Machine Learning

HW#3　電信所 R04942131_蔡銘穎

**Approach and Discussion**

這次的作業要求我們從 cifar-10 的資料庫中的 5000 筆 label data 和 45000 筆 unlabel data ，以兩種方式 train 得 model 後，測試 10000 筆 data。

**(1) Supervised learning**

首先我在 label data 中，每五筆資料取出一筆當作我的 validation data，也就是說在 label 0-9 中，每個 label 都會有 100 筆 data，共 1000 筆，剩下的 4000 筆當作我的 training set。

在此要特別注意的是使用 tensorflow 和 theano 時，input dimension shape 並不相同。

```
model = Sequential()
model.add(Convolution2D(16, 3, 3, input_shape=(3, 32, 32)))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Convolution2D(32, 3, 3))
model.add(AveragePooling2D(pool_size=(2, 2)))
model.add(Convolution2D(64, 3, 3))
model.add(AveragePooling2D(pool_size=(2, 2)))

model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Activation('relu'))
model.add(Dense(10))
model.add(Activation('softmax'))
model.compile(loss='categorical_crossentropy', optimizer=Adam(), metrics=['accuracy'])
```

在此我使用了三層 neuron，filter number 為 16 , 32 , 64 ，這是我測試出最準確的 model，可以將 training data fit 到 0.98 左右，而 validation data 則是 0.8。

**(2) Semi-supervised learning(1)**

Semi-supervised 即是將第一次 train 好的 model 測試 unlabel data 後，拿來擴大自己的資料庫後，在 train 一組新的 model (架構相同)。

```
model.fit(X_train.reshape(X_train.shape[0], 3, 32, 32), Y_train,
          batch_size=32,
          nb_epoch=50,
          validation_data=(x_val, y_val))

Y_unlabel = model.predict(X_unlabel.reshape(X_unlabel.shape[0], 3, 32, 32), batch_size=32, verbose=1)

X = np.concatenate((X_train, X_unlabel), axis=0)
Y = np.concatenate((Y_train, Y_unlabel), axis=0)

model.fit(X.reshape(X.shape[0], 3, 32, 32), Y,
          batch_size=32,
          nb_epoch=25,
          validation_data=(x_val, y_val))
```

在這邊我使用的 batch size 為 32，epoch 為 25

在使用 GPU (GeForce GT 730) 下，可在 10 分鐘內完成。

Fit accuracy 為 0.95 ，validation accuracy 則為 0.76。

**(3) Semi-supervised learning(2)**

第二個我所使用的是 autoencoder，方法是將 input data(3,32,32)先進行 encode，再 decode，而 output 要是自己本身(3,32,32)，而這組 model 我只能 fit 到 mse error 為 0.08 左右

```
# encoder
autoencoder = Sequential()
autoencoder.add(Convolution2D(3, 3, 3, border_mode='same', input_shape=(3, 32, 32)))
autoencoder.add(MaxPooling2D((2, 2)))
autoencoder.add(Activation('relu'))
autoencoder.add(Convolution2D(32, 3, 3, border_mode='same'))
autoencoder.add(MaxPooling2D((2, 2)))
autoencoder.add(Activation('relu'))
x = MaxPooling2D((2, 2))
autoencoder.add(x)
encoder = theano.function([autoencoder.input], [x.output])

# decoder
autoencoder.add(UpSampling2D((2, 2)))
autoencoder.add(Activation('relu'))
autoencoder.add(UpSampling2D((2, 2)))
autoencoder.add(Convolution2D(32, 3, 3, border_mode='same'))
autoencoder.add(Activation('sigmoid'))
autoencoder.add(UpSampling2D((2, 2)))
autoencoder.add(Convolution2D(3, 3, 3, border_mode='same'))

autoencoder.compile(loss='binary_crossentropy', optimizer=Adam(), metrics=['mse'])
X_train, Y_train = load_label(path_name + '/all_label.p')
# X_val, Y_val = validation(X_train, Y_train)
autoencoder.fit(X_train.reshape(X_train.shape[0], 3, 32, 32), X_train.reshape(X_train.shape[0], 3, 32, 32),
                nb_epoch=20,
                batch_size=32,
                shuffle=True)
```

Model 完成之後，我們必須將 input 丟到此 model，而在 encode 時即拿出，不進行 decode，在以此 code 當成 feature 丟到下一組 model 進行 training，故第二組 model 的 I/O 為一組 code (32 , 4 , 4) 以及 label (0-9)，但此方法因為 training 時間長而且並沒有非常準確，最後只達到 fit accuracy 0.6 Validation accuracy 0.52。

**(4) Compare and analyze your results**

其實這兩種方法應該可以有更好的 performance，但因為時間有限，而且一開始並沒有開啟 GPU 進行 training，所以感覺應該還可以調到更好的參數。

方法二是先做了 feature extract ，此方法有利有弊，如果在此即不能 train 出一個好的 model，那後面也不太可能會有好的準確率，在這邊我試過了 SGD，Adam，兩層三層 neuron 等等的變化，始終沒辦法 fit 到一個很好的數字，也使得後面的準確率不盡理想，我想這是一個還可以再進步的地方。