Homeless Population

Ryan Du
Math 199, Spring 2018
Adviser: Michael Lindstrom

Contents

1 Introduction

2 Data Management

For neural networks, some of the data are

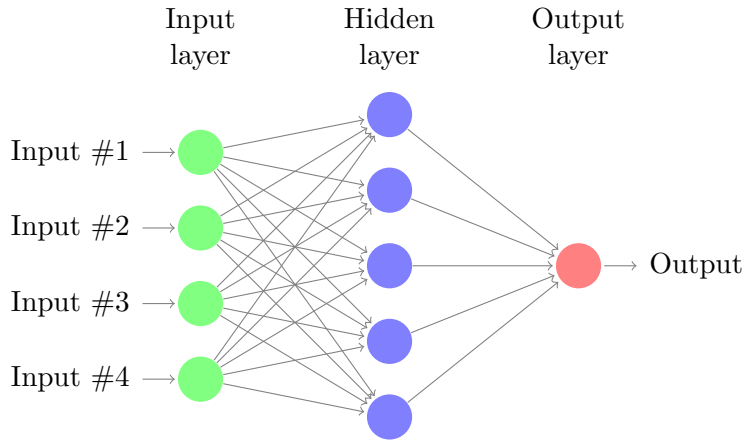
For neural networks, the data used was from several different sources. The numbers of parking citation, coffee shops, restaurants, homeless shelter, crimes, bus stops are obtain by previous work done on the subject [cite]. We introduced new time-dynamic, yearly data into our data set. We introduced the yearly ZRI and ZHVI

3 Techniques and Models

3.1 Topic Modeling

3.2 Neural Networks

We used neural networks to analyze the complex relationship between the time-dynamic data and the change in homeless population. Neural networks is a method that gets it's inspiration from the neurons of the human brain. The basic structure of a neural network is represented by the figure below.



Imagine we have a size m data-set with n features with entries (x_i, y_i), where x_i is the feature value of a record and y_i is the correct output.

Let's first stipulate some notations: We will use W^l_{jk} denote the entry at (j, k) in the weight matrix that connects the $(l-1)^{\text{th}}$ layer to the l^{th} layer. Similarly, we use b^l_j for the bias at the j^{th} position in the l^{th} layer; use h^l for the activation function of the l^{th} layer; use a^l_j for the activation for the activation at the j^{th} position in the l^{th} layer. `More detailed graph and the notation should be reflected on graph`

With these notation, we have the equation of the relation of activation at the j^{th} position in the l^{th} layer to the $(l-1)^{\text{th}}$ layer:

$$z^l_j = W^l_{jk} a^{l-1}_k + b^l_j \qquad\qquad a^l = h^l(z^l) \tag{1}$$

We can rewrite the above equation with vectored form:

$$z^l = W^l a^{l-1} + b^l \qquad\qquad a^l = h^l(z^l)$$

The activation functions used in out models are: relu, sigmoid, and softmax. The relu function output the original input if the input in positive, 0 if the input is negative:

$$\begin{cases} h(z) = z & \text{if } z \leq 0 \\ h(z) = 0 & \text{if } z < 0. \end{cases} \tag{2}$$

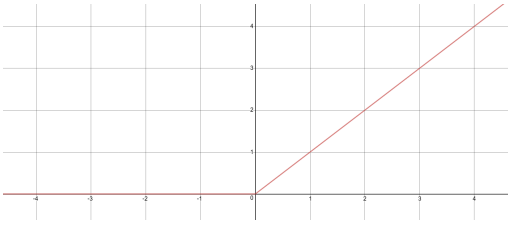


Figure 1: This figure shows the shape of the relu function.

The sigmoid function always output a number between -1 and 1. If has a steeper rate of increase when the input is near 0:

$$h(x) = \frac{1}{1 + e^{-x}} \tag{3}$$

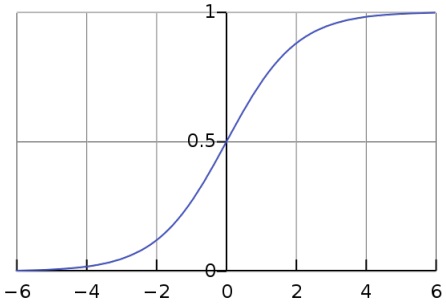


Figure 2: This figure shows the shape of the sigmoid function.

