CS230

Predicting Effective Customer Touchpoint

Ahmed Bux Abro
Stanford University
aabro@stanford.edu

Nagarjuna Rao V S Chakka
Stanford University
nagarjuna@stanford.edu

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 merchandise 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% accuracy and 9.5% loss.

1 Introduction

Deep Learning has touched many aspects of our life and used to solve problems from finance to healthcare. Marketing is looking for ways to adopt Artificial Intelligence to solve their pressing challenges such as customer retention, customer loyalty and improving overall customer relationship. With the evolving technology landscape, customers keep changing their preferred channel to communicate or deal with the firm. Different generations of customers prefer different touchpoints based on their personal interests. While Gen-Z wants to use social platforms such as Instagram and Snapchat to not only socialize but also shop and, in some cases, perform financial transactions, on the other side Millennials (Gen-Y) prefer to use traditional web email touchpoints for their interaction. Firms today invest their marketing resources (budget, etc.) across multiple touchpoints with a fear of missing out (FOMO) the customer opportunity to their competitors as they are unable to predict the most effective touchpoint to engage their customers. We have applied Deep Learning (DL) in our project to address the problem by using deep neural network model to predict the most effective touchpoint for firms to engage with their customer based on the history of past engagements.

While DL has been proven to effective in handling the large volumes of unstructured data (vision, voice, text) and able to predict meaningful patterns, firms use machine learning to forecast revenue predictions with higher accuracy [1]. We believe and test in our project that how Deep Neural Network (DNN) will equally outperform other ML methods and will offer greater performance in processing the structured data. As most firms record their customer information and history in a structured format, we developed a DNN model that analyze the structured data and uses the combination of one-hot and label encoding technique to use categorical data for Google online merchandise store (GStore) customer online visit records and predict the preferred touchpoint for Google customers.

Initially, we planned to combine structured data (firm data) and customer image data (computer vision) but due to the time limitation, we were only able to develop a deep learning program to predict using structured data only. However, in future we plan to combine both computer vision and structured data related into a single model to provide a comprehensive prediction of customer preferred touchpoint across multiple touchpoint types (social media, in-store and others).

2 Related work

Machine learning techniques like Gradient Boosted Tree models are commonly used for predicting tabular or structured data. Recently research started focusing on the use of Deep Neural Network to improve network performance for the structured data. A neural network based, self-adaptive information pattern recognition methodology to solve marketing problem of customer churn was proposed by [2], by applying feedforward NN that can automatically learn features and that is based on error back propagation algorithm, they were able to gain 92% accuracy on structured marketing data. A more recent paper by [3] proposed the use of entity embedding techniques for marketing problem of predicting customer lifetime value (CLTV), the technique is adopted from Natural Language Processing (NLP), it helps improve the prediction performance significantly. Embedding technique has an advantage over one-hot in size (more compact) and dense representation to offer superior performance as it does in NLP. DL research continues to improve the technique to use feed forward networks like Recurrent Neural Network (RNN) with embedding to accurately predict large time-series dataset [4].

3 Dataset and Data Preprocessing

We use Google's structured e-commerce dataset available for Kaggle's Google competition, it includes 1.7 Million past customer online visits that shop on Google online merchandise 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 were exported from Google's e-commerce 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 the following section.

3.1 Dataset Overview

- train_v2.csv - the updated training set - contains user transactions from August 1st 2016 to April 30th 2018.
    - Rows: 903653
    - Features: 55
- test_v2.csv - the updated test set - contains user transactions from May 1st 2018 to October 15th 2018.
    - Rows: 804684
    - Features: 53
- Note that the training data was further split into training and validation data.

Features

Train and dataset matched the columns except that the test dataset did not include the touchpoint categories and used for later predict of preferred touchpoints for the test dataset.

