Multi-Objective Least-Squares

Stephen Boyd

EE103
Stanford University

November 9, 2017

Outline

Multi-objective least-squares

- goal: choose n-vector x so that k norm squared objectives

$$J_1 = \|A_1 x - b_1\|^2, \;\; \ldots, \;\; J_k = \|A_k x - b_k\|^2$$

  are all small
- A_i is an $m_i \times n$ matrix, b_i is an m_i-vector, $i = 1, \ldots, k$
- J_i are the objectives in a *multi-objective optimization problem* (also called a *multi-criterion problem*)
- could choose x to minimize any one J_i, but we want *one* x that makes them all small

Weighted sum objective

- choose positive *weights* $\lambda_1, \dots, \lambda_k$ and form *weighted sum objective*

$$J = \lambda_1 J_1 + \cdots + \lambda_k J_k = \lambda_1 \|A_1 x - b_1\|^2 + \cdots + \lambda_k \|A_k x - b_k\|^2$$

- we'll choose x to minimize J
- we can take $\lambda_1 = 1$, and call J_1 the *primary objective*
- interpretation of λ_i: how much we care about J_i being small, relative to primary objective
- for a bi-criterion problem, we will minimize

$$J_1 + \lambda J_2 = \|A_1 x - b_1\|^2 + \lambda \|A_2 x - b_2\|^2$$

Weighted sum minimization via stacking

- write weighted-sum objective as

$$J = \left\| \begin{bmatrix} \sqrt{\lambda_1}(A_1 x - b_1) \\ \vdots \\ \sqrt{\lambda_k}(A_k x - b_k) \end{bmatrix} \right\|^2$$

- so we have $J = \|\tilde{A}x - \tilde{b}\|^2$, with

$$\tilde{A} = \begin{bmatrix} \sqrt{\lambda_1} A_1 \\ \vdots \\ \sqrt{\lambda_k} A_k \end{bmatrix}, \qquad \tilde{b} = \begin{bmatrix} \sqrt{\lambda_1} b_1 \\ \vdots \\ \sqrt{\lambda_k} b_k \end{bmatrix}$$

- so we can minimize J using basic ('single-criterion') least-squares

Weighted sum solution

- assuming columns of $\tilde{A}$ are independent,

$$\begin{aligned}\hat{x} &= (\tilde{A}^T\tilde{A})^{-1}\tilde{A}^T\tilde{b} \\ &= (\lambda_1 A_1^T A_1 + \cdots + \lambda_k A_k^T A_k)^{-1}(\lambda_1 A_1^T b_1 + \cdots + \lambda_k A_k^T b_k)\end{aligned}$$

- can compute $\hat{x}$ via QR factorization of $\tilde{A}$
- A_i can be wide, or have dependent columns

Optimal trade-off curve

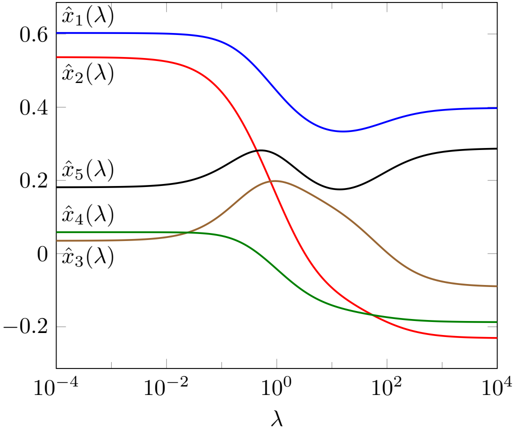
- bi-criterion problem with objectives J_1, J_2
- let $\hat{x}(\lambda)$ be minimizer of $J_1 + \lambda J_2$
- called *Pareto optimal*: there is no point z that satisfies

$$J_1(z) < J_1(\hat{x}(\lambda)), \quad J_2(z) < J_2(\hat{x}(\lambda))$$

  i.e., no other point x beats $\hat{x}$ on both objectives
- *optimal trade-off curve*: $(J_1(\hat{x}(\lambda)), J_2(\hat{x}(\lambda)))$ for $\lambda > 0$

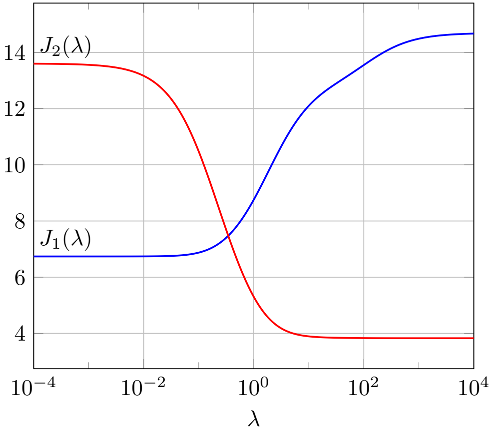
Example

- A_1 and A_2 both 10×5

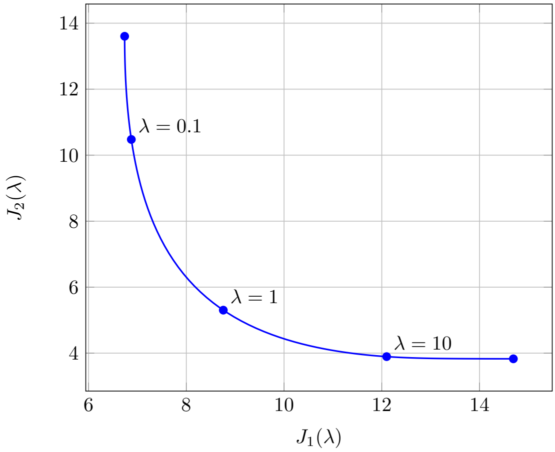


Objectives versus λ

- J_1 (solid); J_2 (dashed)



