Machine Learning Engineer Nanodegree – Project Report

Title:

Mercari Price Suggestion

Author:

Zhipeng Lei, zhipeng.lei@outlook.com

I. Definition

Project Overview

This project is initiated from Kaggle's ongoing competition, named "Mercari Price Suggestion Challenge" (Kaggle, 2018). Mercari is Japan's biggest community-powered shopping application, which is organizing a marketplace where sellers are putting their products to sell. As tons of products are sold online, it is critical for sellers to decide their products' prices, which should be reasonably low to attract customers to buy but be high enough to have profits. One major service provided by Mercari is to offer price suggestions to the sellers. This is challenging because many factors can affect product's prices.

Product's categories would be likely to impact the prices. For example, a smartphone is more expensive than a pen. Another factor is the brand names, as for the same kind of products customers are willing to pay more if a product has a more reputable brand name. Additionally, selling seasons influence clothing prices actively, while product specs affect IT product prices. There is a need to develop an algorithm to automatically suggests the product prices that fit the products' real values.

In the field of business, literature has investigated price dispersion (e.g., Clemons et al., 2002), defined as price dispersion is variation in prices across sellers of the same item, holding fixed the item's characteristics. Degeratu el al. (2000) found that the price dispersion and the brand name influenced customer online purchase behaviors. Li et al. (2012) proposed a log-linear model for book price, in which the features used for prediction the price include book age, retailer, and paperback option but not include text information such as book name, description, or editor's reviews. Wang et al. (2012) applied text mining on customer reviews of online mobile applications to develop the correction between the text information and the application download number.

Problem Statement

This project seeks to solve the problem of building an algorithm for an electronic commerce company to suggest the right product prices based on the information provided by the sellers. The algorithm attempts to quantify the relationship between the product prices and the product details. To train and test the algorithm, I use Mercari's dataset in which product details, like product category name, brand name, item condition, shipping, and user-inputted text descriptions of their products, as well as product prices, are provided. As the product details have the user-inputted text descriptions and the product names, I will also conduct text mining on the text information to derive high-quality information. The accuracy of the algorithm is to be evaluated by calculating the error between the prediction of price and the actual sale price.

The workflow of the algorithm is composed of preprocessing the raw inputs, training a supervised learning model, and evaluating the learning model. The advanced text mining methods, LDA and Word Embedding, are used to process the texts from the item descriptions. The deep learning is applied as the supervised learning model for determining how the independent variables affect the dependent variable.

Metrics

I will use Root Mean Squared Logarithmic Error (RMSLE) as the evaluation metrics, which is also used in Kaggle's Mercari price suggestion challenge. The RMSLE is evaluated with the equation:

$$\varepsilon = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\ln(p_i+1)-\ln(a_i+1))^2}$$

where ε is the RMSLE value, n is the total number of products in the dataset, p_i is the prediction of price, and a_i is the actual sale price for a product i. The target for the algorithm is to obtain a minimum RMSLE value.

In the equation, both the predicting price and the actual sale price are log-transformed, because, as shown in the Data Exploration section, the distribution of the actual price is not normally distributed. After being log-transformed, the predicting price and the actual price are approximately normally distributed, so that their difference (error) is also approximately normally distributed and can be evaluated by least square function. During the log-transformation, the price is added by one to

avoid negative infinity at zero.

II. Analysis

Data Exploration

I use the dataset from Kaggle's Mercari Price Suggestion Challenge for predicting the sale price of a product based on information a user provides for this product. In this competition, although both training and testing datasets are provided, the testing dataset does not contain the actual product prices as the Kaggle competition is in progress. Only the training dataset is used in this project. The original training dataset has 1048575 products, which remarkably large. The text mining techniques that I will use is computationally expensive and time-consuming. Hence, I only use 10% of the original training dataset, i.e., 100001 products, which is still large. The dataset is then randomly divided in to test dataset and training dataset with ration 1:9.

