Machine Learning HW3

b02502108
陶昇永

1. supervised learning：

performance：

大約在 0.6 左右

實作方法：

讀取檔案後，開始建立模型架構，使用的架構為:
16 張 3*3 的 filter, activation =relu
32 張 3*3 的 filter, activation =relu
64 張 3*3 的 filter, activation =relu
之後 flatten
一層 output size = 512, activation = relu
一層 output size = 10, activation = softmax
之後開始 train，train 到 0.9 時停止，詳細的程式碼在 semi-supervised 的部份有涵蓋

2. semi-supervised learning(1)：

Performance:

大約 accuracy 在 0.54 附近擺盪，因為有 random 選 batch size 的關係，加上 dropout，會讓結果有些許不同。

實作方法&code:

```
load label data
process the data format
data.reshape(data.shape[0],3,32,32)
#------------------construct model----------------------------
model = Sequential()
model.add(Convolution2D(16, 3, 3, border_mode='same', input_shape=data.shape[1:]))
model.add(Activation('relu'))
model.add(Convolution2D(16, 3, 3))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Convolution2D(32, 3, 3, border_mode='same'))
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(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(512, activation = 'relu'))
model.add(Dropout(0.5))
model.add(Dense(10, activation = 'softmax'))

model.compile(loss = 'sparse_categorical_crossentropy',optimizer = 'adam',metrics = ['accuracy'])
model.save('emptymodel.h5')                                        #save empty model

model.fit(X_Train, Y_train, batch_size = 100, nb_epoch = 5, shuffle=True)
```

```
score = model.evaluate(X_Train,Y_train)
print 'acc',score[1]
while score[1] < 0.9:                                  #continue to train until the acc arrives 0.9
	model.fit(X_Train, Y_train, batch_size = 100, nb_epoch = 5, shuffle=True)
	score = model.evaluate(X_Train,Y_train)
	print 'acc',score[1]
model.save('epoch.h5')                                 #save trained model
model = load_model('epoch.h5')                         #load trained model
x_train = []                                           #processing data
y_train = []
all_label = pickle.load(open(str(path)+'all_label.p','rb'))
for i in range (10):
	for j in range(500):
		x_train.append(all_label[i][j])
		y_train.append([i])
x_train = np.array(x_train)
unlabel = pickle.load(open(str(path)+'all_unlabel.p','rb'))
unlabel = np.array(unlabel)
while unlabel.shape[0] > 5000:                         #adding unlabeled data
	unlabshape = unlabel.shape[0]
	result = model.predict(unlabel.reshape(unlabshape,3,32,32))
	del model, unlabshape
	addunlabel = []
	uunlabel = []
	X_test = []
	Y_test = []
	for i in range(unlabel.shape[0]):                  #find the class of unlabeled data
		maxp = 0
		predictp = 0
		predictindex = 0
		for j in range (10):
			nextp = float(result[i][j])
			if (nextp > maxp) :
				maxp = nextp
				predictindex = j
				predictp = maxp
		if (predictp > 0.7) :                          #add the confidence data & validation data
			if count < 20000:
				addunlabel.append(unlabel[i])
				y_train.append([predictindex])
			else:
				X_test.append(unlabel[i])
				Y_test.append([predictindex])
		else:
			uunlabel.append(unlabel[i])
	unlabel = np.array(uunlabel)                       #processing data
	addunlabel = np.array(addunlabel)
	X_test = np.array(X_test)
	X_test = X_test.reshape(X_test.shape[0],3,32,32)
	shape = x_train.shape[0]
	x_train = x_train.astype('float32')
	addunlabel = addunlabel.astype('float32')
	X_test = X_test.astype('float32')
	x_train/=255
	X_test/=555
	addunlabel/=255
	x_train = np.concatenate((x_train,addunlabel),axis = 0)
	shape = x_train.shape[0]
	x_train = x_train.reshape(shape,3,32,32)
	model = load_model('emptymodel.h5')
	score = [0,0]
                                                       #train with early stopping
```

```
	model.fit(x_train, np.array(y_train), batch_size = 100, callbacks=[earlystopping], validation_data=(X_test, np.array(Y_test)), nb_epoch = 500, shuffle=True)
	score = model.evaluate(x_train, np.array(y_train))
	print 'label acc ',score[1]
model.save(output_model)
```

