Medical Claim Fraud Detection Using Deep Neural Networks

OHSU - CS/EE 623 Deep Learning

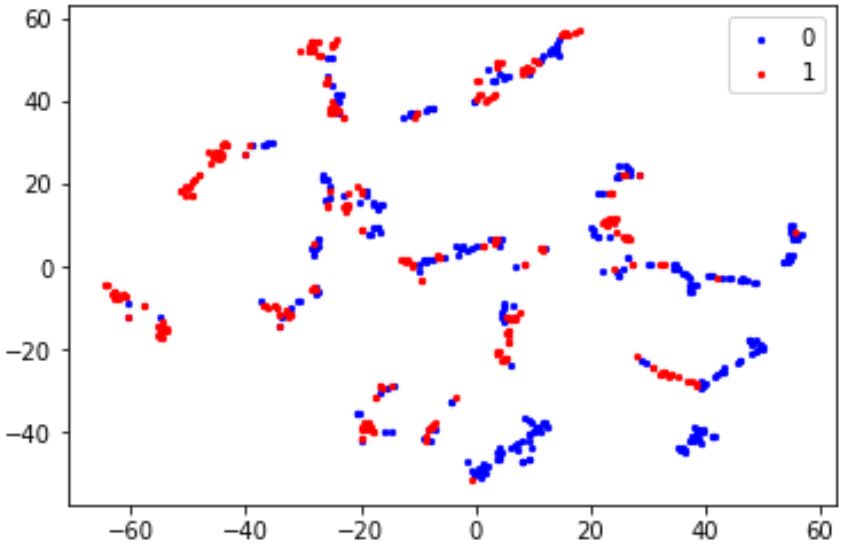
Mohammad Salari

Agenda

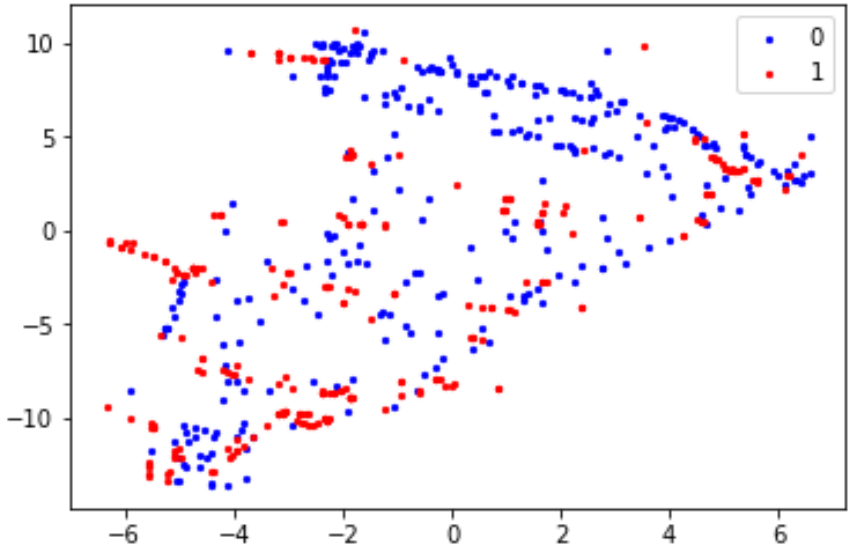
- Goal: build a two class, classifier
- Medical claims from 2015 to 2018 with labels (Supervised learning)
- Data preparation
  - Feature selection
  - Balancing data
  - Normalization
- Architecture
  - Number of hidden layer and number of neurons
- Result

Choosing features

- Visualizing high-dimensional data in a two or three-dimensional map
  - PCA and T-SNE by Laurens van der Maaten and Geoffrey Hinton



t-SNE



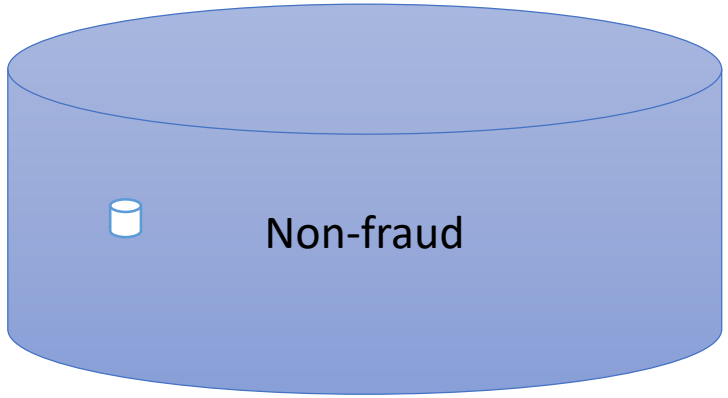
Imbalanced data

Data Set	Number of records
Non-fraud Claims	47944216
Fraud	15379

- Up Sampling ?
  - Didn't get good result at the end.
- Using less non fraud records instead of up-sampling fraud records
- Sub samples must have a fair representation of non fraud data

$$Select * from\ Data$$
$$R(l,u) < A$$

- *R(l,u) as a rand generator function*
- *$l \rightarrow$ lower band*
- *$u \rightarrow$ upper band*
- *$A \rightarrow$ fix number*




```
from ADW_DDW_VW.dbo.VW_Claims_MV as z
WHERE (ABS(CAST((BINARY_CHECKSUM (Claim_Id,Claim_Line_Nbr, NEWID())) as int))
  % 1000) < 2
```

