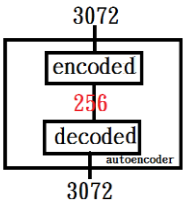
1. supervised learning

2. semi-supervised learning: cluster by autoencoder

2.1 方法

這題的分類方式主要是先藉由將 60000 個 examples (包括 train set 中的 5000 個 labeled examples、45000 個 unlabel examples 與 10000 個 test set 中的 unlabel examples，每個 example 有 3072 個 features，可視其為一個 1*3072 的 vector)降維使其可以用更少的 features 表示(作業中是做成 256 個 features，1*256 的 vector)，再利用這些較少的新 features 對 45000 個 unlabel examples 進行分類，再將分類效果較好的 unlabel examples 加進 train set(一開始的 train set 只有 5000 個 label examples)，再利用新的 train set 去 train 一個 model，最後再用這個 model 預測 test set 的 10000 個 examples。

2.2 Autoencoder



Autoencoder 的架構由一層 encoded 和一層 decoded 所組成：encoded 輸出的每個 data 的長度(output_dim)是 256、輸入的每個 data 的長度是 3072，使用的 activation 為 relu；而 decoded 輸出的每個 data 的長度(output_dim)是 3072、輸入的每個 data 的長度是 256，使用的 activation 為 sigmoid(因為 decoded 是 model 的最後一層)。Autoencoder 在 learning 過程中使用 binary crossentropy 計算 loss 並搭配 Adadelta 調整前進速度，epoch 設 1000、batch size 設 20。因為這裡只需要拿 train 好的 model 去預測 labeled 與 unlabel examples 的 256 codes 是什麼，所以可以單獨把 encoded 從 train 好的 Autoencoder 拿出來轉成 model 的形式(encoder = Model(input=Input(shape=(3072,)), output=encoded))，然後拿來預測 examples 的 codes。

2.3 Cluster

將 5000 筆 labeled examples 以 Autoencoder 產生的 256 個 codes 為 features 拿去 train k-means、random forest(預設產生十棵 trees)和 K Neighbors Classifier(# of neighbors=3)三種不同的分類模型(input data 大小為 5000*256)，再用這三個分類模型個別預測 45000 unlabel examples，若某 unlabel example 被這三種模型預測的類別皆相同，則會被加進 train set。依此條件，可以額外在 train set(for next model)中加入 1011 個原為 unlabel 之 examples。

2.4 Model

```
datagen = ImageDataGenerator(
    featurewise_center=False,  # set input mean to 0 over the dataset
    samplewise_center=False,  # set each sample mean to 0
    featurewise_std_normalization=False,  # divide inputs by std of the dataset
    samplewise_std_normalization=False,  # divide each input by its std
    zca_whitening=False,  # apply ZCA whitening
    rotation_range=0,  # randomly rotate images in the range (degrees, 0 to 180)
    width_shift_range=0.1,  # randomly shift images horizontally (fraction of total width)
    height_shift_range=0.1,  # randomly shift images vertically (fraction of total height)
    horizontal_flip=True,  # randomly flip images
    vertical_flip=False)  # randomly flip images
```

這裡使用的 model 為 keras 提供的 cifar10 CNN 架構範例：Convolution2D→ Convolution2D→MaxPooling2D→Convolution2D→Convolution2D→MaxPooling2D→Flatten→Dense(512)→Dense(10)，過程中三次 dropout 的比例依序為 0.25、0.25 與 0.5，Activation 皆為 relu(最後一次是 sigmoid)，另外 loss 是用 categorical_crossentropy 計算，optimizer 是用 SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)，metrics=['accuracy']，因為 train set 較小所以也有使用 data augmentation 做資料的前處理(設定如左圖)。此 model 使用 cluster 產生的 6011 個 examples 當作 train data，每個 data 使用原本的 3072 features，而每個 data 的 label 有些(1011 個 unlabel examples)是在 cluster 預測的結果或有些是原本就有的。當 epoch=1000、batch_size=50 時，kaggle 的 acc 達 0.64480，而在同樣的 model 架構下，若 train data 只有原來的 5000 個 labeled examples(也就是 supervised learning)，kaggle 的 acc 高達 0.71560，前面第 1 題提到在此 model 下做的 validation 的 acc 與 kaggle 的 acc 差不多，所以幾乎可以確定此分類效果並不太好，很有可能是因為 Autoencoder 產生的 represent code 還不足以清楚地代表這些 train data，所以在分類上出現如此偏頗的錯誤(都已經用 3 種不同分類模型加強驗證了)。

3. semi-supervised learning

這裡實作的方法是用前面提到的 keras 提供的 cifar10 CNN 架構範例做修改，因為有在網路上查到一些做 image 分類的實作方法都使用 LeakyReLU 且 layer 數更多(試過當多加上一層 model.add(Dense(512))時 train set 的進步程度其實

不明顯，但若改為 model.add(Dense(1024))則 train set 的 acc 上升速度會變快)，另外 optimizer 改為 keras 官網提供的 Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.0)的話 train set 的 acc 上升速度會變快。這題的 semi-supervised learning 就是基於這個被修改過的架構反覆做 train the train set→predict unlabel datas→add part of unlabel datas into train set，其中在衡量是否要

1.