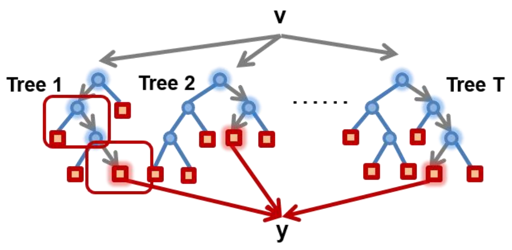
Machine Learning for Computer Vision

Manifold Learning, Online Learning, Discriminant Analysis,Ensemble Learning, Randomised Forests



Python programming.

Experiment with face dataset

Use the provided face data (face.mat), which stores raster-scanned face images (46x56 pixels) in columns. In all questions, you can use any existing toolbox/code, if needed.

Q1.

Eigenfaces

a. Partition the provided face data into your training and testing data, e.g. 8 images and 2 other images of each face identity for training and testing respectively. Apply PCA to your training data, by computing the eigenvectors and eigenvalues of the covariance matrix $S=(1/N)AA^T$ directly. Show and discuss the results, including: the eigenvectors, the eigenvalues, and the mean image, how many eigenvectors with non-zero eigenvalues are obtained and how many eigenvectors are to be used for face recognition. Give insights and reasons behind your answers.

b. Apply PCA to your training data, using the eigenvectors and eigenvalues of $(1/N)A^TA$. Show and discuss the results in comparison to the above, including: if the eigenvectors and eigenvalues obtained are identical, what are the pros/cons of each method. Show respective measurements for your answers.

>> Hereinafter, we use the low-dimensional PCA technique, wherever needed.

Application of Eigenfaces

Use the data partition, which you used in Q1, into training and testing.

a. Perform the face image reconstruction using the PCA bases learnt. Show and discuss the results, while varying the number of bases to use, including: if the reconstruction error (or the distortion measure) obtained is same as in the theory, how good the reconstruction results are for at least 3 images of your choice (e.g. from both the training and testing dataset).

b. Perform the PCA-based face recognition by either the NN classification method or alternative method learnt in the PCA lecture. Report and discuss, including: the recognition accuracy (success rates), example success and failure cases, the confusion matrices, time/memory (and any other aspects you observe), by varying the parameter values/experimental settings you used. Give insights and reasons behind your answers.

Q2. Incremental PCA

Use the same data partition into training and testing as in Q1. Further divide the training data equally into four subsets, each with 104 images (i.e. two images per person). Starting with the first subset, keep adding a more subset into your training.

Perform Incremental PCA, and compare it with the counterpart i.e. batch PCA, and PCA trained only by the first subset, in terms of *training time, reconstruction error, face recognition accuracy (using NN classification).

Show and discuss, including: how accurate your incremental method is, what important parameters in the method are (and how they are set). Provide your own discussions and measurements to support.

*Note: You might ignore all other computations e.g. constructing covariance matrices, orthonormalization, matrix products, (which can be accelerated by proper implementations) than eigen-decompositions.

Q3. LDA Ensemble for Face Recognition

Use the provided face data, and the same data partition into training and testing as in Q1.

Try PCA-LDA and its ensemble learning, along with the NN classifier. Compare and discuss face recognition results.

PCA-LDA

Perform the PCA-LDA based face recognition with the NN classifier. Report and discuss, including:

- recognition accuracies by varying the parameter values, M_pca and M_lda
- ranks of the scatter matrices,
- the confusion matrix, example success and failure cases

Explain your observations and reasons, and discuss the results in comparison to those of **Q1**.

PCA-LDA Ensemble

Show, measure and discuss the results, including:

- randomisation in feature space
- randomisation on data samples (i.e. bagging)
- the number of base models, the randomness parameter,
- the error of the committee machine vs the average error of individual models
- fusion rules
- recognition accuracy and confusion matrix

Observe and discuss the above by varying the parameter values/architectures you used. Give insights and reasons behind all your answers.

Q4. Generative and Discriminative Subspace Learning

PCA is a generative model, by which input images or data can be reconstructed. LDA is a discriminative model, which extracts better features for classification. Say we are interested in subspace learning that fulfils both aspects or controls a balance between the two aspects.

Mathematically formulate the problem (i.e. the objective or goal function to optimise) that learns the subspace for reconstruction and discriminative features at the same time.

And mathematically derive the solution that optimises the defined problem. If needed, you may use Lagrange multiplier formulation, gradient-based optimization, eigenvector-eigenvalues, and/or generalized eigenvector-eigenvalues.

Discuss foreseeable behaviours, and pros and cons, of your method. No programming is needed.

Q5. RF classifier

Train and test Random Forest using the training and testing data set as in Q1. Change the RF parameters (including the number of trees, the depth of trees, the degree of randomness parameter, the type of weak-learners: e.g. axis-aligned, two-pixel test, or any other features you can think of), and show and discuss the results:

- recognition accuracy, confusion matrix,
- example success/failures,
- time-efficiency of training/testing,
- impact of the different types of weak-learners

Discuss the results in comparison to those of **Q1, Q3.**

Q6. Multi-class SVM for Face Recognition

Use the provided face data, and the same data partition into training and testing as in Q1.

Feature vectors **x** are the raw-intensity vectors (obtained by raster-scanning pixel values of face images) or PCA coefficients. Try both and compare the results below.

Train and test multi-class SVM using the feature vectors **x**. You can use any existing toolbox for two-class (or binary-class) SVMs. Note, write your own lines of code for the multi-class extensions of SVM (both one-versus-the-rest and one-versus-one), and provide your code in an appendix of your report. Compare the results of the two multi-class extensions of SVM.

Show, measure and discuss the results, including:

- setting the SVM parameters, i.e. kernel type, kernel parameters, C (underfitting/overfitting),
- recognition accuracy and confusion matrix
- time-efficiency of SVM training/testing
- examples of support vectors and success/failure images
- margin etc.

Discuss the results in comparison to those of **Q1, Q3, Q5.**
Give insights and reasons behind all your answers.