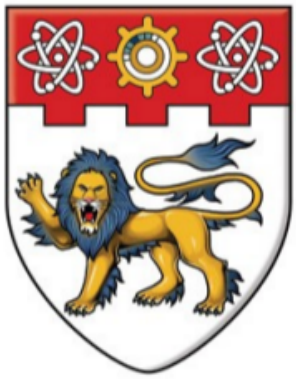

NANYANG TECHNOLOGICAL UNIVERSITY

CZ4041 – MACHINE LEARNING

PETER
ZILLION GOVIN

SCHOOL OF COMPUTER ENGINEERING
2016/17

Contents

List of Tables

List of Figures

1. Introduction

CZ4041 - Machine Learning project aims to expose students to a real life application of machine learning techniques. The main scope of this project is to explore, analyze, and discuss the machine learning methods used by our team in Kaggle Data Science Competition - Rossmann Store Sales.

1.1 Team Members

The team consists of four people:

- Benedita Tanabi
- Peter
- Stefan Artaputra Indriawan
- Zillion Govin

Each group member is in charge of exploring various machine learning techniques to be used in the competition.

1.2 Problem Statement

1.2.1 Challenge

The challenge for Rossmann Store Sales competition is to forecast 6 weeks of daily sales of 1,115 Rossmann stores located across Germany based on the historical data of the stores.

1.2.2 Data

To assist with the challenge, Rossmann has also provided 4 types of comma-separated-value (.csv) files, such as *train.csv, test.csv, store.csv, sample_submission.csv.* However, only *train.csv, test.csv, store.csv* are used in this project.

2. Preprocessing

The problem contains three files, *train.csv*, *test.csv*, and *store.csv*. The file *train.csv* contains historical data including sales for 1115 stores everyday from 1 Jan 2013 to 31 July 2015. While *store.csv* contains supplemental information about each store. Then, *test.csv* contains historical data excluding sales and number of customers everyday from 1 Aug 2015 to 17 Sept 2015.

Each row in *train.csv* contains store ID, day of week, date, number of customers, sales, whether the store is open, whether the store is doing a promo, state holiday, and whether that day is a school holiday. Columns in *test.csv* is almost identical to those of *train.csv*, except that sales and number of customers are unknown in *test.csv*. Each row in *test.csv* contains a submission ID for the purpose of evaluation on prediction result. On the other hand, each row in *store.csv* contains the details of a store, such as store type, assortment type, whether the store has competition, since when the competition exists, competition distance, whether the store is doing *promo2*, and *promo2* period.

Initially, our preprocessing method merges *train.csv* and *store.csv* by store ID. The columns *DayOfWeek* and *StateHoliday* in *train.csv* are transformed into one-hot vector respectively. The columns *StoreType* and *Assortment* in *store.csv* are transformed into one-hot vector respectively as well. The columns *CompetitionOpenSinceMonth* and *CompetitionOpenSinceYear* are substituted into a single column *HasCompetition* which depends on its respective *Date* column. The same thing also applies to the columns *Promo2SinceWeek*, *Promo2SinceYear*, and *PromoInterval*, they are substituted into a single column *IsDoingPromo2* which depends in its respective *Date* column. *CompetitionDistance* is set to the maximum value in the training set if a store does not have a competition in any given date. All store data are retained even when a store is closed on a particular date.

After *train.csv* and *store.csv* has been merged, we proceed to merge *test.csv* and

store.csv by store ID as well. The preprocessing method in this step is identical in the previous paragraph. The difference is that sales and the number of customers are unknown in the test dataset.

3. Experiments

Lorem ipsum dolor sit amet.

3.1 LSTM

LSTM (abbreviation for Long-short Term Memory) is a variant of recurrent neural network. LSTM seeks to solve vanishing gradient and exploding gradient problem in vanilla recurrent neural network. These problems happen when a training sequence is too long. LSTM introduces gating concept in recurrent neural network. An LSTM contains three gates, an input gate, an output gate, and a forget gate. A forget gate determines when to reset the content of the cell memory. An input and output gate control the flows of input and output of the LSTM respectively.

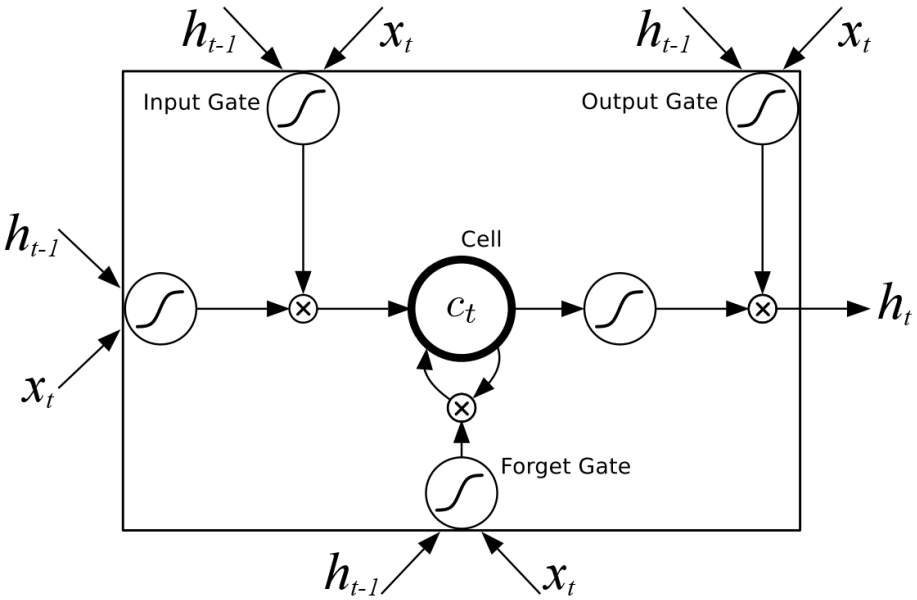

Figure 3.1: An LSTM unit

We use LSTM unit provided by Caffe[1] framework. It requires CUDA 8 and

[1]Caffe is developed by Berkeley Vision and Learning Center (BVLC)

cuDNN v5.1 from Nvidia, which enable Caffe framework to utilize Nvidia GPU to train a neural network. We used Nvidia GTX 850m to train our LSTM model. This GPU is pretty decent to train a medium-sized network.

