
# Deep Learning

## Credit Card Fraud Detection

U Kang
Seoul National University


# In This Lecture

- Credit card transactions
- Fraud detection
- Imbalanced classification
- Feedforward neural networks

# Outline

- **Problem Definition**
- Requirements
- Preprocessing Codes
- Answers
- Autoencoder

# Dataset (1)

- Transactions by credit cards in two days
- Each transaction contains a binary class
  - Normal (class 0)
  - Fraud (class 1)
- Most features are numerical values
  - They are the result of a PCA transformation
  - No information due to confidentiality issues

# Dataset (2)

- Contains 28 numerical features (V1 ~ V28)
- Contains 2 special features ('Time' and 'Amount')
  - 'Time' is the seconds from the start
  - 'Amount' is the transaction amount
  - They are not the result of PCA
- The column 'Class' is the response variable
  - 1 in case of fraud 0 otherwise

# Problem Definition

- To classify transactions into **Fraud** or **Normal**
- Simple classification problem on binary classes
- NOTE: the dataset is highly imbalanced!
  - 492 frauds out of 284,807 transactions (0.172%)

# Selection of an Algorithm

- The dataset has no meaningful structures
  - Images, time-series, …
- Thus, we use feedforward neural networks

# Outline

- ☑ Problem Definition
- ☐ **Requirements**
- ☐ Preprocessing Codes
- ☐ Answers
- ☐ Autoencoder

# Considering Imbalance (1)

- We need to consider the imbalance
  - Otherwise, the model will produce only ‘Normal’
- Two famous techniques when training:
  - Undersampling
    - Select a subset of normal transactions
  - Oversampling
    - Duplicate fraud transactions
    - Adjust weights of the transactions

# Considering Imbalance (2)

- Is *accuracy* enough for the evaluation?
- If we classify all instances as 'Normal', the accuracy will become high
- But, that is definitely not we want
- We need another evaluation metric


# Model Structure

- Implement 3 dense layers
  - Numbers of hidden nodes: (64, 32, 16)
- Use dropout only in the third hidden layer
- Activation function: ReLU
- Optimizer: Adam
- Loss function: cross entropy

# Feature Distribution

- We have the two categories of features
  - Simple numerical features (V1, …, V28)
  - Special features (Time and Amount)
- They are distributed differently
  - Averages, standard deviations, …
- We need to standardize them!

# Outline

- ☑ Problem Definition
- ☑ Requirements
- ☐ **Preprocessing Codes**
- ☐ Answers
- ☐ Autoencoder

# Loading the Dataset (1)

- Read the dataset using the pandas package:

```
import pandas as pd
import tensorflow as tf

df = pd.read_csv('creditcard.csv')
```

- The variable df is a DataFrame instance
- It is a 2D table of instances and features

# Loading the Dataset (2)

- You can check the contents:

```
df.head()
```

| | Time | V1 | V2 | V3 | V4 | V5 | V6 |
|---|---|---|---|---|---|---|---|
| **0** | 0.0 | -1.359807 | -0.072781 | 2.536347 | 1.378155 | -0.338321 | 0.462388 |
| **1** | 0.0 | 1.191857 | 0.266151 | 0.166480 | 0.448154 | 0.060018 | -0.082361 |
| **2** | 1.0 | -1.358354 | -1.340163 | 1.773209 | 0.379780 | -0.503198 | 1.800499 |
| **3** | 1.0 | -0.966272 | -0.185226 | 1.792993 | -0.863291 | -0.010309 | 1.247203 |
| **4** | 2.0 | -1.158233 | 0.877737 | 1.548718 | 0.403034 | -0.407193 | 0.095921 |

# Dividing the Dataset (1)

- Divide the dataset by the class:

```
fraud = df[df.Class == 1]
normal = df[df.Class == 0]
```

- Here [df.Class == 1] selects the normal ones
- This is row indexing (cf. column indexing)

# Dividing the Dataset (2)

- Create training and test sets:

```
X_train = pd.concat([fraud.sample(frac=0.8,
                                  random_state=0),
                     normal.sample(frac=0.8,
                                   random_state=0)],
                    axis=0)
X_test = df.loc[~df.index.isin(X_train.index)]
```

- We divide it with the ratio of 8:2
- We consider each class separately (why?)

# Dividing the Dataset (3)

- Shuffle the resulting instances:

```
from sklearn.utils import shuffle

X_train = shuffle(X_train, random_state=0)
X_test = shuffle(X_test, random_state=0)
```

- This is useful for most ML algorithms

# Creating Labels

- Create the labels for training and test sets:

```
y_train = X_train['Class'].values.astype('float')
y_train = np.stack([1 - y_train, y_train]).T

y_test = X_test['Class'].values.astype('float')
y_test = np.stack([1 - y_test, y_test]).T
```

- Note that each label is one-hot encoded

# Deleting the Labels

- Delete the labels from the DataFrames:

```
fields = ['Class']
X_train = X_train.drop(fields, axis=1)
X_test = X_test.drop(fields, axis=1)
```

- We have 30 features currently
- Is it fine to use ‘Amount’ and ‘Time’ as it is?


# Questions?