Signal Denoising and Regression Models

Goran Banjac

Large-Scale Convex Optimization
ETH Zurich

May 12, 2020

Signal denoising

- in *signal denoising* or *reconstruction*, we have a noisy signal y
- assume that measurement comes from process with slow changes
- approximate with signal x that captures process behavior
- therefore, we want neighboring time-steps to be close to each other
- we have two competing objectives

smooth

  - $x \approx y$
  - x should vary slowly

Signal denoising

- introduce difference operator D

$$D = \begin{bmatrix} 1 & -1 & & \\ & \ddots & \ddots & \\ & & 1 & -1 \end{bmatrix} \implies Dx = \begin{bmatrix} x_1 - x_2 \\ \vdots \\ x_{n-1} - x_n \end{bmatrix}$$

- we want Dx small and $x \approx y$
- we can model this as an optimization problem

$$\text{minimize} \quad \tfrac{1}{2}\|x - y\|_2^2 + \lambda \|Dx\|_2^2$$

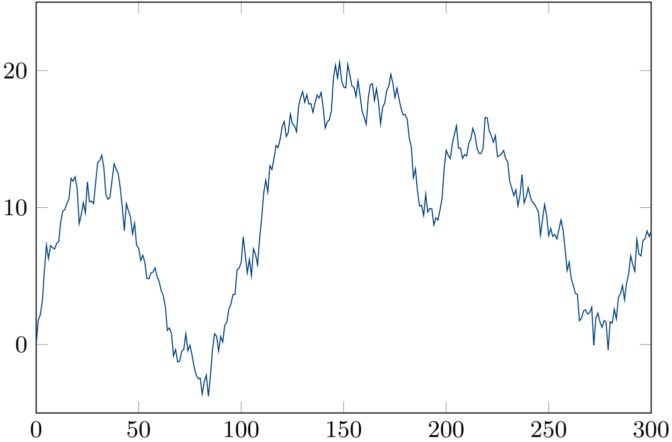
it is convex, sometimes it is still convex when lambda is smaller than zero(look at Hessian)... For L1 norm, no closed form.

where y contains measurements and $\lambda > 0$ trades off objectives

- **example:** $y \in \mathbb{R}^{300}$ constructed by random walk in $\mathbb{R}$

Signal denoising

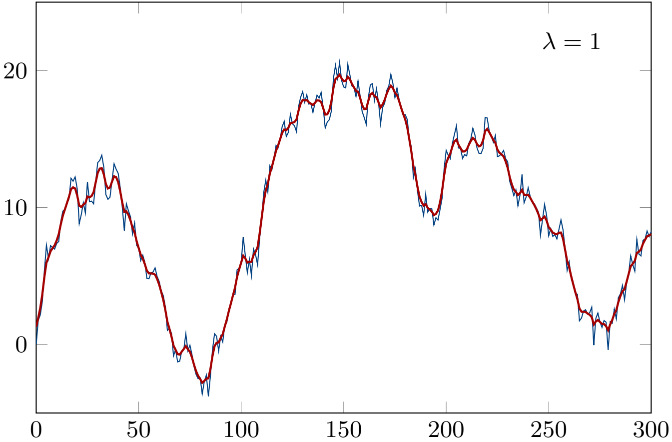
$$\text{minimize} \quad \frac{1}{2}\|x - y\|_2^2 + \lambda\|Dx\|_2^2$$



