ENEE633 Project 2 / Yang Xie

1. Project description

In this project, a Support Vector Machine (SVM) classifier and a Convolutional Neural Network (CNN) are implemented to classify the MNIST dataset.

- Dataset: MNIST dataset from http://yann.lecun.com/exdb/mnist/. It has a training set with 60000 28x28 gray-scale images of handwritten digits (10 classes). The testing set has 10000 images with the same size.
- SVM: The LIBSVM toolbox is used for the SVM classifier. Three different kernels (linear, polynomial, RBF) are evaluated. Before training, the data is reduced to a lower dimension using PCA and LDA methods.
- CNN: The Caffe toolbox is used for building the CNN.

2. SVM

2.1 Introduction

For 2 class problem, given a training set of instance-label pairs $(\mathbf{x}_i, y_i), i=1, \ldots, l$, the SVM find a separating hyper-plane (weight vector $\mathbf{w}$ and threshold b) with maximum margin by solving the following optimization problem.

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\phi(\mathbf{x}_i)+b) \geq 1-\xi_i, \\ & \xi_i \geq 0. \end{aligned}$$

The training vector $\mathbf{x}_i$ can be mapped to a higher dimensional space using kernel functions such as

- Linear: $K(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
- Polynomial: $K(\mathbf{x}_i, \mathbf{x}_j) = (\Upsilon\, \mathbf{x}_i^T \mathbf{x}_j + r)^d$, $\Upsilon>0$
- Radial basis function (RBF): $K(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\Upsilon||\mathbf{x}_i - \mathbf{x}_j||^2)$, $\Upsilon>0$

The LIBSVM toolbox supports these kernel functions.

2.2 From 2-class to multi-class

For k-class problem, a one-against-all approach can be used to train k SVM models, each one separating one class from the rest. During testing, the test data is input to these k SVM models and the LIBSVM toolbox can report the probability estimate of being in a class. The test data is assigned to the class with the highest probability.

2.3 LIBSVM toolbox

The *svmtrain* function and the *svmpredict* function in the LIBSVM toolbox are used for training and testing the SVM model.

```
model = svmtrain(y, X [, 'libsvm_options']);
```

```
        [predict_label, accuracy, decision_values/prob_estimates] =
                svmpredict(y, X, model [, 'libsvm_options']);
```
The options and results of these two functions can be found in the README file of the LIBSVM toolbox and https://www.csie.ntu.edu.tw/~cjlin/libsvm/. *In this project, default values are used for the parameters of each kernel function.*

2.4 Dimensionality reduction

To speed up the training process, the data can be reduced to a lower dimension using PCA or LDA methods. In this project, I try both methods and compare the classification results. For both methods, the dimensionality is reduced from $d = 784$ to $p = 9$.

3. CNN

3.1 Introduction

Convolutional Neural Networks (CNN) exploit spatially local correlation by combining three architectural ideas: local receptive fields, weight sharing and subsampling (pooling). The LeNet CNN architecture is shown in Figure 1.

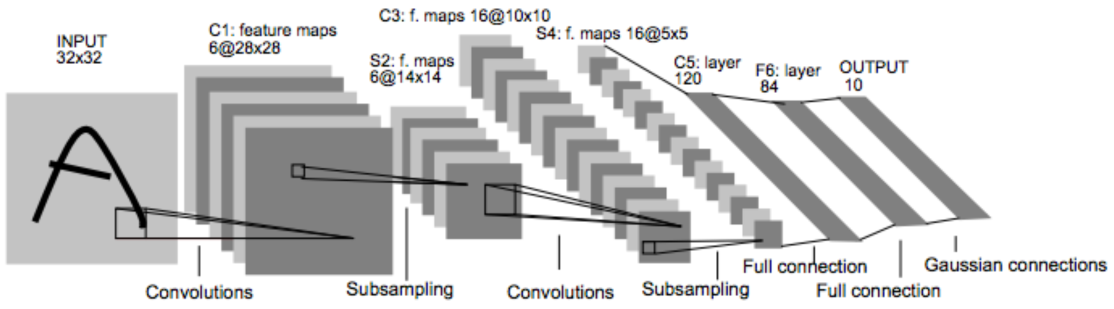

Figure 1

A CNN consists of several layers, which can be one of the four types:

- **Convolutional:** a convolutional layer at layer m consists of multiple (say k) feature maps, which maps the hidden units at layer m-1 to hidden units at layer m with different weight W^i (i=1, ..., k).
- **Sub-sampling:** after each convolutional layer, there may be a subsampling layer which divides a feature map of the convolutional layer into non-overlapping blocks and subsamples it by producing a single output from the block. Max-pooling uses maximum function to perform the sampling.
- **Inner-product:** after several convolutional and max-pooling layers, a fully-connected inner-product layer takes all the hidden units in the previous layer and connect them to every hidden units in this layer. A RELU layer can be added to the output of the fully connected layer in order to use the rectified linear unit $f(x) = max(0,x)$ as the activation function, which was found to greatly accelerate the convergence of stochastic gradient descent compared to sigmoid/tanh functions.
- **Output:** a SOFTMAX layer can be utilized to handle multi-class problem.

3.2 CNN architecture used

The CNN architecture used in this project is the same as the example shown in the Caffe toolbox, which is illustrated in Figure 3.

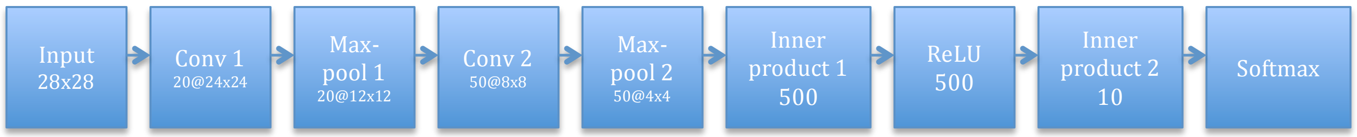

Figure 3

- Conv 1: num_output = 20, kernel_size = 5, stride = 1;
- Max-pool 1: kernel_size = 2, stride = 2;
- Conv 2: num_outputs = 50, kernel_size = 5, stride = 1;
- Max-pool 2: kernel_size = 2, stride = 2;
- Inner-product 1: num_outputs = 500;
- ReLU
- Inner-product2: num_outputs = 10;
- Softmax

4. Experiments and Results

Table I shows the classification results of both the SVM classifier and the CNN classifier. For SVM, both PCA and LDA are implemented to reduce the dimensionality. Besides, three different kernel functions are experimented. For CNN, the architecture setup is described in Section 3.2.

Method	SVM						CNN
	PCA (p=9)			LDA (p=9)			
	Linear	Poly	RBF	Linear	Poly	RBF	
Accuracy	75.43%	91.63%	93.15%	88.40%	88.43%	90.17%	99.03%

Table I

- Comparing PCA and LDA methods, we can see that LDA can have better accuracy only when linear kernel is used. It's because that the LDA linearly discriminates the original data, instead of discriminating the high-dimension data mapped by the kernel functions. Overall, PCA has better performance.
- Comparing different kernel functions, we can see that the linear kernel performs the worst and the RBF function performs the best for both the PCA case and the LDA case. It's because the RBF function maps the data into infinite dimensions and thus can have better performance.
- Comparing the best result of SVM and the result of CNN, we can see that CNN has higher classification accuracy. It's mainly because that the CNN exploits the spatially local correlation of an image.