Signal processing
Lab 4: Denoising with Dictionaries

Maxim Berman – maxim.berman@ecp.fr

Friday 20/03/2015

Abstract

While lab 2 and 3 used a linear filtering approach for image denoising, here we consider a dictionary learning approach. The goal is to learn a dictionary D corresponding to an overcomplete basis of *natural images*, and to keep only a few vectors of this basis to encode the patches of a noisy image, in the hope that the noise is not retained in this sparse representation.

Deliverable: To be submitted before Wednesday 08/04/2015.

1 Image coding with Matching Pursuit

We are interested once more in an image denoising problem. As in the previous labs, we consider a white noise model

$$\mathbf{y} = \mathbf{x_0} + \boldsymbol{\xi} \tag{1}$$

where $\boldsymbol{\xi}$ is a vector of independent gaussian variables of variances σ_n^2. We will now consider a nonlinear filtered technique: denoising using overcomplete basis, or dictionaries.

The idea is that in natural images, small patches in the images are far from being random. By considering local image patches $\mathbf{p}$ of size $w \times w$, one could pick a basis $\mathbf{D}$ that accurately captures this naturally occurring structure, in the hope that a linear combination of a small number of these basis elements

$$\mathbf{p} = \mathbf{D}\boldsymbol{\alpha} \tag{2}$$

yields a good approximation of $\mathbf{p}$, without retaining its noise.

The use of an overcomplete basis (or *dictionary*) of vectors leads to the question of choosing between the many possible approximate decompositions in (2). This is where the crucial criterion of *sparsity* appears: if the basis vectors are good representatives of the structure of the natural patches, then we might suppose that selecting solutions that use the least number of basis vectors yields a natural patch as well, and therefore, a good approximation. This leads to the following problem:

$$\hat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}:\|\boldsymbol{\alpha}\|_\mathbf{0}<\mathbf{k}}{\arg\min} \|\mathbf{p} - \mathbf{D}\boldsymbol{\alpha}\|_2 \tag{3}$$

where $\|\boldsymbol{\alpha}\|_0$ denotes the *support* of α, that is, its amount of non-zero coefficients.

This sparse coding problem is $\mathcal{NP}$-complete in general; however, there has been many successful algorithms developed to tackle it. In this section, you will apply the Matching Pursuit algorithm to decompose an image over a dictionary, and apply this for image denoising.

The general outline of the denoising procedure is as follows:

1. Build a dictionary from patches extracted from the noisy image or from other examples of similar images
2. For each patch extracted on the image on a grid, replace the patch by its sparse approximation over the dictionary
3. Sum the local patches to recover an approximation of the whole image.

Since the noise is independent from the underlying structure of natural images, this encoding will likely not preserve it, while preserving other important features of natural images (such as contours).

In this section, you are provided with a dictionary already. You will implement two different algorithms, Matching Pursuit – Algorithm 1 – and Orthogonal Matching Pursuit – Algorithm 2, in order to compute the sparse approximation.

The provided code already loads the image and the dictionary and densely extracts local overlapping patches densely over the image. It also makes the image bigger by mirroring it at its boundaries, in order to avoid dealing with the edges differently. Be sure to study this code carefully in order to understand its behavior. An important point is that while a local patch is naturally represented as a $w \times w$ array, we work with vector representations of our patches; therefore, the computations are done after stacking the values into a w^2 vector - the same holds for the dictionary, of size $d \times n$, where $d = w^2$ is the dimension of the patch space, and n the number of basis elements.

Algorithm 1 Matching Pursuit (MP) for image coding.

```
1: Inputs:
       dictionary $\mathbf{D} = (\mathbf{d}_1 | \dots | \mathbf{d}_n) \in \mathbf{R}^{d \times n}$, input vector $\mathbf{p} \in \mathbf{R}^d$
       sparseness $k$
2: Output:
       coefficients $\boldsymbol{\alpha}$ such that $\mathbf{D}\boldsymbol{\alpha} \approx \mathbf{p}$
3: Initialize:
       residual $\mathbf{r} \leftarrow \mathbf{p}$; $\boldsymbol{\alpha} \leftarrow \mathbf{0}$
4: while $\|\boldsymbol{\alpha}\|_0 < k$ do
5:     $\hat{j} \leftarrow \arg\max_j |\langle \mathbf{r}, \mathbf{d}_j \rangle|$          ▷ Pick next basis vector
6:     $\alpha_{\hat{j}} \leftarrow \alpha_{\hat{j}} + \langle \mathbf{r}, \mathbf{d}_j \rangle$          ▷ Update weight and residual
7:     $\mathbf{r} \leftarrow \mathbf{r} - \langle \mathbf{r}, \mathbf{d}_j \rangle \mathbf{d}_j$
8: end while
```

Question 1 Implement a function `alpha = MP(D, p, k)` that computes a k-sparse decomposition of a vector $\mathbf{p}$ on the overcomplete basis $\mathbf{D}$ using the Matching Pursuit algorithm.

