networks. To better understand how neural networks work? The best way, I think, is to first understand how perceptron works?

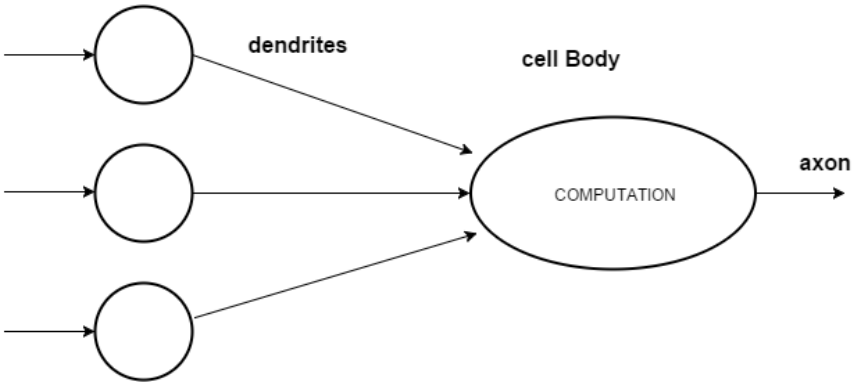


Figure 3.1 Biological neuron representation

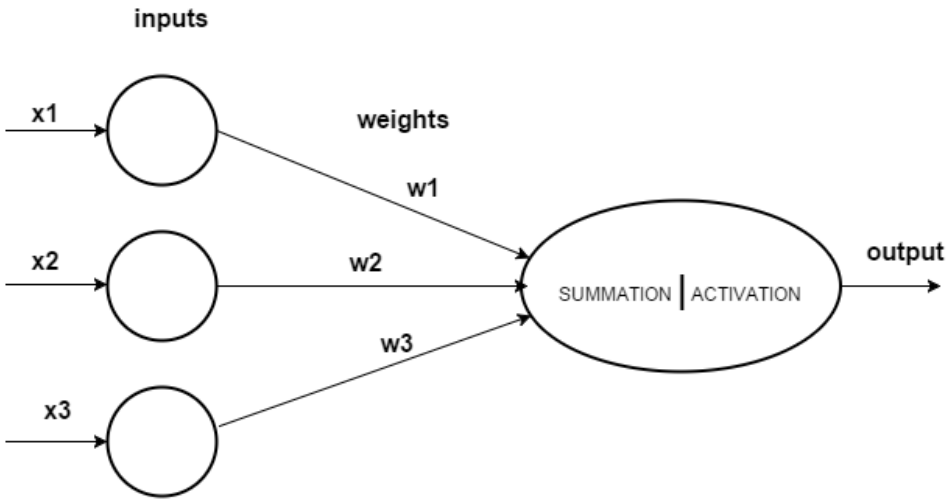


Figure 3.2 Perceptron structure

Figure 3.1 represents the structure of a biological neuron. A simple perceptron structure is as shown above in figure 3.2; it's structured pretty much similar to biological neuron. A biological neuron receives other biological neurons input through dendrites whereas perceptron receives binary inputs through connections and each connection in perceptron has a weight associated with it. Output neuron in perceptron does all the computation and if the computed value is greater than the threshold, perceptron fires, and outputs a binary value which then sends the outcome to other perceptrons through connections.

$$\text{Weighted sum} = \sum_{i=1}^{n} x_i w_i \qquad 3.1.2(a)$$

Given 'n' number of inputs from $x_1,\ x_2, x_3 \ldots.. x_n$ and 'n' corresponding weights from $w_1,\ w_2, w_3 \ldots.. w_n$ then weighted sum is the summation of the product of inputs and its corresponding weights as shown in above equation 3.1.2(a).

Output of the perceptron is given by following equations 3.1.2(b) and 3.1.2(c),

$$\text{Output} = 0: \text{ if } \sum_{i=1}^{n} x_i w_i <= \text{threshold} \qquad 3.1.2(b)$$

And

$$\text{Output} = 1: \text{ if } \sum_{i=1}^{n} x_i w_i > \text{threshold} \qquad 3.1.2(c)$$

Threshold is also called as bias (b) and value of bias is negation of value of threshold,

$$\text{bias (b)} = - \text{ threshold}, \qquad 3.1.2(d)$$

Using this value in above equation 3.1.2(d) of perceptron,

$$\text{Output} = 0: \text{ if } \sum_{i=1}^{n} x_i w_i + b <= 0 \qquad 3.1.2(e)$$

And

$$\text{Output} = 1: \text{ if } \sum_{i=1}^{n} x_i w_i + b > 0 \qquad 3.1.2(f)$$

By changing a number of inputs, values of inputs, values of weights and bias, we can have different models of perceptrons.

3.1.3 Multi-layer perceptrons (MLPS)

Multilayer perceptrons are multiple layers of perceptrons, in which, every unit in the previous layer is connected to every other unit in the next layer, as shown in below figure 3.3. An example multilayer perceptron shown in the figure has three layers, an input layer, a hidden layer and an output layer.

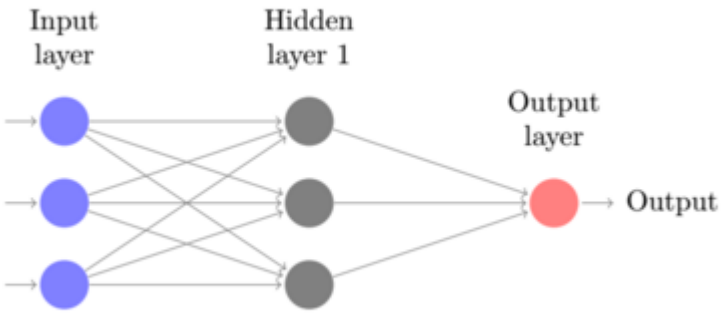


Figure 3.3 MLP structure

Three hidden units in the middle layer can be considered as three separate perceptrons, each perceptron receives three inputs from the input layer and has different weight matrix associated with it. Each hidden unit then does the computation using inputs and weights and makes a decision. Three hidden units in the middle layer make three simple decisions and a neuron in output layer then use outputs from these hidden units to make a higher level decision. In this way, multilayer perceptrons can be used to approximate more complicated functions.

3.1.4 Sigmoid function

Consider a simple perceptron with inputs, weights, and bias as shown in figure 3.4. Using perceptron formulas from equation 3.1.2(e) and 3.1.2(f) we get the output as,

$\sum_{i=1}^{n} x_i w_i$+ b = 8*1 + 10*2 + 20*3 - 90

= 88 - 90

= -2

As $\sum_{i=1}^{n} x_i w_i$+ b is < 0, Output = 0.

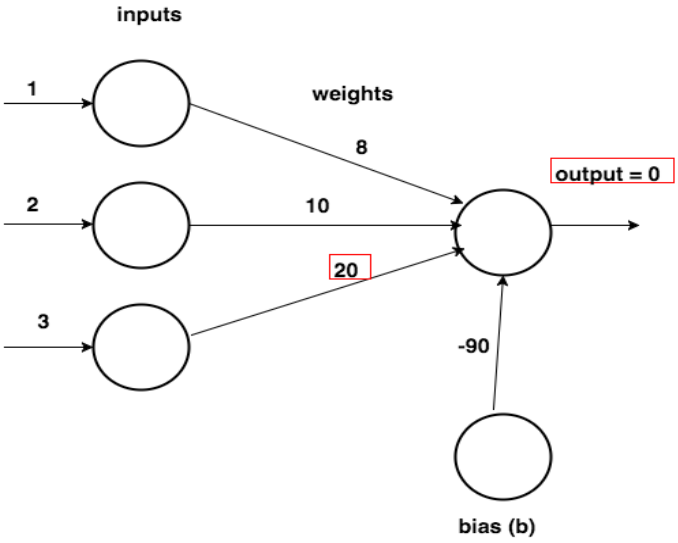


Figure 3.4 Perceptron A

In figure 3.5 below, weight on the third edge has been changed from 20 to 21, now let us see how this small change in the weight affects the final output.

Using perceptron formulas from equation 3.1.2(e) and 3.1.2(f) we get the output as,

$\sum_{i=1}^{n} x_i w_i$+ b = 8*1 + 10*2 + 21*3 - 90

= 91 - 90

= 1

As $\sum_{i=1}^{n} x_i w_i$+ b is > 0, Output = 1.

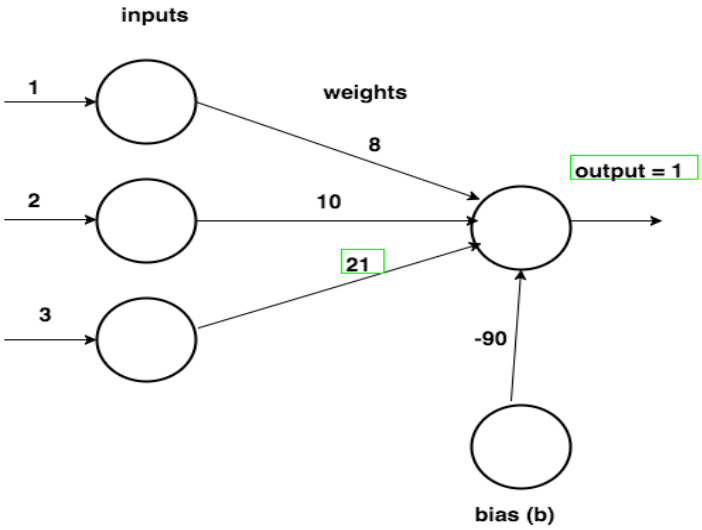


Figure 3.5 Perceptron B

As we can see from the above example, a small change in one of the weights completely reversed the output. As we don't want this to happen we use the continuous nonlinear sigmoid function.

The equation for sigmoid function and the sigmoid function curve is as shown in figure 3.6.

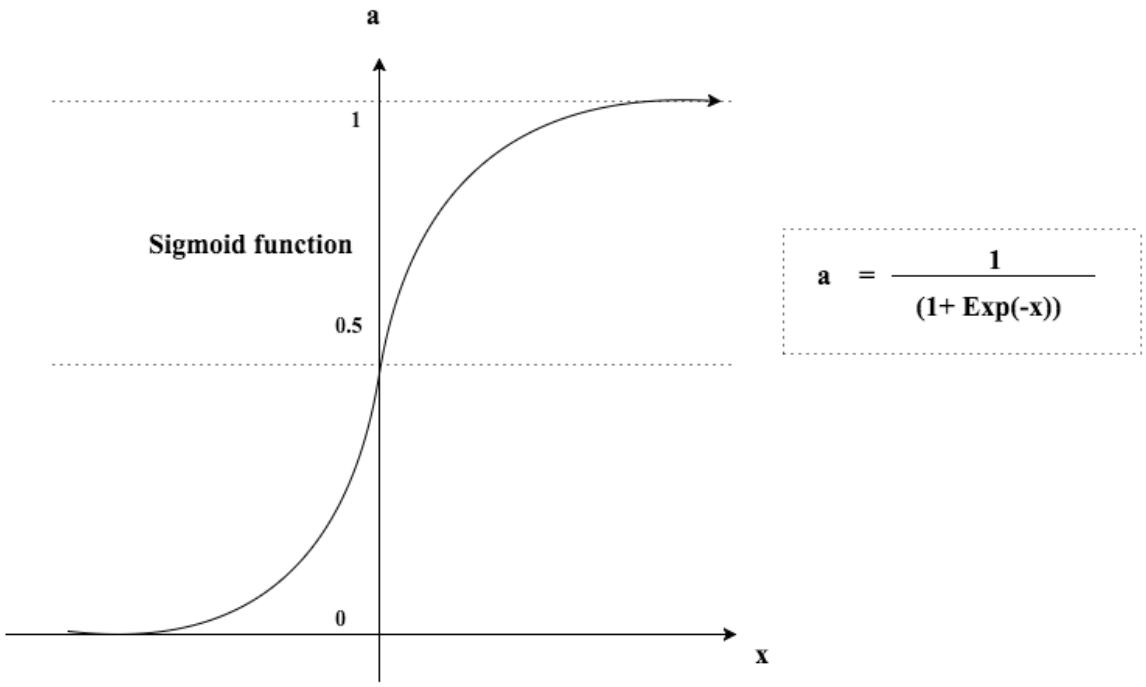


Figure 3.6 Sigmoid function

So, putting the values in the sigmoid equation given in the figure, the output would be,

$$\text{output} = \frac{1}{1 + exp(-\sum_{i=1}^{n} x_i w_i - b)}$$

Similar to perceptron this sigmoid neuron has inputs, weights, and an output. But, inputs and the output of a sigmoid neuron can take any value from 0 to 1, for example, the output value of 0.15 is possible for the sigmoid neuron. Whereas in perceptron, the inputs and the outputs are binary. Hence, the sigmoid function is used to add nonlinearity to a neural network.

