Machine Learning HW3 Report

b03901070 羅啟心

1. Supervised Learning using CNN

   Model

```
65 # building model
66 model = Sequential()
67
68 model.add(Convolution2D(64, 3, 3, border_mode='same',input_shape=(3,32,32)))
69 model.add(Activation('relu'))
70 model.add(Convolution2D(64, 3, 3))
71 model.add(Activation('relu'))
72 model.add(MaxPooling2D(pool_size=(2, 2)))
73 model.add(Dropout(0.25))
74
75 model.add(Convolution2D(64, 3, 3, border_mode='same'))
76 model.add(Activation('relu'))
77 model.add(Convolution2D(64, 3, 3))
78 model.add(Activation('relu'))
79 model.add(MaxPooling2D(pool_size=(2, 2)))
80 model.add(Dropout(0.25))
81
82 model.add(Flatten())
83 model.add(Dense(512))
84 model.add(Activation('relu'))
85 model.add(Dropout(0.5))
86 model.add(Dense(nb_classes))
87 model.add(Activation('softmax'))
88
89 # let's train the model using SGD + momentum (how original).
90 sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
91 model.compile(loss='categorical_crossentropy',
92               optimizer='adam',
93               metrics=['accuracy'])
```

   Arguments:

```
12 # batch_size, nb_classes, nb_epoch, data_augmentation
13 batch_size = 500
14 nb_classes = 10
15 nb_epoch = 60
```

   Fitting:

```
90 #fitting
91 np.random.seed(0)
92 np.random.shuffle(X_train_label)
93 np.random.seed(0)
94 np.random.shuffle(Y_train_label)
95
96 model.fit(X_train_label, Y_train_label,
97         batch_size=batch_size,
98         nb_epoch=nb_epoch,shuffle=True)
```

2. Semi-supervised Learning using CNN

   Same as above, but with extra self-training part :

```
#self training
for r in range(rounds):
    if(len(X_train_unlabel_not_added_id)>10):
      #self learning confident
      prob_of_X_train_unlabel_not_added = model.predict_proba(X_train_unlabel[np.ix_(X_train_unlabel_not_added_id)])
      max_prob_of_X_train_unlabel_not_added = np.amax(prob_of_X_train_unlabel_not_added , axis = 1)

      delete_index = []
      for i in range (len(X_train_unlabel_not_added_id)):
        if(max_prob_of_X_train_unlabel_not_added[i] > confident):
          X_train_unlabel_added_id.append(X_train_unlabel_not_added_id[i])
          delete_index.append(i)
      if(len(delete_index)>0):
        Y_train_unlabel_added = np.vstack((Y_train_unlabel_added,np_utils.to_categorical
                                (np.argmax(prob_of_X_train_unlabel_not_added[np.ix_(delete_index,
                                [j for j in range(10)])],axis=1))))
      X_train_unlabel_not_added_id = np.delete(X_train_unlabel_not_added_id,delete_index)

    #fitting
     model.fit(np.concatenate((X_train_label,X_train_unlabel[np.ix_(X_train_unlabel_added_id,
                 [a for a in range(3)],[b for b in range(32)],[c for c in range(32)])]),axis=0),
                 np.vstack((Y_train_label,Y_train_unlabel_added)),
                 batch_size=batch_size,
                 nb_epoch=nb_epoch_self)
```

With arguments:

```
# batch_size, nb_classes, nb_epoch, data_augmentation
batch_size = 500
nb_classes = 10
nb_epoch = 60
nb_epoch_self = 20
confident = 0.96
rounds = 20
```

3. Semi-supervised Learning using Autoencoder

   To use Autoencoder + agglomerative clustering(constrained, using metric cosine)

   But fail to have this method done.

   Autoencoder

```
#Autoencoder
input_img = Input(shape=(3, 32, 32))

x = Convolution2D(16, 3, 3, activation='relu', border_mode='same')(input_img)
x = MaxPooling2D((2, 2), border_mode='same')(x)
x = Convolution2D(8, 3, 3, activation='relu', border_mode='same')(x)
x = MaxPooling2D((2, 2), border_mode='same')(x)
x = Convolution2D(8, 3, 3, activation='relu', border_mode='same')(x)
encoded = MaxPooling2D((2, 2), border_mode='same')(x)

# at this point the representation is (8, 4, 4) i.e. 128-dimensional

x = Convolution2D(8, 3, 3, activation='relu', border_mode='same')(encoded)
x = UpSampling2D((2, 2))(x)
x = Convolution2D(8, 3, 3, activation='relu', border_mode='same')(x)
x = UpSampling2D((2, 2))(x)
x = Convolution2D(16, 3, 3, activation='relu')(x)
x = UpSampling2D((2, 2))(x)
decoded = Convolution2D(1, 3, 3, activation='sigmoid', border_mode='same')(x)

autoencoder = Model(input_img, decoded)
autoencoder.compile(optimizer='adadelta', loss='binary_crossentropy')
```

```
#Clustering
y_train_unlabel_new=[]
y_train_unlabel=[]
connectivity = [[0 for x in range(5000)] for y in range(5000)]
for i in range(10):
  for j in range(500):
    for k in range(500):
      connectivity[500*i+j][500*i+k]=1

model = AgglomerativeClustering(n_clusters=10,affinity='cosine',connectivity=connectivity,linkage='average',n_clusters = 10)

for i in range(450):

  autoencoder.fit(np.vstack(X_train_label,X_train_unlabel[0:100*i+100],X_test),np.vstack(X_train_label,X_train_unlabel[0:100*i+100],X_test),
                nb_epoch=50,
                batch_size=128,
                shuffle=True,
                validation_data=(X_test, X_test),
                callbacks=[TensorBoard(log_dir='/tmp/autoencoder')])

  model.fit_predict(X_train_unlabel[100*i:100*i+100],y_train_unlabel_new)


  for j in range (len(y_train_unlabel_new)):
    y_train_unlabel.append(y_train_unlabel_new[j])
```

4. Compare and Analysis