Linearly Constrained Separable Optimization using PiecewiseQuadratics.jl and LCSO.jl

Nicholas Moehle **Ellis Brown** Mykel Kochenderfer

BlackRock AI Labs

JuliaCon JuMP-dev Workshop

July 2021

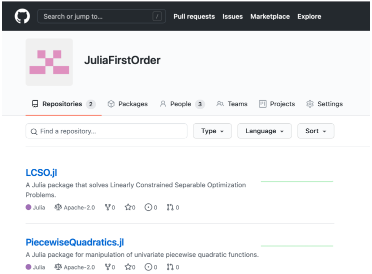
Outline

PiecewiseQuadratics.jl

LCSO.jl

JuliaFirstOrder

Portfolio Optimization



Portfolio Construction as
Linearly Constrained Separable Optimization

Nicholas Moehle Jack Gindi Stephen Boyd
Mykel J. Kochenderfer

March 10, 2021

[2103.05455]

Outline

PiecewiseQuadratics.jl

- Represent and manipulate univariate quadratic functions of the form $f(x) = px^2 + qx + r,\ \forall x \in [lb, ub]$

PiecewiseQuadratics.jl

- Represent and manipulate univariate quadratic functions of the form $f(x) = px^2 + qx + r, \; \forall x \in [lb, ub]$
- Implements several methods useful for optimization
  - sum
  - derivative
  - convex envelope
  - proximal operator
  - ...

PiecewiseQuadratics.jl

- Represent and manipulate univariate quadratic functions of the form $f(x) = px^2 + qx + r,\ \forall x \in [lb, ub]$
- Implements several methods useful for optimization
  - sum
  - derivative
  - convex envelope
  - proximal operator
  - ...
- Applications to cost functions
  - Higher fidelity
  - Computationally tractable

Example

$$f(x) = \begin{cases} x^2 - 3x + 3 & \text{if } x \in [-\infty, 3] \\ -x + 3 & \text{if } x \in [3, 4] \\ 2x^2 - 20x + 47 & \text{if } x \in [4, 6] \\ x - 7 & \text{if } x \in [6, 7.5] \\ 4x - 29 & \text{if } x \in [7.5, \infty] \end{cases}$$

Example

$$f(x) = \begin{cases} x^2 - 3x + 3 & \text{if } x \in [-\infty, 3] \\ - x + 3 & \text{if } x \in [3, 4] \\ 2x^2 - 20x + 47 & \text{if } x \in [4, 6] \\ x - 7 & \text{if } x \in [6, 7.5] \\ 4x - 29 & \text{if } x \in [7.5, \infty] \end{cases}$$

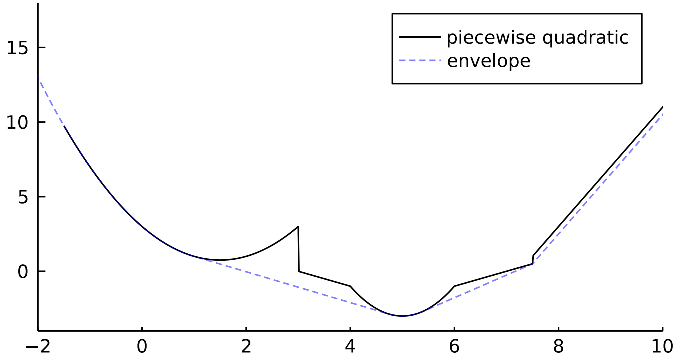
```julia
using PiecewiseQuadratics
f = PiecewiseQuadratic([
  # BoundedQuadratic(  lb,  ub,   p,     q,     r),
    BoundedQuadratic(-Inf, 3.0, 1.0,  -3.0,   3.0),
    BoundedQuadratic( 3.0, 4.0, 0.0,  -1.0,   3.0),
    BoundedQuadratic( 4.0, 6.0, 2.0, -20.0,  47.0),
    BoundedQuadratic( 6.0, 7.5, 0.0,   1.0,  -7.0),
    BoundedQuadratic( 7.5, Inf, 0.0,   4.0, -29.0)
]);
```

Plot

```
using Plots
plot(get_plot(f); ...)
plot!(get_plot(simplify(envelope(f))); ...)
```

Plot

```
using Plots
plot(get_plot(f); ...)
plot!(get_plot(simplify(envelope(f))); ...)
```



