MSc Vibot & Computer Vision

Applied Mathematics

Practical

Désiré Sidibé
dro-desire.sidibe@u-bourgogne.fr

Image representation and compression using SVD

I. Recall on SVD

The SVD of an $m \times n$ matrix A of rank k is given by $A = U\Sigma V^T$, which can also be written as $A = \sum_{i=1}^{k} \sigma_i u_i v_i^T$.
So, the matrix A is a sum of rank one matrices that are orthogonal with respect to the matrix inner product. Truncating the sum at p terms defines a rank p matrix $A_p = \sum_{i=1}^{p} \sigma_i u_i v_i^T$.
If we approximate A with A_p, then we make an error equals to $E_p = A - A_p = \sum_{i=p+1}^{k} \sigma_i u_i v_i^T$. It can be shown that A_p is the best rank p approximation to A.

II. SVD and image compression

SVD can be used for image compression by approximating the matrix I, representing an image, by a low-rank matrix I_c.
In digital image processing, the quality of the compression on an $m \times n$ image I is often measured in term of Peak Signal to Noise Ratio (PSNR) which is defined as

$$PSNR = 10\, log_{10} \frac{(\text{max range})^2}{\sqrt{MSE}},$$

where *max range* is the maximum value a pixel can take and *MSE* is the mean square error defined by

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \|I(i,j) - I_c(i,j)\|^2 .$$

1. Select an image I of your choice (a grayscale image not too big in size, 256×256 pixels for instance).
2. Using SVD, display a rank 1 approximation of I. What can you say about the 'columns' in the rank 1 approximation of the image in terms of linear dependence/independence.
3. Reconstruct your image using different ranks approximation (4, 16, 32, 64, 128). For each approximation, show the absolute difference between the original image and the re-constructed images. Give also the PSNR values.
4. Explain how the reconstruction using rank approximation, provides a compression of the original image. What is the compression ratio?

5. Create a 256×256 random matrix I_2 (use the command *rand*). Convert the random matrix into an intensity image using the command *mat2gray*.

   For the images I and I_2, make a plot which shows how the singular values decay along the diagonal of Σ in the SVD. Explain the difference in the decay for the two images.

6. Another way of performing SVD compression is achieve by subtracting the mean of the original image before performing the SVD. The mean is then added back to the SVD construction to obtain the re-constructed image. This process is called the 'Rank-1 Update'.

   Apply 'Rank-1 Update' to I and I_2. What are the differences with the previous results?

7. What is, from your point of view, the reason why SVD in not a popular image compression tool?

III. Digits recognition

Now, we consider an application of digit classification. We are gievn a set of handwritten digits in the form of binary images. Each digit is represented as a 28×28 image as can be seen in Fig. 1.
The digits can also be represented as a $28 * 28 = 784$ dimensional vector, and all images of a given digit are put as rows of a data matrix X.
For example, the command `load(digit7.mat)` will load a matrix X of size 1028×784, meaning that we have 1028 images of digit 7 each of size 28×28.

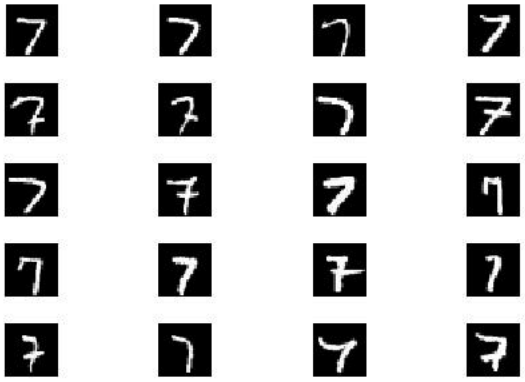

Figure 1: Some example of digit 7.

1. Write a function `show-digits(X, N)`, which shows the N first digits in the matrix X.

   Note that your function should resize the images (from a 784-d vector to a 28 x 28 image).

2. Perform PCA analysis of the data matrix X of digit 7.

   - show the mean image and the 10 first principal components
   - reconstruct a selected image with different number of principal components, for example $[1, 5, 10, 20, 30, 50, 75, 100, 128, 256]$.

     show the reconstructed image and compute the PSNR values.
   - plot the evolution of the PSNR as a function of the number of principal components used.

Digits recognition
Now, we want to recognize digit 1 form digit 7, using a nearest neighbour (NN) classifier.

Nearest neighbour classifier
NN is the simplest classifier we can construct. Assume we have a set of training examples $\mathcal{D} = \{\mathbf{x}^n, c^n\}$, $n = 1, \ldots, N$, where $\mathbf{x}^n$ is an input vector and c^n its corresponding class label ($c^n \in (1, \ldots, C)$).
The strategy of a NN classifier is as follows: *for novel input* $\mathbf{x}$*, find the nearest input in the train set and use the class of this nearest input*.
For example, if we have in our training set only digits 1 and 7, for a new image (not in the training set) whose class is unknown, we find the image $\mathbf{x}^*$ in the training set which is the most similar to the input, this is the nearest neighbour, and assign the label of $\mathbf{x}^*$ (which is 1 or 7 in our example) to the input image.
For vectors $\mathbf{x}$ and $\mathbf{y}$ representing two different datapoints, we measure 'nearness' using a dissimilarity function $d(\mathbf{x}, \mathbf{y})$. A common dissimilarity is the squared Euclidean distance

$$d(\mathbf{x}, \mathbf{y}) = (\mathbf{x} - \mathbf{y})^T (\mathbf{x} - \mathbf{y}).$$

Fig. 2 shows the NN algorithm.

Algorithm 14.1 Nearest neighbour algorithm to classify a vector $\mathbf{x}$, given train data $\mathcal{D} = \{(\mathbf{x}^n, c^n), n = 1, \ldots, N\}$:

1: Calculate the dissimilarity of the test point $\mathbf{x}$ to each of the train points, $d^n = d(\mathbf{x}, \mathbf{x}^n)$, $n = 1, \ldots, N$.
2: Find the train point $\mathbf{x}^{n^*}$ which is nearest to $\mathbf{x}$:

$$n^* = \underset{n}{\operatorname{argmin}}\, d(\mathbf{x}, \mathbf{x}^n)$$

3: Assign the class label $c(\mathbf{x}) = c^{n^*}$.
4: In the case that there are two or more nearest neighbours with different class labels, the most numerous class is chosen. If there is no one single most numerous class, we use the K-nearest-neighbours.

Figure 2: Nearest neighbour algorithm for classification.

1. Load the two digits data, `digit1.mat` and `digit7.mat`.
2. Divide the data into two matrices X_{train} and X_{test} such that X_{train} contains $N = 500$ training examples of each for the two digits. So X_{train} is of size 1000×784.

   X_{test} contains the remaining examples.
3. Classify each example in X_{test} using a NN classifier and report the classification accuracy.
4. Use PCA to reduce the dimensionality of the problem, i.e. represent each training example as a lower dimensional vector using PCA, and perform classification of the test examples.

   Use different number of principal comonents and show how the classification accuray varies.
5. Conclusions ?