EE271 Final Project

Solution to the multi classification problem using CNN, RNN and attention method

1st Hua Yuxiao
dept. Department of Electronic and Electrical Engineering
Southern University of Science and Technology
Shenzhen, China
1628280289@qq.com

***Abstract*—Use a fully connected feedforward deep network, a CNN, an RNN, and an attention network to solve the above 4-class classification problem.**

***Index Terms*—machine learning, classification, CNN, RNN, attention**

I. Introduction

Try to use a fully connected feedforward deep network, a CNN (could be any modern CNN network), an RNN (could be any RNN such as Pyramid RNN, LSTM, GRU, Grid LSTM), and an attention network to solve the above 4-class classification problem. The distribution of the data is highly unbalanced, which requires preprocessing to minimize its negative effects. Finally, we will compare the performance of these models, and conclude the similarities and differences of each other.

II. Preprocessing

Before we build up our model, we shouldpreprocess the data. This includes applying standscaler to normalize the input data, in case of one factor influencing the training process too much, resulting in the weight too large.

```
path_train ='ecg_data.csv'
data_train = pd.read_csv(path_train,header=None)
data_train=np.array(data_train).astype('float32')
x=np.delete(data_train,[-1], axis=1)
y= data_train[:,[-1]]
x_train,x_test,y_train,y_test=
train_test_split(x,y,test_size=0.3,random_state=0)
scaler=StandardScaler()
x_train=scaler.fit_transform(x_train)
x_test=scaler.fit_transform(x_test)
```

In order to process the output data better, we choose to encode the output in onehot way. For example, class 1 refers to 1000, and class 2 refers to 0100.

```
encoder=LabelEncoder()
y= encoder.fit_transform(y)
y= np_utils.to_categorical(y)
```

III. Model of CNN

Convolutional Neural Networks (CNN) is a class of feed-forward neural networks with deep structure that contain convolutional computation, and is one of the representative algorithms of deep learning. Convolutional neural networks have the ability to represent learning and can classify input information by translation invariant according to their hierarchical structure, so they are also called "Shift-Invariant Artificial Neural Networks (SIANN)".

Here is the main body of the CNN model. It includes 6 convolution action and 3 maxpooling process. After that, we flatten the data, and use softmax to obtain the final output.

```
def CNN():
    model =Sequential()
    model.add(Conv1D(16, 3, input_shape=(188, 1)))
    model.add(Conv1D(16, 3, activation='tanh'))
    model.add(MaxPooling1D(3))
    model.add(Conv1D(64, 3, activation='tanh'))
    model.add(Conv1D(64, 3, activation='tanh'))
    model.add(MaxPooling1D(3))
    model.add(Conv1D(64, 3, activation='tanh'))
    model.add(Conv1D(64, 3, activation='tanh'))
    model.add(MaxPooling1D(3))
    model.add(Flatten())
    model.add(Dense(4, activation='softmax'))
    model.compile(loss='categorical_crossentropy',
    optimizer='adam', metrics=['accuracy'])
    return model
model=KerasClassifier(build_fn=CNN)
model.fit(x_train, y_train)
y_predict=model.predict(x_test)
accuracy_score(y_test,y_predict)
print(classification_report(y_test,y_predict))
```

The result of the training of the model is as following:

	precision	recall	f1-score	support
0	0.85	0.90	0.87	1511
1	0.79	0.55	0.65	234
2	0.67	0.65	0.66	728
3	0.54	0.52	0.53	86
micro avg	0.78	0.78	0.78	2559
macro avg	0.71	0.66	0.68	2559
weighted avg	0.78	0.78	0.78	2559
samples avg	0.78	0.78	0.78	2559

It is easy to see that, due to the unbalanced distribution of data, the performance on main class (here it is class 1) is much better than the others.

IV. Model of RNN

Recurrent Neural Network (RNN) is a type of recurrent neural network that takes sequence data as input, recursively recursively in the direction of sequence evolution, and all nodes (recurrent units) are connected in a chain. Recurrent neural networks have memorization, parameter sharing, and Turing-completeness, so they have certain advantages when learning the nonlinear features of sequences. Recurrent neural networks have applications in natural language processing (NLP), such as speech recognition, language modeling, machine translation, and other fields, and are also used in various types of time series forecasting. Recurrent neural networks constructed by introducing convolutional neural networks can deal with computer vision problems involving sequence inputs.