Signal denoising

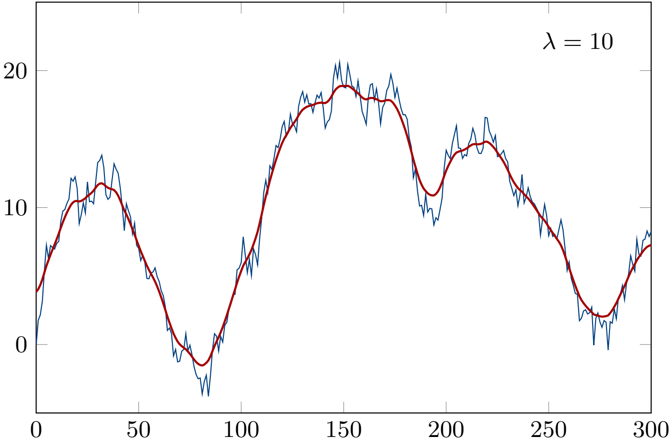
low-pass filter

minimize $\quad \frac{1}{2}\|x - y\|_2^2 + \lambda\|Dx\|_2^2$



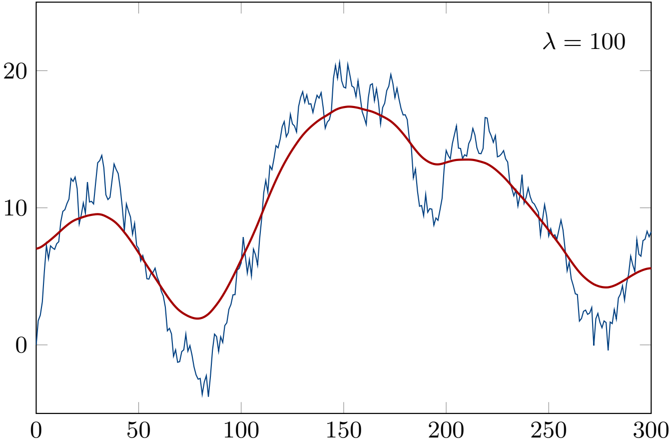
Signal denoising

$$\text{minimize} \quad \frac{1}{2}\|x - y\|_2^2 + \lambda \|Dx\|_2^2$$



Signal denoising

$$\text{minimize} \quad \frac{1}{2}\|x - y\|_2^2 + \lambda\|Dx\|_2^2$$



Piece-wise constant approximation

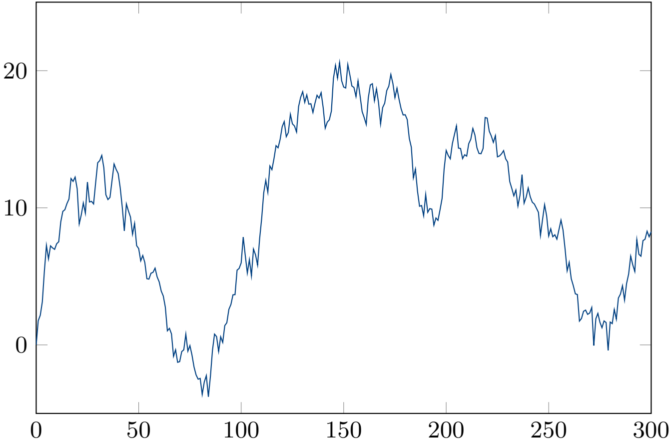
if we have some prior knowledge that signal comes from piece-wise source.

- what if we want instead a piece-wise constant approximation?
- then we want Dx to be sparse
- minimizing cardinality of Dx is a nonconvex problem
- $\|Dx\|_1$ is usually a good convex approximation

actually for sparsity, we want 0-norm (which is the cardinality, but it's difficult to solve. so we use one-norm)

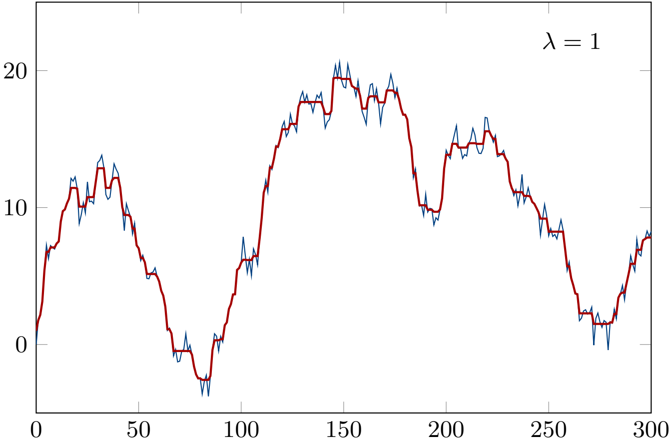
Piece-wise constant approximation

$$\text{minimize} \quad \frac{1}{2}\|x-y\|_2^2 + \lambda\|Dx\|_1$$



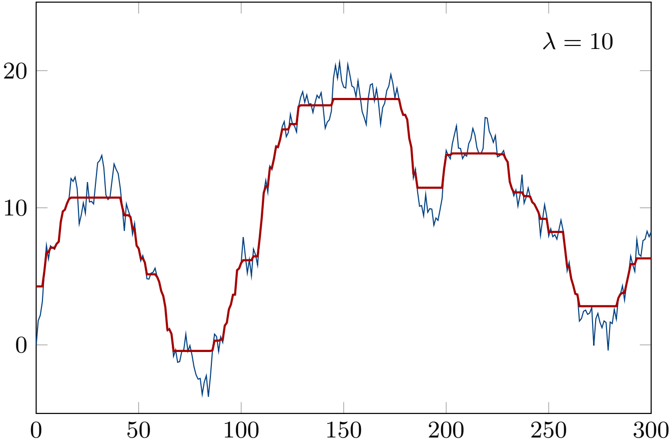
Piece-wise constant approximation

$$\text{minimize} \quad \tfrac{1}{2}\|x-y\|_2^2 + \lambda\|Dx\|_1$$



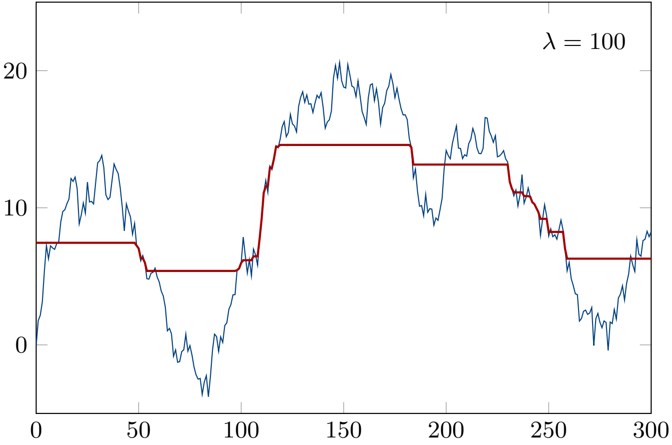
Piece-wise constant approximation

$$\text{minimize} \quad \frac{1}{2}\|x-y\|_2^2 + \lambda\|Dx\|_1$$



Piece-wise constant approximation

$$\text{minimize} \quad \frac{1}{2}\|x - y\|_2^2 + \lambda \|Dx\|_1$$



Piece-wise affine approximation

- maybe we want a piece-wise affine approximation instead
- introduce the second-order discrete difference

