Using a Neural Network to Classify Income by Considering Census Data

Ameer Abdallah
CS 4210.01 (S21) Machine Learning
California State Polytechnic University
Pomona, United States
ayabdallah@cpp.edu

Charles Kypros
CS 4210.01 (S21) Machine Learning
California State Polytechnic University
Pomona, United States
cckypros@cpp.edu

Thuan Le
CS 4210.01 (S21) Machine Learning
California State Polytechnic University
Pomona, United States
tdle@cpp.edu

Paul Vuong
CS 4210.01 (S21) Machine Learning
California State Polytechnic University
Pomona, United States
ddvuong@cpp.edu

***Abstract*—We will use a neural network to establish a prediction model to classify an individual into income bracket in response to input containing life attributes (Marital Status, Location, Educational, Degree Field, Race, Employment, etc...) in the U.S Census.**

***Index Terms*—Machine Learning, Neural Network, Classifying Individual Income, Census**

I. Introduction

There have been several research projects that feature the study of numerous data set for a population. Different techniques in machine learning such as Decision Tree, Naïve Bayes, Random Forest, and Support Vector machine were used to compare efficiency and accuracy between the prediction of such population. In this project, we will use Neural Network Model with hidden layers, neurons and learning rate to predict income bracket of an individual and household based on numerous attributes.

Using the data set from 2019 provided by IPUMS, a non-profit organization that produces democratized access to the world's social and economic data. By testing different hidden layers, neurons and attributes, we were able to graph the correlation between them. We can see that the first several integration with back-propagation provides about 20% increase in accuracy. Different level of hidden layers and neuron do not increase the result. Choosing the right data set improves the algorithm tremendously by 30%. We also found that in our model, by decreasing the learning rate. We were able to get slightly better result.

In the following section, we will first talk about the Data Set we used and how we obtained them, along with what features or attributes we choose to input to the model. Furthermore, the next section will explain the model along with any changes during pre-processing, post-processing phase. Maximize the results and get an accuracy above 80% is our goal. Finally, we will talk about our actual result and what we did to come up with those results.

II. Data Set

The data set for this project was obtained from IPUMS USA, who preserved and catalogued the data from the ACS (American Community Surveys). We specifically parsed the database for instances of data from the year 2019 for our analysis. For those instances of data, there are significant number of features for us to analyze in order to establish a prediction model. The data set that we choose to work with contains a numerous of different attributes. We will work with sex, age, martial status, race, education, school type, field of degree and finally our class label would be income distribution. The available length of this data set will be about 20 thousands instances.

III. Machine Learning Model

The machine learning model that we plan to incorporate will be a *Neural Network*. The neural network will start off with random weights and biases. A prediction will be made on a training data instance to come up with a predicted classification for that instance. The trained data instance already contains a true classification which will be compared against the previous prediction and gradient decent function will be used to adjust the weights and biases to better predict future instances.

The output nodes will correspond to respective income brackets: Under $48,500, $48,500 to $145,500 and over $145,000. A confidence value from $0 \rightarrow 1$ will determine the final prediction (IE the highest value will be the ”best” prediction). We will classify only the personal income contribution. The amount of total layers and hidden layers that the neural network will contain is determined by testing and trials.

With learning rate testing out at 0.01, 0.05, 0.1. Our end classifier should be an income bracket prediction based on multi-variable regression. In these model, we also introduced back-propagation to increase accuracy and adjust the random weight and biases according to the weight means of the neurons. This process is shown in figure 1. For each of these hidden layers, we will use different amount of neurons. In order to use stochastic gradient descent with back-propagation to train these neural networks, an activation function is needed, we choose ReLU activation function in our project. We do not have a deep understanding of ReLU, but given the obstacles that a linear function provides, we decided that a nonlinear function would be necessary and ReLU is the common activation out there. Furthermore, for the final output layers, we will use ”soft-max” function, it is the last activation function of a neural network to normalize the output of a network to a probability distribution over predicted output classes, based on Luce’s choice axiom. Again, our group did not do a lot of research on this, hence we are unable to explain the inner mathematical model of this function. We hope that by incorporate this model into our data set, the weights and biases will change and hence, classification will be more accurate. Produce a good indicator of personal income from life attributes

