Face Recognition

EE5907 CA2

Deadline: November 12, 2021 at 5pm
Submitted via Luminus

Introduction

Goal For the course project, you will work *individually* to construct a face recognition system. You are required to apply Principal Component Analysis (PCA) to perform data dimensionality reduction and visualization, in order to understand underlying data distribution. You 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.

Programming Language You must use 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. You 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/ (if the link is broken, Google "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. Mat- ConvNet can be downloaded via http://www.vlfeat.org/matconvnet/. Manual 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.

Dataset The project will be conducted on the CMU PIE dataset (which can be downloaded from LumiNUS) and the face photos taken by the students themselves. There are in total 68 different subjects. Please choose 25 out of them by yourself and indicate your choice in your report. For each chosen subject, use 70% of the provided images for training and use the remaining 30% for testing. Besides the provided CMU PIE images, please take 10 selfie photos for yourself, convert to grayscale images, 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 selfie-photos into the training set and 3 into the 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 be included in the submission. Add a README file to describe how to run the codes.

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 im- age. 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 neighbour. Report the classification accuracy on the CMU PIE test images and your own photo separately.

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 neighbour classifier. Report the classification accuracy on the CMU PIE test images and your own photo separately.

GMM for clustering

Use the raw face images (vectorized) and the face vectors after PCA pre-processing (with dimensionality of 200 and 80 respectively) as the provided training data. Train a GMM model with 3 Gaussian components on these data. Visualize the clustering results from GMM, i.e., visualize the images that are assigned to the same component.

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 for Classification

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 per- forming 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.