Published in Image Processing On Line on YYYY–MM–DD.
ISSN 2105–1232 © YYYY IPOL & the authors CC–BY–NC–SA
This article is available online with supplementary materials,
software, datasets and online demo at
http://dx.doi.org/10.5201/ipol.YYYY.XXXXXXXX

PREPRINT September 5, 2013

LSSC

Yohann Salaun[1] & Marc Lebrun[2]

[1] Polytechnique, France (yohann.salaun@polytechnique.org)
[2] CMLA, ENS Cachan, France (marc.lebrun@cmla.ens-cachan.fr)

Abstract

1 Overview

2 Theoritical Description

2.1 Notations

In order to keep coherence with [3], the notations used are the same. A picture of n pixels is seen as a column vector in $\mathbb{R}^n$. The noisy picture is noted $\mathbf{y}$ and the denoised one $\mathbf{x}$. The i-th pixel of x is noted $\mathbf{x}[i]$ and the patch centered in $\mathbf{x}[i]$ and of size m is noted $\mathbf{x}_i$.

2.2 Patches

TODO: change name

The first idea behind this method is to decompose the noisy picture into patches of equal size.
TODO: developp patch idea + the fact that there is one patch per pixel

2.3 Learned Sparse Coding

The second idea is to assume that the denoised patches can be approximated by a sparse linear combinations of elements from a basis set that someway sums up the information belonging to the picture.

The basis set is called a dictionary $\mathbf{D} \in \mathbb{R}^{m \times k}$ and is composed of k elements. The linear combination is represented by a vector $\boldsymbol{\alpha} \in \mathbb{R}^k$ called a code.

TODO: formulation is hard.... The denoising problem is then reformulated into a minimization problem where we look for the optimal dictionary and codes solving the equation below:

$$min_{\boldsymbol{\alpha} \in \mathbb{R}^k, \mathbf{D} \in \mathbb{R}^{m \times k}} \sum_{i=1}^{n} ||\boldsymbol{\alpha}_i||_p \text{ s.t. } ||\mathbf{y}_i - \mathbf{D}\boldsymbol{\alpha}_i||_2^2 \leqslant \epsilon \tag{1}$$

$\mathbf{D}\boldsymbol{\alpha}$ is the estimate of the denoised patch which should be similar to the noisy patch and ϵ can be chosen according to the value of the estimated standard deviation of the noise. Once the dictionary and the codes are learned, for each pixel, we have m estimations (from the m patches that contain it) and their averaging give us the denoised information of this pixel:

$$\forall i \in [1, n], \quad \mathbf{x}[i] = \frac{1}{m} \sum_{j=1}^{m} \mathbf{D}\boldsymbol{\alpha}_{\sigma(i,j)} \tag{2}$$

Where $\sigma(i, j)$ is the number of the patch where pixel $\mathbf{x}[i]$ is in j^{th} position.

(1) is usualy minimized with $p = 0$ or 1. It becomes NP-hard to solve when $p = 0$ but a greedy algorithm such as Orthogonal Matching Pursuit [5] can quickly give a good approximation. The problem is convex with $p = 1$ and is efficiently solved with the Least Angle Regression algorithm [1]. Experimental observations [6] have shown that the learning part is better with $p = 1$ and the recomposition part with $p = 0$.

2.3.1 Least Angle Regression

TODO: explain the subject briefly and how to solve it

- For $\mathbf{D} \in \mathbb{R}^{m\times k}$, $\quad \boldsymbol{\alpha}^t = \text{argmin}_{\boldsymbol{\alpha}\in\mathbb{R}^k} ||\boldsymbol{\alpha}||_1$ s.t. $||\mathbf{y}_t - \mathbf{D}^{t-1}\boldsymbol{\alpha}||_2^2 \leq \lambda$
- For $\boldsymbol{\alpha} \in \mathbb{R}^k$, $\quad \mathbf{D}^t = \text{argmin}_{\mathbf{D}\in\mathbb{R}^{m\times k}} \frac{1}{t} \sum_{i=1}^{t} (\frac{1}{2}|\mathbf{y}_t^i - \mathbf{D}\boldsymbol{\alpha}^i||_2^2 + \lambda||\boldsymbol{\alpha}^i||_1)$

2.3.2 Orthogonal Matching Pursuit

TODO: MARC :)

2.4 Simultaneous Sparse Coding

3 Algorithm Description

3.1 Algorithm Overview

The first part consists in initializing a dictionary that will denoise roughly the picture.
Once the picture is denoised a first time, a clustering is made in order to regroup similar patches for further treatment.
Then, iteratively for each clusters, the dictionary is updated using simultaneous sparse coding and the cluster is denoised.

