Handwritten Digit Recognition – EE5907R CA2

Introduction

Weight 20% of final grade

Goal For the course project, students will work individually to construct a pattern recognition system consisting of cutting-edge techniques for handwritten digits recognition. Students will be encouraged to apply Principal Component Analysis (PCA) to perform dimensional reduction and visualization, in order to understand underlying data distribution. Then students will apply two popular classification models – Linear Discriminative Analysis (LDA) and Support Vector Machine (SVM) – to classify the digit images. The models will be introduced in the module. In recent years, deep Convolutional Neural Networks (CNN) have attracted great interest from various research communities and industries, due to their excellent performance in real applications. Students will be encouraged to construct a simple CNN model and apply the model for classifying the digit images. Through the project, students are expected to gain basic and important knowledge of currently popular pattern recognition techniques. and get sense of deep learning methods. Throughout the project, students will use the MNIST dataset for developing models, doing experiments, evaluating the models and making observations. The MNIST dataset contains $60,000$ training images, and $10,000$ testing images. The sample size is large enough for training a complicated model (*e.g.* deep CNN) with many free parameters to learn. It is usually used in practice as a test bed for developing new pattern recognition models and methods, and is good as a starting dataset to play with for pattern recognition practicer and learner.

Programming Language Students may use any language of their choosing for the project, though at least starting with the MATLAB. MATLAB is recommended, because it provides a simple, complete environment for implementing algorithms, running experiments, and visualizing results. Also, there are many off-the-shelf packages for pattern recognition can be used with MATLAB. In particular, following two toolboxes implemented by MATLAB can be found on the web.

Toolbox

- **SVM package**. LibSVM, an excellent SVM classifier toolbox, is widely used in practice. Students are encouraged to learn how to use this toolbox to develop linear SVM classifier as well as non-linear (kernel) SVM models. The toolbox can be downloaded from `https://www.csie.ntu.edu.tw/~cjlin/libsvm/`. Several documents for beginner can also found on the website.

- **CNN package**. Many open-source CNN or deep neural network packages can be found on the web in nowadays. Considering GPU may not be available, students are encouraged to use following two packages that are easy to start with and flexible for creating networks with various architectures.
  - **TensorFlow** `https://www.tensorflow.org/`
  - **PyTorch** `http://pytorch.org/`
  - **MatConvNet**. MatConvNet is easy to use for beginners who are more familiar with MATLAB. But it is rarely used in practical deep learning model development. MatConvNet can be downloaded via `http://www.vlfeat.org/matconvnet/`. Mannual can be found at
    `http://www.vlfeat.org/matconvnet/matconvnet-manual.pdf`.
    Good tutorial for training your own networks (`http://www.vlfeat.org/matconvnet/training/`) is also provided.
  - **Caffe**. Caffe is developed and maintained by Berkeley vision group. The package can be found at `http://caffe.berkeleyvision.org/`. Examples for training your own network is also given at `http://caffe.berkeleyvision.org/gathered/examples/mnist.html`. There is a good community for Caffe users:
    `https://groups.google.com/forum/#!forum/caffe-users`.

Dataset The project will be conducted on the MNIST dataset. The MNIST dataset contains in total $70,000$ handwritten digit images, which are divided into a training set of $60,000$ examples, and a test set of $10,000$ examples. The digits have been size-normalized and centered in a fixed-size image. The raw data and labels can be downloaded from `http://yann.lecun.com/exdb/mnist/`.

Submission Submit your report as a pdf file named YOUR_NAME.pdf. Submit any supplementary material as a single zip file named YOUR_NAME.zip. Add a README file describing the supplemental content. The report should include experimental results and analysis on

- PCA based data distribution visualization
- PCA plus nearest neighbor classification results
- LDA based data distribution visualization
- LDA plus nearest neighbor classification results
- SVM classification results with different parameter values
- CNN classification results with different network architectures (optional).

Source code with good documentation should be submitted in separate folders: including codes for PCA, LDA, SVM respectively. The code for LibSVM toolboxe does not need to included in the submission. Add a README file to describe how to run the codes.

Deadline 2400 GMT+8, Apr 22, 2018.

PCA for Feature Extraction, Visualization and Classification

The size of raw MNIST image is 28×28 pixels, resulting in a 784 dimensional vector for each image. Apply PCA to reduce the dimensionality of vectorized images from 784 to 2 and 3 respectively. Visualize the projected data vector in 2d and 3d plots. Also visualize the eigenvectors of sample covariance matrix used in PCA.

Apply PCA to reduce the dimensionality of raw data from 784 to 40, 80 and 200 respectively. Classifying the test images using the rule of nearest neighbor. Report the classification accuracy.

Denoted the reduced dimension as d with $d \leq 784$. Investigate what value of d preserves over 95% of the total energy after dimensionality reduction. Apply PCA to reduce the data dimension to d and report classification results based on nearest neighbor. Can you devise other criteria for automatically determining the value of d?

LDA for Feature Extraction and Classification

Apply LDA to reduce data dimensionality from 784 to 2, 3 and 9. Visualize distribution of the data with dimensionality of 2 and 3 respectively (similar to PCA). Report the classification accuracy for data with dimensions of 2, 3 and 9 respectively, based on nearest neighbor classifier. Test the maximal dimensionality that data can be projected to via LDA. Explain the reasons.

SVM for Classification

Use the raw digit images (vectorized) and the data vectors after PCA pre-processing (with dimensionality 40, 80 and 200) as inputs to *linear* SVM. Try different values of the penalty parameter C in $\{1 \times 10^{-2}, 1 \times 10^{-1}, 1, 10\}$. Report the classification accuracy with different parameters and dimensions. Explain the effect of data dimension and parameter C on the final classification accuracy.

Apply kernel SVM with radial basis kernel for classification. Tune the parameters and report the parameters providing best classification performance. Compare the results with the one given by linear SVM and explain the observations.

Neural Networks (optional)

Read the documentation for training a simple convolutional neural network (ref. `http://www.vlfeat.org/matconvnet/training/`). Train a CNN with two convolutional layers and one fully connected layer, with the architecture specified as follows: number of nodes: 20-50-500-10. The number of the nodes in the last layer is fixed as 10 as we are performing 10-category classification. Convolutional kernel sizes are set as 5. Each convolutional layer is followed by a max pooling layer with a kernel size of 2 and stride of 2. The fully connected layer is followed by ReLU. Train the network and report the final classification performance. Compare the results with the one reported in the leading board: `http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html`.

Change the network architecture (including the number of layers, type of layers, number of nodes per layer) and report the performance provided by other CNNs with different architectures. Analyze the performance vs. network architecture.

References