Face identification problem

Task

Challenge yourself with image classification task for solving face identification problem on ORL dataset available here.

Goal

The aim of this project is listed (but not limited to) below:

1. Familiarize yourself with EDA (Exploratory Data Analysis) technique on the concrete example with dataset of people' faces.
2. Implement simplest classification algorithm with Python language.
3. Investigate general techniques of normalization and optimization.
4. Experience yourself in report with detailed explanation of applied strategy and the achieved results.

You can familiarize yourself with the common methods in CV (Computer Vision), image processing, and face recognition on the example here.

Outline

1. Retrieve and analyse the dataset. This steps includes visual inspection of downloaded archive - its structure and content. It helps also for deciding what model most appropriate for this type of the input data.
2. Install required software, if needed. After the model is selected, tools for implementation should be enumerated and installed for further using. We suggest to you to use one from popular libraries below:
    a. OpenCV (recommended)
    b. Pillow
    c. Skimage
3. Prepare dataset by loading images from disk and represent it in form of matrix with the dimension m by n, where m - number of examples in the dataset, n - image, unfolded into vector of dimension 1 by n, where n = width * height of image.
    a. Load dataset. Output X - matrix of shape (m, n) with data, Y - vector of labels (m, 1).
    b. Split dataset by 80/20 on train / test dataset.
4. Design and implement an algorithm to solve problem with face identification task (see Algorithm section below).

5. Evaluate implemented model. The evaluation process differs in different models but, at least accuracy must be inspected as a general evaluation metric.
6. Report results (see Report section below).

Algorithm

For start point we suggested to you to use MSE method (i.e., Euclidean distance) for solving face identification problem. This is the simplest, easy to implement, and pretty accurate algorithm for this example. Before to apply it, don't forget to represent image data in the appropriate way (unfold matrix to the vector). As a result, you should get a matrix with the #rows corresponding the #images, where each row represents a single image. Also, create vector of labels for input images.
Next, you need perform next steps:

1. For each sample from **test** set perform:
   a. Calc MSE for each example from **train** set.
   b. Find predicted label for test sample using by calculated mse values (hope you will).
2. Calc accuracy (divergence between predicted labels and ground truth).
3. Add retrieved value to the report.

Optimization / Normalization steps

After you've got your first good results (I hope you did) from applied algorithm, the next step is to improve your model by adding normalization and/or optimization steps to your existing processing steps. Notice, all improvements should have a report. We suggest to you consider next methods:

1. Image normalization (mandatory step). Normalization or feature scaling is a process of preprocessing data features for future processing. It allows such tasks as gradient descent in linear regression models converge more quickly. Results of influencing data normalization should be included into report. There are two approaches to normalize dataset (both of them should be reported):
   a. If we have whole dataset we can for each feature in the dataset calculate mean value and subtract it from the feature.
   b. In case of continuously growing dataset like online learning we normalize each observation (face image in our case) separately: calculate mean from data entry (one face image), subtract mean from original image and divide it by standard deviation of the image.
2. Image alignment (mandatory step). Perhaps, during exploring given ORL dataset you've noticed, that some faces are bit deviated from horizontal position. This deviation increases model' error. To mitigate this error you can align face horizontally before to process it. This is the "normalization" process in your model. As a mean of alignment try to use affine transformations (i.e., image rotation).
3. **PCA** (Principal Component Analysis) (mandatory step). This technique is designed for reducing computational complexity and allows to extract most valuable features from raw input data (i.e., unstructured data the image is kind of); for data compression to increase

computation speedup, decrease disk space / amount of memory. Your algorithm should be parametrized by “pca energy” value, which equals 0.98 by default. In report you should provide summary statistics for different values of pca energy (0.95, 0.98, 0.99, and 1.00).

4. **LBP** (Local Binary Pattern) (**this step is optional**). Here some examples of implementation of this model. This step is the another “normalization” step which helps you to increase your model’ accuracy without altering your model’ implementation. Your model should be also parametrized by trigger variable to switch on/off this step. Learn the basic idea of LBP (Local binary pattern), uniform LBP (most widely used by people). Then test face recognition rate on ORL using LBP using the same setting of PCA. Note LBP uses chi squared distance. PCA is usually used for dimensionality reduction. The feature you used for PCA task is the pixel values. LBP is another feature which replaces pixel values. Then you can use PCA on the LBP features. The extract LBP feature is histogram which needs to be evaluated using chi-squared distance. For LBP, you have 2 free parameters, the number of blocks(divide the images into N*N blocks), the length of radius(R). Test the combinations of (N,R). N=1,2,3...10, R=1,2...5, so 10*50 groups of results. Some parameters are contradict with each other. In the future, you can always set the radius=1. You only tune the parameter: the number of blocks. 1*1, 2*2,....,10*10. Please use the version of uniform LBP. Chi-square distance is used to replace euclidean distance of PCA because LBP is a histogram. Given input I, you can split the input images to 1*1, 2*2, N*N blocks. For example, give 100*100 I, and N=2, then you can split into 4 blocks(sub-images). Each is 50*50. Then you compute LBP histograms (59D) on these 4 blocks to get 4 features, f1,f2,f3,f4. Next, concatenate these features to [f1,f2,f3,f4] (59*4=236D). Use this 236D feature for face recognition. Use chi-squared distance to compare two LBP histograms

Report

Each task must include report - document, described what was done, model was applied, and the results were achieved during solving this task.
The report itself consist from next sections:

1. Problem overview
2. Existing math models for solving this problem
3. Algorithm explanation
4. Results
5. References and sources

Outcome
The result of this challenge should consist from two documents

1. Source code (perhaps jupyter notebook on git gist page).
2. Report document in doc format (google doc will be good).

Link to the results should be posted for peer review. Also, don’t forget to make a review for at least of two of your classmates