Machine Learning Assignment 3

Tung-Chun, Chiang R05922027

November 15, 2016

1 Experiment Setup

We use cifar10 dataset with 5000 labelled images and 45000 unlabelled images. We use 10% labelled data as validation set.

2 Supervised Learning

For supervised learning, we use vanilla convolution neural network with 3 convolution-maxpooling layers, 2 fully-connected layers and each layer has 0.7 probability to be dropped out. Table 1 shows the performance of this model.

$Accuracy_{in}$	$Accuracy_{val}$	$Accuracy_{kaggle}$
0.9013	0.61	0.595

Table 1: Performance of vanilla CNN

3 Semi-supervised Learning 1

We use self-training CNN as Method 1 in semi-supervised learning task.

First, we train a original CNN until validation accuracy(val_acc) reaches 0.55(val_th). Once we reach val_th, we add unlabelled data with top_k probabilities for each label. top_k is set initially 450 and increases 50 after each adding unlabelled data and the val_th is also set to new validation accuracy(best validation accuracy). Note that we re-assign unlabelled data when val_acc is higher than val_th. For example, after the first assigning, total training data size is 4500 + 450*10, and after the second assigning, total training data size is 4500 + 500*10. To save training time cost, we don't re-train model after each assigning.

$Accuracy_{in}$	$Accuracy_{val}$	$Accuracy_{kaggle}$
0.9984	0.628	0.62520

Table 2: Performance of self-training CNN

4 Semi-supervised Learning 2

We use convolutional auto-encoder as Method 2 in semi-supervised learning task. This method has 2 parts: an auto-encoder and a vanilla DNN.

In auto-encoder part, we build encoder with 3 convolution-maxpooling layers and 1 fully-connected layer. So decoder has 1 fully-connected layer and 3 deconvolution-unpooling layers, all weights are tied with corresponding layers of the encoder and use square error. To train an auto-encoder, we use all data: training, validation, testing data as training data of auto-encoder and normalize each image with mean and standard deviation of all training data. To use "relu" activation and enlarge the loss, we use e^{pixel_value} rather than $pixel_value$ so that all value are positive and exponential scaling.

In DNN part, we simply use encoded vectors of labelled data to train a vanilla DNN classifier.

$Accuracy_{in}$	$Accuracy_{val}$	$Accuracy_{kaggle}$
0.9569	0.422	0.43920

Table 3: Performance of Convolutional Autoencoder

5 Comparison and Analysis

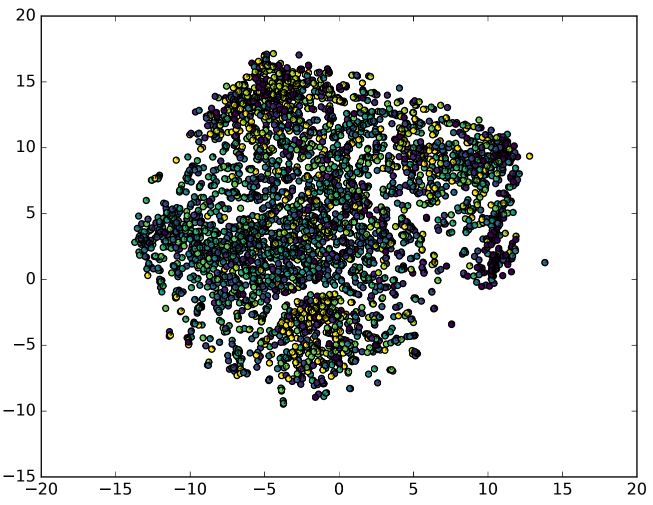

Figure 1: Auto encoder vectors of labelled data

Self-training is a simple method and has good performance, but it has lots of heuristic tricks such as "when to predict and assign unlabelled data", "how many unlabelled data to assign in training data", "whether to re-train model after each assigning unlabelled data", "whether to re-assign unlabelled data", ... It's hard to choose the best heuristic trick.

Auto-encoder is a dimension reduction method. It can also be seen a soft clustering. But it is hard to train a good auto-encoder. We also try PCA to replace auto-encoder, but the performances are almost the same. Figure 1 shows the encoded vectors, and the same color means the same label. Apparently, it is not good enough to use this encoder for semi-supervised learning.