**CSCI 5922: Neural Networks and Deep Learning – Assignment 2**

**INTRODUCTION**

****Public bike sharing is becoming increasingly prevalent in many cities worldwide. Since the first launch in Europe in 1960s , Bike Sharing Systems (BSSs) hit the streets in over 870 cities globally and the number keeps increasing. These BSSs largely fill up the gaps in public transport modes by providing last mile connectivity effectively. Because of their convenience, low price, health, and environment benefits, the shared bicycles are preferred by many travelers for short trips in urban areas.

To further satisfy rider needs with a high-quality service, BSS operators often need to be proactively prepared for daily operation schemes in advance. This greatly motivates them to envision the forthcoming demand for the entire system as well as specific stations. Thus, a reliable forecast of their daily bike usage opens promising avenues for effective and economic system planning, operation (e.g., bike rebalancing), facility maintenance, etc. Many existing BSSs are equipped with automatic rental systems to facilitate accessing and returning bicycles. Through the rental systems, real-time records (e.g., start time and end time) of each trip become available. In addition, real-time data on available docks and bicycles at each station is also available. These data not only offer riders a convenient way to query bicycle information, but also facilitate operators to know their system performance responsively. Given the easy access to the system data, researchers have been leveraging the data to help operators understand their BSS usage predictively.

Over the years, tremendous efforts have been made to the development of models for bike usage prediction. In general, these models attempted to use a set of determinants to explain BSS usage (e.g., pickups and / or drop-offs). For example, BSSs in Montreal incorporated meteorological data, temporal characteristics, and built environment attributes to aid the prediction efforts. Another one predicted the bike demand in rush hours based on a linear regression model that includes taxi usage, weather, and  spatial factors. The success of such models largely lies in the simplified assumptions on the statistical causal relationships between the variables. Despite the simplicity, these tractable mathematical models hardly capture the complexity of the changes in daily usage of BSSs.

Thus, this calls for more reliable approaches that can work in the context of unclear causal relationships and limited explanatory variables. Recently, deep neural networks have shown some impressive results on a variety of challenging tasks such as speech recognition, image classification, and natural language processing. The capability of deep neural networks to automatically learn complex and nonlinear patterns from observed data makes it applicable to a wide range of problems, including those related to transportation systems. Given the accessibility to massive BSS records and other limited data (e.g., weather information), deep learning methods such as convolution neural network and deep belief network hold promise for performance improvements in analyzing the complex scenarios of BSS usage that may have nonlinear and heterogeneous patterns. Thus, with this project I try to examine the potential of using deep learning techniques to forecast the daily usage of BSSs. To be more specific, the predicted “usage” here is the number of pickups.

**ANALYSIS**

**Data gathering, Data Cleaning, Data Preparation**

**Data source background**

Bike sharing systems are a means of renting bicycles where the process of obtaining membership, rental, and bike return is automated via a network of kiosk locations throughout the city. Using these systems, people are able rent a bike from a one location and return it at a different place on as-needed basis.

The data generated by these systems makes them attractive because the duration of travel, departure location, arrival location, and time elapsed is explicitly recorded. Bike sharing systems therefore function as a sensor network, which can be used for studying mobility in a city.

The dataset used in this project was provided by Hadi Fanaee Tork using the data from Capital Bikeshare. This dataset is also hosted on the UCI machine learning repositor.

The historical usage patterns with weather data are made available to accomplish the task of forecasting bike rental demand in the Capital Bikeshare program in Washington, D.C.

**DATA CLEANING AND PREPARATION STEPS**

**Initial EDA of the dataset**

**Data Summary**

No. of rows: 17379

No. of columns: 17

Datatypes of Features available in the dataset

Text

Description automatically generated with medium confidence

**Outliers Analysis**

At first look, "cnt" variable contains a lot of outlier data points which skews the distribution towards right (as there are more data points beyond Outer Quartile Limit).But in addition to that, following inferences can also been made from the simple boxplots given below.

Spring season has got relatively lower count. The dip in median value in boxplot gives evidence for it.  
The boxplot with "Hour of The Day" is quite interesting. The median value is relatively higher at 7AM - 8AM and 5PM - 6PM. It can be attributed to regular school and office users at that time.

Most of the outlier points are mainly contributed from "Working Day" than "Non Working Day". It is quite visible from the plot below.

Chart, box and whisker chart

Description automatically generated

**Correlation Analysis**

One common way to understand how a dependent variable is influenced by features (numerical) is to find a correlation matrix between them. Let’s plot a correlation plot between “count” and [“temp”,“atemp”, “humidity”, “windspeed”] and see how these variables interact with each other

1. temp and humidity features have got positive and negative correlation with count respectively. Although the correlation between them is not very prominent still the count variable has got little dependency on "temp" and "humidity".
2. windspeed is not going to be really useful numerical feature and it's clearly visible from the correlation plot of value with "count"
3. "atemp" is a variable that is not taken into consideration since "atemp" and "temp" has got strong correlation with each other. During model building any one of the variables has to be dropped since they will exhibit multicollinearity in the data.
4. "Casual" and "Registered" are also not considered since they are leakage variables in nature and need to be dropped during model building.

Regression plot in seaborn is one useful way to depict the relationship between two features. Here we consider "count" vs "temp", "humidity", "windspeed".