Table 1 lists the dataset's variables, in which the price is the dependent variable, and others are the independent variables. The average of the price is 26.74. The price distribution will be discussed later. The item condition id is in the range of 1 to 5, and the average of the item condition id is 1.9. Most item condition ids are 1, 2, or 3. The shipping has the value of 0 and 1; it is 1 if shipping fee is paid by seller and 0 by the buyer. The average value of the shipping is 0.45, meaning that there are more cases the buyers pay the shipping fee. About The name, category name, brand name, and item description are input as character strings. The number of different brand names is 2067; the number of different categories is 971.

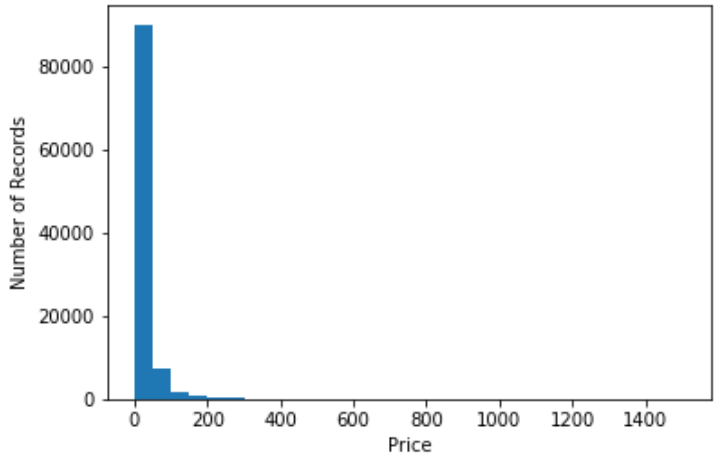
Table 1: the dataset's variables

Variables	**Description**
Name	The title of the item.
Item condition id	The condition of the item provided by the seller.
Category name	Category of the item.
Brand name	Brand.
Price	The price that the item was sold for, which has unit USD and is the target variable that you will predict.
Shipping	1 if shipping fee is paid by seller and 0 by the buyer.
Item description	The full description of the item.

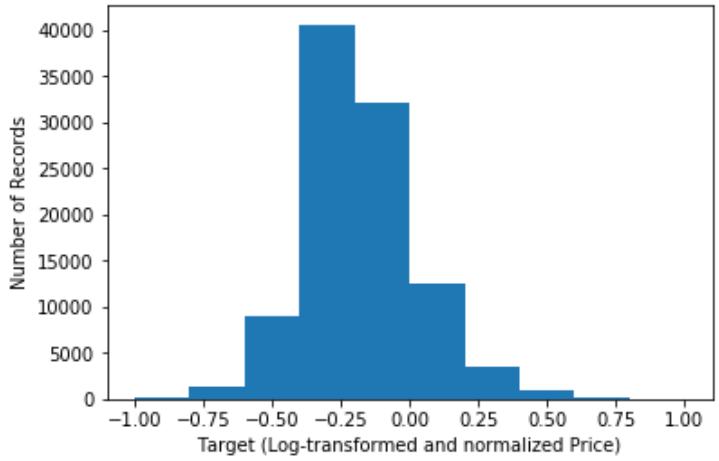
Exploratory Visualization

As shown in Figure 1(a), the distribution of prices is highly skewed to the left, where products' prices are mostly in the range of 0-100, and the highest price is 2009.

Hence, a log-transform of the price variable is necessary. Figure 1(b) shows the distribution of the price after being log-transformed and normalized, which is close to normal distribution.



(a)



(b)

Figure 1: (a) Distribution of the price; (b) distribution of the price after log-transformed

The variables of the name and the item description are considered as the text information. Figure 2 shows Distribution of length of the item description. The maximum and minimum lengths of the item description are 269 and 0; the maximum and minimum lengths of the name are 17 and 0. The average lengths of the item description and the name are 25.8 and 4.4. Thus, these two text variables are relatively short.