Encoding Strings and Standardize Data

- From Scikit Learn Label Encoder
  - How it acts?
    - With empty or null values
    - Frequency of string!
- SQL has been used:
  - Lookup Table

15	Clinical_Edit_Indicator		1
16	Clinical_Edit_Indicator	7	2
17	Clinical_Edit_Indicator	8	3
18	Clinical_Edit_Indicator	9	4
19	Clinical_Edit_Indicator	E	5
20	Clinical_Edit_Indicator	F	6

- Rescaling data to have values between 0 and 1.

achieve this is:

$$x_{new} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

$$x' = \frac{x - \mu}{\sigma}$$

```
#fit_transform is combination of fit() and Transform
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

```
SELECT MAX(Bill_Amt) AS MAX,MIN(Bill_Amt) AS MIN FROM moe_test.dbo.FWA_DATA
```

00 %

Results | Messages

MAX	MIN
216300.00	0.00

Bill_Amt
0.00612112806287
0.00027739251040
0.00064724919093
0.00078169209431
0.00103744798890
0.00005547850208
0.00014794267221
0.00012944983818
0.00064724919093
0.00050855293573
0.00054553860379
0.00082293111419

Data Preparation

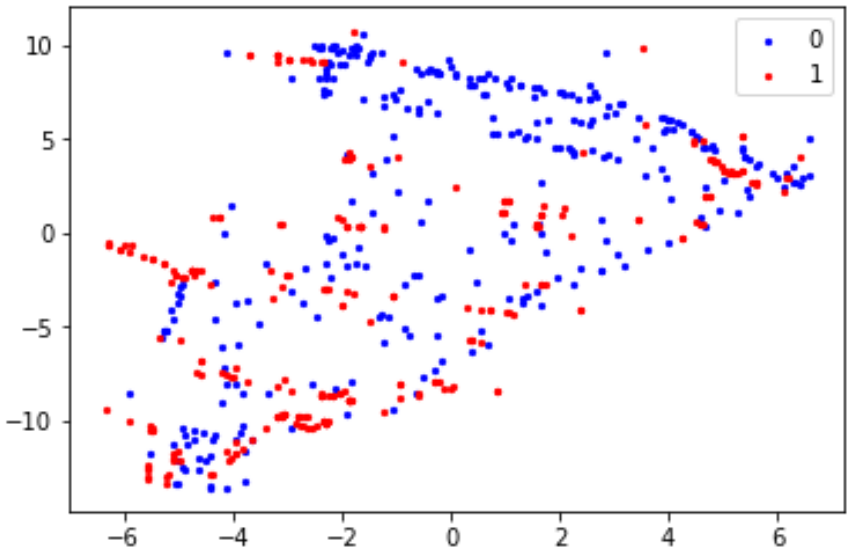
Features Selected

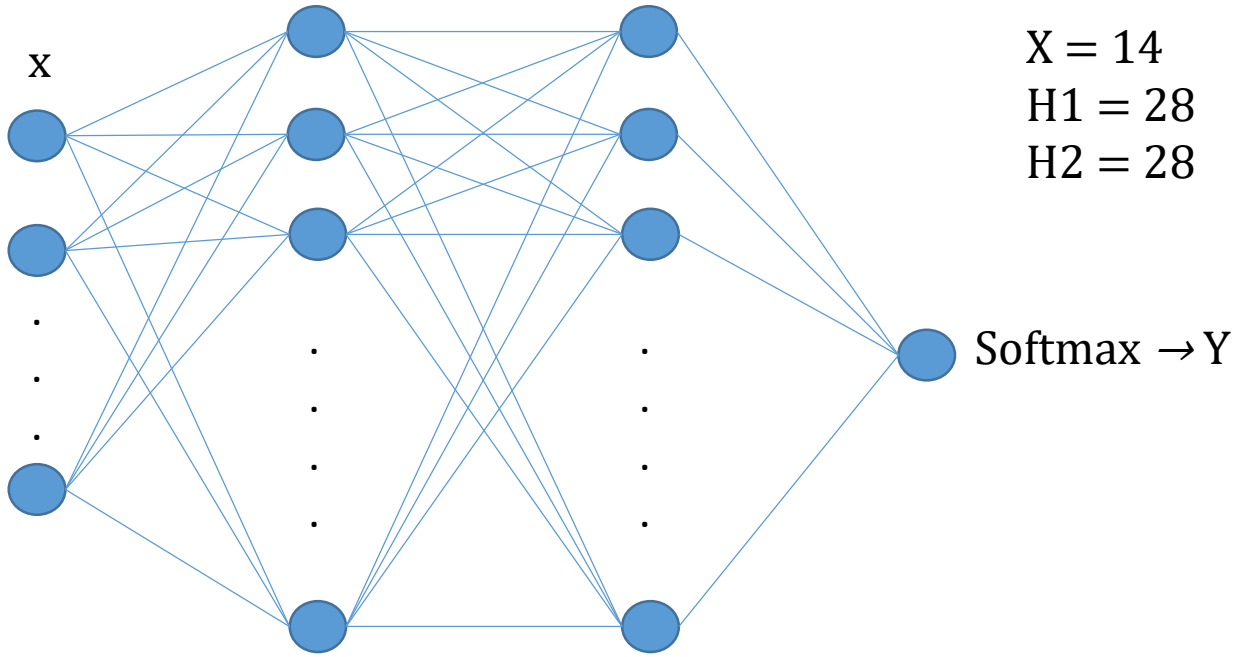
Balance Data

Map to digits

Standardize

Deep Neural Networks Architecture





$X = 14$

$H1 = 28$

$H2 = 28$

DNN Architecture

