JuliaOpt

Optimization packages in Julia

Iain Dunning
Operations Research Center
MIT

http://iaindunning.com
@iaindunning
github.com/IainNZ
What is Optimization?

$$\min_x f(x)$$

subject to $g_i(x) \leq 0 \quad \forall i$
What is Optimization?

What is Optimization?

M. Udell, and S. Boyd, “Maximizing a Sum of Sigmoids”
What is JuliaOpt? Packages for...

- **Modeling**: express optimization problems with in Julia code
- **Solving**: pure Julia routines, and wrappers for external solvers
- **Abstracting**: the “glue” between modeling, solving, and user code
JuliaOpt as an Organization

- Standards for packages: binaries, documentation, tests, integration
- A centralized guide to optimization in Julia
  - http://www.juliaopt.org/
- Examples, documentation, guides
  - http://www.juliaopt.org/notebooks/index.html
Modeling with JuliaOpt - Two Paths

**JuMP**
- General linear, quadratic, nonlinear, integer optimization tool
- Callbacks for advanced integer programming
- Automatic generation of first- and second-order derivatives

**Convex.jl**
- Disciplined Convex Programming
- Linear, second-order, semidefinite, exponential conic optimization
- Automatic validation of model convexity
\[
\begin{align*}
\min_{u, y} & \quad \frac{1}{4} \Delta_x \left( (y_{m,0} - y_0^t)^2 + 2 \sum_{j=1}^{n-1} (y_{m,j} - y_j^t)^2 + (y_{m,n} - y_n^t)^2 \right) + \\
& \quad \frac{1}{4} a \Delta_t \left( 2 \sum_{i=1}^{m-1} u_i^2 + u_m^2 \right) \\
\text{s.t.} & \quad \frac{1}{\Delta_t} (y_{i+1,j} - y_{i,j}) = \\
& \quad \frac{1}{2h_2} (y_{i,j-1} - 2y_{i,j} + y_{i,j+1} + y_{i+1,j-1} - 2y_{i+1,j} + y_{i+1,j+1}) \quad \forall i \in I', j \in J' \\
y_{0,j} = 0 \quad \forall j \in J \\
y_{i,2} - 4y_{i,1} + 3y_{i,0} = 0 \quad \forall i \in I \\
\frac{1}{2\Delta_x} (y_{i,n-2} - 4y_{i,n-1} + 3y_{i,n}) = u_i - y_{i,n} \quad \forall i \in I \\
-1 \leq u_i \leq 1 \quad \forall i \in I \\
0 \leq y_{i,j} \leq 1 \quad \forall i \in I, j \in J
\end{align*}
\]
\[
\begin{align*}
\max_{x_{ijk} \in \{0,1\}} \quad 0 \\
\text{subject to} \quad \sum_{i} x_{ijk} &= 1 \quad \forall j, k \\
\sum_{j} x_{ijk} &= 1 \quad \forall i, k \\
\sum_{k} x_{ijk} &= 1 \quad \forall i, j \\
a+2 \quad b+2 \\
\sum_{i=a} \sum_{j=b} x_{ijk} &= 1 \quad \forall a, b \in \{1, 4, 7\}, k
\end{align*}
\]
\[
\begin{align*}
\max_{x \geq 0} & \quad c^T x \\
\text{subject to} & \quad Ax = b
\end{align*}
\]
m = Model()

@defVar(m, x[i=1:9, j=1:9, v=1:9], Bin)

@addConstraint(m, cols[j=1:9,v=1:9],
    sum{x[i,j,v], i=1:9} == 1)

@addConstraint(m, rows[i=1:9,v=1:9],
    sum{x[i,j,v], j=1:9} == 1)

@addConstraint(m, digits[i=1:9,j=1:9],
    sum{x[i,j,v], v=1:9} == 1)