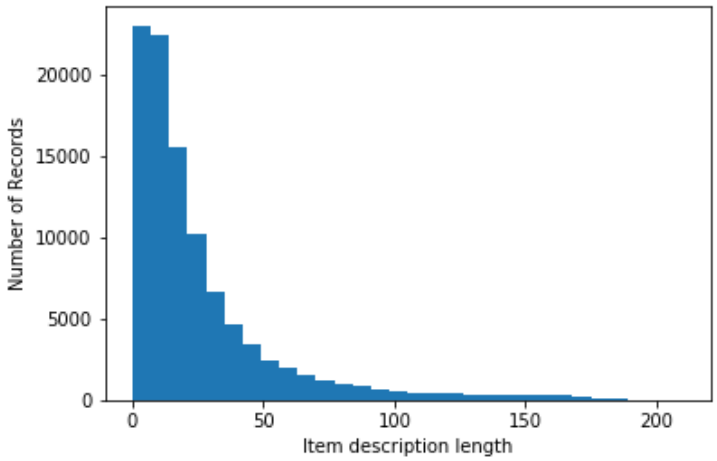
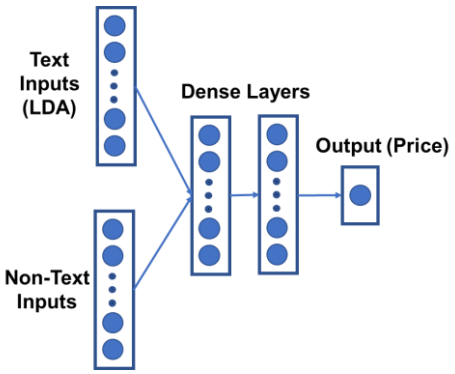


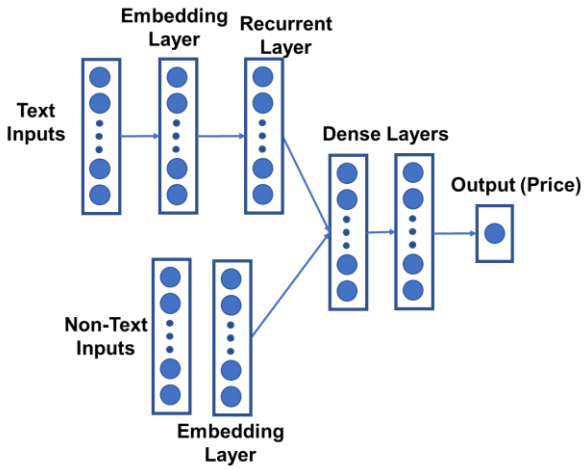
Figure 2: Distribution of length of item description

Algorithms and Techniques

I propose to use the deep neural network (DNN) in the algorithm of predicting prices with inputting variables including item condition, category name, brand name, shipping, and item description. As a kind of artificial neural networks, DNN consists of multiple hidden layers between the input and output layers (LeCun et al., 2015). However, the item descriptions are text files and cannot be directly used as an input variable. To address this, I propose two approaches, the schematics of which are shown in Figure 3. Approach 1 uses Latent Dirichlet allocation (LDA), a text mining technique, to discover topics in raw texts (Liu et al., 2016). For each text, the LDA method calculates topic scores, which are combined with non-text features as the inputs for the DNN schematic. Approach 2 uses Word Embedding method, which is a class of approaches for representing words and documents using dense vectors by projecting word into a vector space. Both the text features and non-text features are processed with an embedding layer, but the text features are further processed with a recurrent layer. Dense layers are added as the hidden layers in the two approaches.



(a)



(b)

Figure 3: Schematic of DNN: (a) Approach 1 with LDA, and (b) Approach 2 with Word Embedding.

Benchmark

Like my proposed model, the benchmark model has two parts. The first part is to transfer an item description to a term-document matrix without further applying advanced text mining method such as LDA. In the term-document matrix, each value is the counts of a term in a document. The second part is to use linear regression to fit independent variables, including the term-document matrix, to the dependent variable, the price. The benchmark model is simple but widely used. Because the independent variables have more than 1000 dimension, the convergence of the linear regression model is very slow. I use the Lasso linear regression model instead, which is trained with L1 prior as a regularizer. The RMSLE error of the benchmark model on test dataset is 0.755.

