Linear Least Squares Modeling in Haskell

Tri Dao

CS 240H
Stanford University

March 16, 2016

Least squares problem

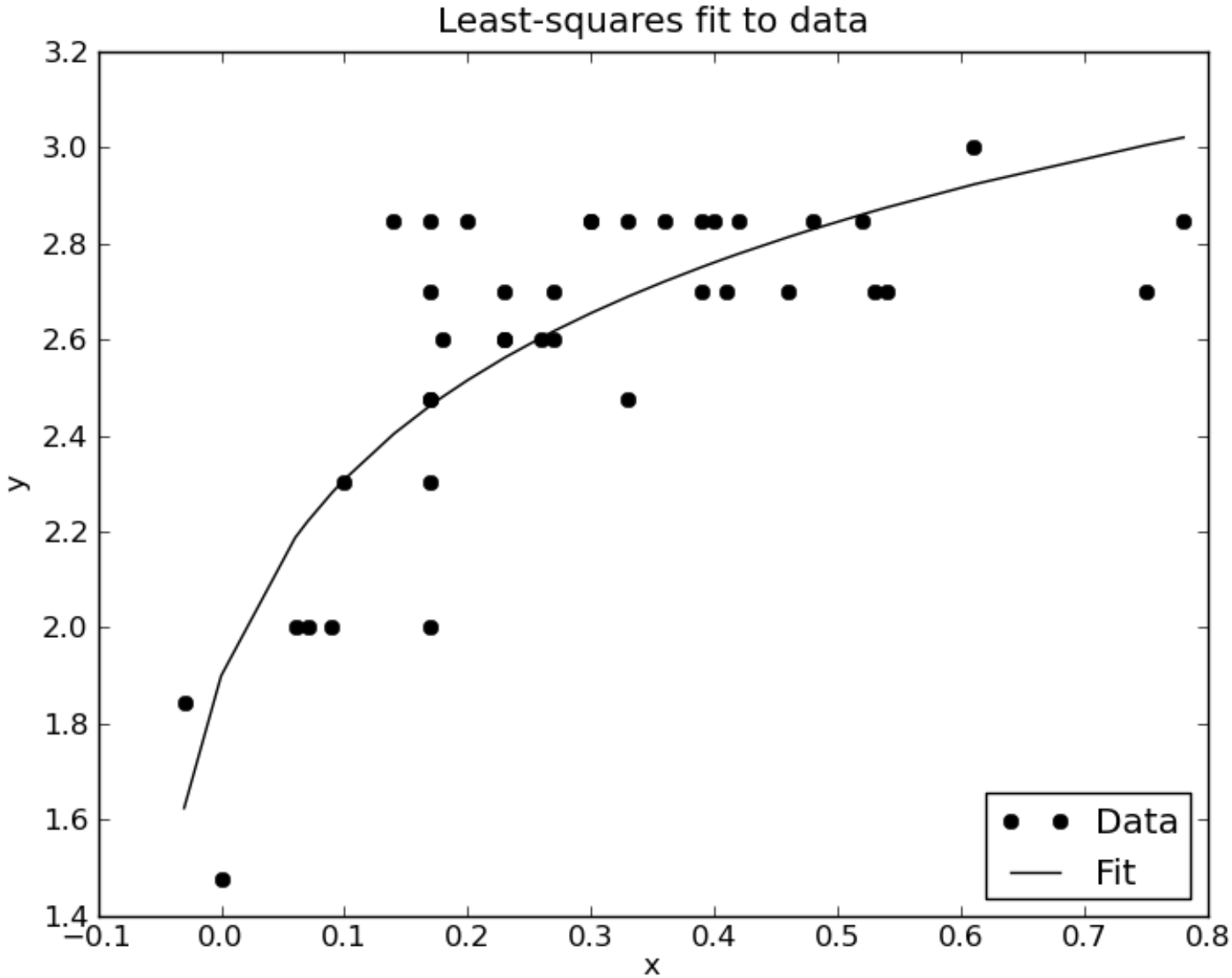
- The *least squares problem* is

$$\text{minimize} \quad \|Ax - b\|^2$$

- Solution of least squares problem is

$$\hat{x} = (A^T A)^{-1} A^T b = A^\dagger b$$

Example


Least-squares fit to data
3.2
3.0
2.8
2.6
2.4
2.2
2.0
1.8
1.6
1.4
y
−0.1
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
x
Data
Fit

Constrained least squares problem

- The (linearly) *constrained least squares problem* is

$$\begin{array}{ll} \text{minimize} & \|Ax - b\|^2 \\ \text{subject to} & Cx = d \end{array}$$

- Solution of the constrained least squares problem is

$$\begin{bmatrix} \hat{x} \\ z \end{bmatrix} = \begin{bmatrix} 2A^TA & C^T \\ C & 0 \end{bmatrix}^{-1} \begin{bmatrix} 2A^Tb \\ d \end{bmatrix}$$

Least squares modeling

```
import Data.Map.Strict (Map)
import Numeric.LinearAlgebra
import LeastSquares

type VarName = String
data LinExpr = Var VarName Int
             | ...
infix 4 :==:
data LinContr = LinExpr :==: LinExpr
data QuadExpr = SumSquares LinExpr
              | ProdQuad Double QuadExpr
              | SumQuad QuadExpr QuadExpr

minimize :: QuadExpr -> [LinContr]
                 -> Map VarName (Vector R)
```

