Exercise 4: Linear Discriminant Analysis

Lecture Information Processing and Communication

Jörn Anemüller, May 2022

Submit solutions until Tuesday 2022-05-17, 23:59h, by uploading to your group's exercise folder on cs.uol.de. You may submit your solutions in groups of at most two students. You are free to write your code in matlab or in python (but we provide the example functions in matlab only).

Summary of exercise 4

The goal of this exercise is to implement classification using the linear discriminant analysis (LDA) algorithm. Use the method from the lecture to implement the LDA algorithm, using the closed-form solution based on within-class and across-class scatter matrices.

List of functions to be completed

`ex4_script.m`: main script for completion of exercise 4

`matrix2vector.m`: convert data from 2-dimensional matrix form into a 1-dimensional feature vector

`vector2matrix.m`: convert back from 1-dimensional feature vector to 2-dimensional matrix

`learn_lda.m`: computes the optimal projection vector for the LDA task

`classify_lda.m`: compute the decision variable vec_y for the test dataset, using the classification model that you have learnt on the training data

`compute_accuracy.m`: compute the classification accuracy, i.e., number of correctly classified images divided by total number of images

List provided helper functions

`mnistRead.m`: Function from the handwritten postal digits dataset from `http://yann.lecun.com/exdb/mnist/`. It loads images and labels for training and testing. Data are stored in the files `train-images-idx3-ubyte.gz`, `train-labels-idx1-ubyte.gz`, `t10k-images-idx3-ubyte.gz` und `t10k-labels-idx1-ubyte.gz`.

`mnistShow.m`: Function to display digits from the dataset

1. Main script for this exercise session

Edit the script `ex04_script.m`, it contains template code for the main steps of this exercise.

2. Loading image data

The mnist dataset is read by the function call `[train_images, train_labels, test_images, test_labels] = mnistRead()` which returns matrices and vectors for training and test images and labels.

3. Converting data matrix to feature vector

Convert each image in the multi-dimensional data array into a feature vector that is used for optimization. Simple reshaping of the data does the job, and should be performed for each image matrix stored in the multi-dimensional data array. The corresponding inverse transformation is needed to visualize the obtained weight vector.

```
function [mat_features, NCol, NRow, NFrame] = matrix2vector(mda_data)
function mat_out = vector2matrix(vec_in, NCol, NRow)
```

5. Perform linear discriminant analysis

Find the optimum projection weight vector and return also the decision variable vec_y for LDA method.

Function definition:

```
function [vec_w_opt_lda, vec_m1_train, vec_m2_train, vec_y_lda] = learn_lda(mat_X, vec_y_true);
```

6. Compute the classification accuracy of the trained classifier on the training and test data

To evaluate the learning success of the classifier, we use the accuracy ("percent correct") of images that have been assigned to the correct class. Do this both on the train and the test set in order to see whether the classifier successfully generalizes from training data to new, previously unseen data. Compare both results.

Function definition:

```
function accuracy_train_lda = compute_accuracy(vec_y_true, vec_y);
```

7. Classify previously unseen data from the test set portion of the dataset, but use the model you trained on the training data

To compare classification accuracy on the train and test set, you first need to take the model (i.e., weight vector) obtained during training and apply it to the test portion of the data. Look also at example plots of digits from the test data and at the estimated class labels.

Function definition:

```
function vec_y_test = classify_lda(vec_w_opt_lda, vec_m1_train, vec_m2_train, mat_X_test);
```