Sparse Representations and Their Applications in Signal and Image Processing

Home Work 3: On-line Dictionary Learning for Denoising

Submitted by:		
	First Student	1234567890
	Second Student	0987654321

due 23.1.2016

Instructions

- Work should be done in pairs and submitted via moodle. Students wishing to submit on their own need to get permission to do so in advance.
- Submission should include a pdf-file containing the report and all Matlab files you created for this exercise.
- You are encouraged (but not obligated) to use LaTeX for your report, and add your answers to the project `.tex` file that is included with the exercise files. *You will get a 5 point bonus for doing so. However, the maximal grade will be 100 in any case for the exercise.*
- Write your own code. This exercise is meant to teach you how the different algorithms operate.
- Indent and highlight the Matlab code you attach to the report. In LaTeX this is done using the command: `\lstinputlisting{<filename>}` from the package `mcode.sty`, which is already included in the homework laTeX file.
- Use sparse type vectors and matrices in your code when it is appropriate.
- Questions regarding the exercise should be asked in the course forum in moodle, e-mails about the exercises will not be answered.

Part A- Problem Description

You will be following the dictionary learning algorithm described in [1]. The process you are going to implement is as follows:

1. Input Image

- Choose an interesting Image with texture and rich content.
- The size of the selected image should be $256x256$ or larger.
- Corrupt the image with noise with AWGN, with variance σ^2 ($\sigma = 20$).
- The image should be decomposed into an array of fully-overlapping patches of size $b \times b$ ($b = 8$).
- Image patches should have *zero-mean*, do this by subtracting the mean from each patch, but save the means for later.
- You can find all these steps implemented in the supplied matlab file.

2. Initial Dictionaries (D_0 in [1])

Use the following dictionaries as initialization for the training process

- Random Gaussian dictionary of size 64×256
- Random Gaussian dictionary of size 64×400
- Truncted wavelet of size 64×256 that is supplied in the file `D0.mat`, taken from [2].

3. Online Dictionary Learning

- Implement the dictionary learning algorithm from [1]
  - Algorithm 1 is the main algorithm. Drawing x from $p(x)$ as mentioned in the algorithm means simply to choose a patch at random from the set of extracted patches (i.e., this can be done by picking a random integer `i` in the range 1 to `size(patches,2)` and setting `x=patches(:,i)`).
  - Algorithm 2 is used by Algorithm 1. In your implementation of this part you only need a single `while` as the `for` step that is used in the description of Algorithm 2 can be implemented simply by a matrix multiplication. Take care also not to divide by 0 (it may happen in the algorithm that you will need to divide by an entry in the matrix that is equal to zero. In this case divide by something other than zero). The stopping condition for this algorithm should be the Frobenius norm between subsequent values of D, i.e., $\|D_j - D_{j-1}\|_F \leq \epsilon$ for some small enough ϵ.

- Your code should support two modes:
  1. Regular version (i.e. $\eta = 1$) as described in algorithm 1 in the paper.
  2. Mini-batch mode that support values of η bigger than 1 (see Equation (11) in [1]) .
- Sparse coding
  - Use OMP instead of LARS (which is used in Algorithm 1 in [1]).
  - Your implementation should support two stopping criteria:
    1. If the sparsity reached a certain threshold: `max_sparsity`
    2. If the error ($\|r\|^2$) is below a `noise_threshold`
  - *Note:* OMP over all the data is also implemented in the supplied `RunMe.m`. You may re-use it if you want.
- Use a parameter named `train_size` to limit the amount of iterations in the algorithm (denoted by T in [1]).
- Your function should look like this in order to use the supplied code: `Dictionary = odl(patches,D0,train_size,max_sparsity,noise_threshold,eta)`

Part B- Implementation & Running

1. Implementation notes

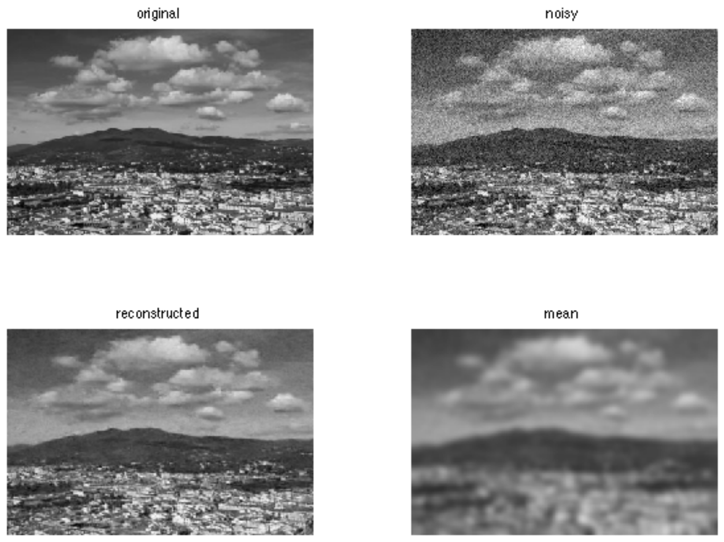
- Set $\sigma = 20$ and $b = 8$.
- Set `max_sparsity=20`. This quantity might need to be reduced due to numerical issues depending on the richness of texture of the image you choose.
- Set the `noise_threshold` to be $C\sigma^2 b^2$, where $C = 1.15$.
- First pick your favorite initial dictionary (the one that gives the best results) when running with $\eta = 1$, and then try to see if a different value of η can help reconstruction accuracy and convergence speed. Pick one of these values as you see fit.
- Every 500 iterations or so of the dictionary learning function compute the *over all* representation error (on all the data) and the average sparsity of the representation. Plot and compare the graphs of the different experiments you perform.
- Take a look at (and put in your report) the dictionary you are getting by plotting the atoms (a 16 by 16 grid of 8 by 8 patches in the case of the truncated wavelets dictionary (and the first Gaussian Dictionary), and a 20 by 20 grid in the case of the second Gaussian dictionary). Your report should include a comparison between how the dictionary looks before and after the training.

- Compare the different initial dictionaries and the running modes (values of η). Compare the root mean square error (RMSE) of the representation and the PSNR before and after the dictionary learning phase. Discuss your results and try to explain them.

2. Supplied Code

You are supplied with a `RunMe.m` file that splits the image into patches and performs pursuit on the extracted patches to get the representation, thus implementing almost anything you need for this exercise, except for the actual dictionary learning function (commented out in block E). Feel free to use it, but if you do, you are required to explain what it does. To this end the code is partitioned into blocks (that are marked by a capital letter, e.g. `%A`). Explain using no more than two lines of comments what each block does.

These are the results after running ODL on a 350×234 pixel image:



```
noisy PSNR = 24.4193, reconstructed PSNR = 30.5649
```

Try to improve upon these.

Wrap Up

- Create the file `RunMe.m` that can be used to run all the code for this exercise and recreate your results

- Submission in moodle should include two files:

- Your report in PDF format, named `SparseRepHw2_<student #1>_<student #2>.pdf`.
- Compressed ZIP file named `SparseRepHw2_<student #1>_<student #2>.zip` containing your report and matlab code.

References

[1] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th annual international conference on machine learning*, pages 689–696. ACM, 2009.

[2] Jeremias Sulam, Boaz Ophir, Michael Zibulevsky, and Michael Elad. Trainlets: Dictionary learning in high dimensions. *IEEE Transactions on Signal Processing*, 64(12):3180–3193, 2016.