Least squares modeling

```
import Data.Map.Strict ((!))
import Numeric.LinearAlgebra hiding ((!))
import LeastSquares

leastSquaresEstimate :: Vector R
leastSquaresEstimate = result ! "x"
  where result = minimize (SumSquares (a*x-b)) [c*x:==:d]
        a = Mat $ (3><2) [1.0, 1.0, 1.0, 4.0, 2.0, -1.0]
        b = Vec $ vector [13.0, 27.0, 1.0]
        c = Mat $ (1><2) [3.0, 7.0]
        d = Vec $ vector [-1.0]
        x = Var "x" 2
```

Modeling language

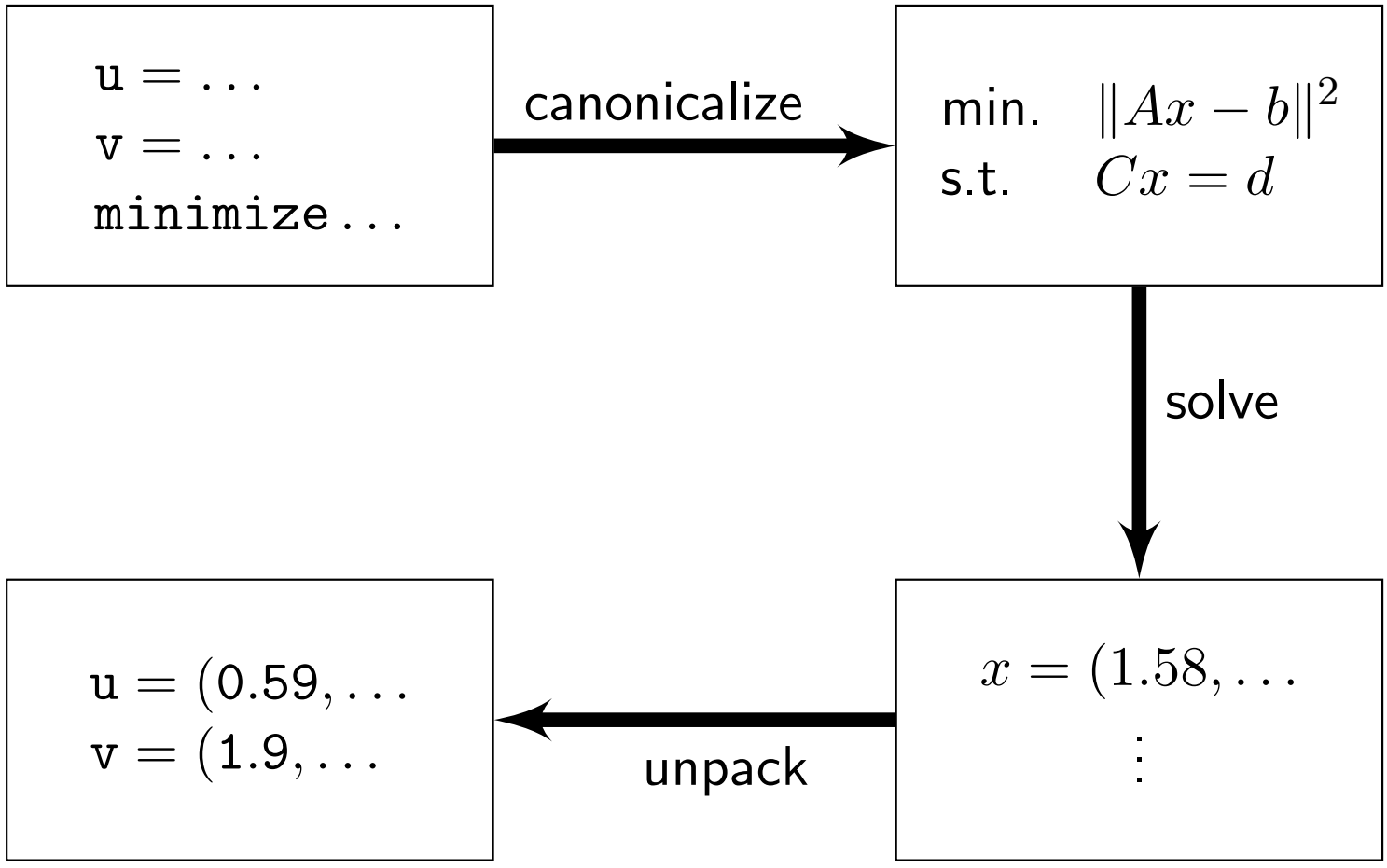


Image de-blurring

- x is an image, A is a blurring operator, and $y = Ax + v$ is a blurred, noisy image
- least-squares de-blurring: choose x to minimize

$$\|Ax - y\|^2 + \lambda(\|D_{\mathrm{v}}x\|^2 + \|D_{\mathrm{h}}x\|^2)$$

  D_{v}, D_{h} are vertical and horizontal differencing operations
- λ controls smoothing of de-blurred image

```
minimize (SumSquares (a*x-b)
          + lambda * (SumSquares (dv*x)
                      + SumSquares (dh*x))) []
```

Example

- left: blurred, noisy image
- right: regularized inversion with $\lambda = 0.007$