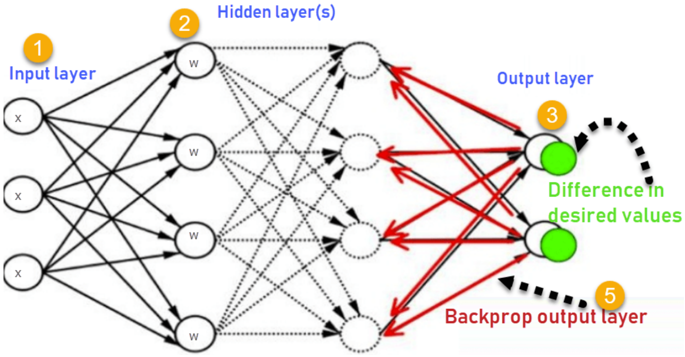


Fig. 1. Example of machine learning model we are implementing

IV. Results

Our final product has a list of 6, 8, 10 hidden layers, 20, 50, 70 as neurons and learning rates as 0.01, 0.05, and 0.1. We used ReLU for hidden layers neurons and soft-max for final activation functions. We divided class label into 3 classifiers, under to $48,500, in the middle from $48,500 to $145,500 and over $145,500. During the pre-processing phase, we removed a total of 13 columns that are not necessary for us and that leaves us with 13 features to test. Sex, Age, Martial Status, Time married, Race, Citizenship Status, Educational attainment, public or private school, field of degree, employment status, class of worker, industry and vision or hearing difficulty.In our training model process, we use a maximum epoch of 1000 which increases our model accuracy but produce over fitting. According to figure 2, more epoch leads to less loss and hence increase accuracy (figure 3).

We also introduced an early stopping function which stop epoch from over-fitting. Using a ratio of 67:33 ratio for training and testing respectively. However, we tested a variety of different amount of hidden layers and neurons. From around 20 neurons up to 900, 2 to 20 hidden layers and different learning rate. The result was different by approximately 2%-3%. Our thoughts is that a better data set would help since these data set has many missing values and data points. Our strongest result came at 78% accuracy when we used 10 hidden layers, 900 neurons and 0.01 as learning rates. We suspect that after all the training provided, the best indicators is in the data set, not about how many layers or neurons it takes.

V. Related work

In another research, a similar comparison with different data set was provided by University of Irvine (UCI). The researcher also used a numerous of different algorithms and methods to classify the label. For example, figure 4 provides a brief summary of his research techniques outcomes.

As we can see, according to this result, all model provides at equal or over 80% accuracy, 75% of precision. This is relatively a good database that the

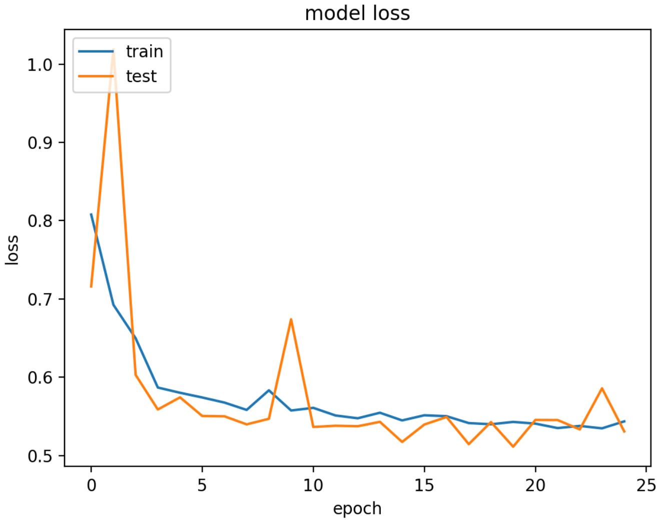


Fig. 2. A comparison of loss vs epoch

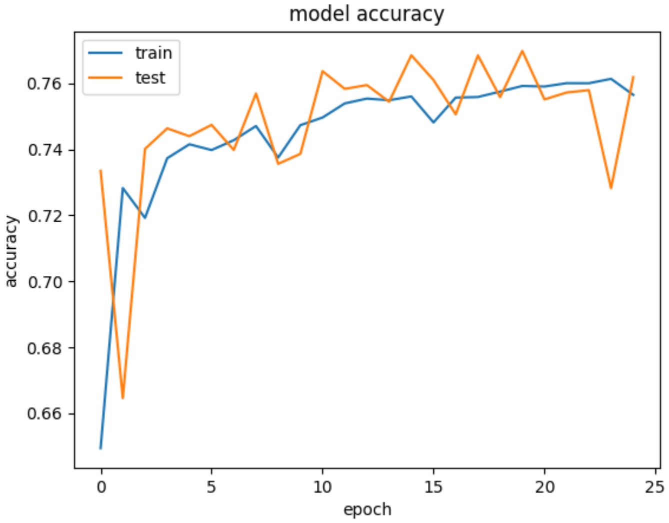


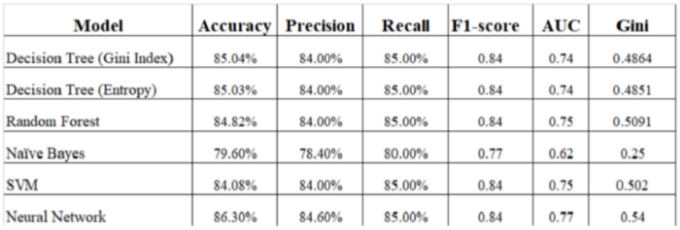
Fig. 3. A comparison of model accuracy with epoch

data collectors from UCI has put together. Comparing that to our actual results based on our current neural network model, our result indicates that we under perform in accuracy measurement.

