Project 2 PCA and Sparseness Representation

Guixiang Zhang

Description:

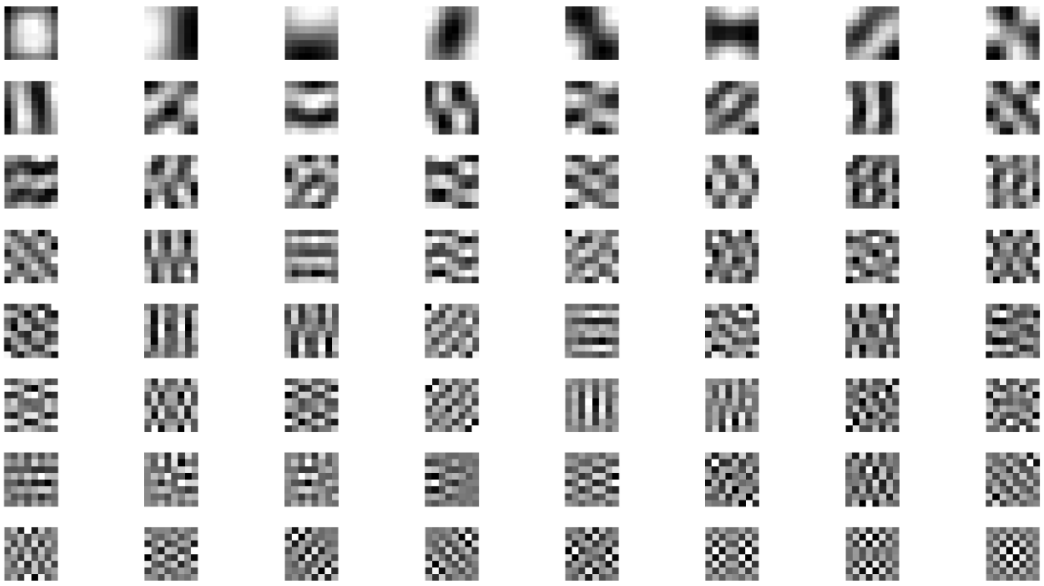
1. PCA

   MATLAB code: Zhang_project2PCA.m

The goal of PCA is to get a set of basic vectors that their linear combination can approximately represent the original data matrix. Generally, the dimensions of the matrix composed of basic vectors is far less than the original matrix. So the PCA can reduce the dimensions of data in order to save them into memory.

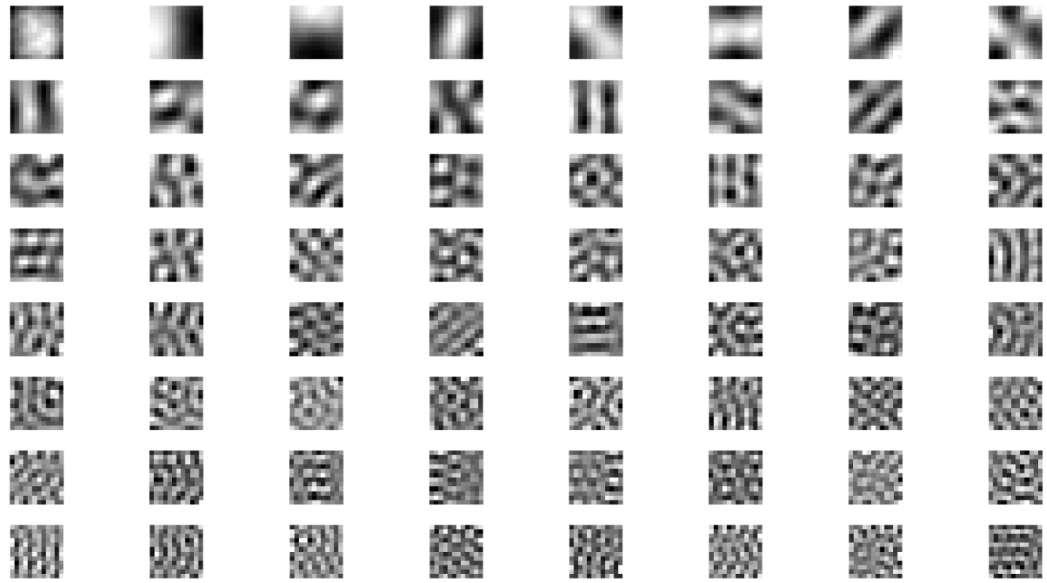
In my code, I first get nine 256*256*3 images, change them to 9 grayscale images. Secondly, I set a window size p, so each window is p*p and I get n windows from nine image. And they do not overlap. Then I reshape them in to 1*(p*p) vectors and save them into a matrix x. For instance, if p=8, so n=9216 and the x will be 9216*64.

