A **Flex**ible Primal-Dual Tool**Box**

Hendrik Dirks

February 4, 2016

1 Introduction

Many variational problems can be written in the form

$$\arg\min_u G(u) + F(Ku), \tag{1}$$

which is generally denoted as the *primal* formulation of the minimization problem. It can be shown that minimizing (1) is equivalentto minimizing the *primal-dual* or *saddle-point* formulation

requirements

$$\arg\min_u \arg\max_p G(u) + \langle y, Ku\rangle - F^*(y). \tag{2}$$

It the recent years, efficient algorithms like ADMM or primal-dual for solving these saddle-point problems have become very popular. In this work we want to present an efficient toolbox that makes use the latter algorithm. The toolbox applies strategies like preconditioning to improve the convergence speed and has an optional C++backend to further reduce the runtime. From the user-side, the toolbox only requires to add the parts of the primal problem. Several operators are already implemented, but the toolbox is also capable of taking arbitrary operators K, as long as they have matrix form.

ref

ref

2 Examples

2.1 Rudin-Osher-Fatemi

The Rudin-Osher-Fatemi model model has very popular applications in image denoising. The primal-formulation reads

$$\arg\min_u \frac{1}{2}\|u - f\|_2^2 + \alpha\|\nabla u\|_{1,2}, \tag{3}$$

where the first part fits the unknown u to the given input image f and the second part refers to the so-called isotropic total variation, which penelized the total number jumps in the solution. Minimizing this problem with FlexBox can be done with the following lines of code

```
1  %Begin: Code example
2  main = flexbox;
3
4  numberU = main.addPrimalVar(size(f));
5
6  main.addTerm(L2dataTerm(0.5,f),numberU);
7  main.addTerm(L1gradientIso(0.08,size(f)),numberU);
8
9  main.runAlgorithm;
10
11 result = main.getPrimal(numberU);
12 %End: Code example
```

Let us begin in line 2, which initializes a **FlexBox** object and saves it in the variable *main*. Line 4 then adds the primal objective variable u which has the same size as the input image f. The toolbox returns the internal number of this primal variable, which is saved in *numberU*.
In line 6 and 7, the L^2-data-term with weight 0.5 and corresponding image f is added, moreover the isotropic TV-term with weight 0.08 is pushed into the framework. Please mention that the function *addTerm* always requires a functional part and the internal number of the corresponding primal variable.
The function call in line 9 finally starts the calculation and once this is finished we transfer the solution into the variable *result* in line 11.

2.2 Optical Flow

2.3 Labeling

3 How-To

4 Arbitrary Operators

Classes: *L1operatorIso, L1operatorAniso, L2operator, frobeniusOperator*
Adding an arbitrary operator in some norm is simple. Let us assume we some term $\alpha\|Ku\|_{1,2}$ with

$$K = \begin{pmatrix} K_1 & K_2 \\ K_3 & K_4 \end{pmatrix}, u = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix},$$

where the norm $\|\cdot\|_{1,2}$ refers to the isotropic L1-norm. We can insert this term into **FlexBox** by calling

```
1  %Begin: Code example
2  main.addTerm(L1operatorIso(alpha,{K_1,K_2,K_3,K_4}),[numberU_1,numberU_2]);
3  %End: Code example
```

The operator K has to be specified row-wise in a cell-array. From the fact that we refer to two primal variables, the toolbox knows that every two elements in the cell-array correspond to one row. Please note that empty blocks in K have to be specified as an empty sparse matrix.

5 Prox-Calculation

5.1 Frobenius-Norm

Minimization problem

$$\min_u \alpha \|Ku\|_F \tag{4}$$

Decuopling the operator K reads

$$F(\tilde{u}) = \alpha \|\tilde{u}\|_F \tag{5}$$

Calculating convex conjugate (Frobenius-norm is self-adjoint)

$$F^*(y) = \delta_{\{\|y/\alpha\|_{F^*} \leq 1\}} = \delta_{\{\|y/\alpha\|_F \leq 1\}} \tag{6}$$

Prox reads:

$$prox_{\sigma F^*}(\tilde{y}) = \min_y \|y - \tilde{y}\|_2^2 + \delta_{\{\|y/\alpha\|_F \leq 1\}} \tag{7}$$

Some calculation yields the compact solution

$$prox_{\sigma F^*}(\tilde{y}) = \frac{\tilde{y}}{\max\{1, \|\tilde{y}/\alpha\|_F\}} \tag{8}$$

5.2 Kullback-Divergence

Minimization problem

Calculate with factor alpha

$$\min_u Ku - f + f \log \frac{f}{Ku} + \delta_{\{\cdot \geq 0\}}(Ku) \tag{9}$$

Decuopling the operator reads

$$F(\tilde{u}) = \tilde{u} - f + f \log \frac{f}{\tilde{u}} + \delta_{\{\cdot \geq 0\}}(\tilde{u}) \tag{10}$$

Calculating convex conjugate

$$F^*(y) = -f \log(\mathbf{1} - y) + \delta_{\{y \geq 0\}}(\mathbf{1} - y) \tag{11}$$

Prox reads:

$$prox_{\sigma F^*}(\tilde{y}) = \min_y \frac{1}{2}\|y - \tilde{y}\|_2^2 - \sigma f \log(\mathbf{1} - y) + \delta_{\{y \geq 0\}}(\mathbf{1} - y) \tag{12}$$

Some calculation yields the compact solution

$$prox_{\sigma F^*}(\tilde{y}) = \frac{1}{2}\left(\mathbf{1} + \tilde{y} - \sqrt{(\tilde{y} - \mathbf{1})^2 + 4\sigma f}\right) \tag{13}$$

5.3 Inner Products

$$\min_u \alpha \langle Ku, b \rangle \tag{14}$$

Decoupling the operator reads

$$F(\tilde{u}) = \langle \tilde{u}, \alpha b \rangle \tag{15}$$

Calculating convex conjugate

$$F^*(y) = \sup_{\tilde{u}} \langle \tilde{u}, y \rangle - \langle \tilde{u}, \alpha b \rangle = \sup_{\tilde{u}} \langle \tilde{u}, y - \alpha b \rangle = \begin{cases} 0 & \text{for } y = \alpha b \\ \infty & \text{else} \end{cases} \tag{16}$$

Prox reads:

$$prox_{\sigma F^*}(\tilde{y}) = \min_y \frac{1}{2} \|y - \tilde{y}\|_2^2 + \sigma \delta_{\{y = \alpha b\}} \tag{17}$$

Compact solution

$$prox_{\sigma F^*}(\tilde{y}) = \alpha b \tag{18}$$

Todo list

requirements 1
ref 1
ref 1
Calculate with factor alpha 3