Algorithm 2 Orthogonal Matching Pursuit (OMP) for image coding.

```
 1: Inputs:
        dictionary $\mathbf{D} = (\mathbf{d}_1 | \ldots | \mathbf{d}_n) \in \mathbf{R}^{d \times n}$, input vector $\mathbf{p} \in \mathbf{R}^d$
        sparseness $k$
 2: Output:
        coefficients $\boldsymbol{\alpha}$ such that $\mathbf{D}\boldsymbol{\alpha} \approx \mathbf{p}$
 3: Initialize:
        residual $\mathbf{r} \leftarrow \mathbf{p}$; index set $J \leftarrow \emptyset$
        $\mathbf{T} \leftarrow$ (empty matrix)
 4: for t = 1 to k do
 5:     $\hat{j} \leftarrow \arg\max_{j \notin J} |\langle \mathbf{r}, \mathbf{d}_j \rangle|$          ▷ Pick next basis vector
 6:     $J \leftarrow J \cup \{\hat{j}\}$; $\mathbf{T} \leftarrow \left(\mathbf{T} \,|\, \mathbf{d}_{\hat{j}}\right)$          ▷ Update selected indices and vectors
 7:     $\hat{\mathbf{a}} = \arg\min_{\mathbf{a}} \|\mathbf{p} - \mathbf{T}\mathbf{a}\|_2$          ▷ Update reconstruction coefficients
 8:     $\mathbf{r} \leftarrow \mathbf{p} - \mathbf{T}\mathbf{a}$
 9: end for
10: We have obtained $\mathbf{a} \in \mathbf{R}^k$ such that $\mathbf{T}\mathbf{a} \approx \mathbf{p}$. Construct $\boldsymbol{\alpha}$ by augmenting $\mathbf{a}$ with
    zero values on the other basis vectors.
```

Question 2 Use the provided code that uses this function to encode every patch on this basis using a k-sparsity constraint. With the dictionary provided, show the denoised image for $k = 4$.

Question 3 Plot the curve of the PSNR of the reconstructed image as a function of the sparsity constraint $k \in [1, 10]$. It should be similar to the red curve of figure 1.

Now we consider the OMP algorithm.

Question 4 We will assume that $\mathbf{T} \in \mathbf{R}^{d \times t}$ has full column rank during the run of the algorithm, i.e. $\mathbf{T}^\top \mathbf{T}$ is invertible. Show that the solution to the least-square problem in line 7 of Algorithm 2 is

$$\arg\min_{\mathbf{a}} \|\mathbf{p} - \mathbf{T}\mathbf{a}\|_2 = (\mathbf{T}^\top \mathbf{T})^{-1} \mathbf{T}^\top \mathbf{p}. \tag{4}$$

Question 5 Implement a function `alpha = OMP(D, p, k)` that computes a k-sparse decomposition of a vector $\mathbf{p}$ on the overcomplete basis $\mathbf{D}$ using the Orthogonal Matching Pursuit algorithm.

Question 6 Plot the curve of the PSNR of the image reconstructed with OMP as a function of the sparsity constraint $k \in [1, 10]$. It should be similar to the blue curve of figure 1.

Although Orthogonal Matching Pursuit has better theoretical properties than Matching Pursuit, we see that in this denoising problem, simple Matching Pursuit gives a better PSNR ratio than Orthogonal Matching Pursuit, in the case of looser sparsity constraints. The dictionary has been trained with a sparsity objective $k = 4$, and the maximum of the PSNR is coherent with this training in the case of OMP. The better experimental performance of MP for looser sparsity constraints can be explained intuitively by the fact that denoising gains from the additional averaging introduced by selecting more dictionary (sub-optimal) atoms.