3.2 Dictionnary Initialization

The initial dictionnary is first learned offline on the 10 000 images of the PASCAL VOC'07 database using the online dictionary learning procedure of [2]. This procedure is then used on the noisy picture in order to improve the dictionary efficiency.
In fact, only a fixed number T of patches in the picture are used to update the dictionary. However they are chosen so that they are independently and identically distributed in the picture.
The algorithm corresponds then to the minimization of eq.1 on the T patches with the l_1 norm using

the LARS [1] algorithm.

Input : number of iterations $\mathbf{T}$, i.i.d. sampling of $\mathbf{T}$ patches of the noisy picture $\mathbf{Y}_T$, initial dictionary $\mathbf{D}^0 \in \mathbb{R}^{m\times k}$, regularization parameter λ
Output : learned dictionary $\mathbf{D}$
Initialization : $\mathbf{A}^0 \in \mathbb{R}^{k\times k} \leftarrow 0$, $\mathbf{B}^0 \in \mathbb{R}^{m\times k} \leftarrow 0$
for *t = 1..T* **do**

$\mathbf{y}_t = \mathbf{Y}_T[\mathrm{t}]$
Sparse coding: compute with LARS algorithm:

$$\boldsymbol{\alpha}^t = \mathrm{argmin}_{\boldsymbol{\alpha}\in\mathbb{R}^k}||\boldsymbol{\alpha}||_1 \text{ s.t. } ||\mathbf{y}_t - \mathbf{D}^{t-1}\boldsymbol{\alpha}||_2^2 \leq \lambda$$

$\mathbf{A}^t \leftarrow \mathbf{A}^{t-1} + \boldsymbol{\alpha}^t\boldsymbol{\alpha}^{tT}$
$\mathbf{B}^t \leftarrow \mathbf{B}^{t-1} + \mathbf{y}_t^t\boldsymbol{\alpha}^{tT}$
Update dictionary from $\mathbf{D}^{t-1}$ to $\mathbf{D}^t$ so that:

$$\mathbf{D}^t = \mathrm{argmin}_{\mathbf{D}\in\mathbb{R}^{m\times k}}\frac{1}{t}\sum_{i=1}^{t}(\frac{1}{2}|\mathbf{y}_t^i - \mathbf{D}\boldsymbol{\alpha}^i||_2^2 + \lambda||\boldsymbol{\alpha}^i||_1)$$

end
return $\mathbf{D}^T$

Algorithm 1: Online Dictionary Learning

Input :input dictionary $\mathbf{D}= [\mathbf{d}^1, \dots , \mathbf{d}^k] \in \mathbb{R}^{m\times k}$,
$\mathbf{A} = [\mathbf{a}^1, \dots , \mathbf{a}^k] \in \mathbb{R}^{k\times k}$, $\mathbf{B} = [\mathbf{b}^1, \dots , \mathbf{b}^k]\in \mathbb{R}^{m\times k}$
Output : updated dictionary $\mathbf{D}$
repeat

for *j = 1..k* **do**

update the j^{th} column:
if *$\boldsymbol{A}$(j,j) = 0* **then**

$$\mathbf{d}^j \leftarrow 0$$

end
else

$$\mathbf{u}^j \leftarrow \frac{1}{\mathbf{A}(j,j)}(\mathbf{b}^j - \mathbf{D}\mathbf{a}^j) + \mathbf{d}^j$$

$$\mathbf{d}^j \leftarrow \frac{1}{\max(||\mathbf{u}^j||_2, 1)}\mathbf{u}^j$$

end

end

until *convergence* *Marc: apparemment, dans le code on a une boucle sur params.updateIteration = 1. Est-ce qu'il ne vaudrait mieux pas calculer l'argmin à chaque boucle et s'arrêter lorsque c'est plus petit qu'une certaine valeur ?Yohann: je n'ai pas trop compris ce que tu voulais faire. Ca c'est le pseudo code prsent par Mairal, qui est donc une sorte de descente de gradient. Ce genre d'algo converge en thorie l'infini, en pratique on prend un grand nombre d'itration. Cependant, dans ce cas on fait une minimisation alterne d'un problme plus global (selon D puis selon alpha et on itre). Du coup, et surement cause de contraintes de temps, Mairal a fix le nombre d'itration 1 mais permet de le changer en paramtre dans son algo. Du coup j'ai voulu faire pareil que Mairal.* ;
return $\mathbf{D}$