One reason why this research demonstrate a slight better result was because the removing redundant values and handle missing values were taken care of, respectively. A strategy that we did not do is filling out the missing values with the most occurring instances in the column. Furthermore, we divided class label data into 3 different classifiers, increasing the number of classifiers in this case lower the accuracy. As with UCI data-set, only 2 classifiers were created which is over and under $50,000.

VI. Conclusion

We performed pre-processing on the data set by creating a class-bound list, This class-bound list sorts the data points into different list based on the true class label. Classifying people who makes under to $48,500 or in the middle from $48,500 to $145,500 and over $145,500 . We implemented the use of a dictionary to



Model	Accuracy	Precision	Recall	F1-score	AUC	Gini
Decision Tree (Gini Index)	85.04%	84.00%	85.00%	0.84	0.74	0.4864
Decision Tree (Entropy)	85.03%	84.00%	85.00%	0.84	0.74	0.4851
Random Forest	84.82%	84.00%	85.00%	0.84	0.75	0.5091
Naïve Bayes	79.60%	78.40%	80.00%	0.77	0.62	0.25
SVM	84.08%	84.00%	85.00%	0.84	0.75	0.502
Neural Network	86.30%	84.60%	85.00%	0.84	0.77	0.54

Fig. 4. A comparison of different model for a different data set

classify these attributes to our model for easier processing. For training the learning model, we integrated a 67:33 training and testing ratio, respectively.

Our performance varies across different quantities of classifications from our data set. For example, 10 classifications provide us with an accuracy of 50%. When reduced to 3 classifications, as stated in the above section, we noticed an increase of almost 30% to our accuracy. Essentially, we modified our model and the direction of our research to instead classify whether an individual earned lower, middle and upper class income as opposed to a more detailed bracket-based technique.

We observed that when we increased epoch, the number of propagation or iteration passes of the entire data-set, the result also increased in accuracy. We suspect that this has to do with our ReLU activation function regarding how it adjusts the weights and biases when more data is available. However, the problem with increasing epoch is that this leads to over-fitting, the result of increased epoch was not getting better but sometimes worse. To counter this, an early stopping function was introduced which minimizes the validation loss and patience equals to 5, which is the number of epochs to wait if no improvement on the progress of the validation set.

Additionally, we also noticed that by increasing our batch size, the amount of data to sends to our graphic card, the larger the batch size, the more accuracy we generally received. Overall, this project has provided a learning experience for all of us, includes some of the more complex mathematical model that we cannot comprehend just yet. We learned the trends of the data and what it takes to see such model being built using the neural network model.

VII. Supplementary Material Section

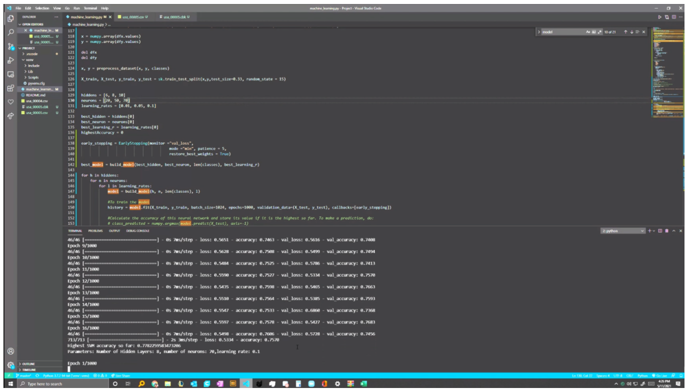
Link to Latex File: https://www.overleaf.com/5987437235jqmgqrytzbgv

Link to Source Code: https://github.com/ameerabdallah/MachineLearningProject

Presentation slides: https://www.canva.com/design/DAEePp_F6Bk/share/preview?token=Ir76RB0HV78K1nFKeBzlSg&role=EDITOR&utm_content=DAEePp_F6Bk&utm_campaign=designshare&utm_medium=link&utm_source=sharebutton

