HW0 Report

R09922110 張煒晟

1. Preprocessing

Bag-of-word：Correct all different words in dataset(all sentences), and similar to one-hot-encoding, combining the different words into a list which can record the occurrence of each words in a sentence.

```
train = []
dic = {}
count = 0
for text in df['text']:
    tmp_text = text.split(' ')
    new_text = []
    for word in tmp_text:
        if(word != ""):
            new_text.append(word)
            if(word not in dic):
                dic[word] = count
                count = count + 1
    train.append(new_text)
```

```
X = []
for data in train:
    x_input = [0 for _ in range(75417)]
    for word in data:
        x_input[dic[word]] = x_input[dic[word]] + 1
    X.append(x_input)
```

```
X = np.array(X)
```

```
X.shape
```

```
(80000, 75417)
```

▲ Figure 1. Transform input sentences to BOW(bag-of-word)

2. Model Structure

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 1024)              77228032
_________________________________________________________________
dropout (Dropout)            (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               262400
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 257
=================================================================
Total params: 77,490,689
Trainable params: 77,490,689
Non-trainable params: 0
_________________________________________________________________
```

▲ Figure 2. DNN structure

Using DNN model to solve this problem. Dropout is built to avoid overfitting. I have tried 1024->256->32->1, 4-layer model and run with 10 epochs. But, it was overfitting. Therefore, I reduced the number of layer and add dropout layer to prevent overfitting, and which had good performance.

```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

▲ Figure 3. Model config

Using binary cross-entropy as loss function, because of binary classification problem. And optimizer is adam which is the best(in most cases) optimizer for now on. Finally, training optimization is using accuracy as metric.

3. Training

```
model.fit(X,Y,validation_split=0.2, epochs=3, batch_size=5)
```

▲ Figure 4. Training config

Splitting 20% of train data as validation set. Running with 3 epochs and batch size was 5 data points.(I have tried 10, 5, 4, 3 epochs and 1, 5, 10, 15, 20 batch size, and 3 epochs, 5 batch size is best with rule of thumbs)

```
Epoch 1/3
12800/12800 [==============================] - 82s 6ms/step - loss: 0.2821 - accuracy: 0.8886 - val_loss: 0.2391 - val_accurac
y: 0.9075
Epoch 2/3
12800/12800 [==============================] - 71s 6ms/step - loss: 0.1785 - accuracy: 0.9361 - val_loss: 0.2581 - val_accurac
y: 0.9056
Epoch 3/3
12800/12800 [==============================] - 71s 6ms/step - loss: 0.1222 - accuracy: 0.9567 - val_loss: 0.3142 - val_accurac
y: 0.9069
```

▲ Figure 5. Training and layout with accuracy and validation accuracy

4. Conclusion

After prediction, the output is probability.

```
array([[1.0000000e+00],
       [1.5408274e-03],
       [9.9999666e-01],
       ...,
       [5.7347955e-07],
       [1.0900800e-02],
       [9.8252099e-07]], dtype=float32)
```

▲ Figure 6. Prediction output

Therefore, I have to decide threshold. In normal, we can split an extra validation set(before training) to optimize threshold. But I just pick 0.5 as threshold which had good performance with 0.9092 in Public score board.

Future work : we can also optimize the threshold with extra validation set. I believe this would improve my prediction. Besides, bag-of-word is not efficient because of lots of unique words in dataset which makes input dimension very large, and bag-of-word would lost relationship between words in a sentence. So maybe there are another encoding algorithms I can use in future to improve my model.