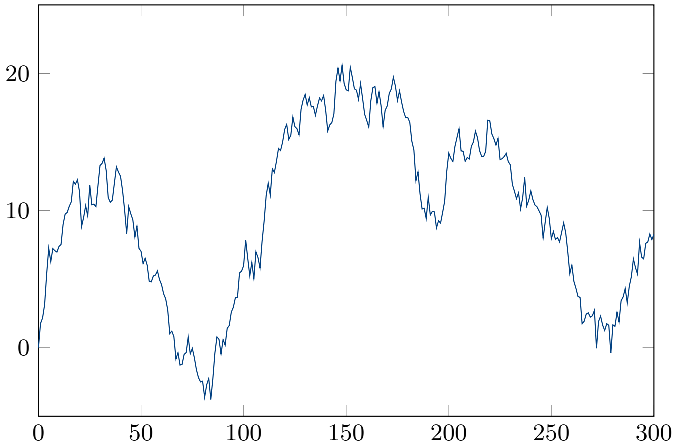
$$D_2 = \begin{bmatrix} 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \end{bmatrix}$$

- $D_2 x$ is zero on any line
- we can model piece-wise affine approximation as

$$\text{minimize} \quad \tfrac{1}{2}\|x - y\|_2^2 + \lambda \|D_2 x\|_1$$

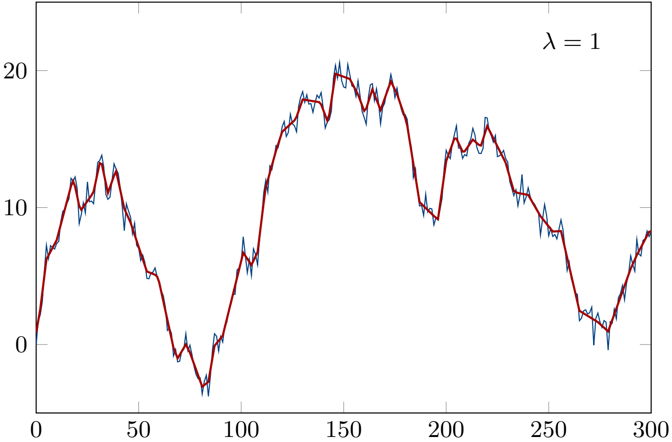
Piece-wise affine approximation

$$\text{minimize} \quad \frac{1}{2}\|x-y\|_2^2 + \lambda\|D_2 x\|_1$$



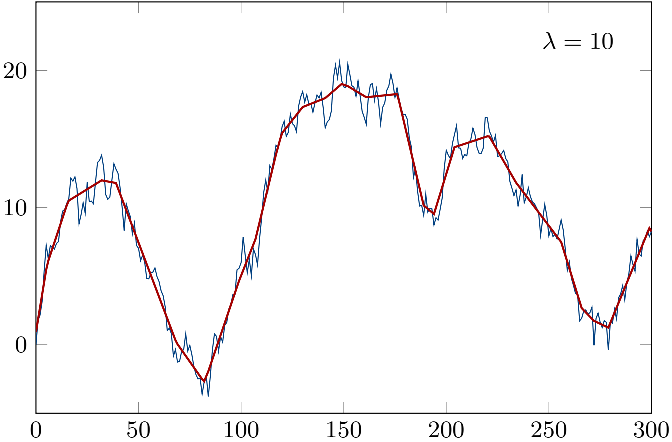
Piece-wise affine approximation

$$\text{minimize} \quad \frac{1}{2}\|x-y\|_2^2 + \lambda\|D_2 x\|_1$$



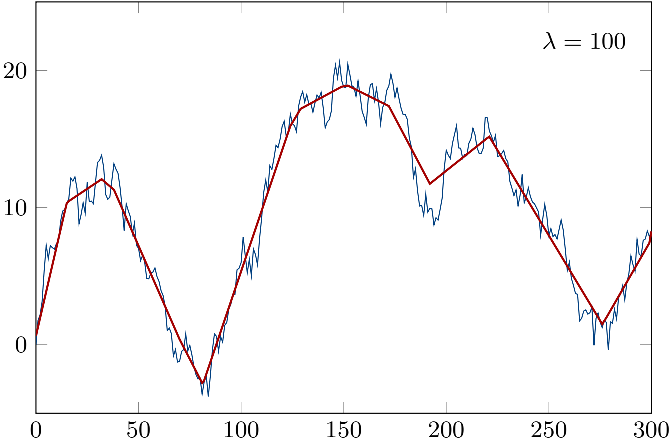
Piece-wise affine approximation

minimize $\quad \frac{1}{2}\|x-y\|_2^2 + \lambda\|D_2 x\|_1$



Piece-wise affine approximation

$$\text{minimize} \quad \frac{1}{2}\|x-y\|_2^2 + \lambda\|D_2 x\|_1$$



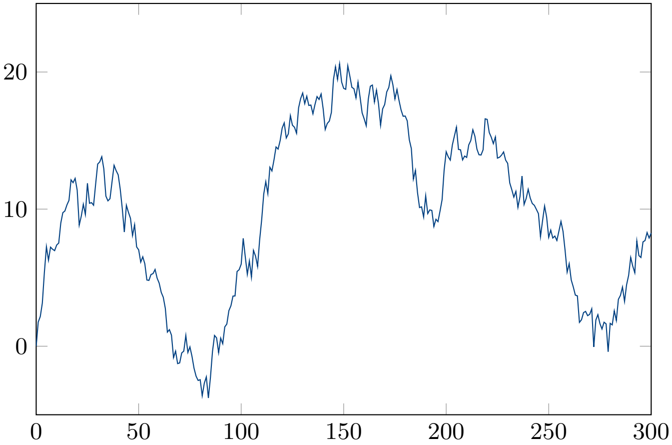
Smooth second derivative

- we might want a smooth second derivative
- we can model this as

