521285S Affective Computing (Laboratory Exercises)

Exercise – I Emotion Recognition from facial expression

Objective

Your task is to extract the spatiotemporal features (i.e., features of local binary pattern from three orthogonal planes, LBP-TOP) from a video. Then, an emotion recognition system is constructed to recognize happy versus sadness facial expressions (binary-class problem) by using a classifier training and testing structure.

The original facial expression data is a sub-set of eNTERFACE (acted facial expression), from ten actors acting happy and sadness behaviors. The used dataset in the exercise includes 100 facial expression samples.

The region of interest (i.e., facial image) is extracted using face tracking, face registration and face crop functions. Basic spatiotemporal features (i.e., LBP-TOP features) are extracted using LBP-TOP.

To produce emotion recognition case, Support Vector Machine (SVM) classifiers are trained. 50 videos from 5 participants are used to train the emotion recognition, use spatiotemporal features. The rest of the data (50 videos) is used to evaluate the performances of the trained recognition systems.

In this exercise, 4 tasks should be finished.

Tip

Use the following website to help the usage of MATLAB functions.
http://se.mathworks.com/help/matlab/
Or type 'doc xxx' (e.g., doc plot) in the command window.

Implementation

The data and toolbox files used in this exercise can be found in the Affective Computing course webpage (see the Noppa system).

Download the code for Exercise 1 ('code4exercise1.zip'). 'lab1_data.mat' containing data variables used in the emotion recognition part of this lab ('example_crop_video', 'example_img', 'model', 'model_im', 'norm_face', 'training_data', 'testing_data', 'training_class', and 'testing_class').

Tip: 'example_img' is used in the first and second tasks. 'model' and 'model_im' is used in the first tasks. 'norm_face' is used in the second tasks. Other variables are used in the third and fourth tasks.

Study the LBPTOP functions 'LBPTOP' ('LBPTOP.m', 'LBP.m'), the Chehra toolbox ('FaceTrack.m') and generic MATLAB functions 'cp2tform', 'imtransform', 'svmtrain', 'svmclassify', 'confusionmat' as they are needed in the exercise.

1. Perform face preprocessing using the provided facial image. Plot the results.

Load 'example_img' variable from 'lab1_data' in MATLAB. Set Chehra_v0.1_MatlabFit, FaceCrop, and LBPTOP files into your MATLAB toolbox path. Tip: Use 'Set Path' to update your toolbox.

Face preprocessing:

- Calculate the facial landmarks using Chehra toolbox. Use 'FaceTrack.m' to locate the 49 facial landmarks. Tip: (1) possibly for some facial images, the landmarks may be not detected. One trick is that scale an image into small one. (2) 'fitting_model' is '.\Chehra_v0.1_MatlabFit\models\Chehra_f1.0.mat', two variables are included (RegMat and refShape), and (3) the type of image input should be double, use 'im2double' to convert it.
- Register facial image into a model using 49 facial landmarks and spatial transform based on local weighted mean. Load 'model' in MATLAB. Tip: use 'cp2tform.m' to obtain the transformation matrix, and then use 'imtransform.m' to register the facial image.
    - Check the usage of 'cp2tform.m' and 'imtransform.m'. In the MATLAB document, there is an example how to use them. Please note that the default points used for local weighted mean (LWM) is 12, you should define it.
- Crop facial image based on the registered face image. Use coordinates around the eyes (landmarks indices: 21, 22, 24, 25, 27, 28, 30, 31) to locate the eye coordinate. Tip: Use 'FaceCrop.m' to crop the face region.

Task 1 Show and check the results:

- Plot facial landmarks on the image.
- Plot an registered image and facial landmarks
- Plot your cropped image. Check that you get the correct cropped image by comparing the gray histogram. Tip for plotting a histogram: (1) 2D to 1D vector, use the example like this 'vector(:)', and (2) use 'double.m' to convert 2D unit8 image into double image, and then 'hist.m' to plot the gray histogram in [0 255].

2. Feature extraction

Extract the following LBP features (R=1, P=8)

- Load 'norm_face' variable from 'lab1_data' in MATLAB.
- Generate the uniform pattern (getmapping.m)
- Construct a histogram of an image by using local binary pattern (lbp.m). Tip: the length of histogram should be 59.

Task 2-1 Show and check the results:

- Check your LBP image and histograms.

Extract the following LBP-TOP features (1x1x1 blocks, R=3, P=8)

- Load ‘example_crop_video’ variable from ‘lab1_data.mat’ in MATLAB
- Set the parameters for block size (1x1x1), radius (R) and the number of neighbors (P)
- Generate the uniform pattern (getmapping.m)
- Construct a histogram of a video by using LBP-TOP (LBPTOP.m)

Task 2-2 Show and check the results:

- Check histograms of XY, XT and YT planes. Tip: the length of each histogram should be 59.

3. Feature Classification

Use the ‘svmtrain’ function to train Support Vector Machine (SVM) classifiers. The ‘training_data’ and ‘testing_data’ matrices contain the calculated LBP-TOP features for the training and testing sets, respectively. The block size for LBP-TOP used for training and testing data are 2x2x1. The ‘training_class’ group vector contains the class of samples: 1 = happy, 2 = sadness, corresponding to the rows of the training data matrices.

- Construct an SVM using the ‘training_data’ and linear kernel

Use the ‘svmclassify’ function (and your trained SVM structures) to classify the ‘training_data’ and the ‘testing_data’ matrices. Then, calculate average classification performances for both training and testing data. The correct class labels corresponding with the rows of the training and testing data matrices are in the variables ‘training_class’ and ‘testing_class’, respectively.

Task 3 Show and check the results:

- Calculate the average classification performances for the training data (‘training_data’) and the testing data (‘testing_data’) using the corresponding trained linear SVM.

4. Plot confusion matrices

Plot confusion matrices for the training and testing data for both classifiers. Tip: Use ‘confusionmat’ function.

Task 4 Show and check the results:

- Calculate the confusion matrix of the linear kernel SVM using the ‘training_data’
- Calculate the confusion matrix of the linear kernel SVM using the ‘testing_data’