@addConstraint(m,
    cells[a=1:3:7,b=1:3:7,v=1:9],
    sum{x[i,j,v], i=a:a+2, j=b:b+2} == 1)

for i in 1:9, j in 1:9
    @addConstraint(m, x[i,j,S[i,j]] == 1)
end

solve(m)
JuMP for nonlinear optimization

\[
\min_x \quad f(x)
\]
subject to \[ g_i(x) \leq 0 \quad \forall i \]

Solvers need *derivative-evaluating* functions \[ f(x), g_i(x) \]
\[ \nabla f(x), \nabla g_i(x) \]
\[ \nabla^2 f(x) + \lambda \sum_i \nabla^2 g_i(x) \]
JuMP + Automatic Differentiation

```plaintext
@setNLOObjective(mod, Min, sum{ (bus[k].p_load +
    sum{ bus_voltage[k] * bus_voltage[branch[i].from] *
      (Gin[i] * cos(bus_angle[k] - bus_angle[branch[i].from]) +
       Bin[i] * sin(bus_angle[k] - bus_angle[branch[i].from])),
      i=in_lines[k] } +

    sum{ bus_voltage[k] * bus_voltage[branch[i].to] *
      (Gout[i] * cos(bus_angle[k] - bus_angle[branch[i].to]) +
       Bout[i] * sin(bus_angle[k] - bus_angle[branch[i].to])),
      i=out_lines[k] } +

    bus_voltage[k]^2*Gself[k] )^2,
  k in 1:nbus;
  bus[k].bustype == 2 || bus[k].bustype == 3})
```

\[
\sum_k \left[ g_k + \sum_m V_k V_m (G_{km} \cos(\theta_k - \theta_m) + B_{km} \sin(\theta_k - \theta_m)) \right]^2
\]
Convex.jl - DCP in Julia

“DCP is a system for constructing mathematical expressions with known curvature from a given library of base functions.”

→ Specify your optimization problem & Convex.jl will analyze the convexity and reduce to a standard form.

http://stanford.edu/~boyd/cvxbook/bv_cvxslides.pdf
Max Volume Inscribed Ellipsoid

“Given polyhedron $C = \{x \mid a_i^T x \leq b_i, \ i = 1, \ldots, m\}$ find ellipsoid $\mathcal{E} = \{Bu + d \mid \|u\|_2 \leq 1\}$ that lies in the interior of $C$ with maximum volume”

\[
\begin{align*}
\text{maximize} & \quad \log \det B \\
\text{subject to} & \quad \sup_{\|u\|_2 \leq 1} I_C(Bu + d) \leq 0
\end{align*}
\]
Max Volume Inscribed Ellipsoid

$$\text{maximize} \quad \log \det B$$

subject to \quad \|Ba_i\|_2 + a_i^T d \leq b_i, \quad i = 1, \ldots, m.$$

Is that objective concave?

→ B is positive definite matrix...

→ \det(B) = \text{product of eigenvalues of } B = +\text{ve}...

→ \log \text{ of positive } x = \text{concave}
Max Volume Inscribed Ellipsoid

using Convex

a = \{ [ 2, 1], [ 2,-1], [-1, 2], [-1,-2] \}
B = Variable(2,2)
d = Variable(2)
p = maximize(logdet(B))
for i in 1:4
    p.constraints += norm(B*a[i]) +
        dot(a[i],d) <= 1
end
solve!(p)
println(B.value)
println(d.value)
Which do I use?

