OFFER PERSONALIZATION USING TEMPORAL CONVOLUTIONS AND MULTI-OBJECTIVE OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Personalized marketing has become important for large scale to small scale e-retail firms due to significant raise in online shopping and market competition. Increase in online shopping and high market competition has led to an increase in promotional expenditure for online retailers. Within these firms, the category manager has to solve the promotion optimization problem for each consumer segment and item combination, to be able to design the best offer for each period in a finite horizon, so as to maximize the retailer's profit. This paper considers the problem of planning sales promotions at the intersection of consumer, item and time in e-retail setting, where we first predict item purchase propensity and then estimate the consumer offer elasticity for solving offer optimization. We build deep neural network model to predict purchase propensity, simulation approach for consumer elasticity and finally offer optimization to compute best offer at consumer item level.

1 METHODOLOGY

We treat each relevant consumer-item as an individual object and shape them into weekly time series data based on historical transactions. In this setup, target value at each time step (week) takes a binary input, 1/0 (purchased/non purchased). *Relevancy* of the consumer-item is defined by items transacted by consumer during training time window. *Positive samples* (purchased/1) are weeks where consumer did transact for an item, whereas *Negative samples* (non purchased/0) are the weeks where the consumer did not buy that item. We apply sliding windows testing routine for generating out of time results. The time series is split into 3 parts - train, validation and test as shown in Table 1. All our models are built in a multi-object fashion, which allows the gradient movement to happen across all consumer-item combinations split in batches. This enables cross-learning to happen across consumers/items. We then perform Feature Engineering over data splits to generate modelling features. Below are some of the feature groups we perform our experiments with:

- **Datetime:** We use transactional metrics at various temporal cuts like week, month, etc. Datetime related features capturing seasonality and trend are also generated.
- **Consumer-Item Profile:** We use transactional metrics at different granularities like consumer, item, consumer-item, department and aisle. We also create features like Time since first order, Time since last order, time gap between orders, Reorder rates, Reorder frequency, Streak - user purchased the item in a row, Average position in the cart, Total number of orders.
- **Consumer-Item-Time Profile:** We use transactional metrics at the intersection of consumer, item and time. We generate interactions capturing consumer behaviour towards items at a given time.
- **Lagged Offsets:** We use statistical rolling operations like mean, median, quantiles, variance, kurtosis and skewness over temporal regressors for different lag periods to generate offsets.

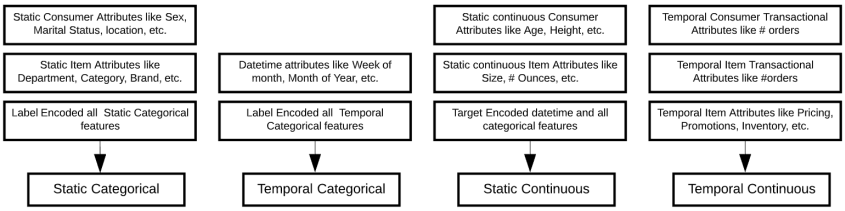
The model we need to build, thus, should learn to identify similarly behaving time series across latent parameters, and take into account consumer and item variations in comparing different time series. A row in time series is represented by

$$y_{\text{cit}} = f(i_{\text{t}}, c_{\text{t}}, .., c_{\text{t-n}}, ic_{\text{t}}, .., ic_{\text{t-n}}, d_{\text{t}}, .., d_{\text{t-n}}) \tag{1}$$

Table 1: Modelling data splits

Data Split	Specifications	Consumer-Item combinations	Max Time-Series length
Train	Model training	50,872	46 weeks
Validation	HyperParameter Optimization	50,888	2 weeks
Test	Reporting Accuracy Metrics	50,910	2 weeks

Figure 1: Data classification for DNN Architectures



where y_{cit} is purchase prediction for consumer 'c' for item 'i' at time 't'. i_t denotes attributes of item 'i' like category, department, brand, color, size, etc at time 't'. c_t denotes attributes of consumer 'c' like age, sex and transactional attributes at time 't'. ic_t denotes transactional attributes of consumer 'c' towards item 'i' at time 't'. d_t is derived from datetime to capture trend and seasonality at time 't'. 'n' is the number of time lags.

