Kaggle-Challenge: San Francisco Crime Classification

Raphael Emberger

January 7, 2019

Date: January 7, 2019
Instructor: Professor Yukawa

Contents

1 Preface

Firstly, I want to express my gratitude to Professor Yukawa for guiding me in this project and to the Kokusaika staff members to arrange my stay here at the Nagaoka University of Technology(subsequently referred to as ”NUT”). I was given the generous opportunity to study at the NUT for one semester, for which I am very grateful. During that time I could choose from the following six Kaggle challenges to work on as project work:

- Toxic Comment Classification Challenge (Kaggle 2017*b*)
- TalkingData AdTracking Fraud Detection Challenge (Kaggle 2018*b*)
- Quora Question Pairs (Kaggle 2017*a*)
- Expedia Hotel Recommendations (Kaggle 2016)
- San Francisco Crime Classification (Kaggle 2015)
- Inclusive Images Challenge (Kaggle 2018*a*)

Of those, I was most interested in the classification of reported crimes (Kaggle 2015), as in my opinion this was an interesting challenge, given the dataset to be only consisting of time and spatial data. As such, this report is dedicated to take on this challenge.

2 Abstract

3 Introduction

3.1 Initial situation

The challenge has been out since roughly 3 years and since then, many teams have participated and submitted their results. This lead the leader-board to fill up with 2335 submissions which were ranked and their results displayed online(see ”Leaderboard” at Kaggle (2015)). The results vary from 34.53877 up to 1.95936, where the sample submission with a value of 32.89183 reaches rank 2241(see 4.1 for the ranking principle).

When searching on the internet for documents about that challenge, there are multiple such projects to be found. For example:

- A paper from Darekar et al. (2016). 2 Naïve Bayes, Decision Tree, Random Forest and Support Vector Machines classifiers were used. Reached highest accuracy of 23.16% with a Decision Tree.

- A blog post from Ramunno-Johnson (2015). In that project, a Bernoulli Naïves Bayes classifier was used. Reached a log-loss score of 2.58.

- A blog post from Murray (n.d.). AdaBoost, Bagging, Extra Trees, Gradient Boosting, K-Nearest Neighbors, Random Forest classifiers and Logistic Regresson were used in this project. The dataset was enriched greatly by adding 9 other datasets(features like house prices, income, police and public transportation stations, healthcare center and homeless shelter locations, altitudes). Highest accuracy achieved with Gradient Boosted Trees, resulting in 45.7%.

3.2 Objective

The objective of this project is to produce a system that is capable of classifiying the type of crime based off of the provided data consisting of date time stamps, the name of the district and street as well as the coordinates of the registered report. To quote Kaggle (2015):

> From 1934 to 1963, San Francisco was infamous for housing some of the world's most notorious criminals on the inescapable island of Alcatraz.
>
> Today, the city is known more for its tech scene than its criminal past. But, with rising wealth inequality, housing shortages, and a proliferation of expensive digital toys riding BART to work, there is no scarcity of crime in the city by the bay.
>
> From Sunset to SOMA, and Marina to Excelsior, this competition's dataset provides nearly 12 years of crime reports from across all of San Francisco's neighborhoods. Given time and location, you must predict the category of crime that occurred.
>
> We're also encouraging you to explore the dataset visually. What can we learn about the city through visualizations like this Top Crimes Map? The top most up-voted scripts from this competition will receive official Kaggle swag as prizes.

Although the Kaggle challenge includes submitting an softmax array of the predictions of the test data, this objectives shifts towards self evaluation on the training set. The reason for this is that the challenge is already over and self evaluation was considered an easier approach to measure the success of the system.

4 Theoretical Principles

4.1 Loss Function

The ranking of the results on the Kaggle leader board are based on the multi-class logarithmic loss function:

$$loss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} \log\left(p_{ij}\right) \tag{I}$$

N : Number of cases in dataset.

M : Number of classes.

y_{ij} : Label for class. 1 if i is in j. Otherwise 0.

p_{ij} : Predicted probability that i belongs to j.

This basically boils down to a format as follows:

Class 1	Class 2	Class 3
0.24	0.48	0.38

With the labels being:

Class 1	Class 2	Class 3
0.00	1.00	0.00

When those values are applied to I, we get a value of 0.49548. Of course, the closer the prediction is to the actual labels, the smaller the loss value will be.