4 Conventional Approaches

4.1 Real-time price-per-square-foot approach

The price-per-square-foot approach is commonly used for house price predictions in Asian countries. The price of a house is calculated using only two features, living area of the house and nearby market price-per-square-foot. Algorithm below describes the price-per-square-foot approach.

Algorithm: Price-per-square-foot

Input: training dataset, test dataset and window size
Output: Predicted selling price for every house in the test dataset

---

1. **For** each data point in the test dataset
2. Get price-per-square-foot for every house in the training dataset
3. Get average price-per-square-foot '**avgPrice**'
4. Predict price for an untested house from the testing dataset
   Predicted price = avgPrice * square footage of the house
   return Predicted price
5. Label the house
6. Add this labeled house data point to training dataset
7. Iterate the training dataset and check sold date of every house
   if ((sold date - current date) > window size)
   remove data point from the training dataset //real-time learning approach

---

Figure 4.1 Price-per-square-foot algorithm

In this, nearby market per-square-footage price for a particular area is multiplied by the total square footage of the house, and the resultant number is the final selling price of that house. Here in the United States, house market is different, even though square footage of a house is the most important and most impacting factor on selling price of a house; the final selling price varies significantly based on many other dependent factors.

In this approach, first, the average price-per-square-feet are calculated for all recent data points from the training dataset. As house market changes with time; price-per-square-feet is calculated in real-time using recently sold houses only. The window size of 6 months is used; this means, at any time, only houses sold in last 6 months will be used to calculate the average. This number is then multiplied by the square footage of a house from testing dataset to get the final selling price prediction. Experimental results show that other advanced approaches outperform naive price-per-square-feet approach.

4.2 Multivariate linear regression approach

Multivariate linear regression is one of the techniques used for predicting one or more dependent variables using multiple independent variables. The approach can also be used to estimate the correlation between dependent and independent variables. Here, input features are referred as independent features whereas predictions are referred as

dependent features. Independent variables can be continuous or categorical or blend of both the types; same goes for dependent variables as well [21].

Now, let us see how linear regression with a set of dependent variables and an independent variable works. Let a_1, a_2, a_3,… a_n be a set of 'n' dependent variables and Y be the dependent variable. Multivariate linear regression equation for a random sample unit would be

$$Y = \beta_0 + \beta_1 a_1 + \beta_2 a_2 + \beta_3 a_3 + \ldots + \beta_n a_n + \varepsilon$$

Where,

ε is an error

β_0 is an intercept

And β_1 to β_n are regression coefficients

Training data points are used to calculate values of regression coefficients, intercept, and error; which then are used for testing.

House selling price prediction is a classic regression problem with selling price as the dependent variable and many independent variables such as number of beds, number of baths, square footage, and lot area etcetera. Here, Microsoft excel regression analysis tool is used to get the analysis and required values of the constants. Experimental results show that other advanced approaches outperform conventional multivariate linear regression approach but still multivariate linear regression has reasonable predictions.

5 Deep Learning

Deep learning is about learning multiple levels of representations [22]. Deep learning was first used in 2006, in recent years it has gained significant attention from the machine learning and artificial intelligence researchers. Ability to automatically learn hidden features from data in layered fashion through both supervised and unsupervised training makes deep learning interesting and useful. Recently, deep learning algorithms, convolutional neural networks, and recurrent neural network have been useful and efficient for image recognition and speech recognition respectively.

5.1 What is deep learning?

As said earlier, deep learning is about learning multiple levels of representations [22]. Deep models learn features from the data in layers, in which, every layer is a more abstract representation of features as compared to features represented by the previous layer.

As an example, if we have a deep neural network model designed for face detection, with raw face image pixels as input neurons, few hidden layers and an output neuron which would be whether the image contains face or not. If one looks into this model and tries to understand what neural network has learned, the person would most likely find some

structure in there. The first hidden layer may just recognize edges, the second hidden layer would learn more abstract features like shapes, and third hidden layer would represent even more abstract features like presence of ear, nose, eyes etcetera.

5.2 How deep learning is different from shallow learning?

Neural networks with more than one hidden layer can be claimed as deep neural networks; whereas, single hidden layer neural networks are shallow neural networks. As we know, technically, if given enough hidden units shallow neural networks can approximate any function with remarkable accuracy, this property is known as universal approximation property [23]. Thus, the question here is why we need deep neural networks then?

What universal approximation property doesn't tell us is how many hidden units shallow neural network needs to approximate any particular function. Deeper neural networks are cheaper means using deep neural networks we may represent a particular function with less number of hidden units as compared to shallow neural networks. Experimental results show that there is a range of families of functions for which we can have an exponential advantage in terms of a number of hidden units [22]. Deeper neural networks take less number of hidden units to represent the function because of the notion of reuse.

In this, lower level features are reused to produce higher level features or more abstract features [24], [25].

To see how notion of reusability works in deep architectures, we take an example of a computer program as explained by Yoshua Bengio in his talk on theoretical motivations of deep learning during deep learning summer school 2015. Figure 5.1 and figure 5.2 are inspired by example figures used in his presentation. From figure 5.1 and 5.2 below we can see how sub codes are shared in deep computer program; whereas in shallow computer program sub codes are not shared.

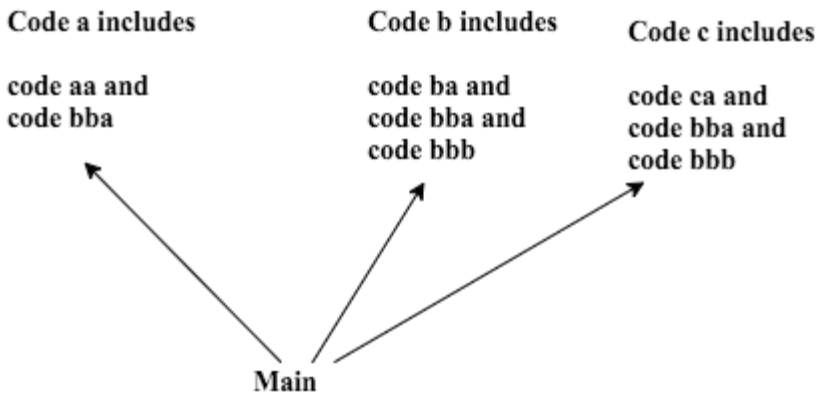


Figure 5.1 Shallow computer program

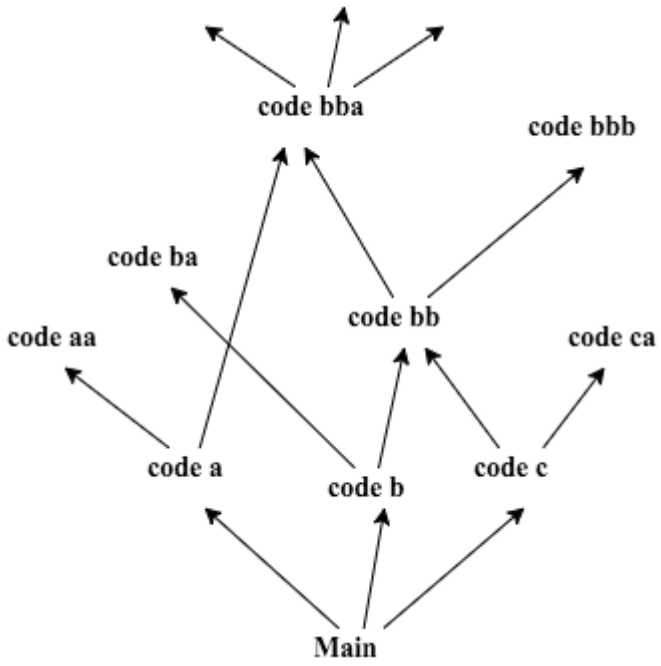


Figure 5.2 Deep computer program

This is how deep architectures help to reuse features of lower layers into higher layers. Power of reusability allows deep architectures to represent a function using fewer number of units as compared to shallow architectures.

5.3 Need of deep learning for house price prediction problem

As house price prediction problem has non-linearities in there, trying advanced neural network approach sounded more reasonable. As a result, we tried neural network approach and experimental results showed that neural network approach actually works

better than conventional multivariate linear regression and naive price-per-square-feet approach.

As we know, there is no one best algorithm for every problem. An algorithm which works best for a certain set of problems may perform worst for another set of problems. So, it's important to study the problem and then choose a set of suitable algorithms to try. The first thing to understand the problem was to understand the underlying semantics of the house data. Thus, we first tried to hand design the structured knowledge of the problem and the structure designed was reasonable as well, but there could be many different possible structures. So, we designed the knowledge structure using the neural network connection weights and as we found that there is a structure in there, we thought deep learning would be a better approach to approximate this sort of function.

5.4 Structured knowledge

As explained in earlier section knowledge structure designed with the help of neural network would be more meaningful and reliable. However, choosing neural network structure is the next task. The neural network structure can be chosen step by step as explained in the algorithm below.

Algorithm: Neural network for getting structured knowledge

Input: Training dataset and test dataset
Output: Neural network for getting structured knowledge

1. Choose (**number of hidden layers**, **number of units in each hidden layer**)
2. Design a neural network model
3. Train the model using the training dataset
4. Test the model on testing dataset and read the accuracy
5. // If accuracy is reasonable, look into the model and calculate number of strong and weak connections
   if (model accuracy >= desired accuracy)
   // threshold is defined to differentiate strong and weak connections
   for each connection 'c' in the model
   if (weight * weight >= threshold * threshold)
   add 'c' to strongConnections
   else
   add 'c' to weakConnections
6. calculate % of strong and weak connections
7. repeat steps 1-6 several times using different number of hidden units and number of hidden layers
8. Choose a neural network with highest percent of weak connections and reasonable accuracy

Figure 5.3 Neural network for getting structured knowledge algorithm

First, choose a simple neural network. Train this model using training dataset and then test it using the testing dataset. If neural network performance is reasonable then we may look into the model to get the insights. If the model accuracy is not reasonable then no need to analyze the model, as that neural network may not represent the best structure. Next step is calculating the number of weak connections and strong connections in the model. A threshold is defined to determine whether the connection is strong or weak. If

the square of the weight of a connection is less than square of the threshold then that connection is added to weak connection list or else added to strong connection list. Weak connections percent is calculated by dividing weak connections by the total number of connections in the network multiplied by 100. Different neural network models with a different number of hidden layers and hidden units are tried and a neural network with reasonable accuracy and highest percent weak connections is chosen for getting the knowledge structure.

Once we have the neural network, next task is to design the structured knowledge. Algorithm for designing structured knowledge is presented below.

Algorithm: Structured knowledge

Input: Neural network structure
Output: Structured knowledge

```
1. for each hidden layer 'l'
2.     for each hidden unit 'i'
           Get list of contributors
                   // threshold is separately defined for each hidden unit based
                   // on the range and distribution of weights
                   define threshold
                   if (weight * weight >= threshold * threshold)
                           add input feature to contributor's list
3.         Name the unit based on contributing features and their
                   corresponding weights
4.         remove unnecessary connections based on the node name
5.         add more connections(if needed) based on the node name
```

Figure 5.4 Structured knowledge algorithm

To design the structure, we first need to look into the neural network and name each hidden unit in it. We go one layer at a time from a lower layer to higher layers. We first get a list of contributors to each hidden unit in the first hidden layer. For that, we define a threshold for each hidden unit based on the range and distribution of the weights. A threshold is chosen so that major contributors would be added to the contributors list. Let's take example of designing structured knowledge for house price prediction problem. We want to name just a single hidden unit as an example, figure 5.5 shows a hidden unit to name and it's contributing input features.