1.1 MODELLING

1.1.1 FEATURE ENGINEERING

From data classification, Figure 1, we can see that data was sub-divided 4 groups:

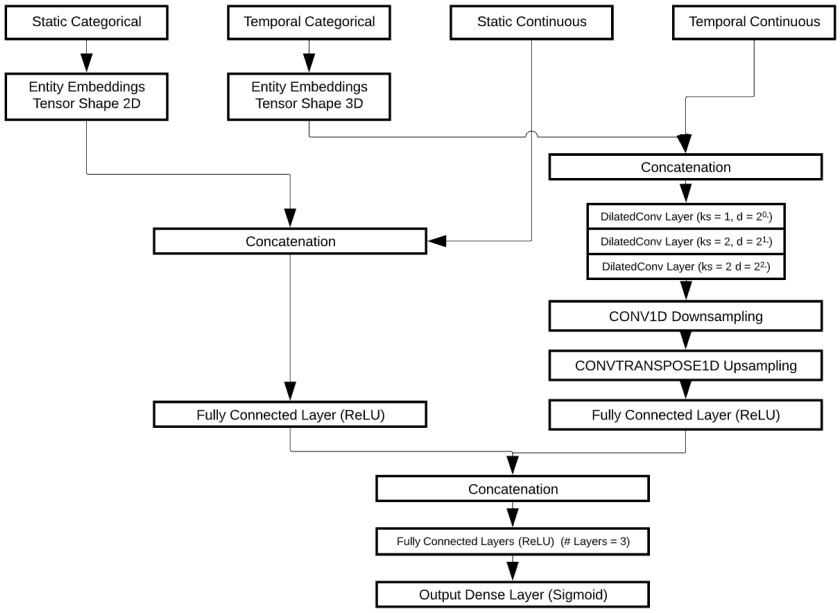
- **Static Categorical:** These are categorical features that do not vary with time. This includes consumer attributes like sex, marital status and location along with different item attributes like category, department and brand.
- **Temporal Categorical:** These are categorical features that vary with time. It includes all the datetime related features like week, month of year, etc.
- **Static Continuous:** These features are static but continuous. This includes certain consumer attributes like age and weight, item attributes like size, and certain derived features like target encoded features.
- **Temporal Continuous:** These are time varying continuous features. All consumer and item related traditional attributes like number of orders, add to cart order, etc. falls under this bucket.

1.1.2 MODEL ARCHITECTURE

As mentioned earlier in this section, traditional machine learning models are not really a suitable choice for modelling f (Equation 1) due to non-linear interaction between the features. We work with Temporal Convolution Network model as shown in Figure Figure 2.

- **Entity Embeddings + Temporal Convolutional Network:** TCN (Figure 2), originally proposed in **?** , is considered a competitive architecture yielding the best results when evaluated over our experimental dataset. This network comprises of 3 dilated Convolutional networks combined with entity embeddings. Similar to LSTM, this architecture, after convolving and concatenating flows into 3 fully connected ReLU based layers yielding to dense layer which has sigmoid activation.

Figure 2: Temporal Convolutional Network (TCN)



1.1.3 HYPERPARAMETER TUNING

Hyper-parameters of tree based models are optimized using Bayesian Hyperparameter Optimization Technique, Hyperopt **?**. For Deep learning, we use documented best practices along with our experimental results to choose model hyperparameters. Hyperparameter Optimization is performed over validation dataset. We list some of the hyperparameters along with the values we tune for Deep learning models.

