PREDICTING EFFECTIVE CUSTOMER TOUCHPOINT

Ahmed Bux Abro, Nagarjuna Rao V S Chakka

CS-230 Stanford University

https://youtu.be/X_dIEZn8Wek

Abstract

Deep learning has conventionally been used for unstructured data. We are using deep learning prediction model to target Google marketing problem, to predict what is the most effective touchpoint (mobile vs desktop vs tablet) for their customers that shop on Google online merchandize store (GStore). The model predicts the touchpoint based on the structured dataset of 1.7 Million customer online visits. Our project leverages the two different encoding techniques for structure data One-hot encoding for inputs and Label Encoding for output (predicting labels) and predicts the 3 output classes with 96.7

Dataset

We use Google's structured ecommerce dataset available for Kaggle's Google competition, it includes 1.7 Million past customer online visits that shop on Google online merchandize store (GStore). Data includes customer transactions information, purchase detail along with the touchpoint and channel used for the transaction. The dataset is split between the training data and test data. Raw data with customer transactions was exported from Google's ecommerce website in .csv format, with many columns aggregated as JSON blobs that required efforts to preprocess the data. Further detail about the dataset provided in following section.

Dataset Overview:

Train.csv: contains customer visits and transactions from August 1st 2016 to April 30th 2018. Rows: 903653, Features: 55

Test.csv: contains customer visits and transactions from May 1st 2018 to October 15th 2018. Rows: 804684, Features: 53

Data Preprocessing

Both training and test datasets included 4 JSON format aggregated columns titled as device, geoNetwork, totals, trafficSouce.

Following phases of data preprocessing were performed on the datasets:

- First, flatten the data and extract all sub-columns from the JSONs
- Second, adding new time features. Also adding new aggregated features (average and sum) grouped by unique fullVisitorId.
- Third, change the data types as appropriate for model, such as converting numerical to floating and device.isMobile from Yes/No to 0/1
- Fourth, Columns were checked for more than 50 percent of null values and were dropped
- Fifth, columns were checked for the null value and filled for its missing and null values
- Finally, getting rid of the columns that are not needed for the model such as trafficSource.adwordsClickInfo.*, trafficSource.*, socialEngagementType, sessionId, device.browser*, visitId, visitStartTime

Data Normalization

We used MinMax Scaling using below formula:

$$\frac{x_i - \min(x)}{\max(x) - \min(x)}$$

Fig. 1: Normalization

Encoding (One-hot and Label Encoding)

Based on our research, we learned that the significance of different encoding types based on the requirements of the model and individual data type for features and labels. One-hot coding was applied to categorical features: channelGrouping, device.isMobile, month, weekday. Each feature coding resulted in number of dummy variables. It dramatically introduced the number of features of one-hot coded columns. We also applied label encoding for selected categorical and labeled data and perform the fit and transform functions on the encoding.

Label Encoding

Food Name	Categorical #	Calories
Apple	1	95
Chicken	2	231
Broccoli	3	50

→

One Hot Encoding

Apple	Chicken	Broccoli	Calories
1	0	0	95
0	1	0	231
0	0	1	50

Fig. 2: Encodings used

Loss Function

Since we use multi-class SoftMax classification for our output layer and our labels are integer encoded, we use sparse categorical cross entropy loss function.

$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$$

Fig. 3: Sparse Categorical Cross Entropy

Model

Our model is a multi-layer perceptron (MLP) using fully connected neural networks with input layer, four hidden layers and multi-class SoftMax classification output layer. Model uses "ReLU" activation for input and hidden layers while the output layer uses "SoftMax".

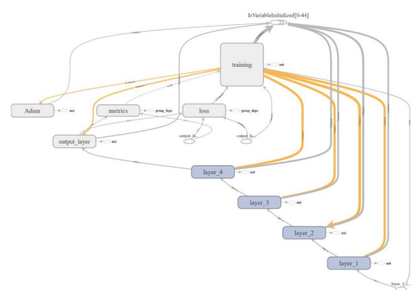

Fig. 4: Model Architecture

Hyperparameter tuning

We tried and tested different model architectures before reaching to our final model architecture that fits our requirements, features and output. After experimenting different mini-batch sizes, we notice that the mini-batch size of 128 was the best performing architecture for our model as there was a significant difference in performance and accuracy with different batch sizes of 32/64/128.

We also tried using large epoch sizes but noticed that there was no much difference in performance beyond 50 epochs. Also experimenting with learning rates of 0.0001/0.0002 and 0.0003, we found that the learning rate for 0.0003 was the best parameter for learning rate. Below is the summary of hyperparameters for final model .

Weights Initialization	Glorot
Activation Function	ReLU (for hidden layers) and SoftMax (for Output)
Mini- Batch Size	128
Optimizer	Adam
Epochs	50
Learning Rate	a = 0.0003
Loss Function	sparse_categorical_crossentropy

Fig. 5: Final working Hyperparameters.

Results

The final model results offered a 96 percent validation accuracy, 13 percent validation loss.

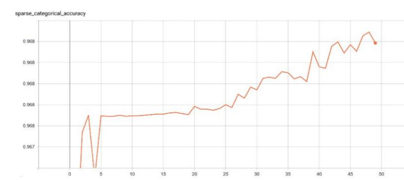

Fig. 6: 50 Epoch Accuracy.

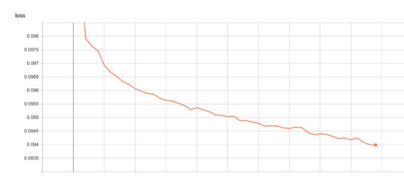

Fig. 7: 0 Epoch Loss