III. Methodology

The whole project is to develop an algorithm to predict the price for online selling products. I create Jupyter Notebook documents for interactive computing and demonstrating the algorithm. The programing language is Python version 3.5; additional packages are imported, including Numpy, Pandas, Matplotlib, and Sk-learn for necessary data analysis, Gensim for text mining, and Keras and Tenserflow for Deep Leaning. Note that the Deep Learning tool, Tenserflow, is equipped with the capability of GPU acceleration.

Data Preprocessing

The Python program first reads the raw dataset. The dependent variable, price, is logged and normalized as shown in Figure 1(b). The independent variables, having missing values, categorical labels, and texts, should be appropriately preprocessed. The program searches and marks the missing values. Because there are 2067 different brand names and 971 different item categories, the appears of brand names and item categories are counted, and they are cut through only maintaining the ones with a high number of appearances. The variables of the category name, brand name, and item condition id are transferred to categorical data.

In Approach 1, the brand name variable is binarized labels in a one-vs-all fashion, the category name variable is transferred to a matrix of token (word) counts, and the item condition id and shipping are transferred to dummy variables. In Approach 2, the brand name and category name are encoded as labels with a value between 0 and number of classes minus 1.

Implementation

I design a DNN architecture consisting of multiple neural network layers between the inputs (including text inputs and non-text input) and output (price). Because of the existing of text input, two advanced text mining methods are used.

As shown in Figure 2 (a), in Approach 1 with LDA, the text information (the product name and item descriptions) is tokenized, and common words are removed (using a stoplist) as well as low-frequency words (i.e., words that appear less than five times in the texts). Also, the words are processed by stemming, the process of reducing derived words to their word stem, base or root form. Then, a term-document matrix is generated to describe the counts of terms that occur in the documents. LDA is used to determine the topics in the whole texts and topic distributions for each text (Hoffman et al., 2010).

As shown in Figure 2 (b), in Approach 2 with Word Embedding, the text information is also tokenized and encoded, but each token (word) is transferred to a dense vector through an embedding layer. Inside the DNN architecture, Recurrent neural networks (RNNs), in which information can flow in forwarding and backward directions, are used, because it has shown power in applications of texting mining (Gers and Schmidhuber, 2001). The dense vectors initiated from the text inputs are fed to an RNN layer, which can be either a Long-Short Term Memory layer or a Gated Recurrent Unit layer to overcome the problem of gradient vanishing or exploding. The non-text inputs also go through an embedding layer and combined with the outcomes

of the RNN layer for the text inputs.

Three densely-connected neural network layers, which are applied with Dropout approach for preventing overfitting, is inserted before the output layer. The DNN architecture is fitted using training dataset. Finally, the fitted model is used to calculate the prediction of price for the products in the testing dataset and then is evaluated by comparing with the benchmark model that uses a linear regression model.

The coding complications I faced in this implementation process mainly come from writing the interface between the Gensim's LDA model and the Keras's deep learning model. The matrix for the topic features from the LDA model has a large size and needs to be combined with matrices of other input features, which also have a large size. The trick is that I used functions of csr_matrix and hstac from scipy.sparse toolbox to merge them.

Refinement

In the initial results of the two approaches, the RMSLE errors on test dataset are around 0.7, which is close to the benchmark error, about 0.75. The hyperparameters that can be tuned are the number of LDA topics, the number of units in each neural layer, the optimizer, the batch size, and the number of epochs. These parameters are adjusting to improve the solution. For example, Figure 4 shows the plot of train and validation errors, in which the overfitting occurs as the number of epochs increases. Further, a dropout layer, which randomly drops weights during training and scales the weights, is added for reducing overfitting. The final models of Approach 1 and Approach 2 have RMSLE errors 0.54 and 0.55.