To calculate examples quickly on the python console, the following code can be used:

```python
import numpy as np
from sklearn.metrics import log_loss
labels = np.array([0.0, 1.0, 0.0])
prediction = np.array([0.04, 0.78, 0.18])
print(log_loss(labels, prediction))
```

Listing 1: Quick Log Loss Calculation

5 Methods

5.1 Dataset

The Kaggle challenge (Kaggle 2015) provides 3 files on their site under ”Data”:

- **sampleSubmission.csv**(884k x 40): A sample file, demonstrating the expected format for submissions to the challenge. Consists of an array of the softmax prediction of each sample in the **test.csv**.

- **test.csv**(884k x 7): The unlabeled sample subset of the data.

- **train.csv**(878k x 9): The labeled sample subset of the data.

The data itself consists of the gathered crime reports of the San Francisco Police Department from January 1st 2003 through May 13th 2015, where the odd weeks belong to the **test.csv** and the even weeks to **train.csv**.

Here are the 10 first rows of the respective data files:

Id	*one column per class*
0	*zeros, except the second last columns being all ones*
1	⋮
2	⋮
⋮	⋮

Table 1: sampleSubmission.csv(first 10 rows)

The two datasets differ slightly in their columns. The training dataset has added the labels(Category) but also Descript and Resolution, which will be ignored for this project.

The labels consist of 39 classes of crimes:

The classes occur in an unbalanced matter: ”Larceny/Theft” is the most predominant recorded crime, taking up nearly 19.92% of the dataset. For this reason, 19.92% is considered the bottom line of accuracy.

Id	Dates	DayOfWeek	PdDistrict	Address	X	Y
0	2015-05-10 23:59:00	Sunday	BAYVIEW	2000 Block of THOMAS AV	-122.39958770419	37.7350510103906
1	2015-05-10 23:51:00	Sunday	BAYVIEW	3RD ST / REVERE AV	-122.391522893042	37.7324323864471
2	2015-05-10 23:50:00	Sunday	NORTHERN	2000 Block of GOUGH ST	-122.426001954961	37.7922124386284
3	2015-05-10 23:45:00	Sunday	INGLESIDE	4700 Block of MISSION ST	-122.437393972517	37.7214120621391
4	2015-05-10 23:45:00	Sunday	INGLESIDE	4700 Block of MISSION ST	-122.437393972517	37.7214120621391
5	2015-05-10 23:40:00	Sunday	TARAVAL	BROAD ST / CAPITOL AV	-122.459023622429	37.7131719025215
6	2015-05-10 23:30:00	Sunday	INGLESIDE	100 Block of CHENERY ST	-122.42561645123	37.7393505144628
7	2015-05-10 23:30:00	Sunday	INGLESIDE	200 Block of BANKS ST	-122.412652039792	37.7397501563121
8	2015-05-10 23:10:00	Sunday	MISSION	2900 Block of 16TH ST	-122.418700097043	37.7651649409646

Table 2: test.csv(first 10 rows)

Dates	Category	Descript	DayOfWeek	PdDistrict
2015-05-13 23:53:00	WARRANTS	WARRANT ARREST	Wednesday	NORTHERN
2015-05-13 23:53:00	OTHER OFFENSES	TRAFFIC VIOLATION ARREST	Wednesday	NORTHERN
2015-05-13 23:33:00	OTHER OFFENSES	TRAFFIC VIOLATION ARREST	Wednesday	NORTHERN
2015-05-13 23:30:00	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Wednesday	NORTHERN
2015-05-13 23:30:00	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Wednesday	PARK
2015-05-13 23:30:00	LARCENY/THEFT	GRAND THEFT FROM UNLOCKED AUTO	Wednesday	INGLESIDE
2015-05-13 23:30:00	VEHICLE THEFT	STOLEN AUTOMOBILE	Wednesday	INGLESIDE
2015-05-13 23:30:00	VEHICLE THEFT	STOLEN AUTOMOBILE	Wednesday	BAYVIEW
2015-05-13 23:00:00	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Wednesday	RICHMOND
2015-05-13 23:00:00	LARCENY/THEFT	GRAND THEFT FROM LOCKED AUTO	Wednesday	CENTRAL

