# Machine Learning Homewrok#3

Wei-Rou Lin, R05922128

## 1 Supervised learning

For this task, I used keras with tensorflow backend to implement a DNN model, and choosing the model trained to produce the best validation accuracy. On Keras, this model give an accuracy slightly better than the baseline as shown in the public leaderboard.

### 1.1 Model

The DNN model consists of 3 convolution layers and 3 dense layers with some activation (ReLU/Softmax), maxpooling, and dropout layers. To train this model, I selected categorical cross-entropy as my loss function, and applying Adam to optimize the loss.

### 1.2 Training

At the training stage, the batch size 300 and epoch number 1000 was pick out, with a checkpointer monitoring the validation accuracy, stopping the training process and recording the current model once the validation accuracy starting to descend.

## 2 Semi-supervised method 1

I use a simple self-training model following the low-density separation assumption, hard labeling all of the unlabeled data using the model trained on the labeled datas. However, this model seems to be too simple to help on getting a better accuracy than the supervised one, even if the whole training data is ten times larger.

### 2.1 Model and Training

At first, the model fit the labeled datas, classifying the unlabeled datas and then training on all of them at last. The layers and the training configuration are equivalent to the supervised ones.

## 3 Semi-supervised method 2

The autoencoder and cluster method had been used. The autoencoder is some DNN models of keras, and the clustering method was described at the class, The autoencoder and cluster method had been used. The autoencoder is some DNN models of keras, and the clustering method was described at the class, implemented with data spreading class in scikit-learn.

I've tried some different models of autoencoder and cluster in this method, but the validation accuracy appeared to be much worse than the above two methods.

### 3.1 Model

**Autoencoder**

- CNN with maxpooling
- Dense layers
- CNN and Dense

Table 1: The accuracy scores

| Method | Kaggle private set | Validation |
|---|---|---|
| Supervised | 0.538 | 0.499 |
| Self-training | 0.519 | 0.429 |
| Label Spread | 0.290 | 0.330 |

The encoders are set from 3-4 CNN and dense layers, i.e. the whole autoencoder models being trained should have 6-8 layers. The binary cross entropy or mean- square-error loss function are used to estimate the models.

**Cluster**

- KNN kernel
- RBF kernel

The propagation repeat for some iterations, in the end of which the process would label some of the most uncertain points estimated with the entropy of the distribution.

### 3.2 Training

**Autoencoder** The batch size is 128, and the Nadam optimizer was applied, other configuration are the same as the above two methods.

**Cluster** The best model propsed by the validation accury among the iterations would be saved in the end of the training process.

## 4 Results and Discussions

The score on the Kaggle and the validation accuracy are in Table 1. The label spread method in this table use RBF kernel and a CNN/Dense mixture model of autoencoder.

A normalization with shifting mean to zero and dividing all by 255 is applied to the first and the second method. The normalization datas yeild smaller loss on both training set and validation set. The third model with the binary-cross- entropy loss function wouldn't been trained with the negative datas, so the shifting was removed.

All of the supervised DNN models, include the autoencoder, could predict better by training the model more carefully because the training set accuracy converged at about 60% (the loss of autoencoder converged at about 0.6), which means the models are just underfitting. The unsupervised training accuracy is much higher because of the large proportion of unlabeled data.

However, as I switch the optimizer, adjusting the learning rate and the batch size, a better trained model had not yet discovered.