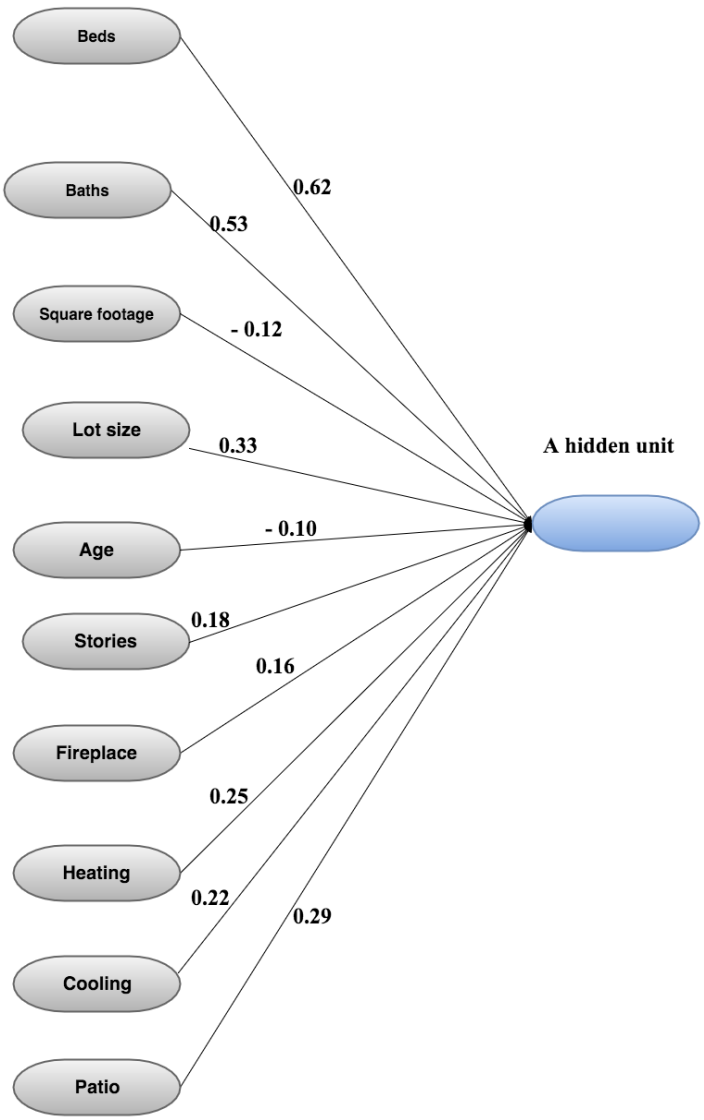


Figure 5.5 Getting structured knowledge

To get the list of contributors we will have to first define threshold. As we can see in the figure 5.5, half of the weights are greater or equal to 0.25. So we define threshold as 0.25. If the square of the weight of a connection is greater than or equal to the square of threshold, then that connection is added to the contributors list. So, contributors would be

beds, baths, lot size, heating, and patio. Based on this selected contributing features we name the node. Four out of this contributing features are interior house features. Hence, we name the hidden unit as ‘interior features’. Now, based on the node name we add other less contributing features if needed. From figure 5.5 we can see that input features ‘fireplace’ and ‘cooling’ can be added as contributing features to node ‘interior features’. Also, feature ‘lot size’ is one of the major contributing feature but it’s not an interior house feature so we consider it as a noise and we remove its connection to the node ‘interior feature’. Resulting structure for this hidden unit is as shown below in figure 5.6.

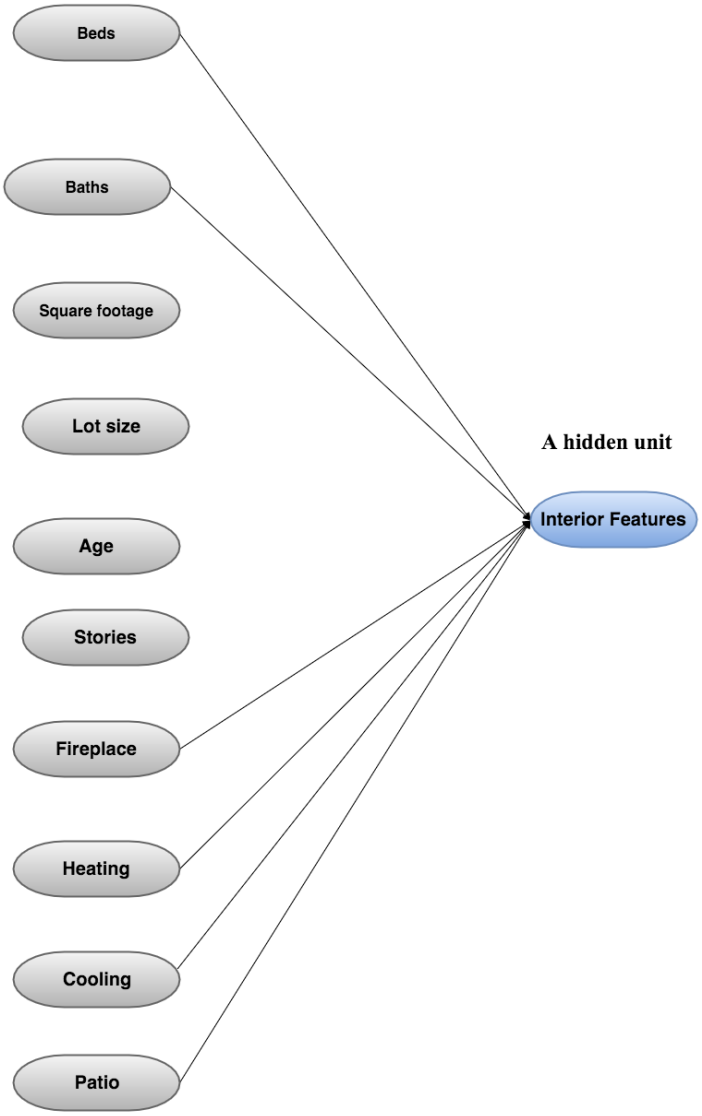


Figure 5.6 Structured knowledge example

So on and so forth we name every hidden unit in the first hidden layer. Once it is done, the first hidden layer would be considered as input layer and above procedure would be repeated for naming every hidden unit in all of the remaining hidden layers. Hence, after

naming every hidden unit and determining contributors to those hidden units we get the final structured knowledge.

6 Structured Deep Neural Network Approach

6.1 Framework for structured deep neural network

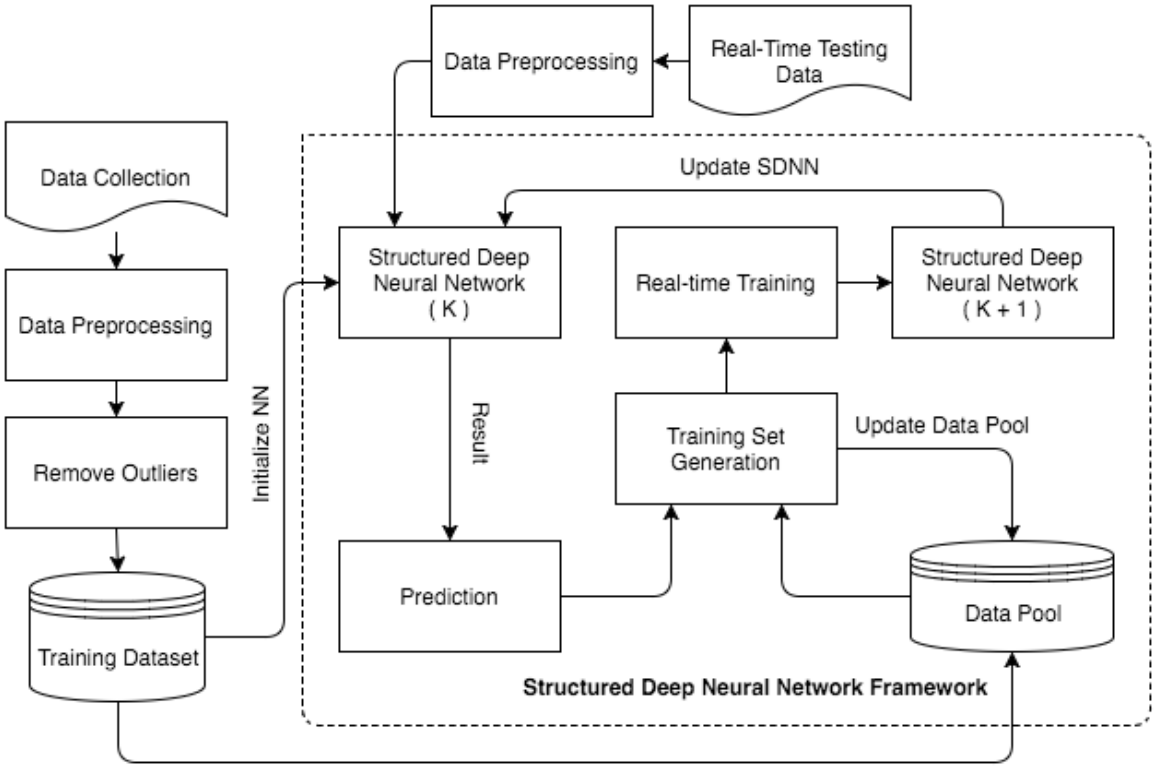


Figure 6.1 Structured deep neural network framework

Framework for structured deep neural network for predictive analytics is as shown in the above figure 6.1. This sub-section would give overall idea and process of predicting future outcomes using historical data and structured deep neural network. Algorithm for the prediction is presented below.

Algorithm: Structured Deep Neural Network Framework

Input: training dataset, test dataset, and window size
Output: Prediction for every datapoint in the test dataset

```
1.  Data collection
2.  Data preprocessing
3.  Remove outliers
4.  Prepare training dataset
5.  for each datapoint 'i' in the testing dataset
6.          train the model using updated training dataset
7.          test 'i'
8.          label 'i'
9.          add 'i' to training set
10.         // update training dataset
            for each data point 'j' in the training dataset
                    if ((timestamp - current time) > window size)
                            remove 'j' from the training dataset
```

Figure 6.2 Structured deep neural network framework algorithm

The very first step is collecting the data. As collected data is raw, it is not in proper format and also it contains lot of unnecessary information. So, we preprocess the data to get necessary fields in the desired format. Also, for few data points, all the required features will not be present or the available information could be wrong as well, and having this wrong information could negatively affect the training of the neural network. To avoid false training, outliers are removed from the training dataset.

Now, we have a training dataset ready, which will be used to train the deep neural network. Algorithm to designing structurally connected deep neural network is presented in section 6.1.1. Before starting the training process, the neural network will be initialized with random weights, and then it would be trained using data points from the training dataset. This trained deep neural network is then used for real-time future predictions. The deep neural network model keeps on predicting outcomes for newly added testing data. Once the datapoint is labeled, it will be added to the data pool and it will be used for training afterward, meanwhile, few old data points will be removed from the data pool to make sure neural network learns from recently labeled data points only. The process is explained in more detail in sub-section 6.1.2. After that, the neural network will be trained again using this new training set and the newly trained neural network will be used for future predictions. So on and so forth, the neural network will keep updating itself continuously to adapt to new market trends, making it a real-time learning deep neural network [19].

6.1.1 Deep neural network structure

The deep neural network is designed to exactly match with the knowledge structure. The structured neural network would have the same number of layers and the same number nodes in each layer as in the knowledge structure. Choosing number of hidden layers and number of hidden units in each layer is one of the most difficult and time-consuming tasks while setting up a neural network model. However, using proposed structure

knowledge to set up a neural network solves the problem. Moreover, once a number of layers and a number of nodes in each of layers are initialized, next task is to structurally connect the neural network.