Optimal trade-off curve



Using multi-objective least-squares

- identify the primary objective
  - the basic quantity we want to minimize
- choose one or more secondary objectives
  - quantities we'd also like to be small, if possible
  - *e.g.*, size of x, roughness of x, distance from some given point
- tweak/tune the weights until we like (or can tolerate) $\hat{x}(\lambda)$
- for bi-criterion problem with $J = J_1 + \lambda J_2$:
  - if J_2 is too big, increase λ
  - if J_1 is too big, decrease λ

Outline

Control

- n-vector x corresponds to *actions* or *inputs*
- m-vector y corresponds to *results* or *outputs*
- inputs and outputs are related by affine input-output model

$$y = Ax + b$$

- A and b are known (from analytical models, data fitting ...)
- the goal is to choose x (which determines y), to optimize multiple objectives on x and y

Multi-objective control

- typical primary objective: $J_1 = \|y - y^{\mathrm{des}}\|^2$, where y^{des} is a given desired or target output
- typical secondary objectives:
  - x is small: $J_2 = \|x\|^2$
  - x is not far from a nominal input: $J_2 = \|x - x^{\mathrm{nom}}\|^2$

Product demand shaping

- we will change prices of n products by n-vector δ^{price}
- this induces change in demand $\delta^{\mathrm{dem}} = E^{\mathrm{d}}\delta^{\mathrm{price}}$
- E^{d} is the $n \times n$ price elasticity of demand matrix
- we want $J_1 = \|\delta^{\mathrm{dem}} - \delta^{\mathrm{tar}}\|^2$ small
- and also, we want $J_2 = \|\delta^{\mathrm{price}}\|^2$ small
- so we minimize $J_1 + \lambda J_2$, and adjust $\lambda > 0$
- trades off deviation from target demand and price change magnitude

Robust control

- we have K different input-output models (a.k.a. *scenarios*)

$$y^{(k)} = A^{(k)}x + b^{(k)}, \quad k = 1, \ldots, K$$

- these represent uncertainty in the system
- $y^{(k)}$ is the output with input x, if system model k is correct
- average cost across the models:

$$\frac{1}{K}\sum_{k=1}^{K} \|y^{(k)} - y^{\mathrm{des}}\|^2$$

- can add terms for x as well, *e.g.*, $\lambda\|x\|^2$
- yields choice of x that does well under all scenarios

Outline

Estimation

- measurement model: $y = Ax + v$
- n-vector x contains parameters we want to estimate
- m-vector y contains the measurements
- m-vector v are (unknown) *noises* or *measurement errors*
- $m \times n$ matrix A connects parameters to measurements
- *basic least-squares estimation*: assuming v is small (and A has independent columns), we guess x by minimizing $J_1 = \|Ax - y\|^2$

Regularized inversion

- can get far better results by incorporating prior information about x into estimation, *e.g.*,
  - x should be not too large
  - x should be smooth
- express these as secondary objectives:
  - $J_2 = \|x\|^2$ ('Tikhonov regularization')
  - $J_2 = \|Dx\|^2$
- we minimize $J_1 + \lambda J_2$
- adjust λ until you like the results
- curve of $\hat{x}(\lambda)$ versus λ is called *regularization path*
- with Tikhonov regularization, works even when A has dependent columns (*e.g.*, when it is wide)

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

Example

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

Regularization path

- $\lambda = 10^{-6}$, $\lambda = 10^{-4}$

Regularization path

- $\lambda = 10^{-2}$, $\lambda = 10^{0}$

Outline

Motivation for regularization

- consider data-fitting model (of relationship $y \approx f(x)$)

$$\hat{f}(x) = \theta_1 f_1(x) + \cdots + \theta_p f_p(x)$$

  with $f_1(x) = 1$
- θ_i is the sensitivity of $\hat{f}(x)$ to $f_i(x)$
- so large θ_i means the model is very sensitive to $f_i(x)$
- θ_1 is an exception, since $f_1(x) = 1$ never varies
- so, we don't want $\theta_2, \ldots, \theta_p$ to be too large

Regularized data-fitting

- suppose we have data $(x_1, y_1), \ldots, (x_N, y_N)$
- express fitting error as $A\theta - y$
- *regularized data-fitting*: choose θ to minimize

$$\|A\theta - y\|^2 + \lambda\|\theta_{2:p}\|^2$$

- $\lambda > 0$ is the *regularization parameter*
- for regression model $\hat{y} = X^T\beta + v\mathbf{1}$, we minimize

$$\|X^T\beta + v\mathbf{1} - y\|^2 + \lambda\|\beta\|^2$$

- choose λ by validation on a test set

Regularized least squares classification

- MNIST digit data set, $n = 785$, $N = 50000$ (train), $N = 10000$ (test)
- we minimize $\|X^T\beta + v\mathbf{1} - y\|^2 + \lambda\|\beta\|^2$
- classifier is $\hat{y} = \mathbf{sign}(x^T\beta + v)$
- build one classifier for each digit, same λ
- vary λ and find train and test classification error
- for $\lambda \approx 10^3$, test error drops a bit to 13.5% (from 14%)

Regularized least squares classification