Technical Disclosure Commons

Defensive Publications Series

April 27, 2018

Machine Learning Prediction System Based on Tensor-Flow Deep Neural Network and its Application to Advertising in Mobile Gaming

Leon Li

MACHINE LEARNING PREDICTION SYSTEM BASED ON TENSOR-FLOW DEEP NEURAL NETWORK AND ITS APPLICATION TO ADVERTISING IN MOBILE GAMING

Background

In recent years, with the development of smart phone technology, mobile games and other applications on mobile devices have become a new mode of entertainment and leisure. According to the current data surveys and research, the global mobile-end game has reached 2.1 billion players, representing a year-on-year growth of 14%. Among them, most players have paid for games or in-game items.

For strategy or simulation-based mobile games, there may be a steep learning curve, particularly with complex games. Because users may need time to understand and master the game operation and mechanics, even if they may be likely to purchase the game or in-game items, this purchase behavior may be delayed compared to other, simpler genres, such as puzzle or action games. As a result, the measurement and optimization of market launch results in great challenges for such games. The same may apply to other genres of games with steep learning curves, as well as complex applications.

Summary

The systems and methods discussed herein provide for prediction of whether an unpaid trial user will convert to a paid user within 28 days after download and installation, and is based on a model trained from a popular mobile strategy game with 6 million daily users and over 500 separate characteristics or behaviors (e.g. achieving a new level, deploying a particular type of troop, loading a map, etc.)

These systems and methods are based around a machine learning architecture or neural network, and is integrated with a content deployment system that may provide additional content items to a user or device based on a predicted conversion rate (e.g. discounts, offers, advertisements, exclusive items, etc.). In practice, one implementation increased the conversion rate by 7 times and the return on investment by 2.6 times, reduce the cost of acquiring paid users by 63%.

Data preparation

The data includes two parts, a prediction target *Y* and a user parameter *X*, wherein the user parameter *X* is further divided into a status parameter *Xs* and a behavior parameter *Xb*. In

practice, rather than user parameters per se, the data may be specific to devices, accounts, MAC addresses, or other device identifiers, and thus may be considered device parameters.

The raw data may include the following fields:

UserID	AliveDays	Payment	Status 1 to Status N	Behavior 1 to Behavior N
123	0	0		
123	1	2		
234	0	1		
345	0	3		
345	1	0		
345	2	2		

Among them, each row of data is a summary of each user's or device's daily behavior parameters and state snapshots. Days that the user is not logged in may be skipped (e.g. there is no need to record the status or behavior of the day). If the user has not purchased the software or in-app or in-game items within the first *n*-1 days, and there is a paid conversion on the *n*th day, the user has *n* rows of data (assuming the user logs in daily). This is shown, for example, for user ID 123, showing a payment on the second day.

The status parameters represent any sort of status identifier of execution of the application or game and include, but are not limited to, in-game character levels, number of gold coins in the game, brand names of landing game handsets, landing game locations, and the like.

Behavioral parameters represent actions of the user or device and include, but are not limited to:

- pay-related behaviors: Open payment windows, click pay button, etc.
- purchasing items with in-game currency: e.g. skins, weapons, lives, maps, etc.
- social behavior: joining teams or guilds, speaking or chatting with other players, sharing social account identifiers, etc.
- discount-related behavior: open discount stores, apply coupons, etc.

After processing gathered data, the following data is obtained (assuming that the model goal is based on predicting whether unpaid users will make an In-App-Purchase (e.g. purchase the application, additional in-App content such as maps, coins, music tracks or other media, or other goods or unlockable items, or make other such purchases or transactions) within 28 days after download, based on seven days of gathered behavior):

• Prediction Goal *y*: may be represented by a binary value or a percentage probability, such as 1 if there are paid conversions within 28 days after the user downloaded the game or application, and 0 otherwise

• User parameter *x*: Given *j* rows of data for user *i* in the original data, the data may be reduced to a single row of data having a user parameter *x*. This parameter may further be divided into:

• State parameter *Xs*: state snapshot of the last day Xs_j

• Behavior parameter *Xb*: sum of *j*-lines of data $\sum_{k=1}^{j} Xb_k$

Note 1: In some implementations, inputs may be limited to actions taken within 7 days of downloading the application or game as the basis of prediction. This can generate a forecast ideally with a short time (e.g. within seven days) after the user downloads and installs the application, and send the prediction signal to a content delivery platform in time to facilitate machine learning in the content delivery platform.

