portal Documentation

Release

Author

March 02, 2017

Contents

Contents:

CHAPTER 1

Installation

Installation from the Python wheel

A Python wheel is provided for an easy installation. Simply download the wheel (.whl file) and install it with pip :

```
pip install --user wheel.whl
```

where *wheel.whl* is the wheel of the current version.

If you are *updating* PORTAL, you have to force the re-installation :

```
pip install --user --force-reinstall wheel.whl
```

Installation from the sources

Alternatively, you can build and install this package from the sources.

```
git clone git://github.com/pierrepaleo/portal
```

To generate a wheel, go in PORTAL root folder :

```
python setup.py bdist_wheel
```

The generated wheel can be installed with the aforementioned instructions.

Dependencies

To use PORTAL, you must have Python > 2.7 and numpy >= 1.8. These should come with standard Linux distributions.

Numpy is the only component absolutely required for PORTAL. For special applications, the following are required :

- The ASTRA toolbox for tomography applications
- `pywt` for Wavelets applications. This is a python module which can be installed with `apt-get install python-pywt`
- `scipy.ndimage` is used for convolutions with small kernel sizes. If not installed, all the convolutions are done in the Fourier domain, which can be slow.

Note : Python 3.* has not been tested yet.

CHAPTER 2

Chambolle-Pock algorithm in PORTAL : a turotial

The Chambolle-Pock algorithm in PORTAL

PORTAL implements a Chambolle-Pock solver for Total Variation regularization. It can solve problems of the type

where A is specified by the user. The advantage of using the Chambolle-Pock algorithm for this kind of problem is that each step is made of simple element-wise operations. This would not have been true for a FISTA Total Variation solver with general operator A :.

Example : Total Variation denoising

Here, the operator A is simply the identity. The syntax of `chambolle_pock_tv` is the following

```
import portal

# Create the noisy image (30% of the max value)
import scipy.misc
l = scipy.misc.lena().astype('f')
pc = 0.3
lb = l + np.random.rand(l.shape[0], l.shape[1])* l.max() * pc

# Define the operator and its adjoint
Id = lambda x : x
K = Id
Kadj = Id
Lambda = 20.

res  = portal.algorithms.chambollepock.chambolle_pock_tv(lb, A, Aadj, Lambda, n_it=101, return_all=Fa
portal.utils.misc.my_imshow([lb, res], shape=(1,2), cmap="gray")
```

If the norm L of $K = \left[A, \nabla\right]$ is not provided, `chambolle_pock_tv` automatically computes it.

The `chambollepock.chambolle_pock_l1_tv` function can also be used for L1-TV minimization. This is useful for noise containing strong outliers (eg. salt & pepper noise)

Fig. 2.1: Lena with gaussian noise, 30% of maximum value

Fig. 2.2: Lena denoised with Chambolle-Pock TV solver

Fig. 2.3: Lena with salt and pepper noise

Fig. 2.4: Lena denoised with Chambolle-Pock TV solver

Example : Total Variation deblurring

Here, the operator A is a blurring operator. It can be implemented with a convolution with a Gaussian kernel. PORTAL implements the convolution operator (with any 1D or 2D kernel) and its adjoint.

```
import portal

sigma = 2.6

# Define the operator A and its adjoint
gaussian_kernel = portal.utils.misc.gaussian1D(sigma) # Faster separable convolution
Blur = portal.operators.convolution.ConvolutionOperator(gaussian_kernel)
A = lambda x : Blur*x
Aadj = lambda x : Blur.adjoint() * x

# Create the blurred image
import scipy.misc
l = scipy.misc.lena().astype('f')
lb = A(l)

Lambda = 5e-2
res  = portal.algorithms.chambollepock.chambolle_pock_tv(lb, A, Aadj, Lambda, n_it=501, return_all=Fa
portal.utils.misc.my_imshow([lb, res], shape=(1,2), cmap="gray")
```

(note that here it takes more iterations to converge, and the regularization parameter is much smaller than in the denoising case).