Link to dataset: https://github.com/ameerabdallah/MachineLearningProject/blob/master/usa_00005.csv

https://github.com/ameerabdallah/MachineLearningProject/blob/master/usa_00005.cbk



```python

x = numpy.array(dfx.values)
y = numpy.array(dfy.values)

del dfx
del dfy

x, y = preprocess_dataset(x, y, classes)

X_train, X_test, y_train, y_test = sk.train_test_split(x,y,test_size=0.33, random_state = 15)


hiddens = [6, 8, 10]
neurons = [20, 50, 70]
learning_rates = [0.01, 0.05, 0.1]

best_hidden = hiddens[0]
best_neuron = neurons[0]
best_learning_r = learning_rates[0]
highestAccuracy = 0

early_stopping = EarlyStopping(monitor ="val_loss",
                                        mode ="min", patience = 5,
                                        restore_best_weights = True)

best_model = build_model(best_hidden, best_neuron, len(classes), best_learning_r)

for h in hiddens:
    for n in neurons:
        for l in learning_rates:
            model = build_model(h, n, len(classes), l)

            #To train the model
            history = model.fit(X_train, y_train, batch_size=1024, epochs=1000, validation_data=(X_test, y_test), callbacks=[early_stopping])

            #Calculate the accuracy of this neural network and store its value if it is the highest so far. To make a prediction, do:
            # class_predicted = numpy.argmax(model.predict(X_test), axis=-1)
```

```
46/46 [==============================] - 0s 7ms/step - loss: 0.5651 - accuracy: 0.7463 - val_loss: 0.5616 - val_accuracy: 0.7408
Epoch 9/1000
46/46 [==============================] - 0s 7ms/step - loss: 0.5628 - accuracy: 0.7508 - val_loss: 0.5499 - val_accuracy: 0.7494
Epoch 10/1000
46/46 [==============================] - 0s 7ms/step - loss: 0.5484 - accuracy: 0.7525 - val_loss: 0.5786 - val_accuracy: 0.7413
Epoch 11/1000
46/46 [==============================] - 0s 7ms/step - loss: 0.5590 - accuracy: 0.7527 - val_loss: 0.5334 - val_accuracy: 0.7570
Epoch 12/1000
46/46 [==============================] - 0s 7ms/step - loss: 0.5435 - accuracy: 0.7598 - val_loss: 0.5465 - val_accuracy: 0.7663
Epoch 13/1000
46/46 [==============================] - 0s 7ms/step - loss: 0.5510 - accuracy: 0.7564 - val_loss: 0.5385 - val_accuracy: 0.7593
Epoch 14/1000
46/46 [==============================] - 0s 7ms/step - loss: 0.5547 - accuracy: 0.7533 - val_loss: 0.6860 - val_accuracy: 0.7368
Epoch 15/1000
46/46 [==============================] - 0s 7ms/step - loss: 0.5597 - accuracy: 0.7578 - val_loss: 0.5427 - val_accuracy: 0.7683
Epoch 16/1000
46/46 [==============================] - 0s 7ms/step - loss: 0.5498 - accuracy: 0.7606 - val_loss: 0.5728 - val_accuracy: 0.7456
713/713 [==============================] - 2s 3ms/step - loss: 0.5334 - accuracy: 0.7570
Highest SVM accuracy so far: 0.7782259583473206
Parameters: Number of Hidden Layers: 8, number of neurons: 70,learning rate: 0.1

Epoch 1/1000
```

Fig. 5. Picture of proof

References

[1] "Backpropagation Calculus | Deep Learning, Chapter 4." YouTube, YouTube, 3 Nov. 2017, www.youtube.com/watch?v=tIeHLnjs5U8.

[2] "But What Is a Neural Network? | Deep Learning, Chapter 1." YouTube, YouTube, 5 Oct. 2017, www.youtube.com/watch?v=aircAruvnKk.

[3] "Gradient Descent, How Neural Networks Learn | Deep Learning, Chapter 2." YouTube, YouTube, 16 Oct. 2017, www.youtube.com/watch?v=IHZwWFHWa-w.

[4] "What Is Backpropagation Really Doing? | Deep Learning, Chapter 3." YouTube, YouTube, 3 Nov. 2017, www.youtube.com/watch?v=Ilg3gGewQ5U.

[5] "IPUMS." U.S. CENSUS DATA FOR SOCIAL, ECONOMIC, AND HEALTH RESEARCH, www.usa.ipums.org/usa/.

[6] "Comparative Study of Classifiers in predicting the Income Range of a person from a census data." towards data science, www.towardsdatascience.com/comparative-study-of-classifiers-in-predicting-the-income-range-of-a-person-from-a-census-data-96ce60ee5a10.