

CS583A: Course Project

Songgaojun Deng

May 12, 2019

1 Summary

We participate in an active competition of predicting total sales for every product and store in the next month. The final model we choose is gradient boosted tree, which produces a prediction model in the form of an ensemble of weak decision tree models. We directly use the XGBRegressor provided by XGBoost. XGBoost stands for eXtreme Gradient Boosting and is an implementation of gradient boosted decision trees designed for speed and performance. We run the code on a workstation with 1 NVIDIA GeForce 1080 GPUs. Performance is evaluated by root mean squared error (RMSE). In the public leaderboard, our score is 0.91718; we rank 738 among the 3085 teams. The result on the private leaderboard is not available until 8 months.

2 Problem Description

Problem. This is a regression problem based on time-series data. Given a challenging time-series dataset consisting of daily sales data, participants are supposed to predict total sales for every product and store in the next month. The competition is at <https://www.kaggle.com/c/competitive-data-science-predict-future-sales/>.

Data. There are 22170 products, 60 shops, 84 categories in total. One kind of product could be appeared in multiple shops and the products of a shop may change over time.

For each product (in different shops), the training data report the unit price in all history days, as well as the number of products sold. In training data, the daily sale of products are given from January 2013 to October 2015. In testing data, participants need to forecast the sales for these shops and products for November 2015.

Challenges. Participants are provided with daily historical sales data, while the task is to forecast the total amount of products sold in every shop for the test set. Also, note that the list of shops and products slightly changes every month. Therefore, designing a robust model that can handle such situations is part of the challenge.

3 Solution

Model. The model we finally choose is the XGBRegressor (XGBoost [1] regression), an implementation of efficient gradient boosted decision trees. A description of XGBoost is online:

<https://xgboost.readthedocs.io/en/latest/>.

Implementation.

- Feature Engineering

1. First, we remove outliers by plotting the distributions of *item_price* and *item_cnt_day*.
2. Then we extract city/type/subtype feature by examining *shop_name* and *item_category_id*.
3. Next, we extract lag features over different combinations of shop, item, type and date. A lag refers to a difference in time between an observation and a previous observation. The opposite of lag is lead. In the feature engineering part, we set the lag value to be 1 2 3 6 12 (month).
4. At last, we extract some price trend, date and other hand-crafted features.

- Training

For training, we directly use the XGBoost, which is an open-source software library which provides a gradient boosting framework for C++, Java, Python, R, and Julia. It works on Linux, Windows, and macOS.

We use the first 32 months data as training data and use the 33rd month data for validation while training. We train our model until validation RMSE hasn't improved in 10 rounds.

We use grid search to find the best combination of hyper-parameters, and select best features by removing some of them and then doing the training.

- Testing

We test our model on the given testing data.

Our code is available at <https://github.com/amy-deng/DeepLearning-HWs/tree/master/Project>. We run the code on a workstation with 1 NVIDIA GeForce 1080 GPUs. Training the model takes less than an hour.

Settings. The loss function is root mean squared error (RMSE). Maximum tree depth for base learners is 8. The min child weight is 300. Number of trees to fit is set to be 1000. Subsample ratio of the training instance (*subsample*) and subsample ratio of columns when constructing each tree (*colsample_bytree*) are 0.8. The learning rate is 0.1.

Cross-validation. We tune the parameters using fixed validation set. Figure 1 plots the convergence curves on 32/33 training data and 1/33 validation data.

4 Compared Methods

We compare our model with several baseline models:

Simple RNN We use time-series features in recurrent neural network models. We collect all the price and the number of items sold in the consecutive historical months as the features. We use two RNN layers in this experiment. We split the data into 0.85 training and 0.15 validation sets.

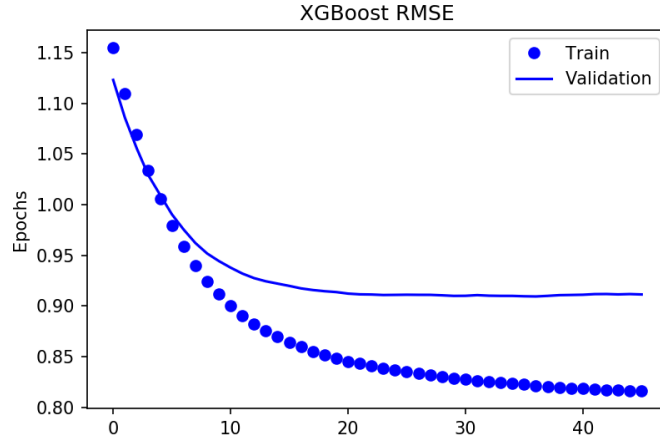


Figure 1: The convergence curves.

LSTM In this baseline, we simply replace the simple RNN module with CuDNNLSTM and follow the same design as in Simple RNN. We implement RNN baselines using Keras.

Linear Regression Linear regression is a very basic machine learning model. We use the same features as our advanced XGBoost model. We use Sklearn to implement it and we do not use validation set here.

Table 1 reports the performance of different model in terms of RMSE on testing set.

Model	RMSE
Simple RNN	1.04695
LSTM	1.02099
Linear Regression	1.00917
XGBoost	0.91718

Table 1: Comparison of model performance.

5 Outcome

We participated in an active competition. Our score is 0.91718 in the public leaderboard. We rank 738/3086 in the public leaderboard. The result on the private leaderboard is not available until 8 months. The screenshots are in Figure 2.

References

- [1] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM, 2016.












728	Zhe LIN		0.91682	1	2mo
729	Eric Wang		0.91682	1	17d
730	ampeeg		0.91682	1	9d
731	ZoanneW		0.91682	2	9h
732	dangdang		0.91684	9	5mo
733	Aleksandr Avdyushenko		0.91689	8	16d
734	Qian Wang		0.91711	4	5mo
735	Jing		0.91712	33	3mo
736	wuwuwu		0.91716	10	10mo
737	jedison		0.91717	1	3mo
738	Amy Deng		0.91718	17	12m

Figure 2: Our ranking in the leaderboard.