Report

R06922074　吳柏威

1. Model description

甲、Rnn

```
model = Sequential()
model.add(Bidirectional(LSTM(512, activation='tanh', return_sequences=True, kernel_initializer= 'glorot_uniform', recurrent_initializer='orthogonal'), input_shape=(timeStep,data_dim)))
model.add(Bidirectional(LSTM(512, activation='tanh', return_sequences=True, kernel_initializer= 'glorot_uniform', recurrent_initializer='orthogonal')))
model.add(Dropout(0.5))
model.add(Bidirectional(LSTM(512, activation='tanh', return_sequences=True, kernel_initializer= 'glorot_uniform', recurrent_initializer='orthogonal')))
model.add(Bidirectional(LSTM(512, activation='tanh', return_sequences=True, kernel_initializer= 'glorot_uniform', recurrent_initializer='orthogonal')))
model.add(Dropout(0.5))
model.add(TimeDistributed(Dense(phoneClass, activation='softmax')))

model.summary()
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

可見於 model_rnn.py 第 33 行

利用 2 層 LSTM、1 層 dropout、2 層 LSTM、1 層 dropout 以及最後的 timedistribute dense 層輸出，使用 arctan 函數作為每一個 LSTM 的 active function，而 return_sequences=True 會使得 LSTM 的每一個 unit 的 output 除了會傳入下一個 unit 之外也會當作傳入下一層 layer 的 input。kernel_initializer= 'glorot_uniform'則是 unit 在接收上一層 layer input 參數初始值會呈現一個 uniform distribution 而且上下界為 `sqrt(6 / (fan_in + fan_out))`，`fan_in` 是 `input unit` 數量，反之亦然。`recurrent_initializer='orthogonal'`則是 `unit` 在接收上一個 `unit` 的 `output` 的參數初始值會會呈現一個 `orthogonal matrix`，最後 `timedistributed` 則是會依序將每一個 `timestep dense` 並 `softmax` 成 `class` 的機率模型。

乙、Cnn + Rnn

```
model = Sequential()
model.add(Reshape((timeStep,data_dim,1), input_shape=(timeStep,data_dim)))
model.add(Conv2D(filters=128, kernel_size=(3,3), padding='same'))
model.add(Conv2D(filters=256, kernel_size=(3,3), padding='same'))
model.add(Reshape((timeStep,data_dim*256)))
model.add(Bidirectional(LSTM(256, activation='tanh', return_sequences=True, kernel_initializer= 'glorot_uniform', recurrent_initializer='orthogonal')))
model.add(Bidirectional(LSTM(256, activation='tanh', return_sequences=True, kernel_initializer= 'glorot_uniform', recurrent_initializer='orthogonal')))
model.add(Dropout(0.5))
model.add(Bidirectional(LSTM(512, activation='tanh', return_sequences=True, kernel_initializer= 'glorot_uniform', recurrent_initializer='orthogonal')))
model.add(Bidirectional(LSTM(512, activation='tanh', return_sequences=True, kernel_initializer= 'glorot_uniform', recurrent_initializer='orthogonal')))
model.add(Dropout(0.5))
model.add(TimeDistributed(Dense(phoneClass, activation='softmax')))

model.summary()
model.compile(loss='categorical_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

可見於 model_cnn.py 第 33 行

在原本的 rnn model 前面加上兩層 convolution 2D layer，其 activate function 設定為無，預設為 linear function。之後再 reshape layer 中將 cnn 輸出的三圍矩陣改成二維矩陣在當成之後 rnn 的 input。

2. How do you improve your performance

甲、在讀取 train.lab 寫入二維矩陣檔之前，先將 label 利用 48_39.map 從 48 種 class 轉換成 39 種之後轉成數字之後再寫入。比起 predict 48 種 class，predict 39 種 class 可以有更高的準確率。

乙、使用 dropout 預防 overfitting 發生。

丙、使用 timedistributed 包著一維輸出的 dense 替代二維輸出的 dense。如果使用 dense predict 出 sequence 的話，因為這樣子 dense 裡面的參數量會非常的高，而 dense 中若有過多的參數將會互相干擾，降低準確率。

丁、Sequence timestep 設定成 123，123 這個數字剛好是一個句子裡面有意義的字互相關聯的最剛好的長度，太短則無法全部利用上下文推斷文字，太長則會被無關的上下文干擾。

戊、Batch 設定成 32，batch 太少則會無法往最佳點移動，而 batch 太高就會發生 overfitting 的事件發生。

己、在 predict 完之後使用一個門檻將機率較低的文字移出，因為最後分數是計算兩個句子的 edit distance，若原本為 aaa 被簡化為 a 的句子被 predict 成 aba，這樣子 distance 會+2，中間的 b 會有非常高的機率是模稜兩可的，故把他剃除。利用門檻將資料做最後處理可大幅降低分數。

庚、在確定最後 model 之後，將所有 data 都當成 training data 丟下去訓練。根據理論，越多的 data 可以讓 model 訓練得更好。

辛、Model 訓練前 initial value 的好壞也會直接影響最後的準確率，所以最後同樣的 model 分別訓練許多次之後產出許多架構一樣的 model，但是準確率可以差到 2%左右。

3. Experiment settings and results

甲、Cnn 與 Rnn 比較

i. 在前處理已轉換 class，batch 設定為 32 且使用門檻的條件下

ii. Cnn 就以上 model 訓練時準確率可到 78%，而 Rnn model 訓練時平均準確率為 80%，而目前最好的 model 可以達到 81.53%。

乙、和其他 model 做比較

i. Input 輸入 sequense 使用 4 層 convolutional layer train 約 46%

ii. Input 輸入 sequense 使用 4 層 timedistributed dense train 的話可以高達 63%

iii. 使用 GRU 以及 relu activate function 取代 LSTM 以及 tanh activated function，雖然 train 的速度快了許多(約 4 倍)，但是約低於 LSTM model 3%的準確率，在測試過 rnn 以及 cnn + rnn model 皆然，適合用來驗證粗略的 model 之後再換成 LSTM 得到更高的準確率。

iv. 在最後如果使用 dense output sequence x class_num 的數量在使用 reshape 去轉換成 sequence，使用之前最好的 rnn model 為 75%，而使用了 timedistributed dense 則可以達到 80%，非常好的進步。