$$\text{minimize} \quad \frac{1}{2}\|x - y\|_2^2 + \lambda \|D_2 x\|_2^2$$

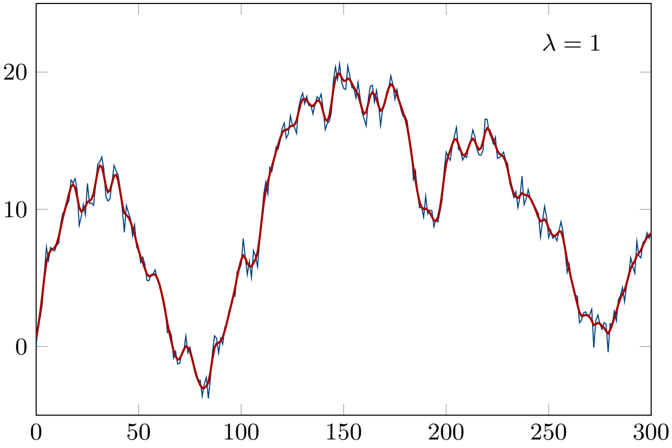
Smooth second derivative

$$\text{minimize} \quad \frac{1}{2}\|x-y\|_2^2 + \lambda\|D_2 x\|_2^2$$



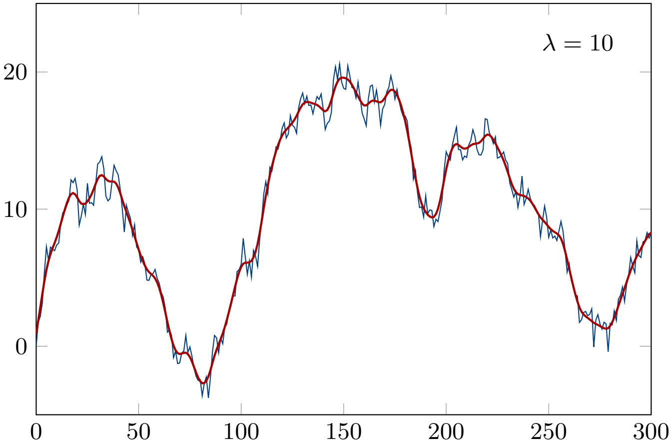
Smooth second derivative

$$\text{minimize} \quad \frac{1}{2}\|x - y\|_2^2 + \lambda\|D_2 x\|_2^2$$



Smooth second derivative

$$\text{minimize} \quad \frac{1}{2}\|x - y\|_2^2 + \lambda \|D_2 x\|_2^2$$



Smooth second derivative

$$\text{minimize} \quad \frac{1}{2}\|x - y\|_2^2 + \lambda\|D_2 x\|_2^2$$

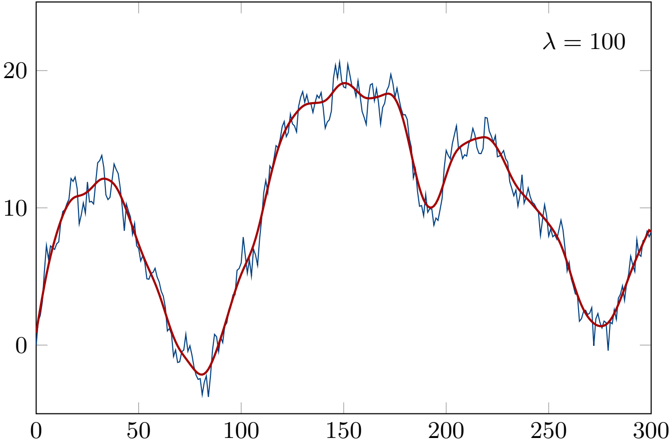
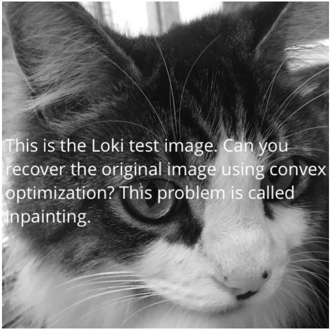


Image reconstruction

- we can also reconstruct images (2D signals)
- example: 512×512 grayscale image ($n \approx 300$k variables)



- reconstruct by minimizing the total variation of X:

$$\mathrm{tv}(X) = \sum_{i=1}^{m-1} \sum_{j=1}^{n-1} \left\| \begin{bmatrix} X_{i+1,j} - X_{i,j} \\ X_{i,j+1} - X_{i,j} \end{bmatrix} \right\|_2$$