- *fullVisitorId*- A unique identifier for each user of the Google Merchandise Store.
- *channelGrouping* - The channel via which the user came to the Store.
- *date* - The date on which the user visited the Store.
- *device* - The specifications for the device used to access the Store.
- *geoNetwork* - This section contains information about the geography of the user.
- *socialEngagementType* - Engagement type, either "Socially Engaged" or "Not Socially Engaged".
- *totals* - This section contains aggregate values across the session.
- *trafficSource* - This section contains information about the Traffic Source from which the session originated.
- *visitId* - An identifier for this session. This is part of the value usually stored as the _utmb cookie. This is only unique to the user. For a completely unique ID, you should use a combination of fullVisitorId and visitId.
- *visitNumber* - The session number for this user. If this is the first session, then this is set to

1

- *visitStartTime* - The timestamp (expressed as POSIX time).
- *hits* - This row and nested fields are populated for any and all types of hits. Provides a record of all page visits.
- *customDimensions* - This section contains any user-level or session-level custom dimensions that are set for a session. This is a repeated field and has an entry for each dimension that is set.
- *totals* - This set of columns mostly includes high-level aggregate data.

Label

We use *device.deviceCategory* as the target label that contains mobile/desktop/tablet as the values that are later encoded to use in the model.

3.2 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 the model, such as converting numerical to floating and *device.isMobile* from Yes/No to 0/1
- Fourth, Columns were checked for more than 50% 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*

3.3 Data Preprocessing: Normalization

Input feature set required the normalization due to the varied types (continuous and categorical) and varied ranges of input across different features. We used MinMax Scaling with below formula:

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

MinMax Scaling helped us keep the features in the same range between 0 and 1. Normalizing feature distribution helped the model to converge faster, reduce the processing time and increase performance.

3.4 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 a 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.

4 Methods

4.1 Framework

We developed our DNN model using the Keras neural network library written in python. It helped us reduces the complexity at TensorFlow level through abstraction and help keep the focus on the actual problem-solving.

4.2 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". Below is the model summary:

```
Layer (type)              Output Shape         Param #          Activation
==================================================================
layer_1 (Dense)           (None, 200)          8800             ReLU
__________________________________________________________
layer_2 (Dense)           (None, 100)          20100            ReLU
__________________________________________________________
layer_3 (Dense)           (None, 100)          10100            ReLU
__________________________________________________________
layer_4 (Dense)           (None, 100)          10100            ReLU
__________________________________________________________
output_layer (Dense)      (None, 3)            303              SoftMax
==================================================================
Total params: 49,403
Trainable params: 49,403
Non-trainable params: 0
```

Batch normalization technique is used as part of the model architecture. It helped speed up the training with fewer steps and improve accuracy. Experimentation using different mini-batches is further discussed in the following section. Following the graphical model architecture design