Confusion matrix with all relevant variables

Chart

Description automatically generated

Correlation plots of counts with respect to temp, windspeed and humidity

Chart, scatter chart

Description automatically generated

**Visualizing Count Vs (Month,Season,Hour,Weekday,Usertype)**

**Chart, histogram

Description automatically generated**

**Preparing the data**

**Creating Dummy Variables for Categorical variables**

Categorical variables like season, weather, month are converted into binary dummy variables so that these can be included in the model, we'll need to make. This is simple to do with Pandas thanks to get\_dummies(). The Screenshots below show the change in features after creating dummy variables

Table

Description automatically generated with medium confidence Table

Description automatically generated

**Scaling features**

As the end goal is to build a deep neural network to make bike usage predictions, it’s extremely important to handle the issue of different scales used for different feature. By doing this, the network can learn the correct weights and biases efficiently. That is, to make training the network easier, we'll standardize each of the continuous variables by shifting and scale of the variables such that they have zero mean and a standard deviation of 1. Also, the scaling factors are saved so that we can go backwards when we use the network for predictions.

**Splitting the data into training, testing and validation sets**

We'll save the data for the last 21 days approximately to use it as a test set after we've trained the network. We'll use this set to make predictions and compare them with the actual number of riders.

We'll also further split the train data into two sets, one for training and one for validating as the network is being trained. Since this is time series data, we'll train on historical data, then try to predict on future data (the validation set).

**Building the Neural Network**

This network will haveone input layer, one hidden layer an output layer with sigmoid as the activation function.

Diagram

Description automatically generated with medium confidence

The hidden layer will use the sigmoid function for activations. The output layer has only one node and is used for the regression, the output of the node is the same as the input of the node. That is, the activation function is f(x)= x. A function that takes the input signal and generates an output signal, but considers the threshold, is called an activation function. Inputs go through each layer of the network, calculating the outputs for each neuron. All the outputs from one layer become inputs to the neurons on the next layer. This process is called *forward propagation*.

We use the weights to propagate signals forward from the input to the output layers in a neural network. We also use the weights to propagate error backwards from the output back into the network to update our weights and this is called backpropagation.

Building the Neural network from scratch involved 4 steps:

1. Implement the sigmoid function to be used as the activation function.
2. Implement the forward pass in the train method.
3. Implement the backpropagation algorithm in the train method, including calculating the output error.
4. Implement the weight update method in the train method.

**RESULTS**

## Training the network[¶](http://localhost:8892/notebooks/Downloads/Bike-Sharing-Prediction-with-NeuralNet-master/Your_first_neural_network.ipynb#Training-the-network)

Hyperparameters tuning for the network takes substantial time and effort. The strategy here is to find hyperparameters such that the error on the training set is low, but we are also not overfitting to the data. If we train the network too long or have too many hidden nodes, it can become overly specific to the training set and will fail to generalize to the validation set. That is, the loss on the validation set will start increasing as the training set loss drops.

We also use Stochastic Gradient Descent (SGD) to train the network and update the weights and biases in the right direction. The idea is that for each training pass, we grab a random sample of the data instead of using the whole data set. We use many more training passes than with normal gradient descent, but each pass is much faster. This ends up training the network more efficiently.

### Choosing the number of iterations

This is the number of batches of samples from the training data we'll use to train the network. The more iterations we use, the better the model will fit the data. However, this process can have sharply diminishing returns and can waste computational resources if we use too many iterations. We want to find a number here where the network has a low training loss, and the validation loss is at a minimum. The ideal number of iterations would be a level that stops shortly after the validation loss is no longer decreasing.

### Choosing the learning rate

This scales the size of weight updates. If this is too big, the weights tend to explode, and the network fails to fit the data. Normally a good choice to start at is 0.1. If the network has problems fitting the data, reducing the learning rate helps. But lower the learning rate is, the smaller the steps are in the weight updates and the longer it takes for the neural network to converge.

### Choosing the number of hidden nodes

In a model where all the weights are optimized, the more hidden nodes there are, the more accurate the predictions of the model will be. However, the more hidden nodes we have, the harder it will be to optimize the weights of the model, and the more likely it will be that suboptimal weights will lead to overfitting. With overfitting, the model will memorize the training data instead of learning the true pattern and won't generalize well to unseen data. Generally the best number of hidden nodes to use, ends up being between the number of input and output nodes.

Hyperparameter chosen for this setup is as shown:

Text

Description automatically generated

**Training results**

****

**Graphical user interface, application

Description automatically generated**

## Checking out the predictions

Using the test data to view how well our network is modeling the data.

Chart, line chart

Description automatically generated

Our network seems to be overestimating bike ridership after December 20th because it has not been fed with enough holiday season training data!

**CONCLUSION**

The prediction of BSS usage continues to attract growing interest. However, many existing models that use a large number of variables to explain the change of bike usage are not flexible enough to envision the real-world data due to the need for restrictive statistical modeling assumptions and exhaustive data acquisition among others. This project used a deep learning approach based on neural network model. The proposed approach applied historical BSS data and very limited external variables that can be measured objectively to predict daily bike usage of bike sharing systems. Using the Capital Bikeshare system data, promising predictions were made that modelled the real-life usage of bikes. To add on it, sensitivity analysis is necessary for the prediction of any BSS because it can help determine most practical and appropriate model parameters. Overall, the benefit of using NN method lies in that it uses transformations and activation functions to account for the complex nonlinear relationship between the target variable and the explanatory variables. Thus, it helps improve the predictive performance for modeling the unknown casual relationships between bike usage and various factors. Despite the promising performance, this network has only seen typical urban data and more diverse data can make this network more robust.

**You can find the cleaned dataset + code here:** [**https://github.com/Gauthami25/5922-Neural-networks**](https://github.com/Gauthami25/5922-Neural-networks)