L1Packv2: Quick guide

Ignace Loris
Mathematics Department, Vrije Universiteit Brussel,
Pleinlaan 2, 1050 Brussel, Belgium

June 1, 2008

Abstract

The functions in this package are useful for the minimization problem

$$\bar{x}(\lambda) = \arg\min_x \|Kx - y\|^2 + 2\lambda \sum_i w_i |x_i|$$

for any given real matrix K, real data y, penalization parameter $\lambda \geq 0$ and weights $w_i \geq 0$.

1 Installation

There are two possibilities:

1. Install in the default directory
   - determine the default directory of your system by executing the following Mathematica command: `$BaseDirectory` (or `$UserBaseDirectory`)
   - copy the file L1Packv2.m to the `Applications` subdirectory of this directory.
   - You can now load the package with the Mathematica command `<<L1Packv2‘`.
2. Install in a directory of your choice
   - copy the file L1Packv2.m to a directory of your choice
   - you can now load the package with the Mathematica commands `SetDirectory["your directory"]` (use double quotes and double backslashes) and then `<<L1Packv2‘`

2 Available functions and options

The following tables list the names of the functions and variables that can be used. Their use is best explained by the examples in section 4.

The following functions are available:

Function	Description
`SoftThreshold`	Soft-thresholding operation
`ProjectionOnL1Ball`	Projection on the ℓ_1 ball of radius R
`FindMinimizer`	Calculate the minimizer $\bar{x}(\lambda)$
`CheckMinimizerList`	Check the list of minimizers and penalties
`MinimizerInterpolationFunction`	Construct a vector-valued interpolation function
`ThresholdedLandweber`	Thresholded Landweber iteration
`ProjectedLandweber`	Projected Landweber iteration
`ProjectedSteepestDescent`	Projected Steepest Descent iteration
`AdaptiveLandweber`	Adaptive Landweber iteration
`AdaptiveSteepestDescent`	Adaptive Steepest Descent iteration

The `FindMinimizer` function takes the following options:

Option	Default	Description
`Weights`	$\{1, 1, \ldots, 1\}$	weights of the components in the penalty.
`ListFunction`	`Null`	make a list of this quantity at every node (`Null` does not make a list, and the minimizer is output by itself).
`MaximumNonZero`	∞	stop when first reaching a node with this many nonzero components
`MaximumL1Norm`	∞	stop at this value for $\sum_i w_i \lvert\bar{x}_i\rvert$
`StoppingPenalty`	0	stop at this value of the penalty parameter λ
`MinimumDiscrepancy`	0	stop at this value of $\lVert K\bar{x} - y\rVert^2$.
`StoppingCondition`	False	stop when this is `True`.
`Verbose`	1	controls the amount of information that is printed while running.

Here are the variable names that can be used in `ListFunction` and `StoppingCondition`:

Name	Meaning
`Minimizer`	$\bar{x}(\lambda)$
`Counter`	index of the node
`DataMisfit`	$y - K\bar{x}$
`Remainder`	$K^T(y - K\bar{x})$
`Penalty`	λ
`Support`	the support of $\bar{x}$
`Time`	elapsed time since start

The iterative algorithms can also take the `Weights`, `ListFunction` options. `StoppingCondition` can be used with `ThresholdedLandweber`, `ProjectedLandweber` and `ProjectedSteepestDescent`.

3 Examples

This loads the package:

```
In[1]:= << L1Packv2`

In[2]:= SoftThreshold[{-3, -2., -1, 0., 1, 2, 3.}, 1]

Out[2]= {-2, -1., 0, 0., 0, 1, 2.}

In[3]:= ProjectionOnL1Ball[{1, 2, 3, 4, 5, 6, 7}, 5]

Out[3]= {0, 0, 0, 0, 2/3, 5/3, 8/3}
```

This is the basic use of `FindMinimizer`; it uses the LARS/homotopy method to find the minimizer:

```
In[4]:= mat = {{2, -1, 1}, {-1, 1, 3}, {0, 2, 4}};
        data = {11, -4, -4};
        FindMinimizer[mat, data]

Out[6]= {3, -4, 1}

In[7]:= FindMinimizer[N[mat], N[data]]

Out[7]= {3., -4., 1.}
```

If the minimize is not unique, the `FindMinimizer`-algorithm gives one possible solution.

```
In[8]:= FindMinimizer[{{1, 1}}, {1}]

Out[8]= {0, 1}

In[9]:= FindMinimizer[{{1., 1.}}, {1.}]

Out[9]= {0.5, 0.5}
```

There are several ways to stop the `FindMinimizer`:

```
In[10]:= x = FindMinimizer[mat, data, MaximumL1Norm → 5];
         {x, Norm[x, 1]}

Out[11]= {{18/5, -7/5, 0}, 5}

In[12]:= x = FindMinimizer[mat, data, StoppingPenalty → 4];
         {x, Max[Abs[Transpose[mat].(data - mat.x)]]}

Out[13]= {{25/7, -29/21, 0}, 4}

In[14]:= x = FindMinimizer[N[mat],
           N[data], MinimumDiscrepancy → 10.];
         {x, Norm[data - mat.x]^2}

Out[15]= {{3.4728, -1.3152, 0}, 10.}

In[16]:= FindMinimizer[mat, data, MaximumNonZero → 2]

Out[16]= {42/11, -17/11, 0}
```