- **Optimizer Parameters:** RMSProp **?** and Adam are used as different trial configurations. The learning rate is experimentally tuned to 1e-3. We also have weight decay of 1e-5 which helps a bit in model Regularization.
- **Scheduler Parameters:** CyclicLR **?** and ReduceLROnPlateau **?** Learning rates are used as different trial configurations. we use 1e-3 as max lr and 1e-6 as base lr for cyclical learning rate along with the step size being the function of length of train loader. ReduceLROnPlateau is tuned at 1e-6 as min lr.
- **SWA:** Stochastic Weight Averaging (SWA) **?** is used to improve generalization across Deep Learning models. SWA performs an equal average of the weights traversed by SGD with a modified learning rate schedule. We use 1e-3 as SWA learning rate.
- **Parameter Average:** This is a method to average the neural network parameters of n best model checkpoints post training, weighted by validation loss of respective checkpoints. The resulting model generalizes better than those with a single best checkpoint model for an unseen data.

Apart from the above parameters we also iterate to tune network parameters like number of epochs, batch size, number of Fully Connected Layers, number of LSTM layers, convnet parameters (kernel size, dilations, padding) and embedding sizes for the categorical features. Binary Cross-Entropy **??** is used as loss function for all the models.

1.2 F_1-MAXIMIZATION

Post stacking, we optimize for purchase probability threshold based on probability distribution at a consumer level using F_1-Maximization. This enables optimal thresholding of consumer level probabilities to maximize F_1 measure **?**. To illustrate the above, let us say we generated purchase

probabilities for 'n' items out of 'b' actually purchased items by consumer 'c'. Now, let us visualize the actuals (2) and predictions (3) of 'n' predicted items for consumer 'c'.

$$A_c = [a_1, a_2, .., a_n] \; \forall \; a_j \in \{0,1\} \tag{2}$$

$$P_c = [p_1, p_2, .., p_n] \; \forall \; p_j \in \; [0, 1] \tag{3}$$

A_c represents the actuals for consumer 'c', with a_j being 1/0 (purchased/non purchased). P_c represents the predictions for consumer 'c' for the respective item, with p_j being probability value. 'n' represents total items for which the model generated purchase probabilities for consumer 'c'. Now we apply Decision rule D() which converts probabilities to binary predictions, as described below in Equation 4.

$$D(Pr_c) : {P_c}^{1 \times n} \rightarrow {P^{'}}_c^{1 \times n} \quad : p^{'}_j = \begin{cases} 1 & p_j \geq Pr_c \\ 0 & \text{Otherwise} \end{cases} \tag{4}$$

$$P^{'}_c = [p^{'}_1, p^{'}_2, .., p^{'}_n] \; \forall \; p^{'}_j \in \{0,1\} \tag{5}$$

$$k = \textstyle\sum_{i=1}^{n} p^{'}_i \tag{6}$$

Pr_c is the probability cut-off to be optimized for maximizing F_1 measure **?** for consumer 'c'. Decision rule D() converts probabilities P_c to binary predictions $P^{'}_c$ such that if p_j is less than Pr_c then $p^{'}_j$ equals 0, otherwise 1. 'k' is the sum of predictions generated post applying Decision rule D(). Now we solve for F_1 measure using equations and formulae described below.

$$V_{Pr_c} = P^{'}_c \; \times \; {A_c}^T \; \Rightarrow \begin{pmatrix} p^{'}_1 & .. & p^{'}_n \end{pmatrix} \times \begin{pmatrix} a_1 \\ .. \\ a_n \end{pmatrix} \tag{7}$$

$$Precision_c = \frac{V_{Pr_c}}{k} \quad and \quad Recall_c = \frac{V_{Pr_c}}{b} \tag{8}$$

$$F_{1_c} = \frac{2 \times Precision_c \times Recall_c}{Precision_c + Recall_c} \quad \Rightarrow \quad 2 * \frac{V_{Pr_c}}{k+b} \tag{9}$$

V_{Pr_c} represents the number of items with purchase probabilities greater than Pr_c which were actually purchased (True Positives). As can be seen, Formulae 8 and 9 are used to calculate Precision, Recall and F_1-score for consumer 'c'.

$$\max_{V_{Pr_c}} \quad 2 * \frac{V_{Pr_c}}{k+b} \quad , \quad \text{subject to:} \quad Pr_c \in \; [0, 1] \tag{10}$$

Equation 10 represents the optimization function we solve to generate purchase predictions (1/0) for each consumer.

1.3 Elasticity Framework

1.4 Multi-Objective Optimization