Then I do the PCA, it is like solving linear least-square problem and the constraint is $||x||_2^2 = 1$.we compute the mean u and get the matrix p=x-u. Do the eigenvalue decomposition of Q=p'*p. Finally, we get the eigenvectors, we reshape them and get the PCA basis. Like below.

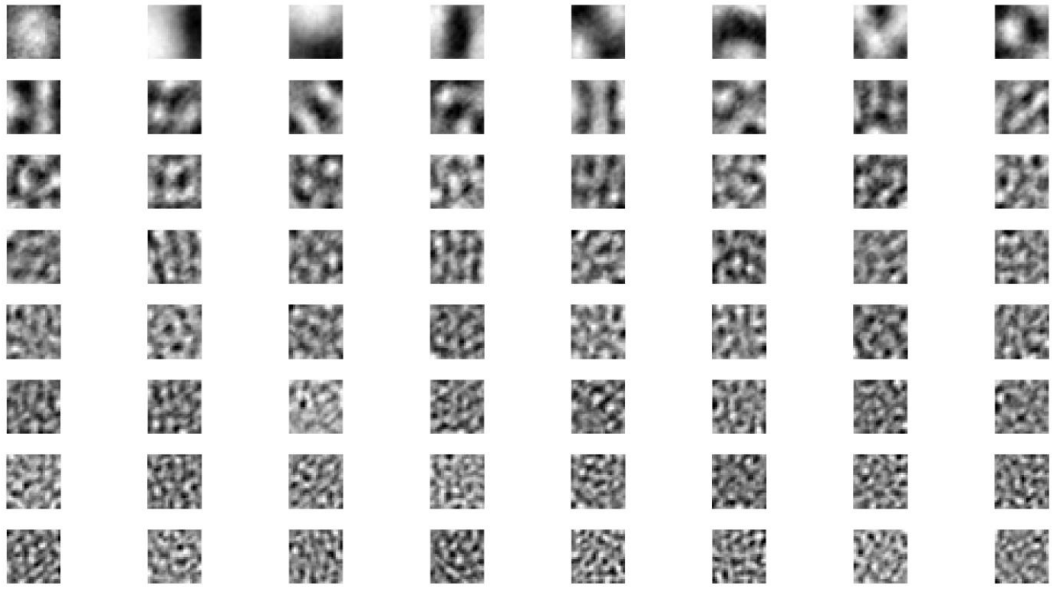


P=8

I also set the p= 12 and 24 to see the differences.



P=12



P=24

In general, they are similar. But as the p increasing, the PCA basis image become more and more blur. It can not show the texture detail clearly.

2. Sparse Representation with OMP and KSVD

   MATLAB code : Zhang_project2OMPKSVD.m

First, I get some windows data from the x matrix above and I randomly select some rows of x to be an initial set of basis functions and call it dictionary. Then normalized the data and dictionary.

To save the computation, I only choose 10 windows in x and set a sparse ratio l=10 which means if want to represent an image using the basis function, up to ten basis will be used, so it is sparse.

Using the data and dictionary matrices, I use an algorithm called Orthogonal Matching

Pursuit(OMP). It is greedy algorithm to iterate the solution of the sparse matrix. The model is

$$min\ ||Y - D\alpha||_F^2\ \ s.t.\ ||\alpha||_0 < l.$$

After that we got the sparse solution A is the matrix of α. Then I use the algorithm K-SVD to update the dictionary. So we get a loop of OMP and K-SVD to iterate until the residual is less than some value. Finally I get the dictionary which is the sparse basis matrix. Like below.

p=8, l=10, 81 basis.

p=8, l=10, 192 basis

p=16, l=10, 81 basis

3. Sparse Representation with fminunc
   MATLAB code : Zhang_project2fminunc

This is the method of my understand of the paper. Like the paper, I choose S(x)=|x|, $\frac{\lambda}{\sigma} = 0.14$.