Steps to connect the neural network are summarized in the algorithm below.

Algorithm: Structurally connected deep neural network

Input: Structured knowledge
Output: Structurally connected deep neural network

1. Initialize number of layers 'L' and number of units in each layer 'N_L' same as in structured knowledge
2. // connect the neural network structurally
   for (layerNum = 1; layerNum < L; LayerNum++)
   for (unitNum = 0; unitNum < N_L ; unitNum++)
   if (structure knowledge contains an edge from 'unitNum' to a unit 'x' in the next layer)
   Connect (unitNum, next layer unit 'x')
3. // add bias to every unit in hidden layers and in output layer
   for (layerNum = 2; layerNum <= L; LayerNum++)
   for (unitNum = 0; unitNum < N_L ; unitNum++)
   Connect (bias, unitNum)

Figure 6.3 Structurally connected deep neural network algorithm

First, we the deep neural network is connected structurally to match with the knowledge structure of the problem, and then we add bias to each layer except to the input layer.

6.1.2 Real-time learning

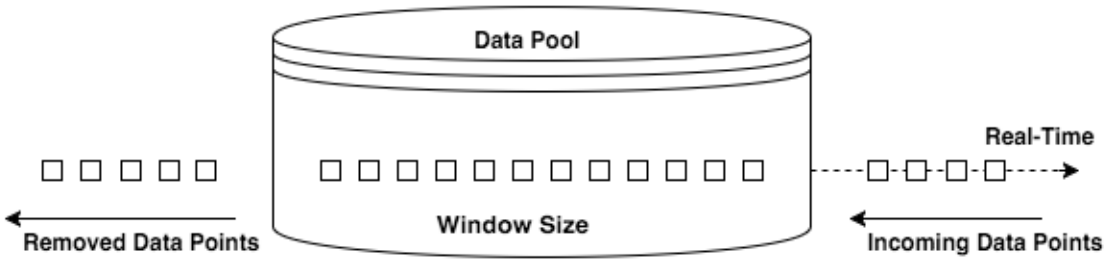


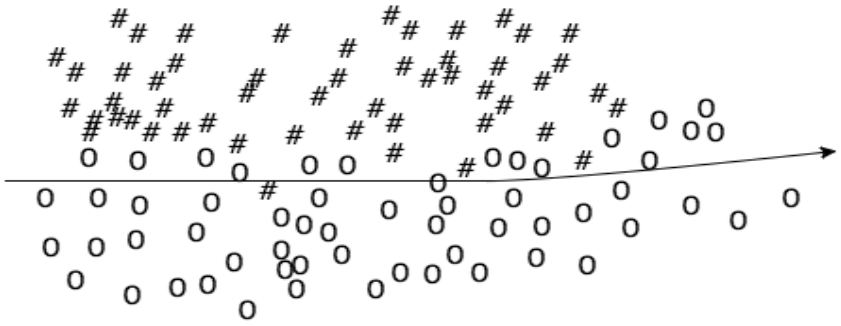
Figure 6.4 Real-time learning

Trends keep changing continuously, in most cases, only recently labeled data points can reflect current market trends accurately. As a result, at any time, only data points labeled no older than defined window size are used to train the deep neural network. The above figure 6.4 is inspired by real-time learning window size figure presented in [19]. As shown in figure 6.4 new data points are added to the training data once they are labeled, and simultaneously, data points which are older than defined window size are removed. This new training data will be used to train the deep neural network again, and this updated neural network will be used to predict future outcomes. Every time new labeled data is available, deep neural network updates itself with the recent data, making it a real-time learning deep neural network.

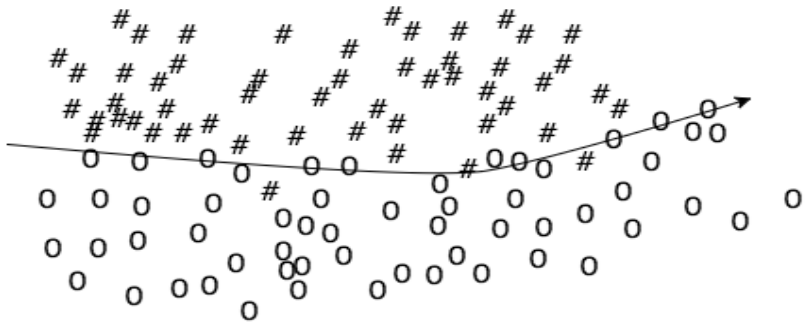
6.2 Fitting the model

Fitting the curve properly is one of the important but difficult tasks while training the neural network. A well fitted neural network gives best results. There is always a chance that one is either underfitting or overfitting the function. Overfitting is a most common but challenging problem in machine learning. In deep learning, the neural networks generally have more hidden layers and a large number of hidden units which corresponds to higher capacity, hence higher chance of overfitting. As a result, overfitting is a serious problem in deep learning.

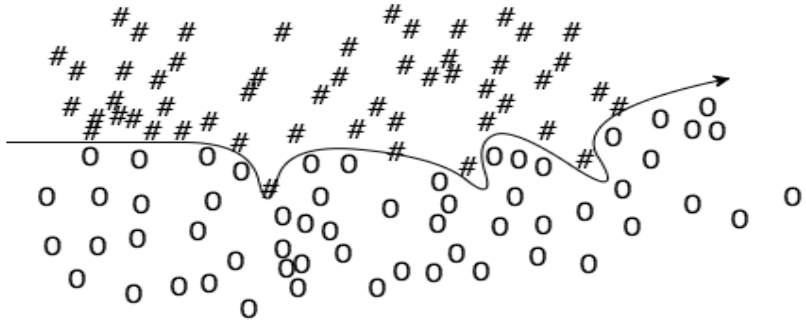
Different fitting techniques like dropout, early stopping, batch training, and regularization etcetera are used to reduce chances of overfitting. Figure 6.5 shows the concept of under fitted, well fitted and over fitted function curve.



Under fitted, Figure (a)



Well fitted, Figure (b)



Over fitted, Figure (c)

Figure 6.5 Fitting the model

Stopping the training too early often leads to under fitted curve as shown in figure 6.5(a). Few of the reasons for over fitting the curve are over training the neural network, training data set size and variety, using the test set to adjust the hyperparameters etcetera. When a testing error is significantly larger than the training error, we know that there is an overfitting problem in the model. Overfitted curve is shown in figure 6.5(c). When the curve achieves the global minima and there is not much difference between training error and testing error, we say that the curve is well fitted and the model has achieved good generalization. Well fitted curve is as shown in figure 6.5(b).

In this research structured deep neural network is used, one of the advantages of using the structured neural network is, it helps to avoid the overfitting problem. When the neural network has a large number of hidden units it could allocate neurons for every single space of the function, which allows the model to overfit. As the structured neural network is structurally connected, it has only necessary connections. Less number of connection weights makes the neural network less powerful and hence less chance of overfitting. Though, over training the model may lead to an overfitting problem. Hence, early stopping is used to avoid overfitting. In early stopping, training is stopped just before the error on validation set starts to climb.

7 Case Study

In the following case study, we use house price prediction problem as an example to demonstrate the usefulness of structured deep neural network for predictive analytics. We present experiments and results in this section.

7.1 Data collection and preprocessing

7.1.1 Data crawling

Data is crawled from leading real estate listings website Zillow.com, a well-known website in real estate market. The website keeps all new and the past house listings data up to date and the site has more house features listed as compared to other house listings websites. Import.io, a data crawling and data scraping tool are used to crawl the data. The website specific and data specific robot is built using import.io, which automatically collects house listings data from the website. We collect data from three different zip codes 02127, 02125 and 02128.

7.1.2 Features selection

All important house features, market features, public records of the house and neighborhood features are collected from the website. Set of total 17 features are defined including number of beds, number of baths, square footage, lot size, built year, yearly tax, sold date, house type, similar houses average sold price, nearby schools average ratings, fireplace_flag, beachfront_flag, number of stories, heating, cooling, patio, and park.

7.1.3 Outliers removal

Studies and experimental results show that outliers have a significantly negative impact on the training of neural networks. Firstly, house data points with insufficient information, for example, a house data point with missing square footage, will not be considered for training. Secondly, data points having wrong information, for example, if a house is sold for $1, then mostly it is a wrong information, this data point will not be considered for training. Also, average selling price-per-square-feet of houses sold in last 6 months is tracked in real-time, and houses with too high price-per-square-feet or too low price-per-square-feet will not be used for the training.

7.2 Knowledge structure

Different types of fully connected deep neural networks were trained to get the structured knowledge through the weights learned by the neural network. Neural network with a maximum number of weak connections between the neurons and with good output predictions would represent the best structure knowledge of the problem. After trying different types of fully connected neural networks, a fully connected network with two hidden layers containing 10 neurons each as shown in figure 7.1 was chosen for getting the structure. In this neural network, input layer has 15 neurons which are features of the house and the single output neuron is the predicted price.

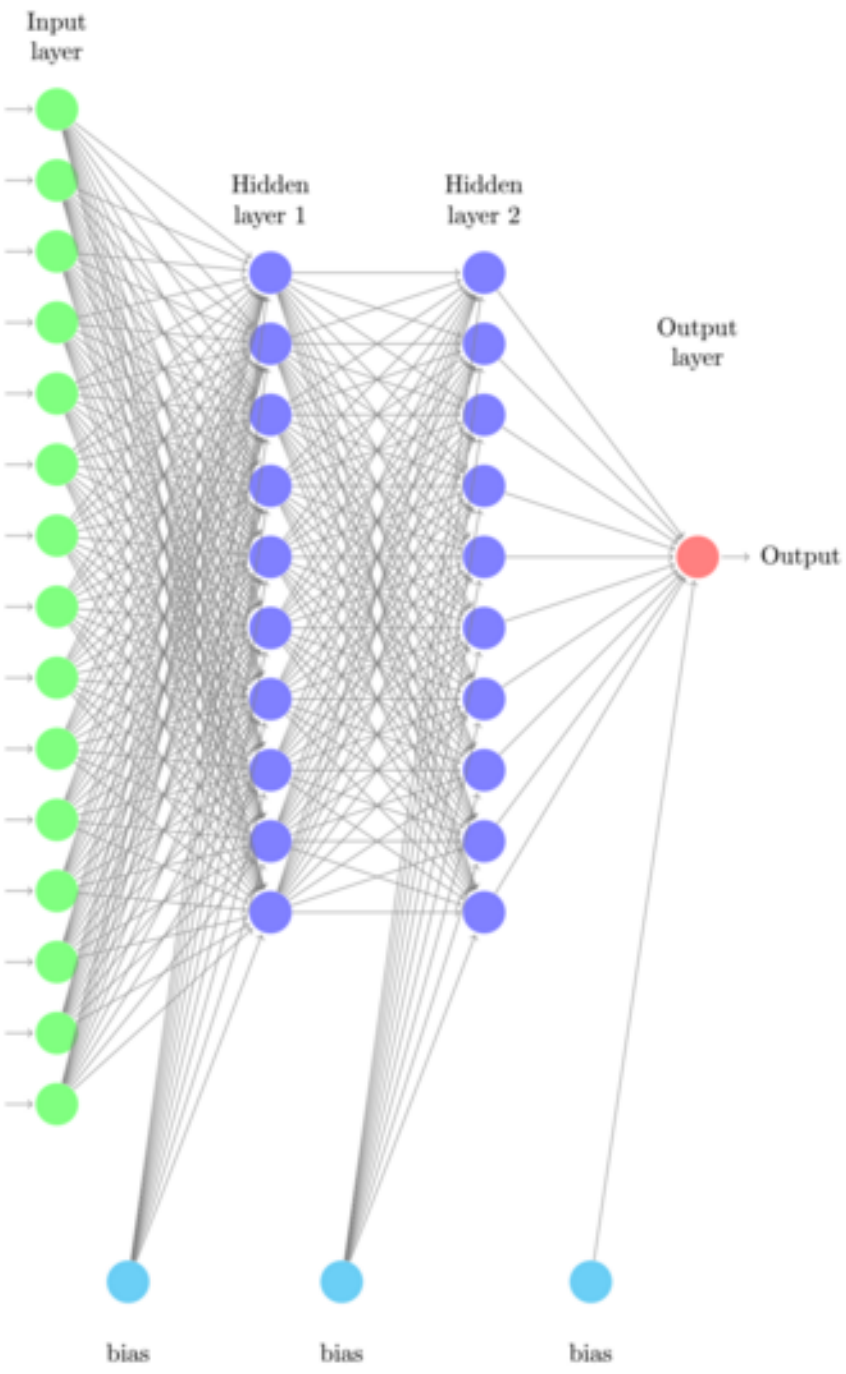


Figure 7.1 Fully connected neural network

Connections weight for the neural network shown in figure 7.1 are listed below.