Note 2: In some implementations, the forecast target may be an in-app conversion or purchase within 28 days after installation. This may provide a reasonable time frame within which to measure the effectiveness of content delivery on the conversion prediction.

Data Cleaning and Feature Engineering

Data may be cleaned or normalized for usage by a prediction engine:

1. Any missing values in the data table may be filled with zeroes or other null values, depending on implementation;

2. Data may be scaled or normalized into a standard range;

3. Principal component analysis (PCA) may be applied to reduce the number of dimensions in the original data by a significant amount, e.g. to half of the original number of dimensions or less

After data cleaning and normalization, the data is processed as follows:

• Divide the processed data into two parts by 1:1, train and test the neural network

• The output value is trained to a predetermined goal value based on whether the conversion event occurs, e.g. *Y_noTransform_train*, *Y_noTransform_test* = 1 if the user pays for the application or game within 28 days of downloading, otherwise 0;

• The corresponding user parameters are transformed, namely *X_transformed_train*, *X_transformed_test*. Each user has one row of data, but after PCA dimension reduction, it may not be immediately apparent what behavior each column in the data set represents.

TensorFlow Deep Neural Network Classifier Modeling

To create the classifier, when designing the system, in some implementations, a tensorflow Python API may be utilized to create a DNNClassifier object directly, eliminating the need to start building the model from the ground up. This may be effective for solving simple classification problems and is a particularly efficient implementation.

classifier = tf.estimator.DNNClassifier();

To train the classifier, the DNNClassifier object's fit function may be called to create a training module.

Classifier.fit(X_transformed_train, Y_noTransform_train, steps=2000, batch_size=5000);

To provide a prediction or forecast for a given data set, the predict function of the DNNClassifier object may be called to create a prediction module. The output may be a binary 0 or 1 value in some implementations, with 0 meaning that the classifier thinks that the conversion probability is lower than 50% (e.g. there will be no payment conversion), and 1 meaning that the classifier thinks that the conversion probability is greater than or equal to 50% (e.g. a large probability of a payment conversion even). 50% may be used as a default threshold of the prediction function. To change this threshold, the *predict_proba* function may be called and the output is the conversion probability, e.g. no longer 0 or 1.

Y_predict = classifier.predict(X_transformed_test);

To verify the classifier's accuracy, precision and recall values may be compared to measure accuracy. For content delivery, recall may be more important than precision, and recall should be increased as much as possible. If the user is operating internally, precision may be more important than recall, and precision should be increased as much as possible.

Function *f1* is a comprehensive consideration of the overall parameters of precision and recall. Given the following parameter values