```
# Importing the Keras libraries and packages
import keras
from keras.models import Sequential
from keras.layers import Dense

#Initializing Neural Network
classifier = Sequential()

# Adding the input layer and the first hidden layer
classifier.add(Dense(output_dim = 28, init = 'uniform', activation = 'relu', input_dim = 14))
# Adding the second hidden layer
classifier.add(Dense(output_dim = 28, init = 'uniform', activation = 'relu'))
# Adding the output layer
classifier.add(Dense(output_dim = 1, init = 'uniform', activation = 'softmax'))
```

- **Uniform, lecun_uniform**, **normal**, **identity, orthogonal**, **one**
- **glorot_normal**: Gaussian initialization scaled by fan_in + fan_out (Glorot 2010)
- **glorot_uniform**
- **he_normal**: Gaussian initialization scaled by fan_in (He et al., 2014)
- **he_uniform**
- **zero**

Usage of initializations: https://faroit.github.io/keras-docs/1.2.2/initializations/

Keras parameters set…

```
# Compiling Neural Network
classifier.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

Optimizer options

- SGD
- RMSprop
- Adagrad
- Adadelta
- Adam
- Adamax
- Nadam

Loss

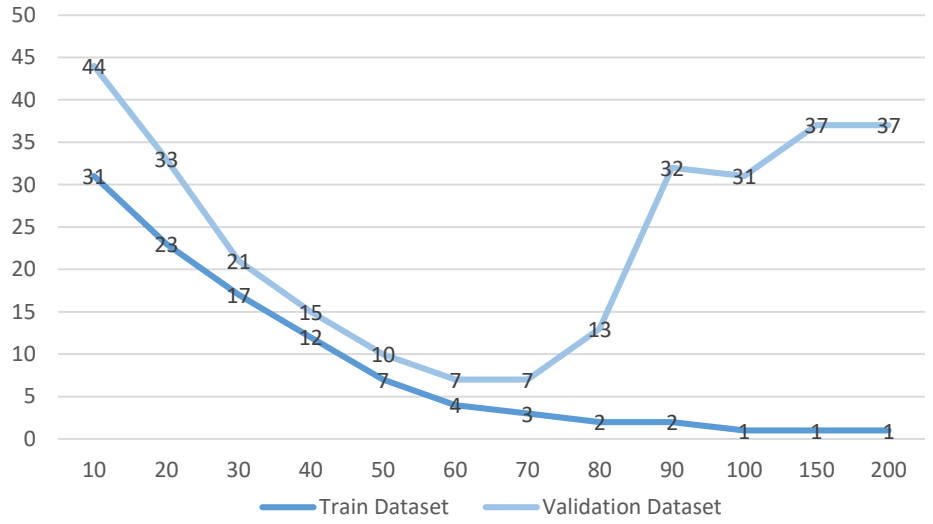
- mean_squared_error
- mean_absolute_error
- mean_absolute_percentage_error
- mean_squared_logarithmic_error
- .
- .
- .
- .
- categorical_crossentropy
- binary_crossentropy

Metric

- Custom Accuracy

A metric is a function that is used to judge the performance of your model. A metric function is similar to a loss function, except that the results from evaluating a metric are not used when training the model.

Number of Epochs




```
# Fitting our model
classifier.fit(X_train, y_train, batch_size = 10, nb_epoch = 65)
```

Result

```
# Predicting the Test set results
y_pred = classifier.predict(X_test)
y_pred = (y_pred > 0.5)
```

If the prediction be more than α then claim is belonging to fraud class.

- *α could be assign by the number of fraud records that expected*
- If α >> and we predict a record as a fraud, the probability of ***being*** true positive is high but we defiantly will loose lots of fraud records with low probabilities.
- Using some pre-knowledge could help up for example:
  - ***y_pred > 0.4 and Amount > $1000***

Thanks