Face Recognition
– EE5907 CA2

Introduction

Weight 20% of final grade

Goal For the course project, students will work *individually* to construct a face recognition system. Students are required to apply Principal Component Analysis (PCA) to perform data dimensionality reduction and visualization, in order to understand underlying data distribution. Then students are required to train and apply three classification models – Linear Discriminative Analysis (LDA), Support Vector Machine (SVM) and Convolutional Neural Networks (CNN) – to classify the face images. Through the project, students are expected to gain basic and important knowledge of currently popular pattern recognition techniques.

Programming Language Students may use any language of their choosing for the project, though at least starting with the MATLAB or Python. MATLAB is recommended, because it provides a simple, complete environment for implementing algorithms, running experiments, and visualizing results. In particular, following toolboxes implemented can be used in the project and 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/`[1]. 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/`

---

[1]If the link is broken, please Google "libsvm".

- **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 CMU PIE dataset[2] and the face photos taken by the students themselves. There are in total 68 different subjects. Please choose 20 out of them by yourself and indicate your choice in your report. For each subject, use 70% of the provided images for training and use the remaining 30% for testing. Besides the provided CMU PIE, please take 10 face photos for yourself, resize them into the same resolution as the CMU PIE images and split them into 7 for training and 3 for testing. Put the 7 self-photos into the CMU PIE training set and 3 self-photos into the CMU PIE test set.

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

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

Deadline   2400 GMT+8, Apr 26, 2019.

---

[2]The data can be downloaded from IVLE.

PCA for Feature Extraction, Visualization and Classification

The size of raw face image is 32×32 pixels, resulting in a 1024 dimensional vector for each image. Randomly sample 500 images from the CMU PIE training set and your own photos. Apply PCA to reduce the dimensionality of vectorized images to 2 and 3 respectively. Visualize the projected data vector in 2d and 3d plots. Highlight the projected points corresponding to your photo. Also visualize the corresponding 3 eigenfaces used for the dimensionality reduction.

Then apply PCA to reduce the dimensionality of face images to 40, 80 and 200 respectively. Classifying the test images using the rule of nearest neighbor. Report the classification accuracy on the CMU PIE test images and your own photo seperately.

LDA for Feature Extraction and Classification

Apply LDA to reduce data dimensionality from to 2, 3 and 9. Visualize distribution of the sampled data (as in the PCA section) 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. Report the classification accuracy on the CMU PIE test images and your own photo seperately.

SVM for Classification

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

Neural Networks

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: 20-50-500-21. The number of the nodes in the last layer is fixed as 21 as we are performing 21-category (20 CMU PIE faces plus 1 for yourself) 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.