Weights from input layer (i) to first hidden layer (o):

// First neuron (i =0) is used as bias

(i=0,o=1)=-0.0064 (i=0,o=2)=-0.2563 (i=0,o=3)=0.2205 (i=0,o=4)=0.0457

(i=0,o=5)=0.1465 (i=0,o=6)=-0.1378 (i=0,o=7)=0.0540 (i=0,o=8)=0.4748

(i=0,o=9)=0.2876 (i=0,o=10)=0.4421

// Number of beds (i=1)

(i=1,o=1)=-0.0436 (i=1,o=2)=-0.3460 (i=1,o=3)=-0.3116 (i=1,o=4)=-0.4250

(i=1,o=5)=0.0349 (i=1,o=6)=0.5401 (i=1,o=7)=-0.1325 (i=1,o=8)=0.0509

(i=1,o=9)=0.2089 (i=1,o=10)=0.3017

// Number of baths (i=2)

(i=2,o=1)=0.4769 (i=2,o=2)=-0.0098 (i=2,o=3)=0.2753 (i=2,o=4)=0.2614

(i=2,o=5)=0.3160 (i=2,o=6)=0.3488 (i=2,o=7)=0.1215 (i=2,o=8)=0.3316

(i=2,o=9)=-0.2727 (i=2,o=10)=-0.4393

// Square footage (i=3)

(i=3,o=1)=0.4620 (i=3,o=2)=0.2055 (i=3,o=3)=0.2363 (i=3,o=4)=-0.3612

(i=3,o=5)=-0.0463 (i=3,o=6)=-0.4973 (i=3,o=7)=-0.4028 (i=3,o=8)=-0.1973

(i=3,o=9)=-0.1059 (i=3,o=10)=0.3211

// Lot area (i=4)

(i=4,o=1)=0.4715 (i=4,o=2)=0.3657 (i=4,o=3)=0.1126 (i=4,o=4)=-0.3210

(i=4,o=5)=-0.2824 (i=4,o=6)=0.3545 (i=4,o=7)=-0.4903 (i=4,o=8)=0.1923

(i=4,o=9)=0.2713 (i=4,o=10)=0.2127

// Age of a house (i=5)

(i=5,o=1)=-0.3678 (i=5,o=2)=0.2506 (i=5,o=3)=0.4098 (i=5,o=4)=-0.5498

(i=5,o=5)=-0.1044 (i=5,o=6)=0.3019 (i=5,o=7)=0.1379 (i=5,o=8)=0.5667

(i=5,o=9)=0.1362 (i=5,o=10)=-0.2989

// Yearly tax (i=6)

(i=6,o=1)=0.1827 (i=6,o=2)=0.2421 (i=6,o=3)=0.1963 (i=6,o=4)=0.3941

(i=6,o=5)=-0.0150 (i=6,o=6)=-0.0635 (i=6,o=7)=0.3733 (i=6,o=8)=-0.0414

(i=6,o=9)=0.2518 (i=6,o=10)=0.3473

// Number of stories (i=7)

(i=7,o=1)=-0.3212 (i=7,o=2)=-0.4943 (i=7,o=3)=-0.4021 (i=7,o=4)=-0.5170

(i=7,o=5)=-0.0130 (i=7,o=6)=0.4456 (i=7,o=7)=0.3580 (i=7,o=8)=0.2403

(i=7,o=9)=-0.1041 (i=7,o=10)=-0.2702

// Fireplace_flag (i=8)

(i=8,o=1)=-0.1755 (i=8,o=2)=0.3817 (i=8,o=3)=-0.3812 (i=8,o=4)=-0.4047

(i=8,o=5)=-0.0784 (i=8,o=6)=-0.2698 (i=8,o=7)=0.5414 (i=8,o=8)=-0.4932

(i=8,o=9)=-0.0575 (i=8,o=10)=0.0342

// Waterfront_flag (i=9)

(i=9,o=1)=0.1806 (i=9,o=2)=-0.2754 (i=9,o=3)=0.3750 (i=9,o=4)=0.0771

(i=9,o=5)=0.3577 (i=9,o=6)=-0.4676 (i=9,o=7)=0.1453 (i=9,o=8)=-0.1133

(i=9,o=9)=0.0132 (i=9,o=10)=0.0988

// Heating(i=10)

(i=10,o=1)=-0.3710 (i=10,o=2)=-0.0192 (i=10,o=3)=0.2871 (i=10,o=4)=-0.1820

(i=10,o=5)=-0.4394 (i=10,o=6)=0.2300 (i=10,o=7)=0.4142 (i=10,o=8)=0.0972

(i=10,o=9)=-0.3565 (i=10,o=10)=-0.1101

// Cooling(i=11)

(i=11,o=1)=0.1825 (i=11,o=2)=-0.3395 (i=11,o=3)=-0.4445 (i=11,o=4)=-0.1823

(i=11,o=5)=0.3946 (i=11,o=6)=-0.2026 (i=11,o=7)=0.0499 (i=11,o=8)=0.1143

(i=11,o=9)=-0.1038 (i=11,o=10)=-0.0728

// Patio(i=12)

(i=12,o=1)=0.0804 (i=12,o=2)=0.2025 (i=12,o=3)=0.4370 (i=12,o=4)=0.0704

(i=12,o=5)=-0.0049 (i=12,o=6)=-0.1880 (i=12,o=7)=0.0965 (i=12,o=8)=0.4145

(i=12,o=9)=0.1054 (i=12,o=10)=-0.3937

// Parking(i=13)

(i=13,o=1)=0.1811 (i=13,o=2)=0.1137 (i=13,o=3)=0.3100 (i=13,o=4)=0.2388

(i=13,o=5)=0.0988 (i=13,o=6)=-0.1182 (i=13,o=7)=0.3073 (i=13,o=8)=-0.3692

(i=13,o=9)=-0.1651 (i=13,o=10)=0.3082

// Similar house average selling price(i=14)

(i=14,o=1)=0.6631 (i=14,o=2)=-0.1345 (i=14,o=3)=0.0790 (i=14,o=4)=-0.1827

(i=14,o=5)=-0.4121 (i=14,o=6)=0.2916 (i=14,o=7)=0.2547 (i=14,o=8)=-0.0916

(i=14,o=9)=0.8617 (i=14,o=10)=0.0094

// Nearby schools rating(i=15)

(i=15,o=1)=-0.3243 (i=15,o=2)=-0.1773 (i=15,o=3)=-0.2083 (i=15,o=4)=0.1809

(i=15,o=5)=-0.3989 (i=15,o=6)=-0.2178 (i=15,o=7)=-0.2616 (i=15,o=8)=-0.1773

(i=15,o=9)=0.2903 (i=15,o=10)=-0.4181

Weights from layer first hidden layer(i=0 to 10) to second hidden layer(o = 0 to 10):

(i=0,o=1)=-0.7149 (i=0,o=2)=0.2971 (i=0,o=3)=-0.0374 (i=0,o=4)=-0.2475

(i=0,o=5)=-0.1545 (i=0,o=6)=-0.6226 (i=0,o=7)=-0.2846 (i=0,o=8)=0.3769

(i=0,o=9)=-0.2086 (i=0,o=10)=0.6517

(i=1,o=1)=0.0699 (i=1,o=2)=0.1532 (i=1,o=3)=-0.2584 (i=1,o=4)=0.0994

(i=1,o=5)=0.4840 (i=1,o=6)=-0.3326 (i=1,o=7)=0.4750 (i=1,o=8)=-0.4917

(i=1,o=9)=-0.2467 (i=1,o=10)=0.1407

(i=2,o=1)=-0.0427 (i=2,o=2)=-0.4511 (i=2,o=3)=-0.1672 (i=2,o=4)=-0.4770

(i=2,o=5)=0.1182 (i=2,o=6)=0.2975 (i=2,o=7)=-0.2374 (i=2,o=8)=0.1325

(i=2,o=9)=-0.4232 (i=2,o=10)=-0.1577

(i=3,o=1)=-0.2495 (i=3,o=2)=0.3987 (i=3,o=3)=0.0158 (i=3,o=4)=0.1394

(i=3,o=5)=-0.0343 (i=3,o=6)=0.2058 (i=3,o=7)=-0.1306 (i=3,o=8)=-0.1528

(i=3,o=9)=-0.2100 (i=3,o=10)=0.0943

(i=4,o=1)=-0.0447 (i=4,o=2)=-0.1235 (i=4,o=3)=0.5280 (i=4,o=4)=-0.0560

(i=4,o=5)=0.3360 (i=4,o=6)=-0.2549 (i=4,o=7)=-0.3742 (i=4,o=8)=0.0817

(i=4,o=9)=0.5219 (i=4,o=10)=-0.3996

(i=5,o=1)=-0.1219 (i=5,o=2)=-0.1347 (i=5,o=3)=0.3843 (i=5,o=4)=-0.0748

(i=5,o=5)=-0.3115 (i=5,o=6)=0.1520 (i=5,o=7)=0.2382 (i=5,o=8)=0.0433

(i=5,o=9)=0.1221 (i=5,o=10)=-0.0897

(i=6,o=1)=-0.0185 (i=6,o=2)=-0.4812 (i=6,o=3)=0.2130 (i=6,o=4)=0.0898

(i=6,o=5)=-0.3610 (i=6,o=6)=-0.0097 (i=6,o=7)=0.1405 (i=6,o=8)=0.3082

(i=6,o=9)=0.2341 (i=6,o=10)=-0.1691

(i=7,o=1)=-0.4383 (i=7,o=2)=-0.0367 (i=7,o=3)=0.0754 (i=7,o=4)=0.3798

(i=7,o=5)=-0.1030 (i=7,o=6)=-0.4028 (i=7,o=7)=-0.0274 (i=7,o=8)=0.4009

(i=7,o=9)=-0.3274 (i=7,o=10)=-0.2211

(i=8,o=1)=0.1561 (i=8,o=2)=-0.2951 (i=8,o=3)=-0.2595 (i=8,o=4)=-0.1324

(i=8,o=5)=-0.2885 (i=8,o=6)=-0.3831 (i=8,o=7)=-0.2681 (i=8,o=8)=0.0770

(i=8,o=9)=-0.3567 (i=8,o=10)=-0.0980

(i=9,o=1)=-0.4540 (i=9,o=2)=-0.1322 (i=9,o=3)=-0.2114 (i=9,o=4)=0.0425

(i=9,o=5)=0.5738 (i=9,o=6)=0.0102 (i=9,o=7)=-0.3074 (i=9,o=8)=0.4289

(i=9,o=9)=0.3290 (i=9,o=10)=0.4975

(i=10,o=1)=-0.3180 (i=10,o=2)=-0.4401 (i=10,o=3)=-0.0967 (i=10,o=4)=-0.1437

(i=10,o=5)=-0.1681 (i=10,o=6)=0.1063 (i=10,o=7)=-0.1104 (i=10,o=8)=0.2115

(i=10,o=9)=-0.1287 (i=10,o=10)=-0.0039

Weights from layer second hidden layer(i=0 to 10) to output layer(o=1):

(i=0,o=1)=-0.2550

(i=1,o=1)=-0.3748

(i=2,o=1)=-0.3513

(i=3,o=1)=-0.1624

(i=4,o=1)=0.3376

(i=5,o=1)=0.3447

(i=6,o=1)=-0.2390

(i=7,o=1)=0.1202

(i=8,o=1)=-0.0093

(i=9,o=1)=-0.0847

(i=10,o=1)=0.3545

Based on connections weight of the input features we name each hidden unit and design a knowledge structure as explained in algorithm shown in figure 5.4. Example knowledge structure designed for house price prediction problem is as shown in figure 7.2. The first layer has 15 input features, the second layer has more abstract 10 features, the third layer

has, even more abstract 7 features which then contribute to the single neuron in the final layer, which is predicted the price of a house.