PORTAL can also help to determine if `A` and `Aadj` are actually adjoint of eachother – an important property for the algorithm.

```
portal.operators.misc.check_adjoint(A, Aadj, lb.shape, lb.shape)
```

Fig. 2.5: Lena with gaussian blur

Fig. 2.6: Lena deblurred with Chambolle-Pock TV solver

Example : Total Variation tomographic reconstruction

Here, the operator A is the forward tomography projector. PORTAL uses the ASTRA toolbox to compute the forward and backward projector. For performances issues (the forward and backward projectors are implemented on GPU), the operators A and A^T are not exactly matched (i.e adjoint of eachother). In practice, this is not an issue for the reconstruction.

```
import portal

# Create phantom
import scipy.misc
l = scipy.misc.lena().astype(np.float32)
ph = portal.utils.misc.phantom_mask(l)

# Create Projector and Backprojector
npx = l.shape[0]
nangles = 80
AST = portal.operators.tomography.AstraToolbox(npx, nangles)

# Configure the TV regularization
Lambda = 5.0

# Configure the optimization algorithm (Chambolle-Pock for TV min)
K = lambda x : AST.proj(x)
Kadj = lambda x : AST.backproj(x, filt=True)
n_it = 101

# Run the algorithm to reconstruct the sinogram
sino = K(ph)
en, res = portal.algorithms.chambollepock.chambolle_pock_tv(sino, K, Kadj, Lambda, L=22.5, n_it=301,

# Display the result, compare to FBP
res_fbp = Kadj(sino)
```

```
portal.utils.misc.my_imshow((res_fbp, res), (1,2), cmap="gray", nocbar=True)
```

Fig. 2.7: Lena reconstructed with 80 projections, Filtered Backprojection

Note : the ASTRA toolbox comes with many available geometries ; but in PORTAL only the parallel geometry has been wrapped.

Mathematical background

Presentation of the algorithm

The Chambolle-Pock algorithm is a very versatile method to solve various optimization problems.

Suppose you want to solve the problem

Or, equivalently

where F and G are convex (possibly non smooth) and K is a linear operator.

The general form of the basic Chambolle-Pock algorithm can be written :

The primal step size τ and the dual step size σ should be chosen such that $\sigma\tau \leq 1/L^2$ where L is the norm of the operator K.

This algorithm is not a proximal gradient descent – no gradient is computed here. This is a *primal-dual* method, performing one step in the primal domain (prox of F) and one step in the dual domain (prox of G^*) ; a kind of combination of Douglas-Rachford (fully primal) and ADMM (fully dual).

Chambolle-Pock algorithm is actually much more versatile than proximal gradient algorithms – an even more flexible algorithm is described here. All you need is defining an operator K, the functions F, G and their proximal. Computing the proximal of F or G is not straightforward in general. When this cannot be done in one step, there are two solutions : re-write the optimization problem (see next section) or split again F and G like in the aforementioned algorithm.

Fig. 2.8: Lena reconstructed from 80 projections with TV minimization

Deriving the algorithm for L2-TV

The Total Variation regularized image deblurring can be written

where A is a linear operator. An attempt to solve this problem with the Chambolle-Pock algorithm would be to write

However, the proximal operator of F is

so it involves the computation of the inverse of $+\tau A^T A$, which is an ill-posed problem. This inverse can be computed if A is a convolution (since it is diagonalized by the Fourier Transform), for example in the deblurring case, but this is not the case in general.

We'll consider the case in which $A^T A$ is not easily invertible. The optimization problem has to be rewritten. We'll make use of the following equalities :

and

so the initial problem can be rewritten :

Noting that

with $K = \begin{bmatrix} A \\ \nabla \end{bmatrix}$

the problem becomes

which is the saddle-point formulation of the problem. Here, the proximal of F is the identity, and the proximal of G^* is separable with respect to q and z (since G^* is a separable sum of these variables). Its computation is straightforward :

where $B_\infty^\lambda z$ is the projection onto the L-∞ ball of radius λ, which is an elementise operation.

Eventually, the Chambolle-Pock algorithm for this problem is :

Prototyping algorithms in the primal-dual framework is more difficult than for proximal gradient algorithms ; but it enables much more flexibility. With PORTAL, the user just has to specify the linear operator for a fixed regularization.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search