The softmax function is a generalization of the logistic function that "squashes" a vector of arbitrary real values to a vector of real values, where each entry is in the range (0, 1), and all the entries adds up to 1:

$$h(z_j) = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}$$

The algorithm learns how to best fit the data-set by trying to minimize a cost function. In our model, we used the cross-entropy cost function: `math needs more accurate`

$$C = -\frac{1}{N} \sum_{i}^{N} \sum_{j} y_{ij} \log(a_{ij}^{L}) \tag{4}$$

The method we used to minimize the cost is gradient descent. All weights and biases are updated against the direction of the cost function gradient for the values at that iteration. The learning rate α is the size of step each iteration takes. For example, from the i^{th} iteration to the $(i+1)^{th}$ iteration, the weight and bias should be updated by:

$$W_{i+1}^{l} = W_{i}^{l} - \alpha \frac{\partial C}{\partial W_{i}^{l}} \qquad\qquad b_{i+1}^{l} = b_{i}^{l} - \alpha \frac{\partial C}{\partial b_{i}^{l}} \tag{5}$$

In order to actually compute the gradient of the cost function against each elements of the weight and bias in each layer, we apply the chain rule repeatedly. This way of finding the gradient is called backpropagation[**NNDL**]. The chain rule can be written out as this:

$$\frac{\partial C}{\partial b_{k}^{l}} = \frac{\partial C}{\partial a^{L}} \frac{\partial a^{L}}{\partial a^{L-1}} \frac{\partial a^{L-1}}{\partial a^{L-2}} \cdots \frac{\partial a^{l+1}}{\partial a^{l}} \cdot \frac{\partial a^{l}}{\partial b_{k}^{l}} \tag{6}$$

$$\frac{\partial C}{\partial W_{pq}^{l}} = \frac{\partial C}{\partial a^{L}} \frac{\partial a^{L}}{\partial a^{L-1}} \frac{\partial a^{L-1}}{\partial a^{L-2}} \cdots \frac{\partial a^{l+1}}{\partial a^{l}} \cdot \frac{\partial a^{l}}{\partial W_{pq}^{l}} \tag{7}$$

We thus need to derive each term in the previous formula.

To make writing these formulas easier, we will be using the Einstein Summation notation where repeated indices are implicitly summed over. For example:

$$a_{ik} a_{ij} \equiv \sum_{i} a_{ik} a_{ij}$$

We will also be using the the Kronecker Delta, i.e:

$$\delta_{ij} = \begin{cases} 1 & \text{when } i = j \\ 0 & \text{when } i \neq j. \end{cases}$$

From the equation of the cost function 4, we can see that: `Work out math`

$$\frac{\partial C}{\partial a_{ij}^{L}} = \frac{-1}{N} \frac{y_{ij}}{a_{ij}} \tag{8}$$

The activation function for the l^{th} layer h^l is a function from z^l to a^l, therefore:

$$\frac{\partial a_{i}^{l}}{\partial z_{j}^{l}} = \frac{\partial h_{i}^{l}(z_{j}^{l})}{z_{j}^{l}} = \mathsf{J}_{\mathsf{ij}}^{\mathsf{l}}$$

$$\frac{\partial a^{l}}{\partial z^{l}} = \mathsf{J}^{\mathsf{l}} \qquad \text{(the Jacobian matrix)} \tag{9}$$

From equation (1), it is obvious that:

$$\frac{\partial z_j^L}{\partial b_k^L} = \delta_{jk} \tag{10}$$

Also from equation (1), we can see that:

$$\begin{aligned} \frac{\partial z_j^l}{\partial W_{pq}^l} &= \frac{\partial}{\partial W_{pq}^l}(W_{jk}^l a_k^{l-1} + b_j^l) \\ &= \delta_{jp}\delta_{kq} a_k^{l-1} = \delta_{jp} a_q^{l-1} \end{aligned} \tag{11}$$