As we can see in figure 7.2, house price function has a layered structured knowledge and many of the hidden features contribute to more than one feature in the next layer. Which means features in the lower level layers are reused by more abstract features in the higher level layers. Thus, the knowledge structure indicates that deep learning is a better approach for house price prediction problem.

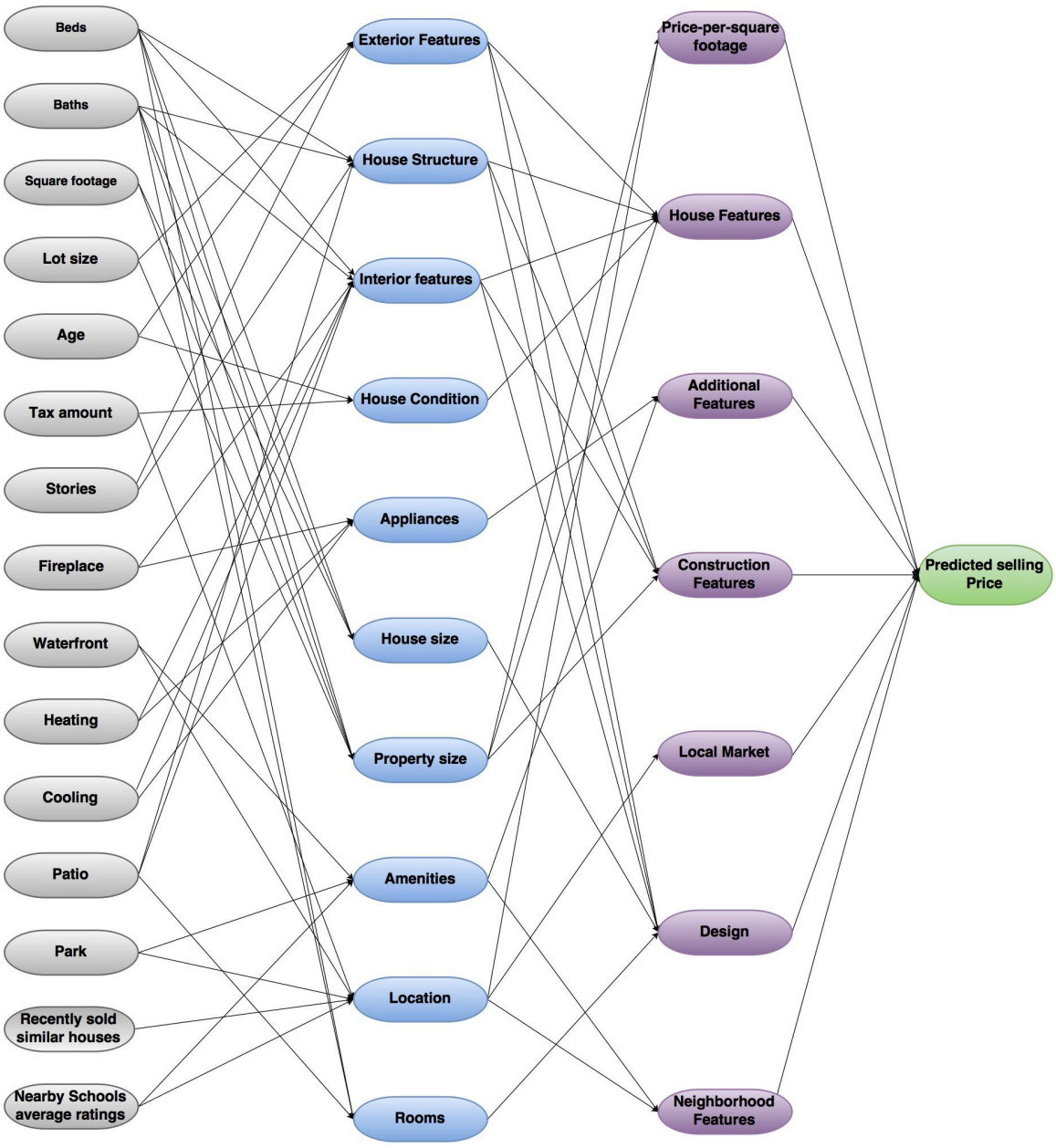


Figure 7.2 Structured knowledge representation

7.3 Deep neural network structure

The deep neural network is designed to exactly match with the knowledge structure; to get the structured neural network, we use algorithm shown in figure 6.3. The designed structured neural network has four layers, an input layer, two hidden layers and an output layer. It is trained using standard feed forward backpropagation algorithm and using problem-specific real-time training, fitting techniques and setting up suitable hyperparameters.

The first layer in the network that is input layer has 15 input features defined as numbers of beds, the number of baths, square footage etcetera as the neurons. We can say that all neurons in the input layer always fire, or in other words, all neurons in the input layer always produce output, as there is no bias (threshold) connected to input layer neurons. The first hidden layer has 10 hidden units and second hidden layer contains 7 hidden units as in the knowledge structure. The output layer has only one neuron, which is predicted selling price of the house.

7.4 Hyper parameters

7.4.1 Weights initialization

All weights in this work are initialized in the range -0.5 to 0.5. Experimental results indicate that better initialization may help to achieve better generalization. As weights on each connection are used during the learning process of the neural network, small weights make a neural network learn slowly while large weights make a neural network learn faster. Experimental results indicate that, if enough data points are available, initializing neural network with small weights helps to get better generalization and hence to achieve better performance.

7.4.2 Learning rate and momentum

Learning rate impacts the learning speed of connections during the training. Learning rate for the first hidden layer is set higher than other layers in the model. As after some point, first hidden layer learns comparatively slower than the second hidden layer[20]. So, it is set to 0.9 for first hidden layer whereas for all other layers learning rate and momentum are set to 0.01. Choosing small learning rate makes small changes in connection weights which make weights learning curve smoother but by this way, it takes a lot of time to

train the model. Whereas, choosing large learning rate value makes model unstable. To speed up the training and at the same time obtaining the stable network, the term momentum is used. 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[26].

7.4.3 Activation function

An activation function is also referred as a transfer function. The name transfer function is more commonly used in signal processing while in neural networks name activation function is more common. Basic use of activation function in neural networks is to project values to a specific range, and more importantly, to introduce nonlinearity to the neural network. If a non-linear activation function is not used in the neural network it would behave as a single layer perceptron, irrespective of how many hidden layers the neural network has, which will result in failure to detect nonlinear features of the function. Many different types of activation functions are available out of which hyperbolic tangent sigmoid activation function is used for this work. The linear activation function is used for input layer and the output layer whereas the hyperbolic sigmoid tangent function is used for both the hidden layers. The function is as shown in figure 7.3.

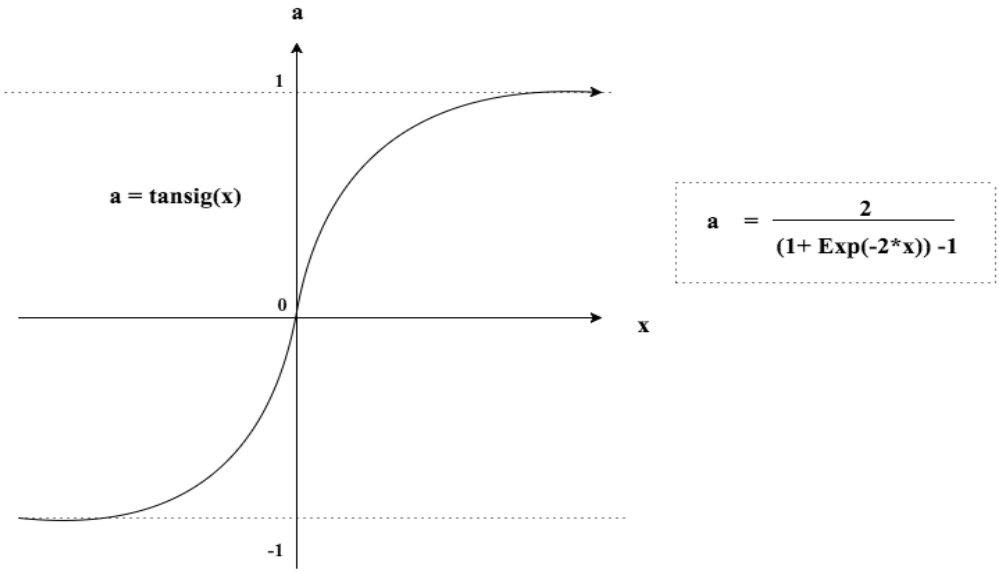


Figure 7.3 Activation function

7.5 Experimental results and analysis

Once we have structured deep neural network and training and testing datasets ready, the classifier uses workflow described in section 6.1 to get the predictions. To demonstrate usefulness of approach for different zip codes, we used data from three different zip codes including 02125, 02127 and 02128. As all of the zip codes used are from the same Boston region, knowledge structure is same for all three zip codes. Though connection weights may differ significantly depending on a zip code. We are designing the knowledge structure with the help of deep neural network, so, as far as deep neural network used is the same, we should have the same knowledge structure. Even though

structured knowledge for all three towns is the same, we train separate structured deep neural network for each town.

7.5.1 Accuracy

This section describes the usefulness of different approaches for house selling price predictions. Prediction accuracy is compared to different approaches including naive price-per-square-foot approach, conventional multivariate linear approach, fully connected deep neural network approach and specialized structured deep neural network approach. Same training dataset and testing datasets are used for all the above methods. Moreover, for all real-time learning methods same window size of 6 months is used. Also, the accuracy of all above methods is compared with house selling price predictions providing leading real estate companies.

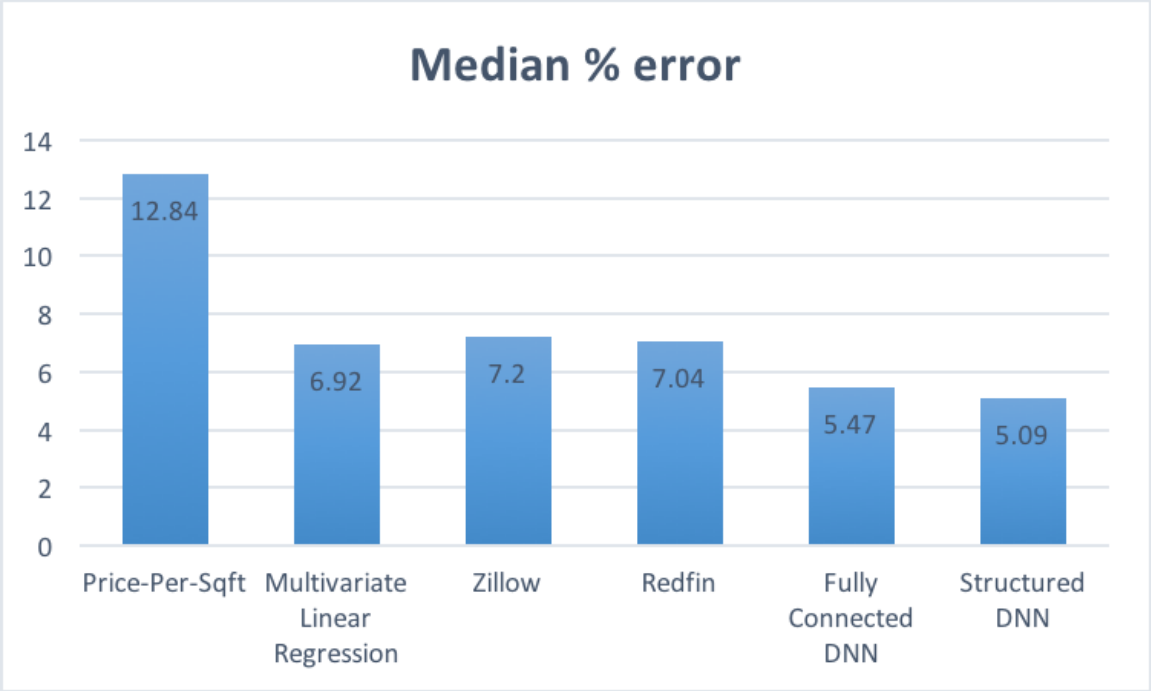


Figure 7.4 Median % error

Figure 7.4 shows the median percent error for house price predictions using different methods. Results show that, structured deep neural network achieves lowest median percent error among all methods.