One can make a list of intermediate data:

```
In[17]:= FindMinimizer[mat, data, ListFunction :>
           {Counter, Time, Norm[DataMisfit]^2, Norm[Minimizer, 1]}]

Out[17]= {{{0, 0., 153, 0}, {1, 0., 345/4, 3/2},
           {2, 0., 725/121, 59/11}, {3, 0., 0, 8}}, {3, -4, 1}}
```

One can verify the result with `CheckMinimizerList`:

```
In[18]:= mat = {{5, 0, 3}, {1, 1, 0}, {-4, -2, 5}};
         data = {0, 3, -2};
         sollist = FindMinimizer[mat, data,
           ListFunction :→ {Minimizer, Max[Abs[Remainder]]}];
         minimizers = Transpose[sollist[[1]]][[1]];
         lambdas = Transpose[sollist[[1]]][[2]];

In[23]:= CheckMinimizerList[mat, data, minimizers, lambdas]

Out[23]= True
```

A vector valued interpolation function is constructed as follows:

```
In[24]:= f[var_] =
          MinimizerInterpolationFunction[minimizers, lambdas, var]

Out[24]= {InterpolatingFunction[{{0, 11}}, <>][var],
          InterpolatingFunction[{{0, 11}}, <>][var],
          InterpolatingFunction[{{0, 11}}, <>][var]}

In[25]:= f[0.5]

Out[25]= {-0.183663, 2.32882, 0.349116}
```

Here's an example of numeric input:

```
In[26]:= numdata = 700;
         numvars = 900;
         mat = Table[Random[Real, {-3, 3}], {numdata}, {numvars}];
         data = Table[Random[Real, {-3, 3}], {numdata}];
         sol = FindMinimizer[mat, data,
           StoppingCondition → Counter > 1400, ListFunction :→
            {Counter, Time, Total[Abs[Sign[Minimizer]]],
             Norm[Minimizer - SoftThreshold[Minimizer + Remainder,
                 Penalty]] / Norm[Minimizer]}];
         minimizationdata = sol[[1]];
         Short[minimizationdata, 10]

Out[32]//Short=
        {{0, 0.016, 0, Indeterminate}, {1, 0.047, 1, 8.54591×10^-14},
         {2, 0.079, 2, 9.35889×10^-13}, {3, 0.094, 3, 2.93272×10^-13},
         <<1394>>, {1398, 449.469, 699, 4.49171×10^-17},
         {1399, 450.032, 698, 4.10856×10^-17},
         {1400, 450.61, 698, 4.42349×10^-17},
         {1401, 451.157, 699, 4.29316×10^-17}}
```

There are also iterative algorithms:

```
In[36]:= mat = {{0.01, -0.23, -0.49, -0.16, -0.46, 0.09},
           {-0.06, 0.18, -0.16, 0.06, -0.16, 0.35},
           {-0.13, -0.11, -0.19, 0.23, 0.32, -0.4},
           {-0.43, 0., 0.5, -0.1, -0.02, -0.4},
           {-0.02, -0.37, -0.03, 0.27, -0.06, 0.04}};
         data = {-0.94, -0.84, -0.79, 0.95, 0.38};
         w = {0.23, 0, 0.58, 0.38, 0.51, 0.92};
         lambda = 0.4;

In[40]:= FindMinimizer[mat, data, Weights → w, StoppingPenalty → 0.4]

Out[40]= {0.046483, -0.931993, 1.97814, 0, 0, 0}

In[41]:= l1nrm = Norm[w * %, 1]

Out[41]= 1.15801

In[42]:= ThresholdedLandweber[mat, data, lambda,
          Weights → w, StoppingCondition → Counter > 100]

Out[42]= {0.0464759, -0.93199, 1.97814, 0., 0., 0.}
```

```
In[43]:= ProjectedLandweber[mat, data, l1nrm,
          Weights → w, StoppingCondition → Counter > 100]
Out[43]= {0.046483, -0.931993, 1.97814, 0., 0., 0.}

In[44]:= ProjectedSteepestDescent[mat, data, l1nrm,
          Weights → w, StoppingCondition → Counter > 100]
Out[44]= {0.046483, -0.931993, 1.97814, 0., 0., 0.}

In[45]:= AdaptiveLandweber[mat, data, l1nrm, 100, Weights → w]
Out[45]= {0.0287798, -0.8965, 1.98516, 0., 0., 0.}

In[46]:= AdaptiveSteepestDescent[mat, data, l1nrm, 100, Weights → w]
Out[46]= {0.0381524, -0.923885, 1.98144, 0., 0., 0.}
```

4 References

1. "L1Packv2: A Mathematica package for minimizing an ℓ_1-penalized functional", Ignace Loris. `http://arxiv.org/abs/0710.3728` and references therein.

2. "A new approach to variable selection in least squares problems", M. R. Osborne, B. Presnell, B. A. Turlach, IMA J. Numer. Anal. 20 (3) (2000) 389–403.

3. "Least angle regression", B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, Ann. Statist. 32 (2) (2004) 407–499.