- Convex but transformation not obvious or is painful? Nonlinear but structured (e.g. GP, exponential cones) → **Convex.jl**

- Complex indexing (3+ dimensions, not 1:n)? Nonlinear, nonconvex? Large scale linear/quadratic, solver callbacks? → **JuMP**
Solving your problems

Cbc.jl
ECOS.jl
Ipopt.jl
NLopt.jl
Optim.jl
LsqFit.jl

Clp.jl
GLPK.jl
KNITRO.jl
SCS.jl

CPLEX.jl
Gurobi.jl
Mosek.jl
CoinOptServices.jl
AmpNLWriter.jl
MathProgBase.jl

- Standard interface for optimization Julia
- Crucial to the success of JuliaOpt
MathProgBase.jl Design & Benefits

- “Don’t try to create interface/abstraction unless you have two or more cases”
  - Callbacks: “many states, one callback” vs “many states, many callbacks”
  - SDP interface

- Multiplier effect of participation
  - JuMP initial consumer, now also Convex.jl
  - Each added solver benefits all
JuliaOpt + MPB for new solvers

1. If making new pure Julia solver, can get problems to your solver easily

2. Compose solvers for new problem classes
   - e.g. “mixed-integer non-convex quadratically constrained optimization”
   - Solve as series of mixed-integer linear problems
   - JuMP → MathProgBase → MySolver → MathProgBase → AnyMILPSolver
Optim.jl

- Pure Julia routines for unconstrained and box-constrained optimization problems
- Line searches, BFGS, Levenberg-Marquadt, Nelder-Mead, etc.
- Can provide function and derivatives, or…
- Ask for derivatives to be provided by autodifferentiation (DualNumbers.jl)

function rosenbrock100(x::Vector)
    out = zero(eltype(x))
    for i in 1:div(length(x),2)
        out += 100*(x[2i-1]^2 - x[2i])^2 + (x[2i-1]-1)^2
    end
    out
end

@time optimize(rosenbrock100, zeros(100),
    method = :l_bfgs, iterations=21)
# elapsed time: 0.003834211 seconds
# Value of Function at Minimum: 3.419262

@time optimize(rosenbrock100, zeros(100),
    method = :l_bfgs, iterations=21, autodiff=true)
# elapsed time: 0.002318992 seconds
# Value of Function at Minimum: 0.000000
Building on JuliaOpt

● Packages using Optim.jl directly
  ○ KernelDensity.jl, KernelEstimator.jl,
    GaussianProcesses.jl, MachineLearning.jl, NLsolve.jl,
    QuantEcon.jl, RegERMs.jl, StochasticSearch.jl,
    TimeModels.jl, TrafficAssignment.jl

● Packages using JuMP (optionally)
  ○ Gadfly/Compose.jl, GraphLayout.jl
JuMP Extensions

- JuMPeR - Robust Optimization [https://github.com/IainNZ/JuMPeR.jl]
- JuMPChance - Chance Constraints [https://github.com/mlubin/JuMPChance.jl]
- StochJuMP - Stochastic Optimization [https://github.com/joehuchette/StochJuMP.jl]
Education, Academia & Industry

- JuliaOpt been used at 5+ universities around the world
- Many papers starting to appear using JuliaOpt
  - Vielma, J, et al. "Extended Formulations in Mixed Integer Conic Quadratic Programming"
  - Gorhan, Mackey. “Measuring Sample Quality with Stein’s Method”
  - Giordano, Broderick, Jordan. “Linear Response Methods for Accurate Covariance Estimates from Mean Field Variational Bayes”
- Several companies, incl. [https://www.staffjoy.com/](https://www.staffjoy.com/)
- See [http://juliaopt.org](http://juliaopt.org) for latest info, email us!
Thanks to our contributors!

blakejohnson
Blake Johnson

carlobaldassi
Carlo Baldassi

cmbcbride
Cameron McBride

davidlizeng
David Zeng

lainNZ
Iain Dunning

jennyhong
Jenny Hong

jfsantos
João Felipe Santos

joehuchette
Joey Huchette

johnmyleswhite
John Myles White

karanveerm
Karanveer Mohan

lindahua
Dahua Lin

madeleineudell
Madeleine Udell

mlubin
Miles Lubin

stevenj
Steven G. Johnson

timholy
Tim Holy

tkelman
Tony Kelman

ulfworsoe
Ulf Worsøe

yesian
Yeesian Ng