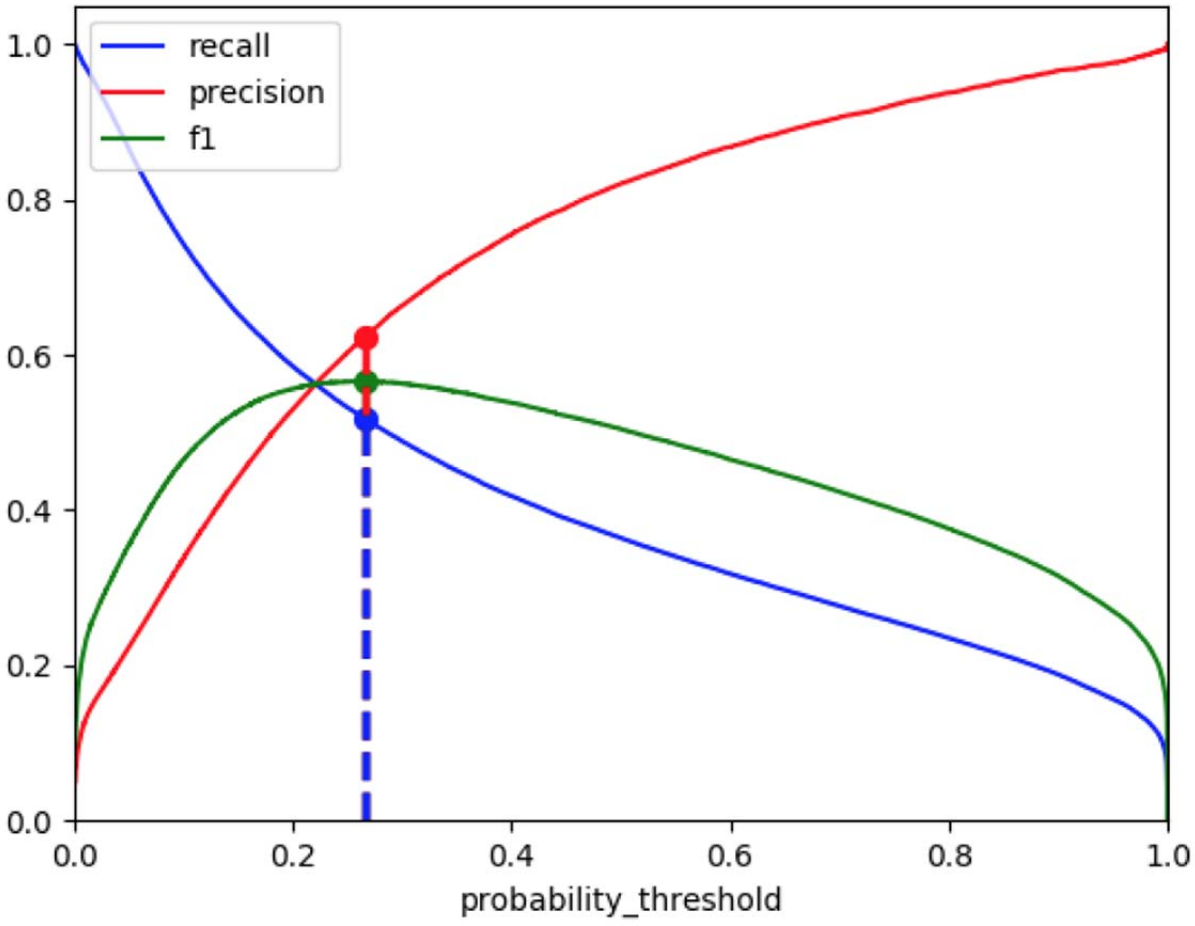
	Y_noTransform_test_0	Y_noTransform_test_1
Y_predict_0	d	c
Y_predict_1	b	a

precision = a / (a+b)

recall = a / (a+c)

f1 = 2*precision*recall / (precision+recall)

After several iterations, the model performance is as follows:

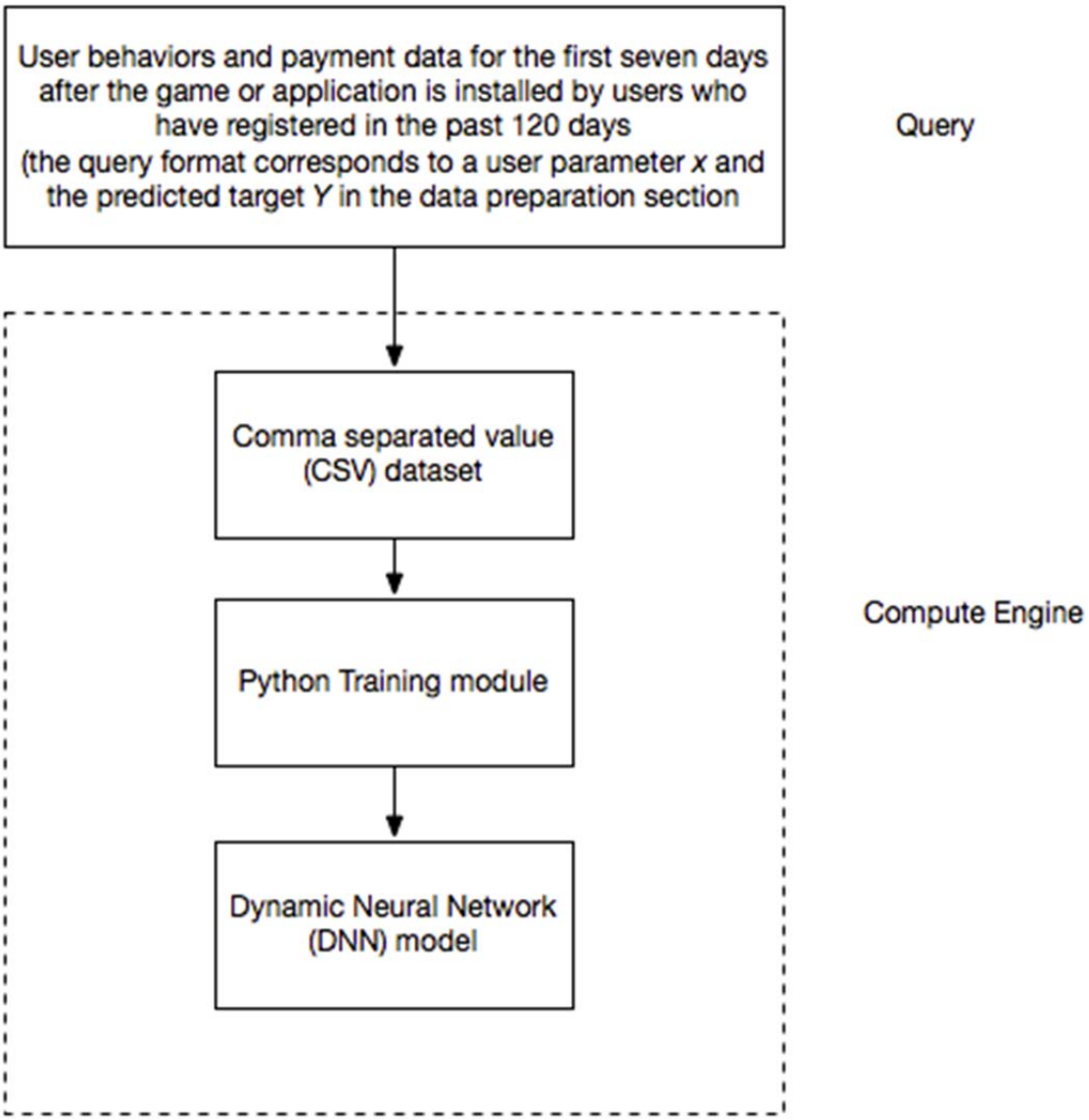


Note: The intersection of the blue vertical dashed line and the x-axis is the value of the prediction probability at which *f1* reaches its highest point.