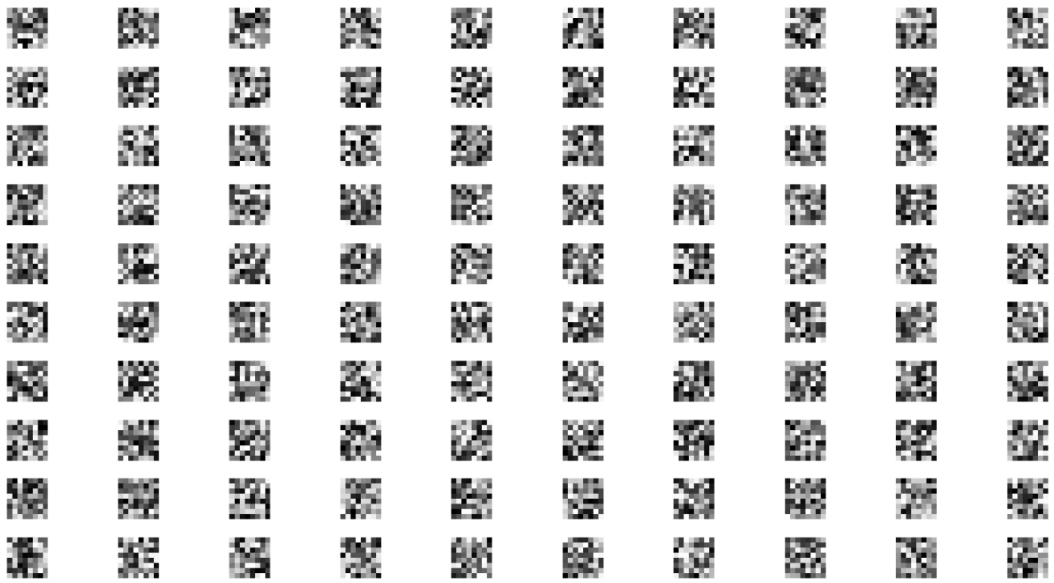
The cost function to be minimized is :

$$E = \sum_{x,y} [I(x,y) - \sum_{i} a_i \varphi_i(x,y)]^2 + 0.14 * \sum_{i} |a_i|$$

I only choose one sample, one window in the image data, called 'img' which is a 8*8 matrix. And reshape it to be a 1*64. I initialize optA=(0,0,0,⋯.0) is the coefficient matrix. And the initial $\varphi_i(x,y)$ is a 64*100 random number matrix 'optbasis'.

Then I do the optimation using fminunc. First I fix the optbasis and img to find the optA minimize E. One time is enough for iterating. Then I fix the img and the new optA to find the optbasis that can further minimize the E using fminunc. This step will take several mins. Then, we get the optA and optbasis. The first part of the E is very tiny, limit to zero. Only the second part have some value which represents the sparseness. The result of the basis vector is showed below.

If we want to further minimize E especially the second part. We should do many time the two steps, the two fminunc.



1 window data, S(x)=|x|, $\frac{\lambda}{\sigma} = \mathbf{0.14}$, iterate 1 time

In my opinion, the images of the optimized basis vectors are similar to the images of the initial basis vectors unless we iterate much many times. I think the data images is preprocessed by some filter otherwise he would not get the result will not like that, something like gabor bank image.

I am sorry not completing the method in the paper Since I have many questions about the equations in it. I don't know that what the equation 5 means. Is that the ai on the left is the gradient use to iterate ai, and we should minus it or plus it? And it also have aj on the right side, if I have already iterate a1 and get a new one, when I update a2, which a1 should I choose, the new one or the one before iterate? And as for the equation 6, which learning rate μ shall I choose, it that need to try many times?