Outline

PiecewiseQuadratics.jl

LCSO.jl

JuliaFirstOrder

Portfolio Optimization

Linearly Constrained Separable Optimiaziton

A *linearly constrained separable optimization* (LCSO) problem:

$$\begin{array}{ll} \text{minimize} & f(x) = \sum_{i=1}^{n} f_i(x_i) \\ \text{subject to} & Ax = b, \end{array} \tag{1}$$

where

- the decision variable is $x \in \mathbf{R}^n$
- the parameters are $A \in \mathbf{R}^{m \times n}$, $b \in \mathbf{R}^m$, and the functions f_i

[1] See: arXiv:2103.05455

Linearly Constrained Separable Optimiaziton

A *linearly constrained separable optimization* (LCSO) problem:

$$\begin{array}{ll} \text{minimize} & f(x) = \sum_{i=1}^{n} f_i(x_i) \\ \text{subject to} & Ax = b, \end{array} \tag{1}$$

where

- the decision variable is $x \in \mathbf{R}^n$
- the parameters are $A \in \mathbf{R}^{m \times n}$, $b \in \mathbf{R}^m$, and the functions f_i

LCSO problems can be solved using ADMM.[1]

- Exactly, if the f_i are convex
- Approximately, if they aren't

[1]See: arXiv:2103.05455

Extended-Form LCSO Problems

An extended-form LCSO problem:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T P x + q^T x + \sum_{i=1}^{n} f_i(x_i) \\ \text{subject to} & Ax = b, \end{array} \tag{2}$$

where

- $q \in \mathbf{R^n}$, $P \in \mathbf{S^n_+}$ is a symmetric positive semidefinite matrix
- f_i are piecewise quadratic functions

[2] See: arXiv:2103.05455

Extended-Form LCSO Problems

An extended-form LCSO problem:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T P x + q^T x + \sum_{i=1}^{n} f_i(x_i) \\ \text{subject to} & Ax = b, \end{array} \tag{2}$$

where

- $q \in \mathbf{R^n}$, $P \in \mathbf{S^n_+}$ is a symmetric positive semidefinite matrix
- f_i are piecewise quadratic functions
- Extended-form LCSO problems can be reduced to standard LCSO problem (using an eigendecomposition)[2]

[2] See: arXiv:2103.05455

LCSO.jl

- Solves LCSO problems using ADMM (more later...)

LCSO.jl

- Solves LCSO problems using ADMM (more later...)
- Works with either standard- or extended-form problems

LCSO.jl

- Solves LCSO problems using ADMM (more later...)
- Works with either standard- or extended-form problems
- Can be used to solve convex and non-convex problems (convergence is only guaranteed for convex problems)

LCSO.jl

- Solves LCSO problems using ADMM (more later...)
- Works with either standard- or extended-form problems
- Can be used to solve convex and non-convex problems (convergence is only guaranteed for convex problems)
- In practice, converges to a moderately accurate solution quickly (even if the f_i are very complicated)

LCSO.jl

- Solves LCSO problems using ADMM (more later...)
- Works with either standard- or extended-form problems
- Can be used to solve convex and non-convex problems (convergence is only guaranteed for convex problems)
- In practice, converges to a moderately accurate solution quickly (even if the f_i are very complicated)
- The separable functions f_i must be piecewise quadratic (see `PiecewiseQuadratics.jl`)

LCSO.jl

- Solves LCSO problems using ADMM (more later...)
- Works with either standard- or extended-form problems
- Can be used to solve convex and non-convex problems (convergence is only guaranteed for convex problems)
- In practice, converges to a moderately accurate solution quickly (even if the f_i are very complicated)
- The separable functions f_i must be piecewise quadratic (see `PiecewiseQuadratics.jl`)
- (In theory, could be extended to any f_i that support a `prox` method)

Example

```
using LCSO
using PiecewiseQuadratics

n = 4  # num features
m = 2  # num constraints
```

Example

```julia
using LCSO
using PiecewiseQuadratics

n = 4  # num features
m = 2  # num constraints

# construct problem data
x0 = rand(n)
X = rand(n, n)
```

Example

```
using LCSO
using PiecewiseQuadratics

n = 4  # num features
m = 2  # num constraints

# construct problem data
x0 = rand(n)
X = rand(n, n)

# ensure P is PSD
P = X'X
q = rand(n)
A = rand(m, n)
b = A * x0
```