Resolution	Address	X	Y
”ARREST, BOOKED”	OAK ST / LAGUNA ST	-122.425891675136	37.7745985956747
”ARREST, BOOKED”	OAK ST / LAGUNA ST	-122.425891675136	37.7745985956747
”ARREST, BOOKED”	VANNESS AV / GREENWICH ST	-122.42436302145	37.8004143219856
NONE	1500 Block of LOMBARD ST	-122.42699532676599	37.80087263276921
NONE	100 Block of BRODERICK ST	-122.438737622757	37.771541172057795
NONE	0 Block of TEDDY AV	-122.40325236121201	37.713430704116
NONE	AVALON AV / PERU AV	-122.423326976668	37.7251380403778
NONE	KIRKWOOD AV / DONAHUE ST	-122.371274317441	37.7275640719518
NONE	600 Block of 47TH AV	-122.508194031117	37.776601260681204
NONE	JEFFERSON ST / LEAVENWORTH ST	-122.419087676747	37.8078015516515

Table 3: train.csv(first 10 rows)

ARSON	ASSAULT	BAD CHECKS
BRIBERY	BURGLARY	DISORDERLY CONDUCT
DRIVING UNDER THE INFLUENCE	DRUG/NARCOTIC	DRUNKENNESS
EMBEZZLEMENT	EXTORTION	FAMILY OFFENSES
FORGERY/COUNTERFEITING	FRAUD	GAMBLING
KIDNAPPING	LARCENY/THEFT	LIQUOR LAWS
LOITERING	MISSING PERSON	NON-CRIMINAL
OTHER OFFENSES	PORNOGRAPHY/OBSCENE MAT	PROSTITUTION
RECOVERED VEHICLE	ROBBERY	RUNAWAY
SECONDARY CODES	SEX OFFENSES FORCIBLE	SEX OFFENSES NON FORCIBLE
STOLEN PROPERTY	SUICIDE	SUSPICIOUS OCC
TREA	TRESPASS	VANDALISM
VEHICLE THEFT	WARRANTS	WEAPON LAWS

Table 4: Crime classes

5.2 First Approach

For the first approach a Keras (n.d.) model on top of Tensorflow (n.d.) was chosen. For this, the first step was to pre-process the dataset to standardize it and properly feed it to the neural network.

5.2.1 Pre-Processing

To handle CSV files properly, a `class CsvFile` class was created that represents a single csv file. When instantiated, it loads the csv file as a Pandas `DataFrame`. Apart from an abstract `def parse(self)` method, it implements per data field methods that prepare the respective column for a conversion to a numerical representation(i.e. `def _prepare_date(self, date: datetime) -> datetime`). It also defines `def toNpArray(self) -> ndarray`, which allows to access the data as a `numpy` array.

From this basic class, three other classes were derived:

- `class TestDataCsvFile`
  This class represents the `test.csv` file. It implements the missing `parse(self)` method as follows:

```
def parse(self):
    self.df = self.df_orig.copy()
    self.log.debug('Parsing Dates')
    self._transform_date()
    self.log.debug('Parsing Day of the week')
    self.df['DayOfWeek'] = self.df['DayOfWeek'].apply(self._prepare_day)
    self.log.debug('Parsing District')
    self.df['PdDistrict'] = self.df['PdDistrict'].apply(self._prepare_district)
    self.log.debug('Parsing Address')
    self.df['Address'] = self.df['Address'].apply(self._prepare_address)
    self.log.debug('Parsing Longitude')
    self.df['X'] = self.df['X'].apply(self._prepare_longitude)
    self.log.debug('Parsing Latitude')
```

```
    self.df['Y'] = self.df['Y'].apply(self._prepare_latitude)
    self.log.info('Parsed dataframe')
```

Listing 2: Parse method if the TestDataCsvFile class

- `class TrainDataCsvFile`
  This class represents the sample part of the `train.csv` file. It implements the `parse(self)` method in a similar fashion.

- `class TrainLabelsCsvFile`
  This class represents the label part of the `train.csv` file. When instantiating, it can make a link to an already existing `TrainDataCsvFile` class, to prevent loading the same csv file a second time. It implements the `parse(self)` method in a similar fashion.