詳述實作方法：

先 load label data，建立 model 之後先存一個空 model，再將他 train 到 0.9 之後存另一個 model，再來 load unlabeled data，用 train 好的 model predict unlabeled data， 若機率最高的 class，其機率>0.7 則加進 label data 中，加完之後分出一組 validation set 來做 early stopping。之後開始 train，用 early stopping 來 train model，train 完之後重新 predict 剩下還沒加進去的 unlabeled data，重複以上循環，直到大於 40000 筆 unlabel data 被加進 label data 中去 train。

3.semi-supervised learning(2)：

Performance:

大約 accuracy 在 0.27 附近擺盪，表現不甚理想。

實作方法&code:

```
x_label = []
y_label = []

all_label = pickle.load(open(str(path)+'all_label.p','rb'))

for i in range (10):                                    #將 RGB 合成灰階
	for j in range(500):
		x_un = []
		for k in range(1024):
			x = all_label[i][j][k]*0.587
			x = x + all_label[i][j][k+1024]*0.114
			x = x + all_label[i][j][k+2048]*0.299  # gray value(GBR)
			x = round(x,2)
			x_un.append(x)
		x_label.append(x_un)
		y_label.append(i)
x_label = np.array(x_label)
x_label = x_label.astype('float32')/255

unlabel = pickle.load(open(str(path)+'all_unlabel.p','rb'))
unlabel = np.array(unlabel)

for i in range(1024):                                   #將 RGB 合成灰階
	unlabel[:,i] = unlabel[:,i]*0.587+unlabel[:,i+1024]*0.114+unlabel[:,i+2048]*0.299
unlabel,a,b = np.hsplit(unlabel,np.array([1024,2048]))
unlabel = unlabel.astype('float32')/255

for i in range(10):                          #加入 unlabel 整張照片中灰階差值方均根最相近的 class
	for j in range(450):
		minvalue = 30000
		minindex = 50
		for k in range(5000):
			temp = np.dot(unlabel[i*10+j]-x_label[k],unlabel[i*10+j]-x_label[k])
			if temp < min:
				minvalue = temp
				minindex = y_label[k]
		x_label = x_label.tolist()
		x_label.append(unlabel[i*10+j])
		x_label = np.array(x_label)
		y_label.append(minindex)
y_label = np.array(y_label)
```

```
model = Sequential()                                    #以 DNN 來 train data
model.add(Dense(100, input_dim = 1024, activation = 'sigmoid'))
model.add(Dense(100, activation = 'sigmoid'))
model.add(Dense(10, activation = 'softmax'))
model.compile(loss = 'sparse_categorical_crossentropy',optimizer = 'adam',metrics = ['accuracy'])
model.fit(x_label, y_label, batch_size = 50, nb_epoch = 30)
score = model.evaluate(x_label,y_label)
model.save(output_model)
```

詳述作法：

將 label & unlabeled 的照片整張轉灰階，這樣可以節省 1/3 的 memory 空間，接著計算 unlabeled data 整張照片每個 pixel 與哪張 label data 的差值方均根直最接近，其 unlabeled data 的 class 就定為最接近的 label data 的 class，接著用 DNN 的架構 train 這些 data。

4.整體比較：

令人意外的結果是，使用 label data 所 train 出來的 model，在 predict public set 居然最高，而我在做 self-training 時，分數卻總是無法提升，我想可能是因為 unlabeled data 多少有錯誤的 data 加進來，因此擾亂了整個 label data set 的準度，因為 label set 的 model 在 public set 上的準度是 0.6，那應該可以合理的推斷，label data 在 unlabel data 的準度也約略是 0.6 左右而已，然而，我卻發現用 confidence > 0.7 的 data 加入，幾乎加進了所有 unlabeled data(約加入 44000 筆)，因此可以合理推論是有錯誤 label 的 data 被加入的，這也是錯誤可能的來源之一。另外，method2 的部份，因為時間因素無法實作 auto encoder，因此也注定這個方法無法有有效的 performance，因為要將照片分群其實是不太容易的。