- known pixels are set to correct value

Image reconstruction

- example: text over image

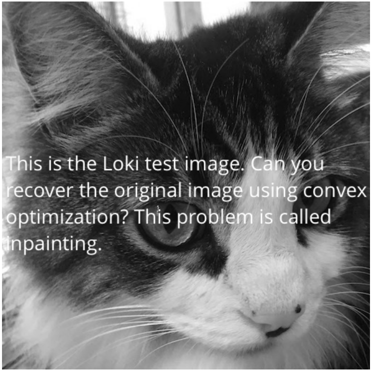


Image reconstruction

- example: text over image

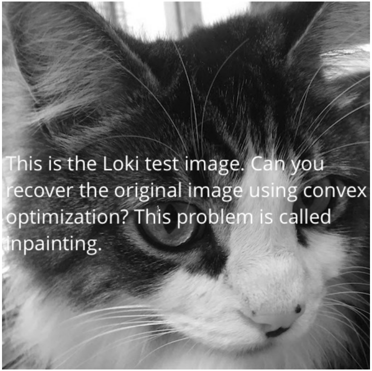


Image reconstruction

- example: 90% of pixels in image lost

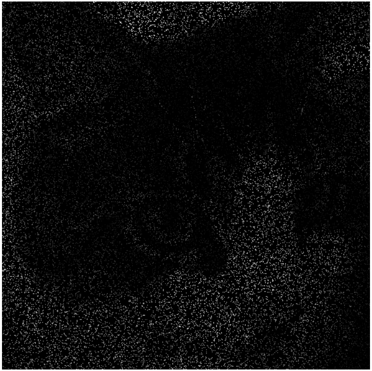


Image reconstruction

- example: 90% of pixels in image lost

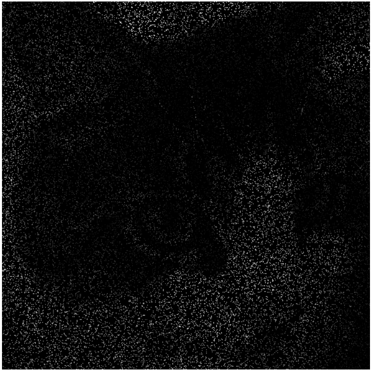


Image reconstruction

- example: 70% of pixels in image lost

Image reconstruction

- example: 70% of pixels in image lost

Image reconstruction

- example: 50% of pixels in image lost

Image reconstruction

- example: 50% of pixels in image lost

Image reconstruction

- example: 30% of pixels in image lost

Image reconstruction

- example: 30% of pixels in image lost

Image reconstruction

- comparison to ground truth

Modeling idea

- if you want something to be small, use $\|\cdot\|_2^2$
- if you want something to be sparse, use $\|\cdot\|_1$
- if you want to really enforce something, use constraints

Learning from data

- we have data from which we want to draw conclusions
- the data are represented as points x_i in a Euclidean space
- we let $X = [x_1, \ldots, x_n]$ be the data matrix
- every row in X is called an *example*
- every column in X is called a *feature*
- in supervised learning we also have *response variables* y_i, which can be real-valued (regression) or integer-valued (classification)
- **objective:** create model of unknown function $x \mapsto y(x)$
  (x data vector and y response variable)

think classification as a span filter

Linear model

- we start with a linear model for the mapping $x \mapsto y(x)$
- we have
  - data $X = [x_1, \ldots, x_n]$
  - real-valued responses $y = (y_1, \ldots, y_m)$ ($y_i \in \mathbb{R}$)
- create estimator $\hat{y}$ with

$$\hat{y}(x) = b + s^T x$$

- objective: minimize prediction error on data

$$\text{minimize} \quad \sum_{i=1}^{m} (\hat{y}(x_i) - y_i)^2$$

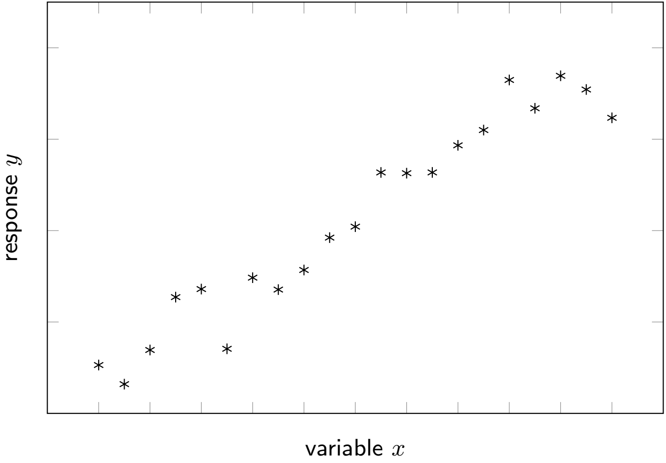
- letting $\theta = (s, b)$, the problem becomes

$$\text{minimize} \quad \sum_{i=1}^{m} \left(s^T x_i + b - y_i\right)^2 = \|\Phi\theta - y\|_2^2$$

- least squares problem

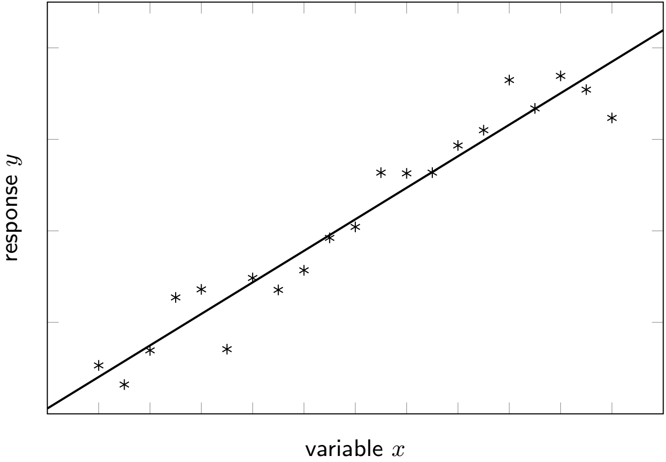
Least squares

- find affine function parameters that fit data
- data points (x, y) marked with $*$



