KL-Regularized Least Squares

Nicholas Barnield Michael P. Friedlander Tim Hoheisel

2024-05-06

```
Precompiling KLLS
  ✓ ArrayInterface
  ✓ OffsetArrays
  ✓ OffsetArrays → OffsetArraysAdaptExt
  ✓ ArrayInterface → ArrayInterfaceStaticArraysCoreExt
  ✓ ArrayInterface → ArrayInterfaceReverseDiffExt
  ✓ StaticArrayInterface
  ✓ StaticArrayInterface → StaticArrayInterfaceStaticArraysExt
  ✓ StaticArrayInterface → StaticArrayInterfaceOffsetArraysExt
  ✓ CloseOpenIntervals
  ✓ LayoutPointers
  ✓ VectorizationBase
  ✓ SLEEFPirates
  ✓ LoopVectorization
  ✓ LoopVectorization → SpecialFunctionsExt
  ✓ LoopVectorization → ForwardDiffExt
  ✓ Octavian
        Info Given KLLS was explicitly requested, output will be shown live
WARNING: using NLPModels.grad in module KLLS conflicts with an existing
identifier.
WARNING: using NLPModels.hess in module KLLS conflicts with an existing
identifier.
  ✓ KLLS
  ✓ Octavian → ForwardDiffExt
 18 dependencies successfully precompiled in 21 seconds. 202 already precompiled.
  1 dependency had output during precompilation:
┌ KLLS
│  [Output was shown above]
└
```

Load physics data

The file `PhysicsData.npz` contains the matrix and vectors generated by the script `PhysicsData.py`, where we get the output from `generate_data`:

- `A` is the first output argument `A`
- `b` is the second output argument `data0`

- `x0` is the third input argument, which seems to be the ground truth distribution.

```
data = npzread("PhysicsData.npz", ["A", "b", "x0"])
@unpack A, b, x0 = data
klprob = KLLSData(A, b)
```

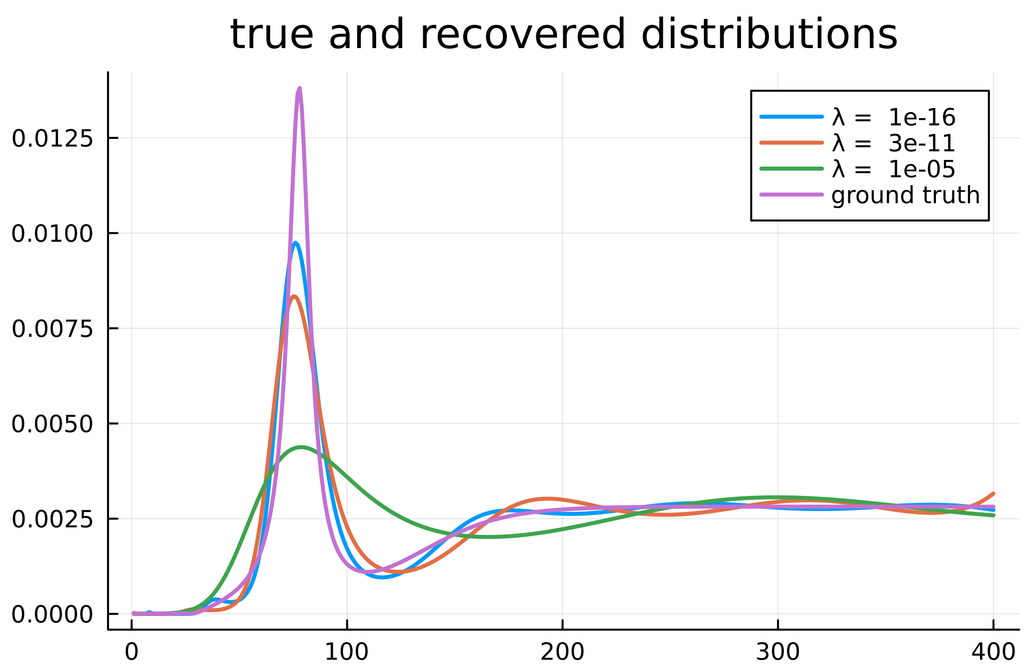
Solve over a range of regularization parameters

Solve the problem for a range of logarithmically spaced regularization parameters λ between 10^{-8} and 1.

```
stats = map(exp10.(range(-16, stop=-5, length=3))) do λ
          klprob.λ = λ
          p, y, stats = newtoncg(klprob)
          (λ=λ, p=p, iters=stats.iter, ∇dNrm=stats.dual_feas)
        end;
```

Plot the recovered distributions for each value of λ and overlay the ground truth distribution x_0.

```
lab = hcat([@sprintf("λ = %6.0e", λ) for λ in getfield.(stats, :λ)]...)
default(lw=2, title="true and recovered distributions")
plot(getfield.(stats, :p), label=lab)
plot!(x0, label="ground truth")
```



The curve corresponding to the smallest parameter λ (1e-8) best approximates the modes of the ground-truth distribution, but smaller values of λ don't help.

These tests use a uniform prior. Does the data generator make a prior available? If so, this could be used to improve the results.

Accuracy of the solution

Now try to solve the problem as accurately as possible. Set $\lambda = 0$ and the tightest tolerance reasonable.

```
klprob.λ = 0.0
p, y, stats = newtoncg(klprob, atol=1e-12, rtol=1e-12, verbose=0)
```

```
m, n = size(A)
@printf("%20s: %11.2e\n", "rms(p-x0)", norm(p - x0)/√n)
@printf("%20s: %11.2e\n", "rms(Ap-b)", norm(A*p - b)/√m)
@printf("%20s: %11f\n", "Solve time", stats.elapsed_time)
@printf("%20s: %11d\n", "Number of iterations", stats.iter)
plot([p x0], lab=["p" "x0"], title="true vs recovered distributions")
```

```
           rms(p-x0):    3.35e-04
           rms(Ap-b):    2.52e-10
          Solve time:   30.000023
Number of iterations:      196578
```