5.2.2 Keras Model

To build the model, a dedicated `Model` class was created. This class operates as a Keras model factory, using the `def get_model()` method to either create and train a model or load it's weights and parameters from a file and return the model.

The layers of the model changed greatly over time. This is the the last version of the model:

```
model = keras.Sequential([
    keras.layers.Dense(128, input_shape=(train_data.shape[1],), activation='relu'),
    keras.layers.Dense(128, activation='relu'),
    keras.layers.Dense(39, activation='softmax')
    # keras.layers.Flatten(input_shape=(28, 28)),
    # keras.layers.Dense(128, activation=tf.nn.relu),
    # keras.layers.Dense(10, activation=tf.nn.softmax)
])

optimizer = keras.optimizers.Adam(lr=0.01)
model.compile(optimizer=optimizer,
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
self.log.info("Compiled model")
```

```python
model.fit(train_data, train_labels, epochs=5, batch_size=20)
self.log.info("Trained model")
```

Listing 3: Keras model

5.2.3 Classification process

The classification process was easily written using the classes created in 5.2.1 and 5.2.2:

```python
trainsamplesfile = TrainDataCsvFile()
trainlabelfile = TrainLabelsCsvFile(trainsamplesfile)
testsamplesfile = TestDataCsvFile()
for file in [trainsamplesfile, trainlabelfile, testsamplesfile]:
    if args.prep_data or not file.prep_file_exists():
        file.parse()
        file.save()
    else:
        file.load()

if args.train:
    mdl = Model().get_model(
        trainsamplesfile.toNpArray(),
        trainlabelfile.toNpArray()
    )
else:
    mdl = Model().get_model()

# predictions = mdl.predict(testfile.toNpArray())
predictions = mdl.predict(trainsamplesfile.toNpArray())
for i in range(0, 19):
    predicted =
    trainlabelfile.CATEGORIES[np.argmax(predictions[i])]
    actual = trainlabelfile.get(i)
    logging.info("{} ?= {}".format(actual, predicted))
```

Listing 4: main.py

When using this setup, training for even 25 epochs did not raise the accuracy in any way. The accuracy value usually hovered barely below 20%, which coincides with the bottom line described in 5.1.

After multiple unfruitful tries, this approach had to be abandoned because of the lack in progress due to lack of knowledge and experience with neural networks.

6 Results

7 Conclusion

8 Listings

References

Darekar, R., Dandona, R. & Sureshbabu, V. (2016), 'Predicting and Analysis of Crime in San Francisco'.
URL: *https://www.slideshare.net/SameerDarekar1/san-francisco-crime-analysis-classification-kaggle-contest* 4

Kaggle (2015), 'San Francisco Crime Classification'.
URL: *https://www.kaggle.com/c/sf-crime* 2, 4, 7

Kaggle (2016), 'Expedia Hotel Recommendations'.
URL: *https://www.kaggle.com/c/expedia-hotel-recommendations* 2

Kaggle (2017*a*), 'Quora Question Pairs'.
URL: *https://www.kaggle.com/c/quora-question-pairs* 2

Kaggle (2017*b*), 'Toxic Comment Classification Challenge'.
URL: *https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge* 2

Kaggle (2018*a*), 'Inclusive Images Challenge'.
URL: *https://www.kaggle.com/c/inclusive-images-challenge* 2

Kaggle (2018*b*), 'TalkingData AdTracking Fraud Detection Challenge'.
URL: *https://www.kaggle.com/c/talkingdata-adtracking-fraud-detection* 2

Keras (n.d.), 'The Python Deep Learning library'.
URL: *https://keras.io/* 9

Murray, M. (n.d.), 'Classifying San Francisco Crime Incidents'.
URL: *http://mattmurray.net/classifying-san-francisco-crime-incidents/* 4

Ramunno-Johnson, D. (2015), ‘Machine learning to predict San Francisco crime’.
URL: *http://efavdb.com/predicting-san-francisco-crimes/* 4

Tensorflow (n.d.), ‘An open source machine learning framework for everyone’.
URL: *https://www.tensorflow.org/* 9

List of Figures

List of Tables

Listings

A Appendix