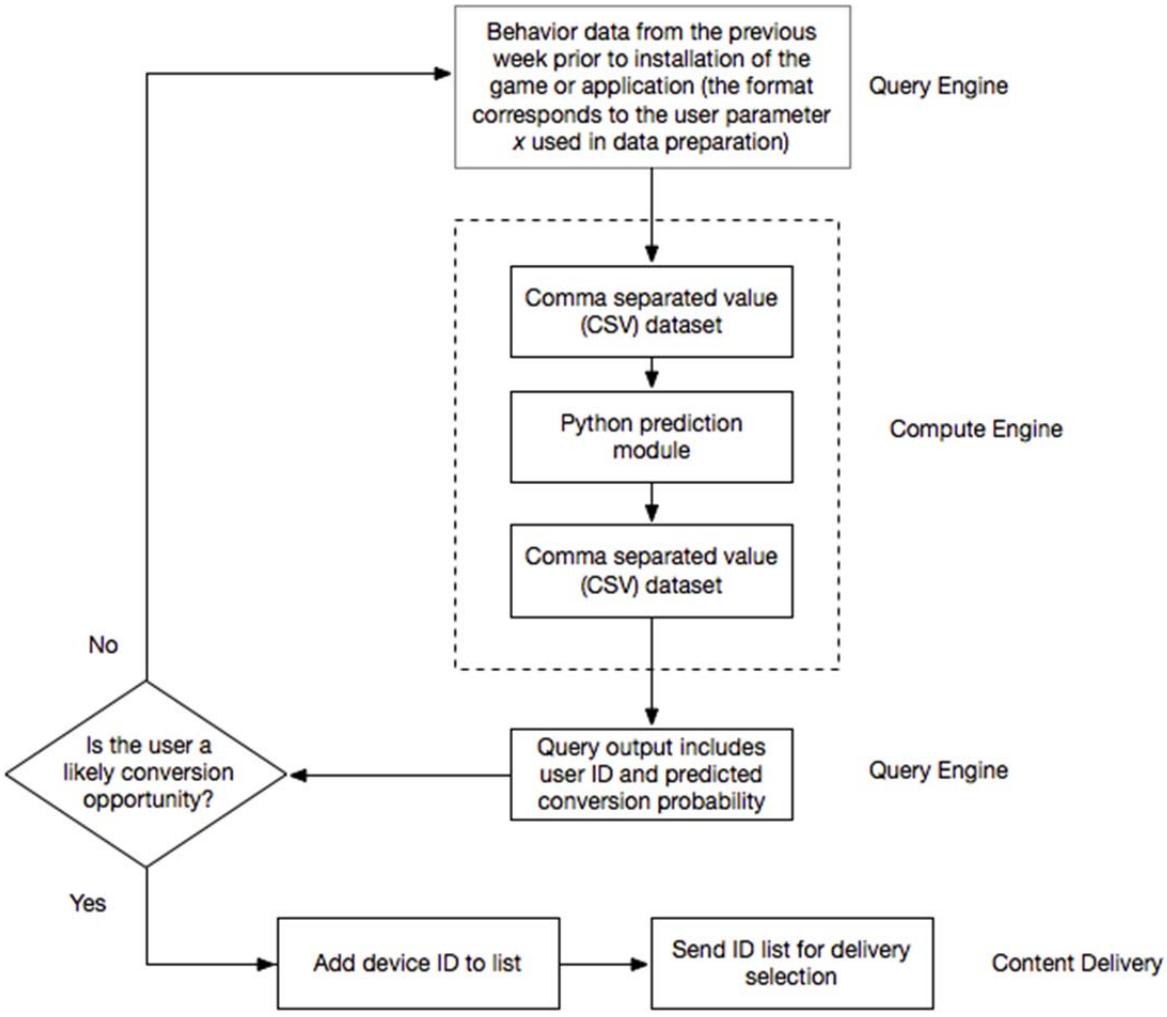
System design

In some implementations, the system may include a Query Engine, a Compute Engine, and Storage, which may be dynamically instantiated by a cloud service in some implementations. Every day, the “prediction module” is run based on newly generated data in the game or application, each user or device may be evaluated and a prediction made as to whether a conversion event will happen. Periodically, such as quarterly, the "training module" is run based on user data registered in the previous quarter, and the model is updated. In some implementations, a hybrid dataset approach may be used with 90 days of new data, and 30 days of old data to ensure smooth collaboration between the model and the content delivery system.