Here we just post the core codes as following. In an RNN model, the training process includes dropout. This means some unimportant information will be discarded, which will improve the efficiency of the model.

```
def RNN():
    model = Sequential()
    model.add(Embedding(
    input_dim=189,output_dim=64))
    model.add(SpatialDropout1D(0.2))
    model.add(LSTM(100,dropout=0.2,
    recurrent_dropout=0.2))
    model.add(Flatten())
    model.add(Dense(4, activation='softmax'))
    model.compile(loss='categorical_crossentropy',
    optimizer='adam', metrics=['accuracy'])
    return model
model=KerasClassifier(build_fn=RNN)
model.fit(x_train, y_train)
y_predict=model.predict(x_test)
accuracy_score(y_test,y_predict)
print(classification_report(y_test,y_predict))
```

The result of the training of the model is as following:

	precision	recall	f1-score	support
0	0.86	0.88	0.87	1511
1	0.68	0.65	0.66	234
2	0.68	0.61	0.64	728
3	0.41	0.60	0.49	86
micro avg	0.77	0.77	0.77	2559
macro avg	0.66	0.68	0.67	2559
weighted avg	0.77	0.77	0.77	2559
samples avg	0.77	0.77	0.77	2559

It can be seen from the chart that, the classification performance is still not so good. Comparing to the previous CNN model, the recall rate is a little bit higher. CNN is spatial expansion, neurons and feature convolution. While RNN is time extension, neurons with multiple time output calculations. In this project, the time extension feature does not make much difference.

V. Model of Attention

RNN machine translation has its own weaknesses, and Focus came into being to overcome this weakness. Attention is essentially pooling: an avg-pooling of power. That is, a variable-length matrix, pooling as a fixed-length process. The difference between different Attention is only in the way the weights are calculated/pooled objects.

The follwing is the main part of a classical attention model. It is similar to the previous RNN model. The main difference between them is the additional attention part. The hidden state is updated each time, and then all the weighted hidden state will be sumed up to attain the output.

```
def attention():
    model = Sequential()
    input = Input(shape=(10, 188))
    model= LSTM(64, return_sequences=True)(input)
    model= Attention(units=32)(model)
    model.add(Flatten())
    model.add(Dense(4, activation='softmax'))
    model.compile(loss='categorical_crossentropy',
    optimizer='adam', metrics=['accuracy'])
    return model
model=KerasClassifier(build_fn=attention)
model.fit(x_train, y_train)
y_predict=model.predict(x_test)
accuracy_score(y_test,y_predict)
print(classification_report(y_test,y_predict))
```

Here is the training result of the attention model.

	precision	recall	f1-score	support
0	0.80	0.94	0.87	1511
1	0.79	0.63	0.70	234
2	0.71	0.53	0.61	728
3	0.73	0.38	0.50	86
micro avg	0.78	0.78	0.78	2559
macro avg	0.76	0.62	0.67	2559
weighted avg	0.77	0.78	0.77	2559
samples avg	0.78	0.78	0.78	2559

Better than the previous model. However, due to the extremely unbalanced distribution of data, the classification performance is still not so good, especially the recall rate of class 3 and class 4.

VI. Conclusion

In this project, we try to solve a multi classification problem using different kinds of convolution model, CNN, RNN and attention. These models share many similarities and differences.

Similarities: Extensions of traditional neural networks; Forward calculation produces results, reverse calculation model update;Each layer of neural network can coexist with multiple neurons horizontally, and there can be multi-layer neural network connections vertically.

Differences: CNN is space expansion, neurons and feature convolution; RNN is time extension, neurons with multiple time output calculations; RNNs can be used to describe the output of continuous states in time, with memory function, and CNNs are used for static output; CNN advanced 100+ depth; RNN depth is limited.

VII. REFERENCE

[1] ”[Deep Learning Article]—Combination and Comparison of CNN and RNN, Case Explanation - Zhihu,” .

[2] ”Pytorch-LSTM+Attention Text Classification - Saltier Fish - Blog Park,” .

[3] ”Python Deep Learning 12 - Keras Implements Self-Attention Mechanism Chinese Text Sentiment Classification (Detailed Annotations) -Johngo Senior,” .

[4] ”LSTM Deals with Multi-Classification Problems-Sleeping Bear Awakening Blog-CSDN Blog-lstm Classification,” .