Project: The least squares face-off

By now you are familiar with representing an image that is p pixels by q pixels as a $p \times q$ matrix with entries from 0 to 255. A $p \times q$ matrix can also be viewed a $pq \times 1$ vector. In fact, this is how it is stored in computer memory, which is addressed linearly. In MATLAB, you can convert between an $p \times q$ matrix $\boldsymbol{Z}$ and its vector form $\boldsymbol{z}$ as follows:

```
z = Z(:);
Z = reshape(z,p,q);
```

The vector point of view has certain advantages. For example, we can collect a library of n identically sized $p \times q$ images as columns of a single $pq \times n$ matrix $\boldsymbol{A}$. If another image comes along as the pq-vector $\boldsymbol{z}$, we might want to know how much it resembles other images in the library, for instance as part of a facial recognition application.

A natural way to attack this problem is to solve $\boldsymbol{A}\boldsymbol{x} \approx \boldsymbol{z}$ for $\boldsymbol{x}$ using linear least squares. If $\boldsymbol{z}$ is well represented as a linear combination of images in the library, then we should see a relatively small residual $\|\boldsymbol{z} - \boldsymbol{A}\boldsymbol{x}\|$. Furthermore, suppose J is a set of column indices in $\boldsymbol{A}$ all representing images of a single person. We could define, say,

$$s_J = \sum_{j \in J} x_j, \tag{1}$$

and use s_J as an indicator of the similarity of $\boldsymbol{z}$ to that person. However, there are many other ways one could define s_J based on the x_j, and you are not required to use (1).

For this project you will need data available online as `attfaces.zip`. This file has a README describing 400 face images of size 112×92. The file also has two MATLAB functions: `loadface`, which accepts a subject and pose number and returns the corresponding image in column form, and `showfaces`, which takes a matrix of image vectors and displays them in a grid.

Your team will be given a list of 25 subject numbers and 6 pose numbers, implying a total of 150 images. Thus, your library matrix $\boldsymbol{A}$ will have $n = 150$ columns, each of length $m = (112)(92) = 10304$. You should also create an n-vector `subj` that records the subject number whose image appears in the corresponding column. Because you will be solving many different least squares problems with the same matrix, you should perform a reduced (not full!) factorization $\boldsymbol{A} = \hat{\boldsymbol{Q}}\hat{\boldsymbol{R}}$; see section 3.3 on how this factorization is used to solve the least squares problem.

Your goal is to write a function

```
function [s,conf] = identiface(Q,R,subj,z)
```

The inputs `Q` and `R` are the reduced QR factors of $\boldsymbol{A}$, and `subj` is described above. The other input is another image `z` in vector form. The only information you are permitted to use are the input arguments and the solution vector $\boldsymbol{x}$ of the linear least squares problem $\min \|\boldsymbol{z} - \boldsymbol{A}\boldsymbol{x}\|_2$. The first output `s` is the subject number (not column number) that is judged to be the most likely match for `z`, and the second output `conf` is a number between 0 (lowest) and 1 (highest) that indicates the level of confidence in the identification. For example, calling

```
identiface(Q,R,subj,A(:,j))
```

should return `subj(j)` and 1 as outputs. On the other hand, if the image is not from one of the subjects in your collection (or not even a face at all!), then `conf` should ideally be small.

To measure the performance of the algorithm, suppose T is a collection of test images. For each $z_i \in T$, let the returned confidence score be w_i. Then the performance is

$$p(T) = \left(\sum_{i \in \text{HITS}} w_i \right) - \left(\sum_{i \in \text{MISSES}} w_i \right), \tag{2}$$

where HITS and MISSES are index sets that divide the successful subject identifications from the errors. You should perform tests using your own choice(s) of T, including images from the library, images of subjects in poses not used in the library, images of other subjects in the database, and images of things other than faces, until you are satisfied that changes to your algorithm are unlikely to improve the overall performance.

Submission requirements

You should turn in all code needed to run and test the `identiface` function. (Submit as separate files; there is no need to zip them.) Also turn in a short PDF report (no more than 2 pages) describing in mathematical terms how your function computes its outputs. Justify your algorithm with mathematical or experimental arguments. Mathematical notation in the report must be made using LaTeX or a high-quality equation editor in a word processing application.