Example

```julia
using LCSO
using PiecewiseQuadratics

n = 4  # num features
m = 2  # num constraints

# construct problem data
x0 = rand(n)
X = rand(n, n)

# ensure P is PSD
P = X'X
q = rand(n)
A = rand(m, n)
b = A * x0
```

Recall the extended-form (2)

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}x^T Px + q^T x + \sum_{i=1}^{n} f_i(x_i) \\ \text{subject to} & Ax = b. \end{array}$$

Example

```
# x_1 has to be ∈ [−1, 2] ∪ [2.5, 3.5]
#   with quadratic penalty ∈ [−1, 2]
#   and linear penalty ∈ [2.5, 3.5]
f1 = PiecewiseQuadratic([
  BoundedQuadratic( -1,  2,1,0,0),
  BoundedQuadratic(2.5,3.5,0,1,0)
])
```

Example

```
# x₁ has to be ∈ [−1, 2] ∪ [2.5, 3.5]
#   with quadratic penalty ∈ [−1, 2]
#   and linear penalty ∈ [2.5, 3.5]
f1 = PiecewiseQuadratic([
  BoundedQuadratic( -1,  2,1,0,0),
  BoundedQuadratic(2.5,3.5,0,1,0)
])

# x₂ has to be ∈ [−20, 10]
f2 =  PiecewiseQuadratic([
  BoundedQuadratic(-20,10,0,0,1)
])
```

Example

```
# x_1 has to be ∈ [−1,2] ∪ [2.5,3.5]
#   with quadratic penalty ∈ [−1,2]
#   and linear penalty ∈ [2.5,3.5]
f1 = PiecewiseQuadratic([
  BoundedQuadratic( -1,  2,1,0,0),
  BoundedQuadratic(2.5,3.5,0,1,0)
])

# x_2 has to be ∈ [−20,10]
f2 =  PiecewiseQuadratic([
  BoundedQuadratic(-20,10,0,0,1)
])

# x_3 has to be ∈ [−5,10]
f3 = indicator(-5, 10)
```

Example

```
# x₁ has to be ∈ [−1,2] ∪ [2.5,3.5]
#   with quadratic penalty ∈ [−1,2]
#   and linear penalty ∈ [2.5,3.5]
f1 = PiecewiseQuadratic([
  BoundedQuadratic( -1,  2,1,0,0),
  BoundedQuadratic(2.5,3.5,0,1,0)
])

# x₂ has to be ∈ [−20,10]
f2 =  PiecewiseQuadratic([
  BoundedQuadratic(-20,10,0,0,1)
])

# x₃ has to be ∈ [−5,10]
f3 = indicator(-5, 10)

# x₄ has to be exactly 1.231
f4 = indicator(1.231, 1.231)
```

Example

```
# x₁ has to be ∈ [−1,2] ∪ [2.5,3.5]
#   with quadratic penalty ∈ [−1,2]
#   and linear penalty ∈ [2.5,3.5]
f1 = PiecewiseQuadratic([
  BoundedQuadratic( -1,  2,1,0,0),
  BoundedQuadratic(2.5,3.5,0,1,0)
])

# x₂ has to be ∈ [−20,10]
f2 =  PiecewiseQuadratic([
  BoundedQuadratic(-20,10,0,0,1)
])

# x₃ has to be ∈ [−5,10]
f3 = indicator(-5, 10)

# x₄ has to be exactly 1.231
f4 = indicator(1.231, 1.231)
```

```
f = [f1, f2, f3, f4]
```

Example

```julia
# x₁ has to be ∈ [−1,2] ∪ [2.5,3.5]
#   with quadratic penalty ∈ [−1,2]
#   and linear penalty ∈ [2.5,3.5]
f1 = PiecewiseQuadratic([
  BoundedQuadratic( -1,  2,1,0,0),
  BoundedQuadratic(2.5,3.5,0,1,0)
])

# x₂ has to be ∈ [−20,10]
f2 =  PiecewiseQuadratic([
  BoundedQuadratic(-20,10,0,0,1)
])

# x₃ has to be ∈ [−5,10]
f3 = indicator(-5, 10)

# x₄ has to be exactly 1.231
f4 = indicator(1.231, 1.231)
```

```julia
f = [f1, f2, f3, f4]

params = AdmmParams(P, q, A, b, f)
```