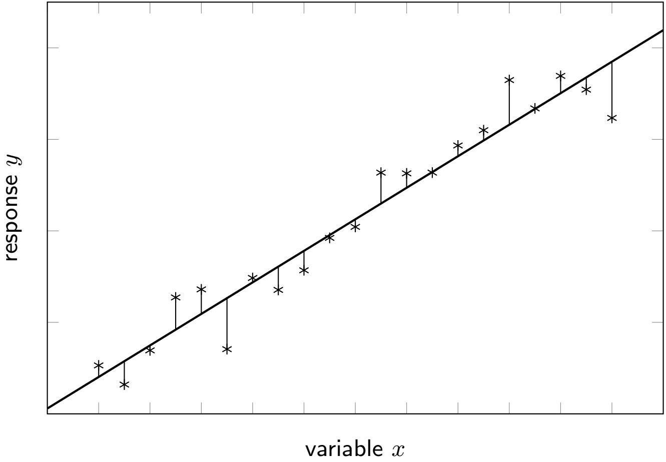
Least squares

- find affine function parameters that fit data
- data points (x, y) marked with $*$



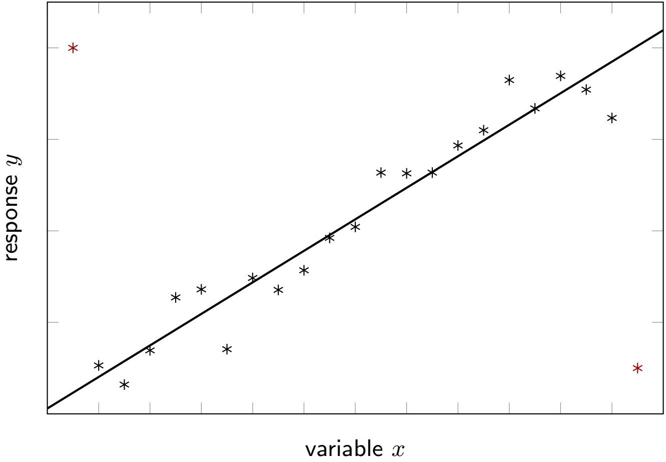
Least squares

- find affine function parameters that fit data
- data points (x, y) marked with $*$



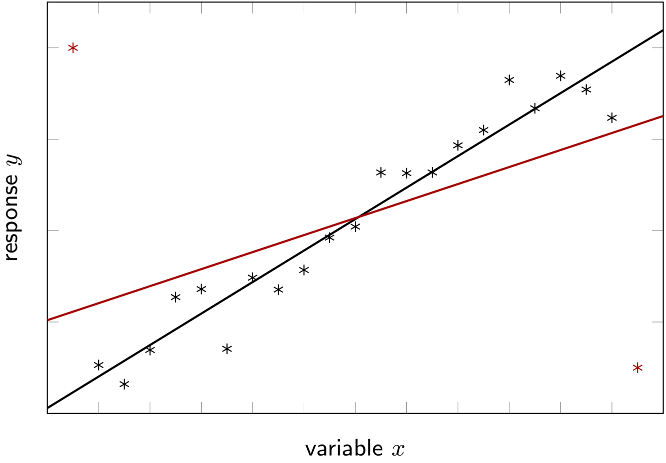
Least squares

- find affine function parameters that fit data
- data points (x, y) marked with $*$, outliers with $*$



Least squares

- find affine function parameters that fit data
- data points (x, y) marked with $*$, outliers with $*$



Huber fitting

- least squares is sensitive to outliers in problem data
- *Huber fitting* or *robust least squares* uses the Huber penalty function $\phi_{\mathsf{hub}} : \mathbb{R} \to \mathbb{R}$ defined as

$$\phi_{\mathsf{hub}}(u) = \begin{cases} u^2 & |u| \leq M \\ 2Mu - M^2 & |u| > M \end{cases}$$

- the fitting problem can be written as

$$\boxed{\text{minimize} \quad \sum_{i=1}^{m} \phi_{\mathsf{hub}}\left(\hat{y}(x_i) - y_i\right)}$$

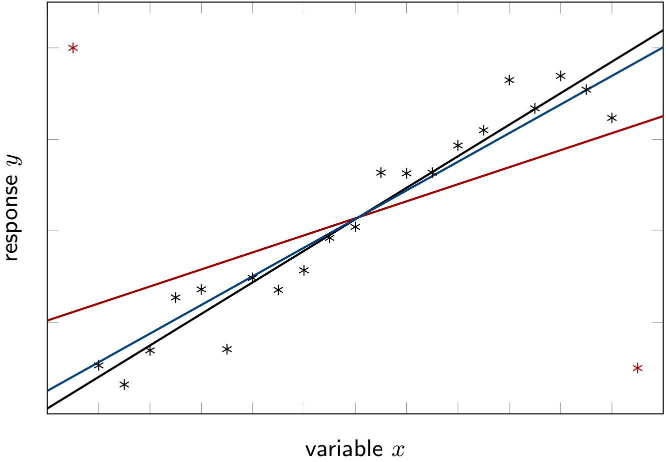
- the problem can be reformulated as a quadratic program

thus can be solved efficiently.