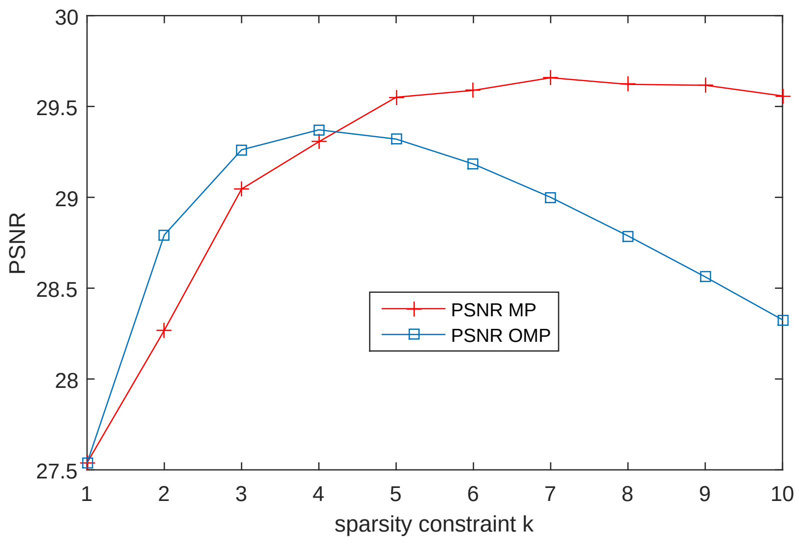

Figure 1: PSNR obtained with Matching Pursuit and with Orthogonal Matching Pursuit as a function of the sparsity constraint k.

Note: Orthogonal Matching pursuit can be made more efficient by using more advanced optimizations of the algorithm (Choletsky decomposition) and slightly modifying the selection of the next basis vector. It becomes superior to MP after these optimizations. We will not, however, go into those details.

2 Dictionary learning

We will now find out how to learn a dictionary suited to our image denoising problem. We will consider a dictionary of $n = 200$ atoms. The dictionary will be learned from the noisy image, with the hope that the noise is not retained in this representation.

Question 7 To create a suited dictionary of $n = 200$ atoms, one first idea is to randomly extract $n = 200$ patches from the noisy image and to take these patches as dictionary atoms. Show experimentally that this "fast training" method leads to bad denoising performance: create a dictionary $\mathbf{D}_{\text{fast}}$ from 200 random patches extracted from the image, and show the reconstructed image and its PSNR using the code from the first section, with a Matching Pursuit algorithm. You may use the provided function `extract_patches` to extract the patches randomly from the image.

Questions 8, 9, 10 are optional, accounting for 1 bonus point, but highly recommended, for a better understanding of dictionary methods.

The simplistic approach of stacking up examplar vectors is thus not sufficient. Learning a dictionary can be formulated and solved as an learning optimization problem. Remember the sparsity-constrained objective of dictionary encoding

$$\hat{\boldsymbol{\alpha}} = \underset{\boldsymbol{\alpha}:\|\boldsymbol{\alpha}\|_0<\mathbf{k}}{\arg\min} \|\mathbf{p} - \mathbf{D}\boldsymbol{\alpha}\|_2 \tag{5}$$

Suppose we have $m = 1000$ example patches $\mathbf{p}^{(1)}, \mathbf{p}^{(2)}, \ldots, \mathbf{p}^{(m)}$, extracted from the noisy image, and we want to find the dictionary that will give the best encoding for these example patches. This leads to the minimization problem

$$\hat{\mathbf{D}} = \arg\min_{\mathbf{D}} \left\{ \sum_{i=1}^{m} \min_{\boldsymbol{\alpha}^{(\mathbf{i})}:\|\boldsymbol{\alpha}^{(\mathbf{i})}\|_{\mathbf{0}}<\mathbf{k}} \|\mathbf{p}^{(i)} - \mathbf{D}\boldsymbol{\alpha}^{(i)}\|_2 \right\} \tag{6}$$

where we try to minimize the sum of the reconstruction errors for each of these example patches.

Since the same reconstruction error can be obtained by multiplying the atoms in $\mathbf{D}$ and dividing a vector $\boldsymbol{\alpha}^{(i)}$ by the same amount, this problem is ill-defined. It can be made well-defined by constraining the atoms in $\mathbf{D}$ to have unit norm, yielding the constraint $\mathbf{D} \in \mathcal{D}$ with

$$\mathcal{D} = \left\{ \mathbf{D} \in \mathbf{R}^{d\times n}, \forall i = 1, \dots, n, \|D_{\cdot,i}\|_2 = 1 \right\}. \tag{7}$$

In order to solve the resulting joint minimization problem

$$\min_{\mathbf{D}\in\mathcal{D},\|\boldsymbol{\alpha}^{(i)}\|_0<k} \left\{ \sum_{i=1}^{m} \|\mathbf{p}^{(i)} - \mathbf{D}\boldsymbol{\alpha}^{(i)}\|_2 \right\}, \tag{8}$$

we will use an iterative procedure were we alternate between picking the best $\boldsymbol{\alpha}^{(i)}$ for a fixed $\mathbf{D}$, and computing the optimal $\mathbf{D}$ for fixed $\boldsymbol{\alpha}^{(i)}$. At first, the dictionary is initialized to $\mathbf{D}_{\text{fast}}$, computed above, normalized so that its column vectors have unit norm.

Question 8 Write a function `D = normalize_dict(D)` that takes a dictionary $\mathbf{D}$ as input and divides each of its column vectors by their norm, so that the returned normalized dictionary $\mathbf{D}$ has unit-norm column vectors.