Example

```julia
# x₁ has to be ∈ [−1,2] ∪ [2.5,3.5]
#   with quadratic penalty ∈ [−1,2]
#   and linear penalty ∈ [2.5,3.5]
f1 = PiecewiseQuadratic([
  BoundedQuadratic( -1,  2,1,0,0),
  BoundedQuadratic(2.5,3.5,0,1,0)
])

# x₂ has to be ∈ [−20,10]
f2 =  PiecewiseQuadratic([
  BoundedQuadratic(-20,10,0,0,1)
])

# x₃ has to be ∈ [−5,10]
f3 = indicator(-5, 10)

# x₄ has to be exactly 1.231
f4 = indicator(1.231, 1.231)
```

```julia
f = [f1, f2, f3, f4]

params = AdmmParams(P, q, A, b, f)

# solve
vars, stats = optimize(params)
```

Example

```julia
# x_1 has to be ∈ [−1,2] ∪ [2.5,3.5]
#   with quadratic penalty ∈ [−1,2]
#   and linear penalty ∈ [2.5,3.5]
f1 = PiecewiseQuadratic([
  BoundedQuadratic( -1,  2,1,0,0),
  BoundedQuadratic(2.5,3.5,0,1,0)
])

# x_2 has to be ∈ [−20,10]
f2 =  PiecewiseQuadratic([
  BoundedQuadratic(-20,10,0,0,1)
])

# x_3 has to be ∈ [−5,10]
f3 = indicator(-5, 10)

# x_4 has to be exactly 1.231
f4 = indicator(1.231, 1.231)
```

```julia
f = [f1, f2, f3, f4]

params = AdmmParams(P, q, A, b, f)

# solve
vars, stats = optimize(params)

print(vars.x)  # optimal x
# [-0.0493, 1.218, -1.932, 1.231]
```

.

Applications

- Portfolio optimization

Applications

- Portfolio optimization
- Radiation treatment planning

Applications

- Portfolio optimization
- Radiation treatment planning
- Dynamic energy management

Applications

- Portfolio optimization
- Radiation treatment planning
- Dynamic energy management
- ...

Outline

JuliaFirstOrder

- GitHub organization for first-order methods in Julia
  - https://github.com/JuliaFirstOrder
- Plan to migrate several related packages
  - ProximalOperators.jl
  - ProximalAlgorithms.jl
  - StructuredOptimization.jl
  - ...
- Thank you to Miles Lubin for the introductions!

Outline

Portfolio Optimization

The original mean–variance portfolio optimization problem of Markowitz, *i.e.*, "Risk Minimization"

$$\begin{array}{ll} \text{minimize} & \gamma x^T \Sigma x - \mu^T x \\ \text{subject to} & \mathbf{1}^T x = 1, \end{array} \qquad (3)$$

where

- $x \in \mathbf{R}^n$ is the fraction of the portfolio value in each of n assets

Portfolio Optimization

The original mean–variance portfolio optimization problem of Markowitz, *i.e.*, "Risk Minimization"

$$\begin{array}{ll} \text{minimize} & \gamma x^T \Sigma x - \mu^T x \\ \text{subject to} & \mathbf{1}^T x = 1, \end{array} \tag{3}$$

where

- $x \in \mathbf{R}^n$ is the fraction of the portfolio value in each of n assets
- $\mu \in \mathbf{R}^n$ is the expected return forecast for the n assets, meaning $\mu^T x$ is the expected portfolio return

Portfolio Optimization

The original mean–variance portfolio optimization problem of Markowitz, *i.e.*, "Risk Minimization"

$$\begin{array}{ll} \text{minimize} & \gamma x^T \Sigma x - \mu^T x \\ \text{subject to} & \mathbf{1}^T x = 1, \end{array} \tag{3}$$

where

- $x \in \mathbf{R}^n$ is the fraction of the portfolio value in each of n assets
- $\mu \in \mathbf{R}^n$ is the expected return forecast for the n assets, meaning $\mu^T x$ is the expected portfolio return
- $\Sigma \in \mathbf{S}^n_{++}$ is the asset return covariance matrix, meaning $x^T \Sigma x$ is the variance of the portfolio return

Portfolio Optimization

The original mean–variance portfolio optimization problem of Markowitz, *i.e.*, "Risk Minimization"

$$\begin{array}{ll} \text{minimize} & \gamma x^T \Sigma x - \mu^T x \\ \text{subject to} & \mathbf{1}^T x = 1, \end{array} \tag{3}$$

where

- $x \in \mathbf{R}^n$ is the fraction of the portfolio value in each of n assets
- $\mu \in \mathbf{R}^n$ is the expected return forecast for the n assets, meaning $\mu^T x$ is the expected portfolio return
- $\Sigma \in \mathbf{S}^n_{++}$ is the asset return covariance matrix, meaning $x^T \Sigma x$ is the variance of the portfolio return
- $\gamma > 0$ is the risk aversion parameter

...With Separable Costs!