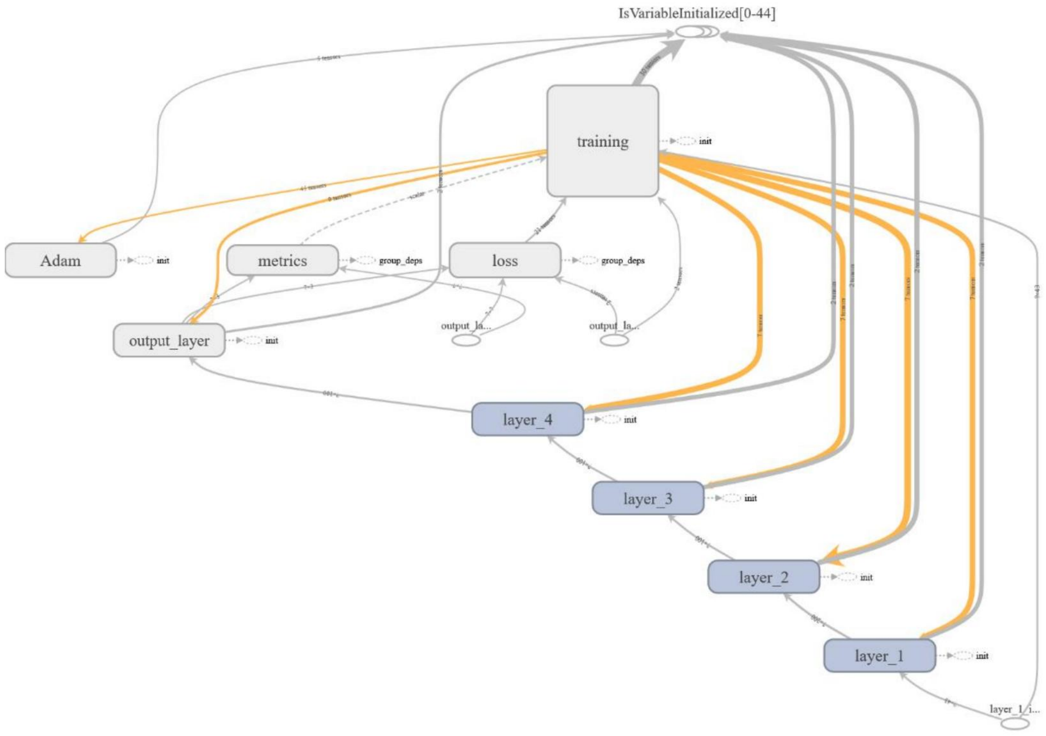

Figure: Graphical Model Architecture Design

4.3 Cost Function

Since we use multi-class SoftMax classification for our output layer and our labels are integer encoded, we chose to use "sparse_categorical_crossentropy" loss function. It uses the same equation as categorical cross entropy, with the only benefit that it allowed us to keep output as integers:

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

5 Experiments and Results

5.1 Experiments and 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 with 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 the final model

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

Table: Hyperparameters for final model

5.2 Results

The final model results offered a 96% validation accuracy, 13% validation loss. However, we noticed that the results for validation accuracy indicate that the model was able to gain greater accuracy during initial epochs and running the model with longer epochs didn't help much in improving the model learning. We plan in the future to revisit and test new model architecture, hyperparameters and also add new image features in the model to see if the model learning improves any further with new changes.

Below is the snapshot for last epoch and related loss and accuracy for training and validation:

"765707/765707 [==============================] - 32s 42us/step - loss: 0.0940 - sparse_categorical_accuracy: 0.9677 - val_loss: 0.1285 - val_sparse_categorical_accuracy: 0.9574"

We ran tensorboard for graphical representation of the model training and output. Below are the graphs from tensorboard for a model run:

