ENEE439M - Project 2

Anurag Bansal

Maryland, United States

Abstract

The aim of this project is to implement different conventional classifiers to achieve hand-written digits recognition. The two classification techniques used in this project are Support Vector Machines and LeNet a Convolutional Neural Network. The dataset used is the hello world from the CNN technique, that is the 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.

1. Support Vector Machine

1.1. Implementation Theory

1.1.1. Introduction

A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. In two dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side. The way the SVM works is, it tries to maximize the distance of the separating boundary between the two classes, by maximizing the distance of the separating hyperplane from each of the feature vector irrespective of the class. We won't be discussing the underlying math in this report we will only be highlighting the results of the various experiments. However, this is a constrained optimization problem which can be solved using Lagrange multiplier.

1.1.2. Regularization Parameter

The Regularization parameter (often termed as C parameter in LibSVM library tells the SVM optimization how much you want to avoid misclassifying each training example.

For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane mis-classifies more points.

1.1.3. Gamma

The gamma parameter defines how far the influence of a single training example reaches, with low values meaning far and high values meaning close. In other words, with low gamma, points far away from plausible separation line are considered in calculation for the separation line. Where as high gamma means the points close to plausible line are considered in calculation.

1.1.4. Kernel Trick

Some common kernels include:

- Polynomial (homogeneous): $k(\vec{x_i}, \vec{x_j}) = (\vec{x_i} \cdot \vec{x_j})^d k(\vec{x_i}, \vec{x_j}) = (\vec{x_i} \cdot \vec{x_j})^d$
- Polynomial (inhomogeneous): $k(\vec{x_i}, \vec{x_j}) = (\vec{x_i} \cdot \vec{x_j} + 1)^d k(\vec{x_i}, \vec{x_j}) = (\vec{x_i} \cdot \vec{x_j} + 1)^d$
- Gaussian radial basis function: $k(\vec{x_i}, \vec{x_j}) = \exp(-\gamma \|\vec{x_i} - \vec{x_j}\|^2) k(\vec{x_i}, \vec{x_j}) = \exp(-\gamma \|\vec{x_i} - \vec{x_j}\|^2)$, for $\gamma > 0 \gamma > 0$. Sometimes parametrized using $\gamma = 1/2\sigma^2 \gamma = 1/2\sigma^2$

1.2. 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.

1.3. Toolbox and Helper Function

The toolbox used for the SVM implementation is the LIBSVM by R.-E. Fan, P.-H. Chen, and C.-J. Lin. Libsvm is a simple, easy-to-use, and efficient software for SVM classification and regression. It solves C-SVM classification, nu-SVM classification, one-class-SVM, epsilon-SVM regression, and nu-SVM regression. The toolbox can be downloaded from https://www.csie.ntu.edu.tw/ cjlin/libsvm/. I have also used the helper functions from Stanford to read the data files. http://ufldl.stanford.edu/wiki/index.php/Using_the_MNIST_Dataset

1.4. Experiment

We run several experiments using the MNIST dataset. We use different kernels to test the performance and we also use the dimensionality reductions techniques like Linear Discriminant Analysis and Principal Component Analysis to reduce the dimensionality of the data before feeding it for training or testing. The dimensions are reduced to 9 (c-1) to perform the tests. The number 9 is chosen specifically due to LDA, as the MNIST dataset has 10 (c) classes.

1.5. Results

Method	Linear	Poly	RBF
PCA	82.79%	90.89%	93.62%
LDA	89.28%	86.43%	88.5%

1.6. Discussion

Comparing PCA and LDA methods, we can see that LDA can have better accuracy only when linear kernel is used. Its 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. Its because the RBF function maps the data into higher dimensions and thus can have better performance. Difference can also be noticed on training times. Where the RBF takes much more time as compared to Linear method. Using the Polynomial kernel we have a less efficient performance using this kernel function with a high training cost and low error rate for the extracted features.

2. Convolutional Neural Net

2.1. Implementation Theory

2.1.1. Introduction

Convolutional Neural Networks are are a special kind of multi-layer neural networks. Like almost every other neural networks they are trained with a

version of the back-propagation algorithm. Where they differ is in the architecture.
Convolutional Neural Networks are designed to recognize visual patterns directly from pixel images with minimal preprocessing. They can recognize patterns with extreme variability (such as handwritten characters), and with robustness to distortions and simple geometric transformations.
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 below -

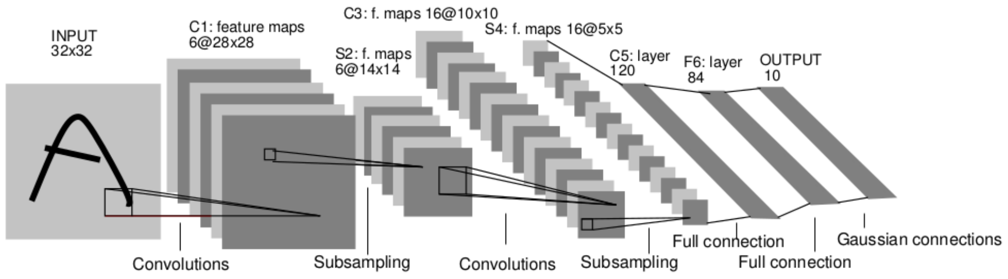

Figure 1: Network Architecture

2.1.2. Caffe and Classification Model

We are using Caffe to execute this part of the project. Before we begin to train model for MNIST, wed better spend some time on the philosophy and key concepts of Caffe. The philosophy of Caffe is expression, speed, modularity, openness and community. The models and optimizations in Caffe are defined as plaintext schemas instead of code. Caffe supports many types of data layer. Here I access data from LMDB, one of the lightning and efficient key-value databases. Comparing to HDF5, another type of data layer, it uses memory-mapped files, and doesnt need to load whole dataset in memory, so its suitable for large datasets.
lenet_train_test.prototxt is almost the direct translation of the LeNet model on paper except the slight difference in the output layer. We replace the Gaussian connections to Rectified Linear Unit(ReLU) activation.

2.1.3. Network Architecture

The CNN architecture used in this project is the same as the example shown in the Caffe toolbox, which is illustrated in figure below -

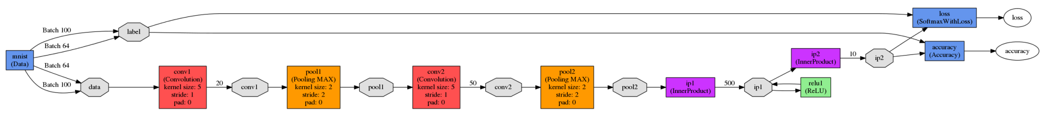

Figure 2: Network Architecture

2.2. Experiment

We ran the MNIST example from the Caffe toolbox to obtain the results. No changes were made to the prototxt file.

2.3. Result

The experiment gave constant result varying from test net out = 99.03% to 99.08% and loss = 0.026% after performing the tests repeatedly.

2.4. Discussion

Comparing the best result of SVM and the result of CNN, we can see that CNN has higher classification accuracy. Its mainly because that the CNN exploits the spatially local correlation of an image.
LeNet is known as its ability to classify digits and can handle a variety of different problems of digits including variances in position and scale, rotation and squeezing of digits, and even different stroke width of the digit. Meanwhile, with the introduction of LeNet, LeCun et al. (1998b) MNIST database, is the standard benchmark in digit recognition field.