Machine Learning: HW3 Report

November 19, 2016

Professor Hung-Yi Lee

電機三 B03901018 楊程皓

1. Supervised learning

參考 keras example 的 layers 以創建 model, code 如下:

```

data = []
val_data = []
answers = np.zeros((4000, 10), dtype=np.int)
val_answers = np.zeros((1000, 10), dtype=np.int)

# parsing data, answers
for ...

# transform to numpy array
data = np.array(data)
val_data = np.array(val_data)
data = data.astype('float32')
val_data = val_data.astype('float32')
data /= 255
val_data /= 255

model = Sequential()
model.add(Convolution2D(32, 3, 3, border_mode='same', input_shape=data.shape[1:]))
model.add(Activation('relu'))
model.add(Convolution2D(32, 3, 3))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Convolution2D(64, 3, 3, border_mode='same'))
model.add(Activation('relu'))
model.add(Convolution2D(64, 3, 3))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(512))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(nb_classes))
model.add(Activation('softmax'))

# Use various optimizer, in this case, rmsprop
model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['↩
```

```
    accuracy'])

# train with 4/5 labeled data
model.fit(data, answers, batch_size=batch_size, nb_epoch=nb_epoch, shuffle=True, validation_data=(val_data, val_answers))

# save model to model_path
model.save(model_path)
```

經過各種 optimizer 測試後，此 model 以 rmsprop 為最佳， nb_epoch = 30，僅以 labeled data train 就可以達到public data 0.63 (private 0.64) 的準確率。然而以這個 model 做 semi supervised training 卻會得更差的結果。

2. Semi-supervised learning(1)

將 supervised learning 後的 model 做 unlabeled data prediction 後的結果加入原本的 labeled data 作為進一步 training model 的資料，並以 predict 的屬性值線性作為 sample_weight 的參數，並加入 image preprocessing 來優化結果。code 如下:

```
data = []
training_data = []
answers = []

# parsing labeled data, unlabeled data, answers

# load supervised learning model from model_path
model = load_model(model_path)

# predict the classes of unlabeled data
results = model.predict(training_data, verbose=1)

# parsing new answers and weights

# preprocessing and data augmentation
datagen = ImageDataGenerator(
    featurewise_center=False,
    samplewise_center=False,
    featurewise_std_normalization=False,
    samplewise_std_normalization=False,
    zca_whitening=False,
    rotation_range=0,
    width_shift_range=0.1,
    height_shift_range=0.1,
    horizontal_flip=True,
    vertical_flip=False)

datagen.fit(data)

# train the model on the batches generated by datagen.flow()
model.fit_generator(datagen.flow(data, answers,
                    batch_size=batch_size),
                    samples_per_epoch=data.shape[0],
                    nb_epoch=nb_epoch)

# save semi-supervised model to ouput_model_path
model.save(output_model_path)
```

這裡的 supervised learning model 有用同樣的方式做 preprocessing，但因為單一 supervised learning 結果沒有上述的方式好，故沒有多做敘述。
這裡的 optimizer 選的是 adam, nb_epoch = 30，predict score threshold = 0.8 (不到 threshold 的 data 就不加入下一輪 training 的資料了)，public data 準確率為 0.64 (private 0.65)，進步幅度其實並不大...

3. Semi-supervised learning(2)

以 autoencoder 做 semi-supervised learning, 只取前 5000 個最有可能的資料作為下一輪新的資料。code 如下:

```
model = Sequential()
# dim 3072 -> encoding_dim ( = 512)
model.add(Dense(encoding_dim, activation='relu', input_shape=(3072,)))
# add numOfLayers ( = 5) layers
for i in range(numOfLayers):
    model.add(Dense(layer_dim, activation='relu'))
# dim layer_dim ( = 512) -> 3072
model.add(Dense(3072, activation='linear'))

# use optimizer rmsprop
model.compile(loss='mse', optimizer='rmsprop', metrics=['accuracy'])
model.fit(  data, data,
            batch_size=batch_size_auto,
            nb_epoch=nb_epoch_auto,
            verbose=1)
            # validation_data=(x_test[0:3000], x_test[0:3000]))
score = model.evaluate(x_test[0:3000], x_test[0:3000])
print('score', score)

# define encoder as the middle layer's data
encoder = K.function([model.layers[0].input], [model.layers[2].output])

# encode labeled data
data_encoded = encoder([data])[0]


average_data = np.zeros((10, layer_dim), dtype=np.float)

# calculate average of each cluster of each 512 dim
for ...

# encode unlabeled data
x_test_encoded = encoder([x_test])[0]

# calculate min error of each unlabeled data of each cluster
for ...

# sort unlabeled data by error
results = sorted(results, key = getValue)

# pick first max_size ( = 5000) of results, linear weighted with rank
for i in range(max_size):
    ...

# load supervised learning model from model_path
old_model = load_model(model_path)

# train the model with added data
```

```
old_model.fit(data, answers, batch_size=batch_size, nb_epoch=nb_epoch, shuffle=True, sample_weight=weights)

# save semi-supervised model to ouput_model_path
old_model.save(output_model_path)
```

這裡的 supervised model 我取 adam optimizer 與 nb_epoch = 30, 新一輪的 optimizer 與 nb_epoch 取一樣，但 private score 僅約為 0.46，可能參數與 layers 還有待調整。

4. Compare & Analyze

可能因為對 deep learning 了解不夠深，沒有設計出好的 model 來 train, 故無法作到如 supervised learning 就有 0.8 accuracy 的水準; 而 semi-supervised learning 感覺也沒有抓到核心的關鍵，加入 weighted 與 threshold 後只比試出的最佳 supervised learning 高一些些 accuracy 而已。 autoencoder 的確如老師所言不是很好做， accuracy 在 train 完後反而更糟糕了。

之後如要改進 accuracy 首要就是設計出更好的 model, 並用更好的 clustering 做 semi-supervised。