Master Degree in Artificial Intelligence
Statistical and Mathematical Methods for Artificial Intelligence

2019-2020

Homework 3

Singular Value Decomposition and Linear Least Squares

1. SVD for matrix approximation and image compression
   - Write a Matlab script/function that:
     - Reads a grayscale image
     - Displays the image
     - Computes the Singular Value Decomposition of the image matrix A and plots the singular values
     - Approximates the matrix A with the matrix A_k obtained by summing the k components of rank one associated with the largest singular values
     - Displays the matrix A_k
     - Approximates the matrix A with the matrix $\tilde{A}_k$ obtained by summing the k components of rank 1 associated to the k smallest singular values
     - Displays $\tilde{A}_k$
     - Computes the relative error in the Frobenius norm bewteen the exact and the approximated image.
     - computes the following compression factor for an image of size $m \times n$:

$$c = k(\frac{1}{m} + \frac{1}{n})$$

   - Test the program on 4-5 grayscale images with different features (er.g. sharpness, grayscale levels, size of objects ,....). Complete the following table for the considered images by reporting the number of components needed to get the different errors reported in the columns of the table:

image	error< 1%	error < 5%	error< 10%

   - Plot a graph of the relative error as a function of the number k of components considered
   - Plot a graph of the compression factor as a function of the number k of components considered

2. Linear least-squares for data approximation. Given the least-squares problem:

$$min\|Ax - b|_2^2, \tag{1}$$

   - Write a Matlab script/function that
     - Computes and displays the matrix condition number.
     - computes the solution with the normal equations

$$A^T Ax = A^T y.$$

     - computes the solution by using the pseudoinverse of A

- GIven a data set of m data $(x_i, y_i), i = 1, \dots m$, determine the least-squares polynomial approximation of degree $n = 1, \dots 5$ of the data.
- Plot the data and the computed functions
- Analyse with plots the residuals
- Compute the R^2 value:

$$R^2 = 1 - \frac{\sum_{i=1}^{m}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{m}(y_i - \mu)^2}$$

  where $\hat{y}_i = f(x_i)$ are the fitted values in x_i and μ is the arithmetic mean of the avlues y_i.
- consider the following data sets:
  a) (1.0, 1.18), (1.2, 1.26), (1.4, 1.23), (1.6, 1.37), (1.8, 1.37), (2.0, 1.45), (2.2, 1.42), (2.4, 1.46), (2.6, 1.53), (2.8, 1.59), (3.0, 1.50).
  b) (10, 1), (10.1, 1.2), (10.2, 1.25), (10.3, 1.267), (10.4, 1.268), (10.5, 1.274).
  c) At least one data set with difficulty=lower from the NIST website: http://www.itl.nist.gov/div898/strd/lls/lls.s
  d) The data set `time_series.txt` from the course website.

3. Image Classification with SVD.

- Consider the three image data sets `coil20,orlfaces,yalefaces` (or choose three image data sets from the web) for a classification problem with n_p classes. (*Tips*: all the images must have the same size $m \times n$).
- Divide all the images in n_p classes and order them in different n_p directories.
- Split all the n_p directories in training test and test set. (*Tips*: Training set of different classes must have the same dimension. Test set of different classes must have the same dimension).
- Construct the design tensor (3D-matrix) A for each of the n_p directories, taking all the images from the training set.
  Tips:

```
for k=1:np
t=1;
for image in training_set(k)
[m,n]=size(image)
A(:,t,k)=double(reshape(image,[m*n,1]));
t=t+1;
end
end
```

- For $k = 1 \dots n_p$ compute the SVD for each of the n_p matrices $A(:,:,k)$ and store the matrices U_k. (*Tips:* Use $[U_k, S_k, V_k] = svd(A(:,:,k), 0)$).
- Take an image z from the test set and store it as a vector.
- For $i = 1 \dots n_p$ construct the projections $\tilde{z}_i$ of the previous vector over the n_p vector spaces genereted by the column vectors of the n_p matrices U_k. Classify z as a member of the class P where $norm(z - \tilde{z}_P)$ is the minimum values. (*Tips:* use the Orthogonal Decomposition Theorem to compute the projections).