We then iterate between the two steps

- *α*-**update :** Matching Pursuit is used to find the optimal $\boldsymbol{\alpha}^{(i)}$ for each of the example patches $i = 1, \dots, m$
- *D*-**update :** Projected Gradient Descent is done to find the best $\mathbf{D}$ for fixed $\boldsymbol{\alpha}^{(i)}$.

Matching Pursuit is used in the $\boldsymbol{\alpha}$ because it is faster than OMP (for a naïve implementation) and still gives good results, as seen in the previous section. We now detail the *D*-**update** step. The optimization in $\mathbf{D}$

$$\min_{\mathbf{D}\in\mathcal{D}} \left\{ \sum_{i=1}^{m} \|\mathbf{p}^{(i)} - \mathbf{D}\boldsymbol{\alpha}^{(i)}\|_2 \right\} \tag{9}$$

can be rewritten more compactly as

$$\min_{\mathbf{D}\in\mathcal{D}} \left\{ \|\mathbf{P} - \mathbf{D}\mathbf{A}\|_2^2 \right\} \tag{10}$$

where $\mathbf{P}$ is a $d \times m$ matrix containing each of the example patch vectors, and $\mathbf{A}$ is a $n \times m$ matrix containing each of the $\boldsymbol{\alpha}^{(i)}$ – and the $\|\cdot\|_2$ matrix norm is the Frobenius norm

$$\|\mathbf{M}\|_2^2 = \sum_{i,j} |M_{i,j}|^2.$$

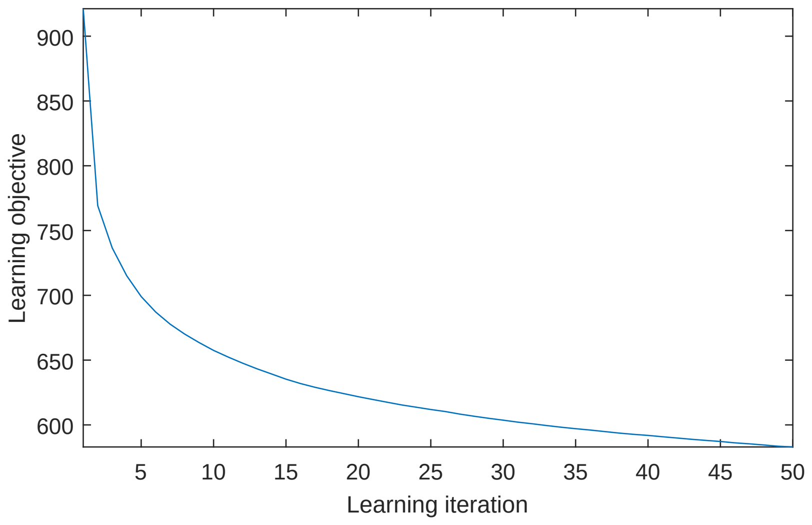

Figure 2: Decrease of the learning objective with the number of iterations.

This constrained minimization problem can be solved by a few iterations of gradient descent, which consists in following the opposite direction of gradient of the objective in $\mathbf{D}$ towards the minimum of this objective. At each step of this gradient descent, one updates $\mathbf{D}$ according to

$$\mathbf{D} \leftarrow \mathbf{D} - \tau \underbrace{(\mathbf{D}\mathbf{A} - \mathbf{P})\mathbf{A}^\top}_{\text{gradient of the objective in } \mathbf{D}} \tag{11}$$

where τ is the step-size. After each step of gradient descent, we renormalize $\mathbf{D}$ using `normalize_dict(D)`, so that its column vectors keep unit norm (hence it is a *projected* gradient descent).

Implementation. The initialization, the first $\boldsymbol{\alpha}$-update, and the first D-update are already implemented. For the gradient descent in D, we take the step-size $\tau = 1/\|PP^\top\|$ – it can be shown to have good convergence properties.

Question 9 We have so far done 1 iteration of dictionary learning. Put the A and D update in a loop to perform 50 iterations of dictionary learning. Plot the decrease of the objective (11) as a function of the learning iteration. You should obtain a figure similar to Figure

Question 10 Denoise the image using your own learned dictionary. Report the denoised image and the PSNR obtained.

Question 11 *(not optional.)* Why is the denoising method presented here much more suited to a real application than the methods seen in labs 2 and 3?

Question 12 is an optional super-bonus.

Question 12 The dictionary learning method seen here used random patches extracted from the noisy image as training examples. Another way of learning the dictionary is to train it on patches extracted from other natural images (other than the ground truth). Learn a dictionary on 1000 patches extracted randomly from 5 grayscale images found on the Internet. Compare the denoising performance of the dictionaries trained by the two different methods.