We can thus derive the function for the gradient: $\frac{\partial a_i^l}{\partial b_k^l}$ and $\frac{\partial a_i^l}{\partial W_{pq}^l}$
From (6) and (7):

$$\frac{\partial a_i^l}{\partial b_k^l} = \left(\frac{\partial a_i^l}{\partial z_j^l}\right)\delta_{kj} = \frac{\partial a_i^l}{\partial z_k^l} \tag{12}$$

From (6) and (8):

$$\frac{\partial a_i^l}{\partial W_{pq}^l} = \left(\frac{\partial a_i^l}{\partial z_j^l}\right)\delta_{jp} a_q^{l-1} = \left(\frac{\partial a_i^l}{\partial z_p^l}\right) a_q^{l-1} \tag{13}$$

To derive the expression for $\dfrac{\partial a^l}{\partial a^{l-1}}$, we have:

$$\frac{\partial a_i^l}{\partial a_k^{l-1}} = \frac{\partial a_i^l}{\partial z_j^l}\frac{\partial z_j^l}{\partial a_k^{l-1}} \tag{14}$$

According to (1):

$$\frac{\partial z_j^l}{\partial a_k^{l-1}} = W_{jk}^l \tag{15}$$

Therefore we can use (6) and (12) to calculate:

$$\frac{\partial a_i^l}{\partial a_k^{l-1}} = \mathsf{J}_{\mathsf{ij}}^{\mathsf{l}} W_{jk}^l \tag{16}$$

Now we have all the elements to calculate (3) and (4)

For neural network, it is also good practice to normalize the input data so that the data is on the same scale. There are two method that we used: Z-score and Principle Component Analysis (PCA). The data matrix X is of dimension $\mathbb{R}^{m\times n}$, each row is a data point (m of them) and each point has n features.
Z-score normalization makes each feature (the column of our matrix) has 0 mean and 1 standard deviation. We first calculate the mean of of each column:

$$M_j = \frac{\sum_i X_{ij}}{m} \tag{17}$$

Then we can calculate the standard deviation of the column:

$$S_j = \sqrt{\frac{\sum_i (X_{ij} - M_j)^2}{m-1}} \tag{18}$$

The Z-score of the data in a column would be:

$$Z_{ij} = \frac{X_{ij} - M_j}{S_j} \tag{19}$$

We do this calculation for each number in our matrix X, we can get the Z-score normalized data matrix Z, where the data in each column has a mean of 0 and the standard deviation of 1.

Principal component analysis (PCA) is a procedure that uses an orthogonal transformation to convert matrix variables into a set of linearly independent vectors called principal components. This transformation is defined in such a way that the first principal component has the largest possible variance (i.e.: accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors are an orthogonal basis. PCA is sensitive to the relative scaling of the original variables, therefore we use the normalized Z matrix derived above. `[cite]`

We use T to denote the full principal components decomposition matrix of Z. It is common nowadays to use single value decomposition (SVD) to derive the matrix T.

$$Z = U\Sigma W^T \tag{20}$$

$$T = ZW \tag{21}$$

And each column of T is the principle component of Z, and column with smaller index explain more variance in the data. `More detailed math? Like about why they have the largest variance`

There are 818 census tracks that all the data we desire to use are reliable. The features we used are in two groups:

- Static: Citations, Coffee, Restaurants,Shelters, Crime Count, Bus Stops, Tract size in Square Miles;
- Time Dynamic: Available Housing Units, Total Housing Units, Total Vacant Units, Unemployment Rate, Below Poverty Rate, Medium Rent As Percent Of Gross Income, Total Population, Medium Household Income, Medium Rent, Medium Value, Medium Monthly Housing Costs, ZRI, ZHVI.

For data in the static group, we took the value of 2016 and assumed that there is not much yearly changes. For data in the time dynamic group, we have data from 2014, 2015, and 2016.

We tried a ternary classification for the change of numbers of homeless populations in a census track. We implemented the neural networks in Tensorflow, a popular machine learning system `[cite]`.