The main idea of this approach is to train each store separately. Training an LSTM model for the entire training set is infeasible as the network model required would be large and our GPU is not powerful enough for it. Since there are 1115 stores, we are going to train 1115 small LSTM models for each store.

There are few preprocessing steps that have to be taken before training an LSTM model. First, we need to split the training dataset by store ID since we are training one model for one store. Then, we need to drop *Customers* column as this column is not available in the test dataset. Next, we have to drop *StoreType*, *Assortment*, *HasCompetition*, and *CompetitionDistance* as we are going to train an LSTM model for each store. Finally, normalize the date to $[0, 1]$, where 0.0 indicates 1 Jan 2013, and 1.0 indicates 17 Sept 2015.

The network architecture consists of, in this order, an input layer with 15 features, an LSTM unit, a fully-connected layer, and an output layer. The LSTM unit contains 50 hidden units, and the fully-connected layer contains 50 neurons. The training algorithm used is RMSProp with decaying learning rate. The learning rate update equation is as follows.

$$\alpha_{t+1} = \frac{\alpha_t}{1 + \gamma t}, \text{ where } \gamma \text{ is the decay rate.} \tag{3.1}$$

Base learning rate α_0 is 0.001, learning rate decay γ is 10^{-6}, and RMS decay is 0.9. The LSTM model is trained for 200 epochs, and the last 50 days of training data of each store are used as validation. At the end of the training, we have 1115 LSTM models. The training duration is approximately four hours.

The performance result by this model according to Kaggle, is shown in the table below.

Table 3.1: LSTM per store result

Private score	0.15578
Public score	0.15904

3.2 Neural Network

Our neural network model requires Caffe framework as well, along with CUDA 8 and cuDNN v5.1. This model is also trained on the same Nvidia GTX 850m as before.

There are few preprocessing steps that are specific to this approach. First, we add four extra features, maximum sales, average sales, maximum customers, and average customers. Note that maximum sales are only specific to each store, i.e., maximum sales is not a global max, instead it is maximum sales of a store in the training dataset. The same thing applies for average sales, maximum customers, and average customers. Then, we drop *Customers* column as it is not available in the training dataset. Finally, normalize non-boolean features, such as maximum sales, average sales, maximum customers, average customers, and competition distance.

Our neural network model consists of, in this particular order, an input layer, an embedding layer, two fully-connected layers, and an output layer. The first fully connected layer has 128 neurons, and the second fully connected layer has 64 neurons. Not all features in the input layer are embedded. Features that are embedded include *StoreID*, *DayOfWeek*, *Day*, *Month*, *Year*, *StateHoliday*, *StoreType*, and *Assortment*.

The training algorithm used is Nesterov's accelerated gradient descent with decaying learning rate. The learning rate decay equation is identical to equation 3.1. The momentum is set to 0.9, and base learning rate is 0.01. The model is trained for 35 epochs with 10000 random training samples set aside for validation.

The performance result by this model according to Kaggle, is shown in the table below.

Table 3.2: Neural network with embedding layer's result

Private score	0.20356
Public score	0.20298

3.3 Lasso Regression

Besides neural network, we explored linear regression methods to solve the given challenge as well.

3.3.1 Background

Lasso regression is an extension of Ordinary Least Square (OLS) regression. The most notable difference between Lasso and OLS is the existence of L1 regularization term in Lasso.

$$\lambda \sum_{i=0}^{N} |w_i| \tag{3.2}$$

The L1 regularization term is defined as the summation of the absolute of the weight for each features, multiplied by the penalty weight of λ. The L1 regularization term is included to the error term in OLS regression. The objective of defining a regularization term is to constraint the value of w in order to avoid overfitting the model

3.3.2 Implementation

The implementation of Lasso regression model is done using Scikit Learn Library in Python, along with supporting libraries such as Pandas and Numpy.

3.3.3 Model Training & Testing

During the training and testing phase, the Lasso model was trained with different training methods, and the difference was observed based on the model's performance in the competition leaderboard.

Merged and Normalized Train Test

This phase trains the Lasso model using the preprocessed training and testing data in chapter 2 of this report, where *StateHoliday, Storetype, Assortment* are all converted into one hot encoding.
Additionally, the *Sales* column is normalized (using global mean and standard deviation of *Sales*) during training and denormalized during testing.
The performance of this training type is measured with RMSPE according to Kaggle, as shown in table 3.3

Table 3.3: Merged Normalized Lasso Result

Private score	0.24082
Public score	0.22302

Train Test per Store

The data used in this phase is the merged training and testing data from Section 2. In addition, the *Sales* data is normalized with regards to the mean and standard deviation of each store. Furthermore, a unique predictive model is defined for each store ID.
The result of this model can be observed in table 3.4.

Table 3.4: Lasso Result per Store

Private score	0.23920
Public score	0.23479

K-Fold Validation

In addition to training and testing per store, this step performs a 5-fold validation on each predictive model during the training phase. Using the model with the best validation score, the model is then used to predict the sales of the respective store.

Table 3.5: 5-Fold Validation Lasso Result

Private score	0.25260
Public score	0.25426

4. Results

Lorem ipsum dolor sit amet.

5. Conclusion

Lorem ipsum dolor sit amet.