Below figure 7.5 shows the mean percent error using various methods. Zillow and Redfin are not included in the figure below because the companies do not provide mean percent error for their predictions. The figure shows that structured neural network has the lowest mean % error.

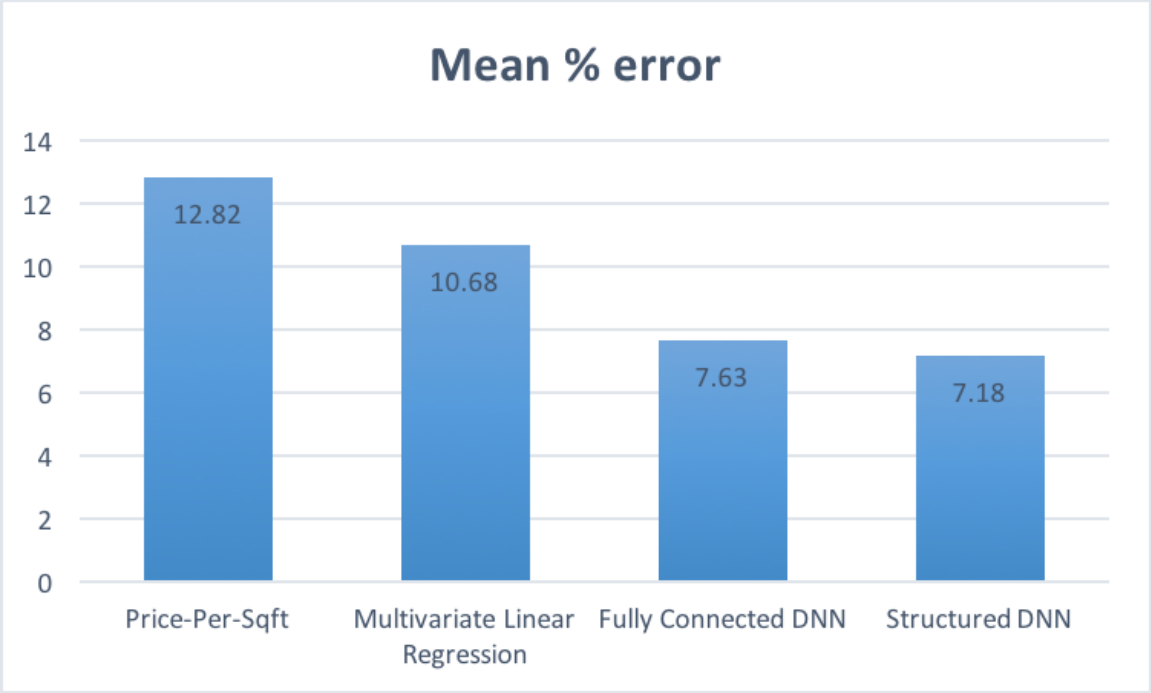


Figure 7.5 Mean % error

Figure 7.6 shows percent error range for all different methods. Three different ranges are defined which includes within 5%, within 10%, and within 20%. For example, if a model predicts selling price of a house as 108K and if the house is sold for 100K then percent error for the prediction using that model is 8%, which means the prediction percent error is within 10% error range. Results shown in figure 7.4, 7.5, and figure 7.6 show that structured deep neural network performs better than other methods.

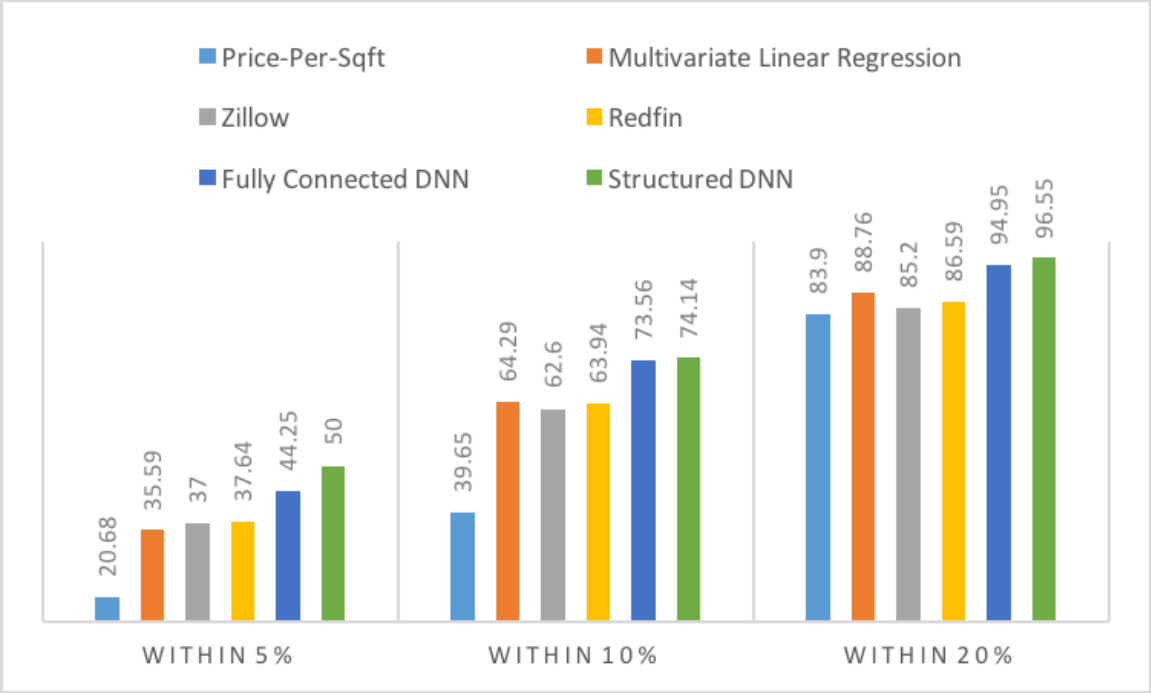


Figure 7.6 Percent error range

We used data from different zip codes to verify the usefulness of the approach. Above results are from zip code 02127 (South Boston) using house data points from previous year only (2015-2016). Figures below show the results for other zip codes. For following results, we used house sales data within previous 5 years for zip codes 02125, 02127 and 02128.

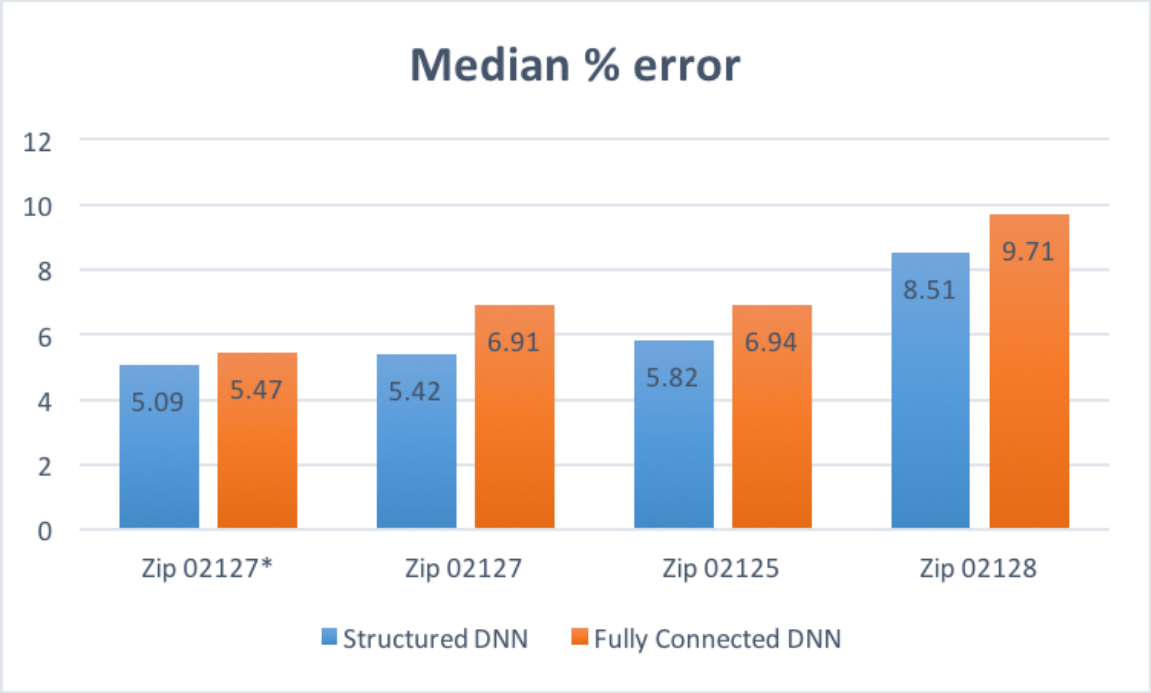


Figure 7.7 Median % error for different zip codes

02127* = Data points used for training and testing are within previous 1 year

02125, 02127 and 02128 = Data points used for training and testing are within previous 5 years.

Above figure 7.7 shows median % error for zip codes 02127, 02127 and 02128.

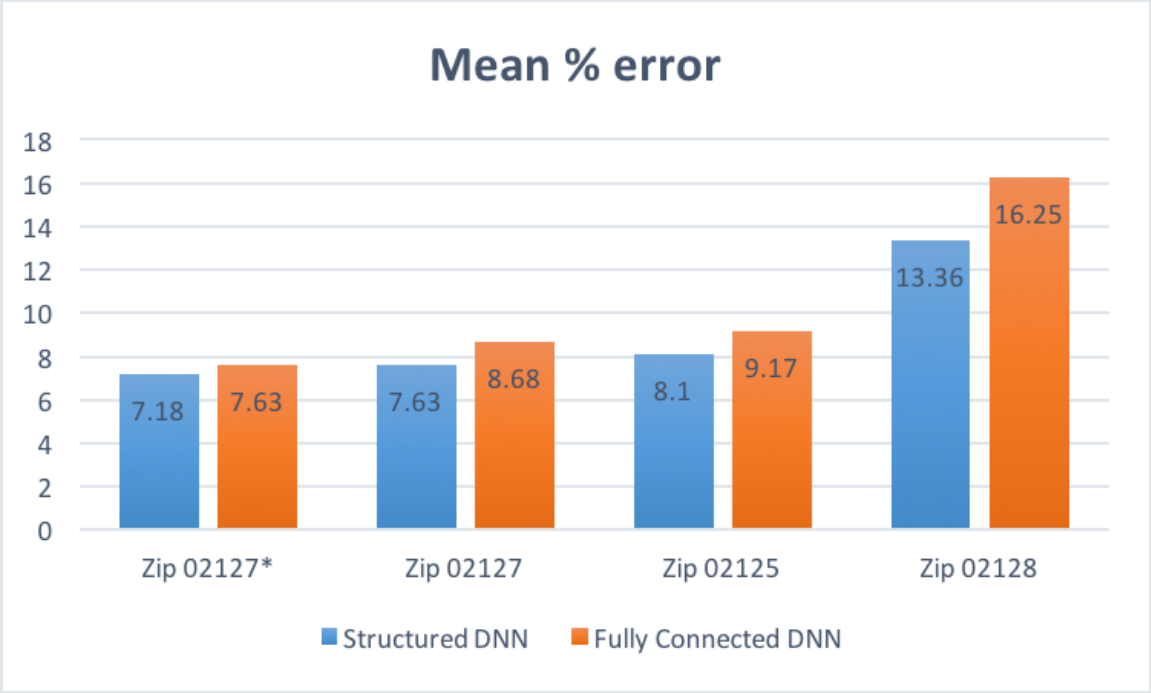


Figure 7.8 Mean % error for different zip codes

02127* = Data points used for training and testing are within 1 year

02125, 02127 and 02128 = Data points used for training and testing are within previous 5 years.