First, we divide the change in homeless population into three buckets. Roughly, the lower bucket represent a significant decrease in homeless population, the middle bucket represent a small, insignificant variation in the homeless population, and the higher bucket represent a significant increase in homeless population.

We used static data and time dynamic data (single year and change between years), to try to classify the change in homeless population. More specifically, for the time dynamic data, we used

the change from 2014 to 2015 and the data of 2015 to classify the change in homeless population from 2015 to 2016; and the change from 2015 to 2016 and the data of 2016 to classify the change in homeless population from 2016 to 2017. We used the change of features of previous years to predict the change in homeless population because (1) ACS did not release the data for 2017 yet, and (2) the homeless population data is gathered during January, changes in previous years are reflected in changed in the year later `[cite]`. We compiled the data of changes in two years together and split them up with a ratio and 70% and 30%, 70% of the data is used as the training set and 30% of the data is used as the testing set.

We tried different method of reprocessing the data:

- Normalization
- Principal Component Analysis (PCA)

We tried different cut off point for the buckets.
We tested different neural net work structure. We did trials on:

- Number of layers
- Number of neurons in each layer
- Activation function
  - Relu
  - Sigmoid
  - Softmax
- Learning rate
- Optimizer
  - stochastic gradient descent
  - Adam

4 Results

4.1 Neural Networks

Because of the uneven distribution of the bins, we use different measurement to reflex how well the algorithm do. Apart from the accuracy, we have confusion matrix and precision. Accuracy can be calculated by:

$$A = \frac{\text{\# of correct predictions}}{\text{\# of all predictions}} \tag{22}$$

A confusion matrix C is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known. The (i, j) entry of the matrix represents the number of point whose true value is the i^{th} class and it was predicted to the j^{th} class. Precision$i$ is the ratio of true predictions among all the points that was predicted to be in the i^{th} class:

$$\text{Pred i} = \frac{C[i,i]}{\sum_j C[i,j]} \tag{23}$$

The table below shows the best result of the ternary classification for bucket division: [-400,-20), [-20,20), [20,600).

Model	Preprocessing	Accuracy	Training Cost	Testing Cost	Hidden	Output
A	PCA	0.725	0.359	0.501	-	Sigmoid

With a confusion matrix and precision:

-	True 0	True 1	True 2
Pred 0	4	19	2
Pred 1	46	346	61
Pred 2	2	4	3

Precision0	Precision1	Precision2
0.160	0.764	0.333

We need to take more care when we look at the precision numbers. Because of the imbalance of the numbers of data points in each bin, if we have a purely chance predictor, the precision will be the ratio of the number of data point in that bin to the number of data point in total. Thus a pure chance model will have a precision:

Precision0	Precision1	Precision2
0.070	0.758	0.136

Thus even thought the precision number looks bad by their face value, all of them in fact is bigger than the change model. Therefore our model is performing better than chance at predicting change in homeless population.

We change the cutoff point so that the three bins to have roughly equal amount. The table below shows the best result of the ternary classification for bucket division: [-400,-3), [-3,6), [6,600).

Model	Preprocessing	Accuracy	Training Cost	Testing Cost	Hidden	Output
B	Normalization	0.389	0.397	0.914	relu	Sigmoid

With a confusion matrix and precision:

-	True 0	True 1	True 2
Pred 0	60	49	60
Pred 1	58	79	48
Pred 2	48	35	51

Precision0	Precision1	Training Precision2
0.355	0.427	0.380

A pure chance model for this one would result in a precision of:

Precision0	Precision1	Training Precision2
0.333	0.333	0.333

The precision of our model is higher than each of it's corresponding entry in the chance model. Therefore our model performs better than chance.

5 Summary and Future Work

Future research should explore more method of prepossessing the data. Considering the error in public database, we could put the feature data into bins as well (e.g.: income can be slit into 5 bins according to it's standard deviation). Convolutional neural network is also promising in the context of analyzing the change in homeless population because homeless population move around frequently. Change in one area will change the number of homeless population in other areas.

6 Acknowledgments

We thank our adviser and mentor, Professor Michael Lindstrom, for all of his advice and help on this project.