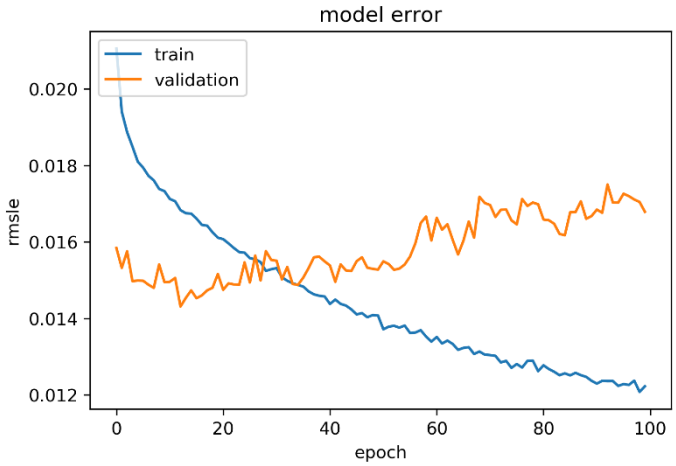


Figure 4: Plot of train and validation errors.

IV. Results

Model Evaluation and Validation

A validation dataset is used to evaluate the two approaches, the schemes of which are present in Figure 3. I select the final architectures and hyperparameters as they have the lowest validation errors among the different combinations.

In Approach 1, I generate 100 LDA topics is 100. Table 1 list five of 100 topics. From the weights and terms in each topic, the semantic meaning can be found. For example, the topic 20 is related to the bra. Approach 1 has two dense layers with 128 and 64 units, respectively, and dropout layers with 0.1 rate. The optimizer is "rmsprop". The batch size is 3000; the epochs number is 5. The RMSLE error on validation and test datasets are 0.53 and 0.54, respectively.

Table 1: five of 100 topics, displaying as weights and terms.

Topic number	Weights and terms
7	'0.326*"face" + 0.132*"north" + 0.098*"wedg" + 0.074*"candi" + 0.037*"rue" + 0.037*"bronzer" + 0.036*"beig" + 0.033*"clay" + 0.031*"round" + 0.029*"shimmer"
57	'0.183*"sexi" + 0.099*"big" + 0.091*"coutur" + 0.089*"juici" + 0.079*"lingeri" + 0.076*"sweet" + 0.068*"parti" + 0.052*"fox" + 0.052*"strip" + 0.043*"wed"
20	'0.468*"bra" + 0.073*"push" + 0.053*"bow" + 0.048*"pad" + 0.047*"dot" + 0.047*"sequin" + 0.040*"strapless" + 0.032*"size" + 0.031*"clip" + 0.027*"inch"
46	'0.244*"sleev" + 0.243*"long" + 0.082*"mous" + 0.063*"figur" + 0.058*"summer" + 0.049*"maroon" + 0.036*"trainer" + 0.026*"shower" + 0.025*"cold" + 0.024*"foam"
32	'0.321*"slime" + 0.104*"coral" + 0.081*"buy" + 0.080*"state" + 0.064*"stamp" + 0.041*"singl" + 0.039*"sticki" + 0.037*"random" + 0.034*"defin" + 0.032*"leav"

In Approach 2, I apply embedding layers for the name, item description, brand name, category name and item condition. The output unit numbers of these embedding layers are 50, 50, 10, 10, and 5, respectively. The RNN layers for the item description and name are 16 and 8. Approach 1 has two dense layers with 128 and 64 units, respectively, and dropout layers with 0.1 rate. The optimizer is "adam". The batch size is 5000; the epochs number is 5. The RMSLE error on validation and test datasets are 0.56 and 0.58, respectively.

From the RMSLE error, Approach 1 is slightly better than Approach 2. However, the LDA technique in Approach is time-consuming, as it is not supported by GPU acceleration.

For verifying the robustness of the two approaches, I collect another 10% of the

Kaggle's original training dataset, i.e., 100001 observations. After applying the final models of the two approaches, the RMSLE errors on test dataset are 0.57 and 0.53, respectively, showing consistency of accuracy with different datasets. Thus, I can conclude that the final models are not sensitive to small changes in the data, and we can trust the models.