Above figure 7.8 shows mean % error for zip codes 02127, 02127 and 02128. From figure 7.7. And 7.8 we can observe that for zip code 02127, structured neural network performs better when data within previous 1 year is used as compared to when data within previous 5 years is used. This could be because of instability in the market in earlier years. Instable market is hard to predict, hence, it reduces prediction accuracy. Also, we can see that results for zip code 02128 are not so accurate as compared to other zip codes. This is because of unavailability of enough data points for training the model.

Due to insufficient data points used for training, the model is undertrained, hence, results are not so accurate for that particular zip code.

Experimental results presented above show that structured neural network outperforms all other methods as well as leading real estate companies zillow.com and redfin.com.

7.5.2 Fitting

Now, let's see how using structured DNN helps to reduce chances of overfitting. Table below shows mean testing error with respect to maximum mean training error for both structured DNN and fully-connected DNN. Maximum mean training error is a number used as a threshold for the training; which means, training of the neural network will not be stopped until mean batch error is below the defined threshold. For example, if maximum mean training error is set to 9.0 then neural network would train each batch till the mean error is less than 9.0.

Table 7.1 Mean testing error

Maximum mean training error(training threshold)	**Structured DNN mean testing error**	**Fully-connected DNN mean testing error**
7.00	7.38	7.78
7.10	7.36	7.66
7.25	**7.32**	**7.56**
7.40	7.41	7.63
7.50	7.43	7.69

We choose window size of a year so as have more training samples per batch. From the table above we can observe that, mean testing error achieved by structured DNN is less as compared to fully-connected DNN. Which implies that structurally connected DNN significantly reduce chances of overfitting.

7.5.3 Time efficiency

To measure time efficiency, we set a percent mean accuracy number which can be achieved by both fully-connected DNN and structured DNN. Best achievable mean % testing error using fully-connected DNN is 7.56 and it takes 8 seconds for fully-connected DNN to achieve this testing error. Whereas structured DNN achieves it in less than a second. This shows that structured neural network is time efficient. Time efficient model is not so useful for this example because we are using small data sets for the

experiments which takes just a few seconds to compute. But, when large datasets are used, time efficient model would be certainly useful.

7.5.4 Space efficiency

The structurally connected nature of the model reduces number of connections significantly. In this work, the structured neural network designed for the case study has 70% less connections as compared to fully-connected neural network. In neural networks, a lot of memory is required to save connection weights between the neurons. As the proposed structured DNN reduces the number of connections significantly, it saves space, hence, making the model space efficient.

7.5.5 Number of training data points needed for training

For training neural networks, it's good to have as many data points as possible, more data helps to achieve better generalization. However, minimum number of training data points required for training a neural network depends on the complexity of a function to be learned and size of a neural network. Large sized neural networks generally take more data points for training as compared to smaller or structured neural networks. For training, one should have at least as many number of data points as the number of free parameters (connection weights and biases) in a neural network. Consider a simple

example of calculating bias and connection weights of a simple perceptron as shown in below figure.

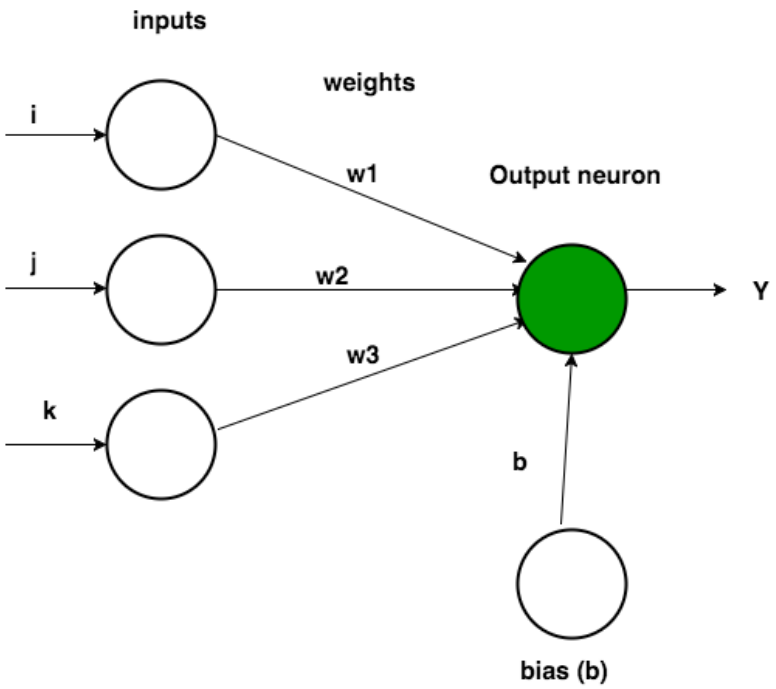


Figure 7.9 Perceptron with weights and bias

Using perceptron formulas from equation 3.1.2(e) and 3.1.2(f) we get the output as,

$$Y = \sum_{i=1}^{n} x_i w_i + b$$

$$= i*w_1 + j*w_2 + k*w_3 + b \quad (7.5.5)$$

Here, given values of inputs i, j, k and output Y, our goal is to find values of connection weights w_1, w_2, w_3 and bias b. As we can see, equation 7.5.5 has four unknown variables so we will need at least four equations to calculate values of these variable. To get four

different equations, we will need four data points (four sets of values of i, j, k and Y). Which implies that we may need at least as many number of data points as the sum of number of weights and biases in a neural network.

As the proposed structured neural network significantly reduces number of connections, it may take less data points for training as compared to fully connected neural network.

8 Conclusions and Future Work

In the era of big data and machine learning, predictive analytics is the next interesting thing. A number of big data analytics tools and algorithms are already in the market and deep learning algorithms are the most popular these days. Because of the large structure of deep architectures, large amount of data is required to train the models and hence deep models takes longer for training, making the models time inefficient. Large deep architectures give deep learning models enough power to overfit. Hence, making models time efficient and getting rid of the overfitting problem are two major problems while training deep architectures. Also, if the data is big, storage and management are major issues in predictive data analytics.

In this thesis, we propose a structured deep learning approach for predictive analytics. We presented the algorithm to get knowledge structure of the problem and designed a structured deep neural network out of it. The structured deep neural network designed is structurally connected which makes it time efficient, space efficient and the model can be trained using fewer data points as compared to the fully connected deep neural network. Also, as the model contains fewer connections, the use of the model significantly reduces chances of overfitting. Even though the structured deep neural network presented here is tested for house selling price prediction problem, the idea would be helpful to efficiently implement on many other predictive analytics problems.

Thus, in this thesis, we presented a systematic approach for getting structured knowledge and to design structured deep neural network out of it. We use house price prediction as an example problem to demonstrate the usefulness of the proposed approach. Our experimental results show that house price prediction accuracy is significantly improved.

One of the most interesting things to do in the future would be automating the process of designing the structured knowledge and the structurally connected deep neural network. In this approach although we have successfully proposed the algorithms for these processes, in practice few of the tasks in the process are manual. For example, if we want to add new input feature to the house price prediction problem, then maybe we will have to design a new knowledge structure and a new structured neural network, which would be a time-consuming task. Instead, we may update the process so that if we just give training data to the algorithm, it would automatically come up with the structure and the structurally connected neural network. Also, we may add the functionality of automatically changing the neural network structure to allow the model to better adapt to new market changes and it would also help to make the model scalable.

Also, trying the presented idea of structured deep neural network on other predictive analytics problems from different fields such as stock market, healthcare, business, security, marketing, operations, risk, customer support, and finance etcetera would be an interesting thing to do.

References

[1] "Predictive analytics: what it is and why it matters." [Online]. Available: http://www.sas.com/en_us/insights/analytics/predictive-analytics.html. [Accessed: 05-Apr-2016].

[2] C. Nyce and A. Cpcu, "Predictive analytics white paper," American Institute for CPCU. Insurance Institute of America, pp. 9–10, 2007.

[3] W. W. Eckerson, "Predictive analytics," Extending the value of your data warehousing investment. TDWI best practices report, vol. 1, pp. 1–36, 2007.

[4] Wikipedia contributors, "Predictive analytics," Wikipedia, The free encyclopedia, 04-Apr-2016. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Predictive_analytics&oldid=713573902. [Accessed: 05-Apr-2016].

[5] IBM, P. Zikopoulos, and C. Eaton, Understanding big data: Analytics for enterprise class hadoop and streaming data, 1st ed. McGraw-Hill Osborne Media, 2011.

[6] "Deep Learning ..moving beyond shallow machine learning since 2006!" [Online]. Available: http://deeplearning.net/. [Accessed: 10-Jan-2016].

[7] J. Frew and G. Jud, "Estimating the value of apartment buildings," Journal of Real Estate Research, vol. 25, pp. 77-86, 2003.

[8] Hu Xiaolong, X. Hu, and M. Zhong, “Applied research on real estate price prediction by the neural network,” in 2010 The 2nd Conference on Environmental Science and Information Application Technology, 2010.

[9] B. Mitchell and J. Sheppard, “Deep structure learning: Beyond connectionist approaches,” in Machine Learning and Applications (ICMLA), 2012 11th International Conference on, 2012, vol. 1, pp. 162–167.

[10] Y.-H. Liao, H.-Y. Lee, and L.-S. Lee, “Towards structured deep neural network for automatic speech recognition,” in 2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU), Scottsdale, AZ, USA, pp. 137–144.

[11] S. van den Dries and M. A. Wiering, “Neural-fitted TD-leaf learning for playing Othello with structured neural networks,” IEEE Trans Neural Netw Learn Syst, vol. 23, no. 11, pp. 1701–1713, Nov. 2012.

[12] M. V. D. Steeg, M. M. Drugan, and M. Wiering, “Temporal Difference Learning for the Game Tic-Tac-Toe 3D: Applying Structure to Neural Networks,” in Computational Intelligence, 2015 IEEE Symposium Series on, 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 Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on, 2014, pp. 6849–6853.

[14] N. Nghiep and C. Al, “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.

[15] 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 Artificial Intelligence (MICAI), 2011 10th Mexican International Conference on, 2011, pp. 175–181.

[16] Z. X. Li, “Using Fuzzy Neural Network in Real Estate Prices Prediction,” in 2007 Chinese Control Conference, Zhangjiajie, China, 2006, pp. 399–402.

[17] 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, 2007, pp. 173–182.

[18] R. E. Lowrance, “Predicting the Market Value of Single-Family Residential Real Estate,” Thesis, New York University, 2015.

[19] B. J. Ford, H. Xu, and I. Valova, “A Real-Time Self-Adaptive Classifier for Identifying Suspicious Bidders in Online Auctions,” Comput. J., Mar. 2012.

[20] M. A. Nielsen, “Neural Networks and Deep Learning,”, Determination Press, 2015.

[21] R. Maitra, “Multivariate Regression.” [Online]. Available: http://www.public.iastate.edu/~maitra/stat501/lectures/MultivariateRegression.pdf. [Accessed: 28-Mar-2016].

[22] I. Goodfellow, Y. Bengio and A. Courville, “Deep Learning.”, Book in Preparation for MIT Press, 2016.

[23] M. Stinchcombe and H. White, “Universal approximation using feedforward networks with non-sigmoid hidden layer activation functions,” in Neural Networks, 1989. IJCNN., International Joint Conference on, 1989, pp. 613–617 vol.1.

[24] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” Nature, vol. 521, no. 7553, pp. 436–444, May 2015.

[25] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: a review and new perspectives,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 35, no. 8, pp. 1798–1828, Aug. 2013.

[26] G. Batres-Estrada, “Deep learning for multivariate financial time series,”, Thesis, Soderberg & Partners, 2015.