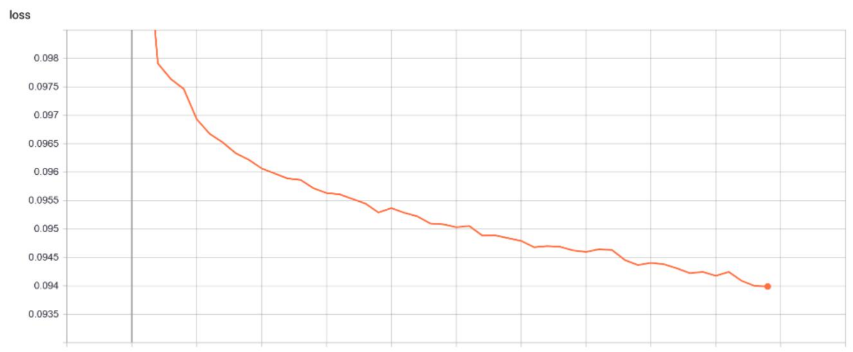

Figure: Loss with 50 Epochs

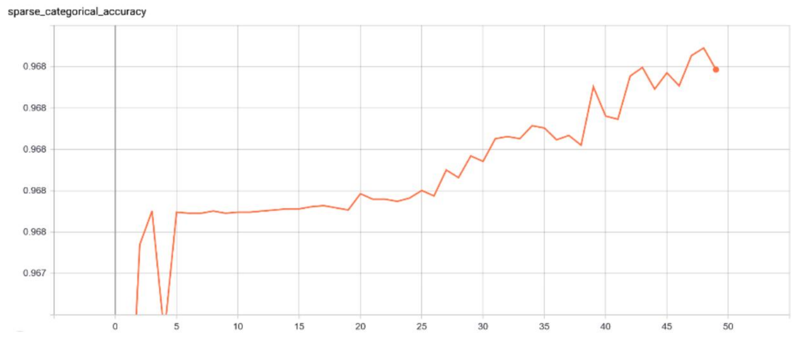

Figure: Sparse Training Accuracy with 50 Epochs

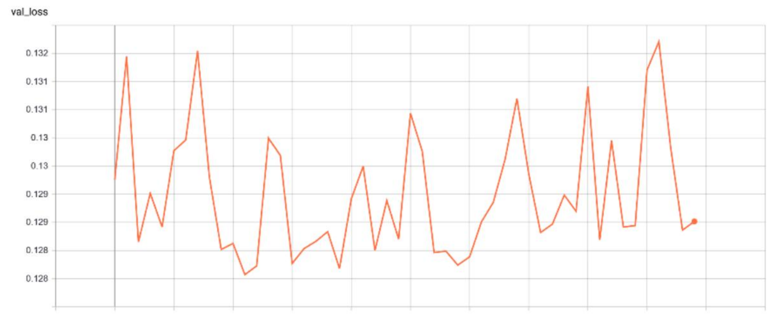

Figure: Validation Loss with 50 Epochs

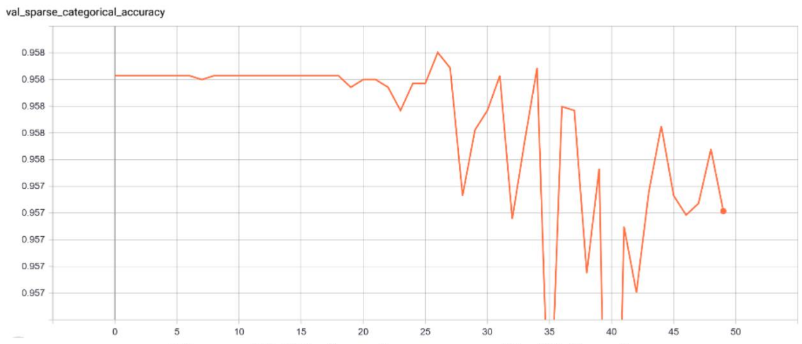

Figure: Validation Accuracy with 50 Epochs

6 Conclusion and Future Work

This project helped us validate the outstanding performance of Deep Neural Networks for the structured dataset, its ability to predict complex marketing data patterns with up to 96% accuracy, and its impact on solving pressing marketing problems. We acknowledge the possible gaps in the current model. However we believe our project is introducing just a baseline model and probably an easy task for deep learning compares to its capability to process complex data. However, as future task, we can challenge and enhance this model by adding computer vision (product catalog images, customer images, live video) and Text (social media, reviews). We planned to introduce a future enhancement to model, adding transfer learning using Inception v4 and RNN. After these enhancements model would offer better use of DNN techniques introduced in this project and provide more meaningful predictions for our data. Finally, we think structured data is as critical as non-structured data to solve complex business problems as firms mostly record their customer data in a structured format. By adopting deep learning techniques, we can address many current challenges with outstanding performance, and higher accuracy compare to many other ML techniques.

7 Contributions

All team members involved in this project have a contribution. While Nagarjuna focused on dataset management and data preprocessing, Ahmed focused on model architecture, hyperparameter selection and tuning.

8 References

1. Gajewar, A. and G. Bansal, *Revenue forecasting for enterprise products.* arXiv preprint arXiv:1701.06624, 2016.
2. Sharma, A., D. Panigrahi, and P. Kumar, *A neural network based approach for predicting customer churn in cellular network services.* arXiv preprint arXiv:1309.3945, 2013.
3. Chamberlain, B.P., et al. *Customer lifetime value prediction using embeddings*. in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2017. ACM.
4. Grob, G.L., et al. *A Recurrent Neural Network Survival Model: Predicting Web User Return Time*. in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. 2018. Springer.