The portfolio optimization problem with separable costs:

$$\begin{array}{ll} \text{minimize} & \gamma x^T \Sigma x - \mu^T x + \sum_{i=1}^n f_i(x_i) \\ \text{subject to} & \mathbf{1}^T x = 1, \end{array} \tag{4}$$

where

- f_i are asset-level penalties, like taxes and trading costs, and are piecewise quadratic

Separable cost examples

- Trading costs:

$$f_i^{\mathtt{trd}}(x_i) = s_i|x_i - x_{\mathrm{init},i}| + \begin{cases} 0 & x_i = x_{\mathrm{init},i} \\ c_i^{\mathtt{trd}} & \text{otherwise} \end{cases}$$

Separable cost examples

- Trading costs:

$$f_i^{\mathrm{trd}}(x_i) = s_i|x_i - x_{\mathrm{init},i}| + \begin{cases} 0 & x_i = x_{\mathrm{init},i} \\ c_i^{\mathrm{trd}} & \text{otherwise} \end{cases}$$

- Holding cost:

$$f_i^{\mathrm{hold}}(x_i) = \begin{cases} 0 & x_i = 0 \\ c_i^{\mathrm{hold}} & \text{otherwise} \end{cases}$$

Separable cost examples

- Trading costs:

$$f_i^{\mathrm{trd}}(x_i) = s_i|x_i - x_{\mathrm{init},i}| + \begin{cases} 0 & x_i = x_{\mathrm{init},i} \\ c_i^{\mathrm{trd}} & \text{otherwise} \end{cases}$$

- Holding cost:

$$f_i^{\mathrm{hold}}(x_i) = \begin{cases} 0 & x_i = 0 \\ c_i^{\mathrm{hold}} & \text{otherwise} \end{cases}$$

- Position limits:

$$f_i^{\mathrm{lim}}(x_i) = \begin{cases} 0 & x_{\mathrm{lb},i} \leq x_i \leq x_{\mathrm{ub},i} \\ \infty & \text{otherwise} \end{cases}$$

Separable cost examples

- Trading costs:

$$f_i^{\mathrm{trd}}(x_i) = s_i|x_i - x_{\mathrm{init},i}| + \begin{cases} 0 & x_i = x_{\mathrm{init},i} \\ c_i^{\mathrm{trd}} & \text{otherwise} \end{cases}$$

- Holding cost:

$$f_i^{\mathrm{hold}}(x_i) = \begin{cases} 0 & x_i = 0 \\ c_i^{\mathrm{hold}} & \text{otherwise} \end{cases}$$

- Position limits:

$$f_i^{\mathrm{lim}}(x_i) = \begin{cases} 0 & x_{\mathrm{lb},i} \le x_i \le x_{\mathrm{ub},i} \\ \infty & \text{otherwise} \end{cases}$$

- Combinations of these:

$$f_i(x_i) = f_i^{\mathrm{trd}}(x_i) + f_i^{\mathrm{hold}}(x_i) + f_i^{\mathrm{lim}}(x_i)$$

Relaxation

- Problems are often nonconvex
- One way to handle this: **Relax** the problem by replacing f_i with its **convex envelope**

$$f_i^{**}(x) = \sup\{g(x) \mid g \text{ is convex and } g(x) \leq f_i(x),\ x \in \mathbf{dom}(f_i)\},$$

then use the result to recover a solution to the original problem

Relaxation

- Problems are often nonconvex
- One way to handle this: **Relax** the problem by replacing f_i with its **convex envelope**

$$f_i^{**}(x) = \sup\{g(x) \mid g \text{ is convex and } g(x) \leq f_i(x),\ x \in \mathbf{dom}(f_i)\},$$

  then use the result to recover a solution to the original problem
- Easy to compute when f_i are piecewise quadratic
- Supported by `PiecewiseQuadratics.jl`

Questions?