Each quarter:



Daily:



Prediction results and Content Delivery

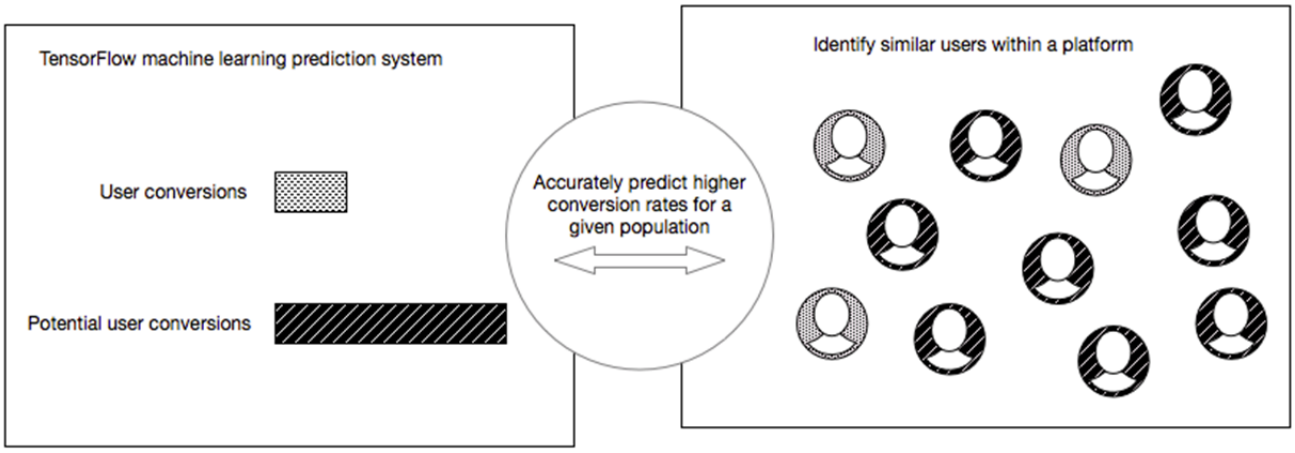
Based on the predicted results of conversions, content may be selected and provided to devices for presentation. For example, coupons, discounts, or other such content may be provided to devices when the prediction model identifies a high likelihood of an eventual conversion event. Conversely, content suggesting other applications or games for which conversions may be more likely may be provided to devices when the prediction model identifies a low likelihood of an eventual conversion event for this game or application.

As described in the previous section, the final predicted result may include two columns of data, a DeviceID or other identifier and the prediction label (0 or 1), having been adjusted after the prediction to a binary value. Depending on the difference in the predicted probability threshold,

the number and accuracy of users who are marked as having a 1 (with high payment propensity) will also be different. The higher the threshold, the more stringent the requirement of being marked as 1, the fewer users or devices may be considered likely to have a conversion event. While this increases accuracy, the resulting increase in false negatives may cannibalize conversions. For example, when the threshold reaches the highest 100%, the predicted events are equivalent to the actual conversion events; when the threshold reaches the lowest 0%, the predicted event is equivalent to application installation. Accordingly, the threshold may be adjusted as in the figure shown above to maximize the function *f1*.

A large number of experimental studies have found that, for content delivery systems, it may be preferable to adjust the threshold lower and sacrifice some precision, in order to make recall as high as possible.

As shown in the figure below, the working principle of the combination of TensorFlow based machine learning prediction system and a content delivery platform is illustrated. On the left is the forecasting system described herein; and on the right is a content delivery platform. Through analysis and forecasting, devices or users are identified that may have a potential conversion event, the number of seed populations to the content delivery system is expanded, and the effective feedback time from application installation to effective conversion is shortened, thereby improving the effect of machine learning and content delivery.



Conclusions and future prospects

This disclosure analyzes the background of the industry and proposes solutions to the challenges faced by complex applications or games, such as strategy games. From the perspective of the system architecture, this disclosure describes how to clean up data, identify salient features, model predictive classifiers, and automate analysis and selection for content delivery. Data

collection and processing have a great influence on the accuracy and applicability of the model. Four kinds of behavioral parameters (or more) may be used to predict conversion events, with PCA used to reduce the dimensions utilized as inputs to the model. In addition, by adjusting the threshold of the predicted conversion probability, a trade-off can be made between accuracy and breadth, so that the prediction results of the model can be applied to different fields.