Analysis - Log

Deep Neural Network Application on Ted Talk Data Set – Predicting number of views and comments (popularity)

May 4, 2021, 2021-05-04, Tuesday.

Need to organize all the python scripts and jupytor notebooks.

Colour Key: Complete, In Progress,

1. Cleaning the data + Exploratory Data Analysis.
   1. Corresponding scripts, code, and jupyter notes:

Univariable Analysis using Neural Network (unn)

1. Model 1: Univariable Shallow Neural Network (usnn)
   1. Create a univariable shallow neural network (usnn)
   2. Perform a hyperparameter search for the usnn using the following the search options for the hyperparameters:
   3. Visual the results of the usnn hyperparameter search and find the optimal hyperparameter choice with the production error.

Multivariable Analysis using Neural Network (mnn)

1. Model 2: Multivariable Shallow Neural Network (msnn)
   1. Create a multivariable shallow neural network (msnn) with only 1 hidden layer.
   2. Perform a hyperparameter search for the msnn using the following the search options for the hyperparameters:
   3. Visual the results of the msnn hyperparameter search and find the optimal hyperparameter choice with the production error.
2. Model 3: Multivariable Deep Neural Network with 3 hidden layers (mdnn3hd)
   1. Create a multivariable deep neural network with 3 hidden layers (mdnn3hd).
   2. Perform a hyperparameter search for the msnn3hd using the following the search options for the hyperparameters:
   3. Visual the results of the msnn3hd hyperparameter search and find the optimal hyperparameter choice with the production error.
3. Model 4: Multivariable Deep Neural Network with 6 hidden layers (mdnn6hd)
   1. Create a multivariable deep neural network with 6 hidden layers (msnn6hd).
   2. Perform a hyperparameter search for the msnn6hd using the following the search options for the hyperparameters:
   3. Visual the results of the msnn6hd hyperparameter search and find the optimal hyperparameter choice with the production error.
4. Model 5: Multivariable Deep Neural Network with 12 hidden layers (mdnn12hd)
   1. Create a multivariable deep neural network with 12 hidden layers (mdnn12hd).
   2. Perform a hyperparameter search for the msnn12hd using the following the search options for the hyperparameters:
   3. Visual the results of the msnn12hd hyperparameter search and find the optimal hyperparameter choice with the production error.
5. Model 6: Multivariable Deep Neural Network with 24 hidden layers (msnn24hd)
   1. Create a multivariable deep neural network with 24 hidden layers (msnn24hd).
   2. Perform a hyperparameter search for the msnn6hd using the following the search options for the hyperparameters:
   3. Visual the results of the msnn24hd hyperparameter search and find the optimal hyperparameter choice with the production error.
6. Model 7: Multivariable Deep Neural Network with 50 hidden layers (msnn50hd)
   1. Create a multivariable deep neural network with 50 hidden layers (msnn50hd).
   2. Perform a hyperparameter search for the msnn50hd using the following the search options for the hyperparameters:
   3. Visual the results of the msnn50hd hyperparameter search and find the optimal hyperparameter choice with the production error.

Data

There are several different types of conferences in the TED Talks dataset: Ted talks and non-Ted talks. And within Ted Talks there are several different types:

TED Conference (the main conference)

TEDGlobal (international sister conference)

TED Translators (formerly The Open Translation Project (OTP))

TEDx (independent TED conferences)

TED Fellows

TED-Ed

TED Interview

TEDMED

TEDWomen

Of these the following are present in the dataset:

Use all the data so as to improve the prediction. However, make the event type either TED event or not as a variable.

We have done univariate prediction of number of views using the number of comments as predictor variable (feature).

We will now automate this process with different permutations of the different hyperparameter choices, by writing function, loops, and creating data frame from the results.