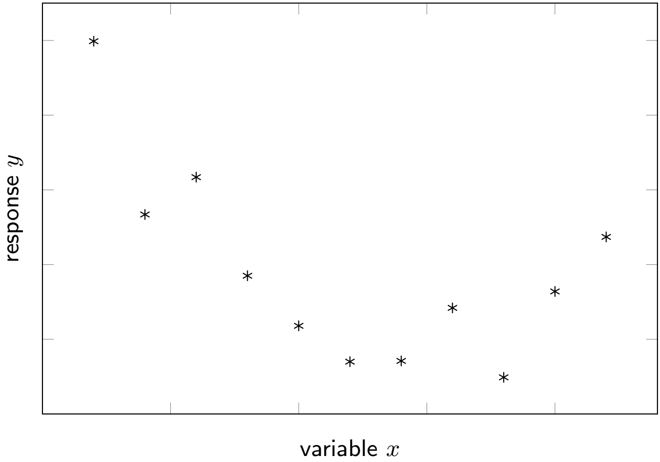
Huber fitting

more robust to outliers: blue line

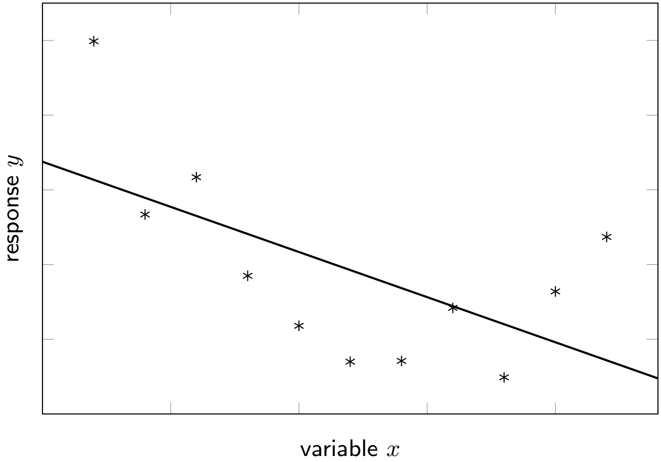
- find affine function parameters that fit data
- data points (x, y) marked with $*$, outliers with $*$



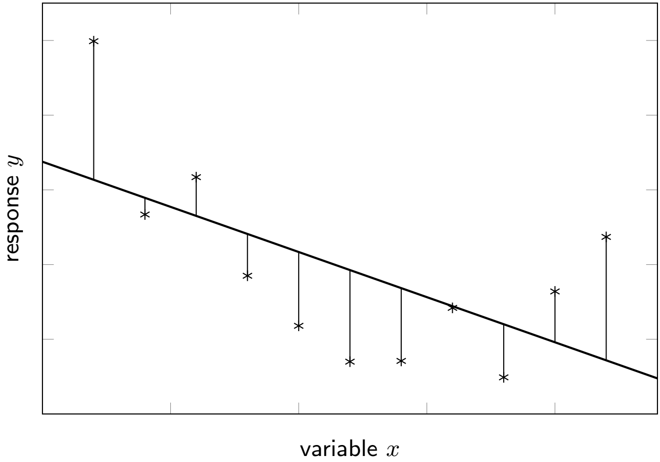
Nonaffine example



Nonaffine example



Nonaffine example



Polynomial models

- a linear model may not be accurate enough to model the mapping
- try, *e.g.*, a quadratic model

$$\hat{y}(x) = b + s^T x + \sum_{i=1}^{n} \sum_{j=1}^{i} w_{ij} x_i x_j$$

- for $x \in \mathbb{R}$, this becomes

$$\hat{y}(x) = b + sx + wx^2 = \phi(x)^T \theta$$

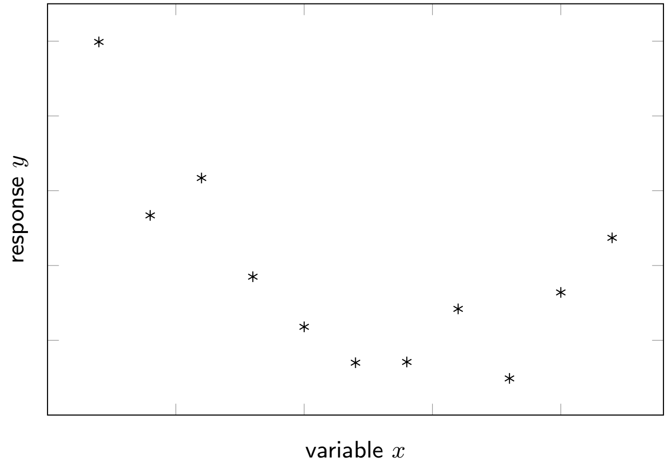
  where $\phi(x) = (1, x, x^2)$ and $\theta = (b, s, w)$
- the LS problem can be written as

$$\text{minimize} \quad \|\Phi\theta - y\|_2^2$$

- lift problem to higher dimensional LS problem
- obviously, higher order models can be used as well