Justification

The same test dataset is used to evaluate the benchmark model, Approach 1 model, and Approach 2 model. The latter two have the significant lower RMSLE errors (0.54 and 0.58) than the one of benchmark model (0.75). I can conclude that the final solution is significant enough for solving the problem of Mercari price suggestion.

V. Conclusion

Free-Form Visualization

The important quality of this project of predicting prices is that there are only six independent variables. For better prediction, the text variables (the item description and name) have to be used in the price prediction. Figure 5 plots the word cloud of the item descriptions, in which terms that frequently show in the texts have larger fonts than other terms do. The terms, "brand new", "never used", "free shipping", "great condition", etc., are frequently occur in the item description and very likely related to the price. This project demonstrates two methods of text mining, LDA and word-embedding. Both these two methods confirm power in price prediction. The LDA has the advantage of providing topics that consist of terms and can be interpreted, while the word-embedding has the advantage of speed because of GPU acceleration (the running time for the LDA approach is about 1 hour and the one for the word-embedding approach is ten minutes).

Figure 5: Word cloud of item descriptions.

Reflection

In this project, I develop two algorithms to predict prices in an online market of Japan. Using advanced text mining techniques, the good accuracy is achieved in both algorithms. For the extension of the algorithm usage, we need to explore the business insight, such as how the features of text information affect the price so that sellers in the online market can better input information to increase their products' sells as well as their profits. The LDA method can discover the topics inside texts, but the word-embedding method is incapable. Another concern is that the deep learning architecture used in this project cannot display the significance of each input feature. This weakens the deep learning capabilities in business analysis. For feature significance, we may have to switch back to the linear regression model or the boosting tree model.

The most interesting aspect of this project is to design and train a deep learning model to predict a continuous dependent variable that is price in this project. Deep learning method can overcome the curse of dimensionality and can generation accurate predictions. The biggest challenge is how to transfer raw inputs to independent variables that can be imported into a deep learning architecture. In this project, I explore various data processing techniques, especially text mining methods. Through the project, I have learned that the data in most real world applications is far from perfectly clean, but there should always be techniques, existed in Python community, to process the data.

Improvement

In the ongoing Kaggle's Mercari competition, the top teams can reach an RMSLE error below 0.45, lower than my models' results. The reason for the gap may come

from the fact that I only use 10% of the dataset from Kaggle. However, I am one of the first to adopt LDA topic model in the Mercari competition. In future research, I will improve the models including more dataset. For increasing the model's accuracy, I will try other deep learning techniques, such as Bidirectional RNN, stack RNN, and Batch Normalization.

Reference:

Clemons, E. K., Hann, I. H., & Hitt, L. M. (2002). Price dispersion and differentiation in online travel: An empirical investigation. Management Science, 48(4), 534-549.

Degeratu, A. M., Rangaswamy, A., & Wu, J. (2000). Consumer choice behavior in online and traditional supermarkets: The effects of brand name, price, and other search attributes. International Journal of Research in Marketing, 17(1), 55-78.

Gers, F. A., & Schmidhuber, E. (2001). LSTM recurrent networks learn simple context-free and context-sensitive languages. IEEE Transactions on Neural Networks, 12(6), 1333-1340.

Hoffman, M., Bach, F. R., & Blei, D. M. (2010). Online learning for latent dirichlet allocation. In Advances in Neural Information Processing Systems (pp. 856-864).

Kaggle (2018). Mercari Price Suggestion Challenge. https://www.kaggle.com/c/mercari-price-suggestion-challenge

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.

Li, X., Gu, B., & Liu, H. (2013). Price dispersion and loss-leader pricing: Evidence from the online book industry. Management Science, 59(6), 1290-1308.

Liu, L., Tang, L., Dong, W., Yao, S., & Zhou, W. (2016). An overview of topic modeling and its current applications in bioinformatics. SpringerPlus, 5(1), 1608.

Wang, Y., Aguirre-Urreta, M., & Song, J. (2016). Investigating the Value of Information in Mobile Commerce: A Text Mining Approach. Asia Pacific Journal of Information Systems, 26(4), 577-592.