Hypothesis:

Hypothesis 1: Batch size vs. epochs: The batch sizes should be small enough that each epoch run is like running independent data sets. Hence, my hypothesis is that it is better to have smaller batch\_sizes and many more epochs. I.e. batch\_size << number of epochs. The results of independently training many weights and pooling the results in the end I think will out perform training data sets that are more dependant as they share more of the data points as the batch size is large. Further, I believe that batch\_size x epochs should equal at least 5 times the size of the training set.

batch\_size << epochs

batch\_size = epochs

epochs

epochs

batch\_size >> epochs

batch\_size

Batch\_size

Hyperparameter list:

1. Number of neurons in the input layer: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
   1. Question 1: Keras has number of neurons AND input\_dimension variables. Input\_dimension is for specifying the number of variables to be passed into the neural network. I am not sure what the number of neurons does in the input layer. I do not think it needs to be equal to the input\_dimensions and I do not know if there needs to be any alignment between the two.
2. Number of hidden layers: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.
   1. The more there are the longer the compute time, also supposedly, the neural network can self-select and identify more features in the data.
3. Number of neurons per hidden layer: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10.
   1. Question 2. Overall structure: Should there be any restriction, either upper or lower limit, for how many neurons per layer there should be? One question is that is there some advantage of having a monotonically decreasing neural network structure – meaning the number of neurons in every subsequent layer can only be equal or less than in the previous one.
   2. Question 3: How does having more neurons in the first hidden layer than either in the input\_dimension or the number of neurons in the input layer affect the performance, computation, and efficiency of the neural network.
4. Activation function: relu, leakyRelu, sigma,
5. Optimizer: SDG, adam, nadam,
6. Loss function for regression: mse, mae (mae is more robust to outliers than mse because of the absolute value taken), Huber Loss - The Huber loss combines the best properties of MSE and MAE. It is quadratic for smaller errors and is linear otherwise (and similarly for its gradient). It is identified by its delta parameter – source: <https://www.analyticsvidhya.com/blog/2019/08/detailed-guide-7-loss-functions-machine-learning-python-code/> among other loss functions.
7. Evaluation metric of the neural network:

Experiment aborted around 5400 data points. Total data points should have been 14400 which was the number of all hyperparameter combinations.

Realized logically incorrect hyperparameter values as well as it was taking long to complete. Output layer had more than one output neuron as hyperparameter values (1,2,3,4,5,6,7,8,9,10) which does not make sense because this is a regression problem and there should only be one neuron in the output layer so that we have one numeric output value. Five or ten will give 5 or 10 different regression results. Further, the y is only a vector, not a matrix, hence, how can y be compared/assessed/evaluated against more than 1 output like 5 or 10.

# Results of Experiment 1, 2, 3

Hyperparameter search for predicting views using number of comments. A very simple model: a univariable model. The correlation between views and number of comments is 0.528903.

So there is some positive correlation but they are not very strongly correlated. Clearly there are other variables that increase or decrease the number of views.

We will divide the reasons behind the number of views into three categories:

* 1. The merits of the talk affecting the views.
  2. The logistical reasons affecting the views.
  3. All other reasons.

Lastly, we will also try to understand and predict which is the best Ted Talk.

Structure of the Neural Network Experiment 3:

# The Model

model = Sequential()

model.add(Dense(neurons\_in\_inputlayer, input\_dim = 1, activation = activation\_fn)) # Input Layer

model.add(Dense(neurons\_in\_hiddenlayer,activation = activation\_fn)) # Hidden Layers

model.add(Dense(1, activation = output\_activation\_fn)) # Output Layer

# Compiling the model

model.compile( loss=loss\_fn, optimizer=optimizer,

metrics=['mse', 'mae', 'mape', CosineSimilarity(), RootMeanSquaredError() , MeanSquaredLogarithmicError()] )

# Train the model and make predictions

model.fit(X\_train, y\_train, epochs=epochs, batch\_size=batch\_size, verbose = verbose)

score = model.evaluate(X\_test, y\_test, verbose = verbose) # Evaluate the model

## Results of experiment 1

Summary of results

Which metrics predict which combination of hyperparameters.

How much variation is there in the metrics by choosing the different hyperparameter values.

## Results of experiment 2

Summary of results

## Results of experiment 3

Summary of results

## Completed Experiment 4 today 07/04/2021 (Wednesday, April 7, 2021)

## Results of experiment 4

Summary of results