Algorithm 2: Dictionary Update updateDictionary Marc: Algo validé

Marc: TODO : choisir entre les indices ou les crochets pour les indices

Input : Input dictionary $\mathbf{D} \in \mathbb{R}^{m \times k}$, noisy patch $\mathbf{y} \in \mathbb{R}^m$, constraint $\lambda \in \mathbb{R}$
Output : code $\boldsymbol{\alpha} \in \mathbb{R}^k$
—INITIALIZATION—
G $\in \mathbb{R}^{k \times k} \leftarrow \mathbf{D}^T\mathbf{D}$
normPatch $\in \mathbb{R}^+ \leftarrow ||\mathbf{y}||_2^2$
$\boldsymbol{\alpha} \in \mathbb{R}^k \leftarrow \mathbf{0}$
$\mathcal{A} \in [0; k]^k \leftarrow \mathbf{0}$
Most correlated element
$\hat{\mathbf{c}} \in \mathbb{R}^k \leftarrow \mathbf{D}^T\mathbf{y}$
$C \in \mathbb{R}^+ \leftarrow \max_{j=1..k}(|\hat{\mathbf{c}}_j|)$
currentInd $\leftarrow j$ s.t. $\hat{\mathbf{c}}_j = C$
newAtom $\leftarrow$ **True**
if *normPatch* $< \lambda$ **then**
 return 0
end
for *i = 1..k* **do**
 LOOP, see below
end
return $\boldsymbol{\alpha}$

Algorithm 3: LARS algorithm - Mairal Version computeLars

Marc: TODO : choisir entre les indices ou les crochets pour les indices

—NEW ATOM—

if *newAtom* **then**

  $\mathcal{A}[i] \leftarrow$ currentInd

  $G_A \in \mathbb{R}^{k\times i}, G_S \in \mathbb{R}^{i\times i}$

  $G_A[i^{th}$ column$] \leftarrow G[\text{currentInd}^{th}$ column$]$

  $G_S[i^{th}$ line$] \leftarrow G_A[\text{currentInd}^{th}$ line$]$

  symmetrize G_S

  UPDATE G_S^{-1}

end

—VARIABLES UPDATES—

$\mathrm{u} \in \mathbb{R}^i \leftarrow G_S^{-1}(\mathrm{sgn}(\hat{\mathbf{c}}_{\mathcal{A}[j]}))_{j\in[1;i]}$

$C \leftarrow |\hat{\mathbf{c}}[\mathcal{A}[1]]| = \max_{j=1..k}(|\hat{\mathbf{c}}_j|)$

$\gamma \in \mathbb{R}_*^+ \leftarrow \min^+\left(\frac{C+\hat{\mathbf{c}}_j}{1+(G_Au)[j]}, \frac{C-\hat{\mathbf{c}}_j}{1-(G_Au)[j]}\right)_{j \text{ s.t. } \mathcal{A}[j]=0}$

currentInd $= j$ s.t. $\gamma = \frac{C\pm\hat{\mathbf{c}}_j}{1\pm(G_Au)[j]}$

ratio $\in \mathbb{R}^i \leftarrow \left(-\frac{\boldsymbol{\alpha}[\mathcal{A}[j]]}{u_j}\right)_{j\in[1;i]}$

stepDownDate $\in \mathbb{R}_*^+ \leftarrow \min^+$(ratio) ($\min^+$ is the minimum between strictly positive values only)

downDateInd $\in [1;k] \leftarrow \mathcal{A}[j]$ s.t. ratio[j] = stepDownDate

—POLYNOMIAL RESOLUTION—

$a \in \mathbb{R} \leftarrow \sum_{j\in[1;i]} \mathrm{sgn}(\hat{\mathbf{c}}[\mathcal{A}[j]])u[j]$

$b \in \mathbb{R} \leftarrow \sum_{j\in[1;i]} \hat{\mathbf{c}}[\mathcal{A}[j]]u[j]$

$c \in \mathbb{R} \leftarrow$ normPatch - λ

$\Delta \in \mathbb{R} \leftarrow b^2 - ac$

stepMAX $\in \mathbb{R}^+ \leftarrow \min(\frac{b-\sqrt{\Delta}}{a}, C)$

—FINAL STEP & BREAK—

$\gamma \leftarrow \min(\gamma$, stepDownDate, stepMAX$)$

for $j = 1..i$ **do**

  $\boldsymbol{\alpha}[\mathcal{A}[j]] \leftarrow \boldsymbol{\alpha}[\mathcal{A}[j]] + \gamma\mathrm{u[j]}$

end

$\hat{\mathbf{c}} \leftarrow \hat{\mathbf{c}} - \gamma G_A u$

normPatch $\leftarrow$ normPatch $+ a\gamma^2 - 2b\gamma$

if $|\gamma| < 10^{-6}$ **or** $\gamma = stepMAX$ **or** $normPatch < 10^{-6}$ **or** $normPatch - \lambda < 10^{-6}$ **then**

  break

end

if $\gamma = stepDownDate$ **then**

  DOWNDATE G_S^{-1} w.r.t downDateInd

  $\mathcal{A}$[downDateInd] $\leftarrow$ 0

  $\boldsymbol{\alpha}$[downDateInd] $\leftarrow 0$

  newAtom $\leftarrow$ **False**

  i $\leftarrow$ i - 1

end

else

  newAtom $\leftarrow$ **True**

  i $\leftarrow$ i + 1

end

Algorithm 4: LARS algorithm - Mairal Version computeLars Marc: Algo correspondant, mais à valider. - LOOP

Input : Gram matrix $G_S \in \mathbb{R}^{i \times i}$, and its former inverse to update $G_S^{-1} \in \mathbb{R}^{i-1 \times i-1}$
Output : updated $G_S^{-1} \in \mathbb{R}^{i \times i}$
if $i = 1$ **then**
  return $\frac{1}{G_S}$
end
$u \leftarrow G_S^{-1} G_S[i^{th} \text{ line}]$
$\sigma \leftarrow \frac{1}{G_s(i,i) - u.G_S[i^{th} \text{ line}]}$
$G_s^{-1}(i,i) \leftarrow \sigma$
$G_s^{-1}[i^{th} \text{ line}] \leftarrow$ -σu
return $G_s^{-1} \leftarrow G_s^{-1} + \sigma u u^T$

Algorithm 5: Update invert algorithm updateGram Marc: Algo correspondant, mais à valider.

Input : pseudo-Gram matrix $G_A \in \mathbb{R}^{k \times i}$ Gram matrix $G_S \in \mathbb{R}^{i \times i}$, and its inverse $G_S^{-1} \in \mathbb{R}^{i \times i}$, criticalInd $\in [1; k]$, current iteration i
Output : downdated matrices $G_A \in \mathbb{R}^{k \times i-1}, G_S, G_S^{-1} \in \mathbb{R}^{i-1 \times i-1}$
$\sigma \leftarrow \frac{1}{G_S^{-1}(\text{criticalInd,criticalInd})}$
$u \leftarrow G_S^{-1}[\text{criticalInd}^{th} \text{ line}]$ without its criticalIndth coefficient
for *j= criticalInd:i-1* **do**
  $G_A[j^{th} \text{ column}] \leftarrow G_A[(j+1)^{th} \text{ column}]$
  for *k= 1:criticalInd-1* **do**
    $G_S(j,k) \leftarrow G_S(j+1,k)$
    $G_S^{-1}(j,k) \leftarrow G_S^{-1}(j+1,k)$
  end
  for *k= criticalInd:i* **do**
    $G_S(j,k) \leftarrow G_S(j+1,k+1)$
    $G_S^{-1}(j,k) \leftarrow G_S^{-1}(j+1,k+1)$
  end
end
$G_s^{-1} \leftarrow G_s^{-1} - \sigma u u^T$

Algorithm 6: Downdate invert algorithm downdateGram Marc: Algo correspondant, mais à valider.

Glossary

Image Credits

© IPOL (there's no need to credit this image, here is used as an example.)

4 References

References

[1] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani *Least angle regression*. Ann. Statist., 32(2):407499, 2004.

[2] J.Mairal, F. Bach, J. Ponce, and G. Sapiro *Online dictionary learning for sparse coding*. ICML, 2009.

[3] J. Mairal, F. Bach, J. Ponce, G. Sapiro and A. Zisserman *Non-local Sparse Models for Image Restoration*. International Conference on Computer Vision, 2009.

[4] J. Mairal *Reprsentations parcimonieuses en apprentissage statistique, traitement dimage et vision par ordinateur*. PhD thesis, 2010.

[5] S. Mallat and Z. Zhang *Matching pursuit in a timefrequency dictionary*. IEEE T. SP, 41(12):33973415, 1993.

[6] M. Elad and M. Aharon *Image denoising via sparse and redundant representations over learned dictionaries*. IEEE T. IP, 54(12):37363745, 2006.