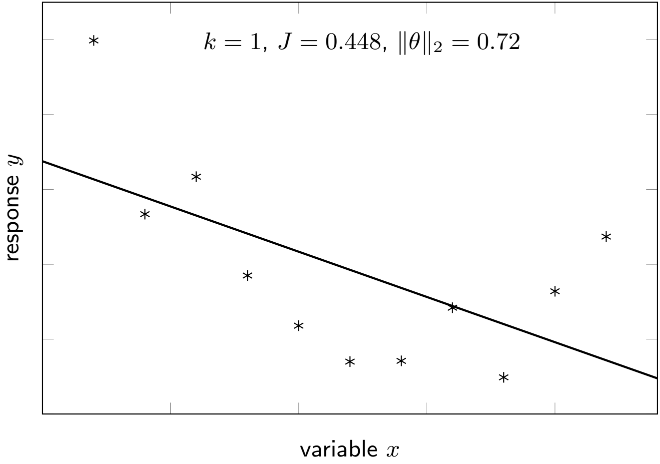
Polynomial models

- k: polynomial order; J: LS cost; $\|\theta\|_2$: norm of weights



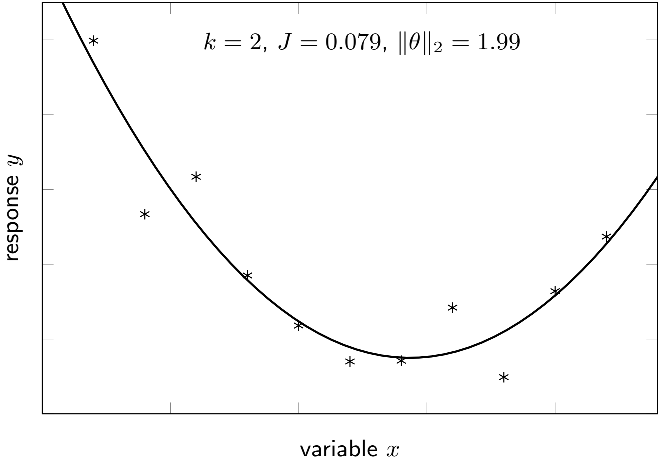
Polynomial models

- k: polynomial order; J: LS cost; $\|\theta\|_2$: norm of weights



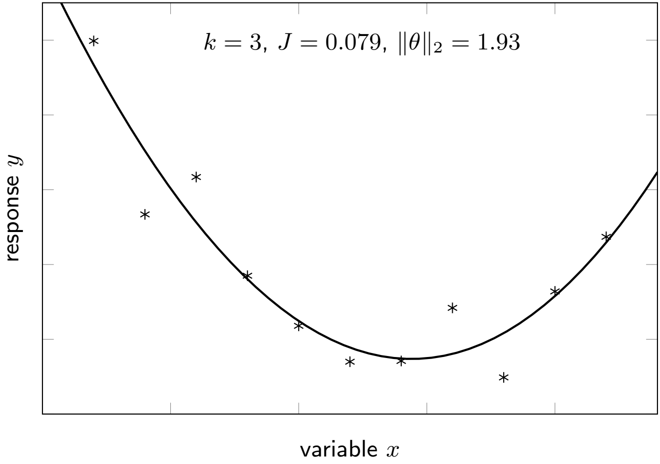
Polynomial models

- k: polynomial order; J: LS cost; $\|\theta\|_2$: norm of weights



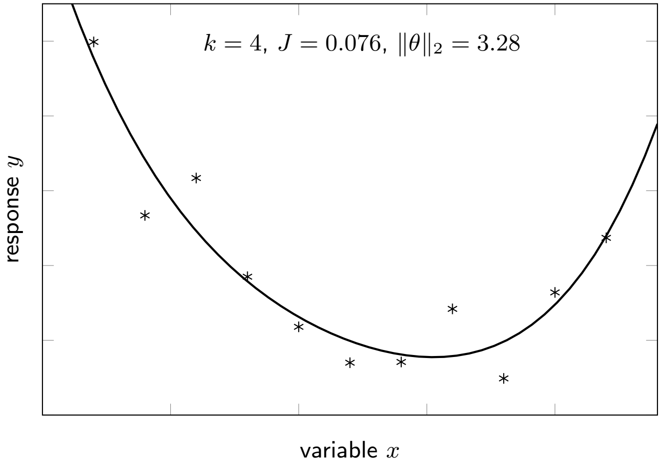
Polynomial models

- k: polynomial order; J: LS cost; $\|\theta\|_2$: norm of weights



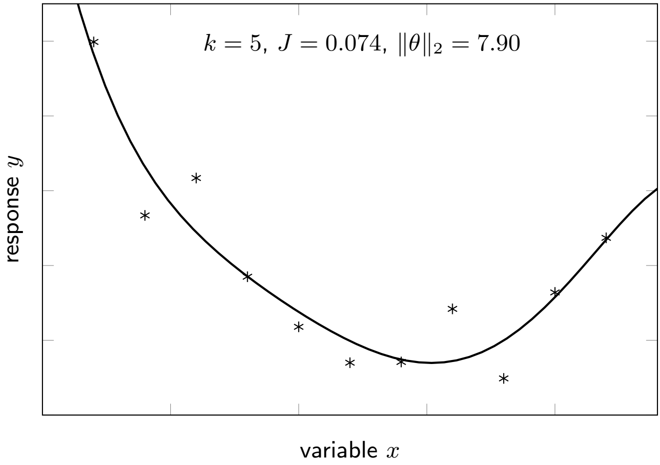
Polynomial models

- k: polynomial order; J: LS cost; $\|\theta\|_2$: norm of weights



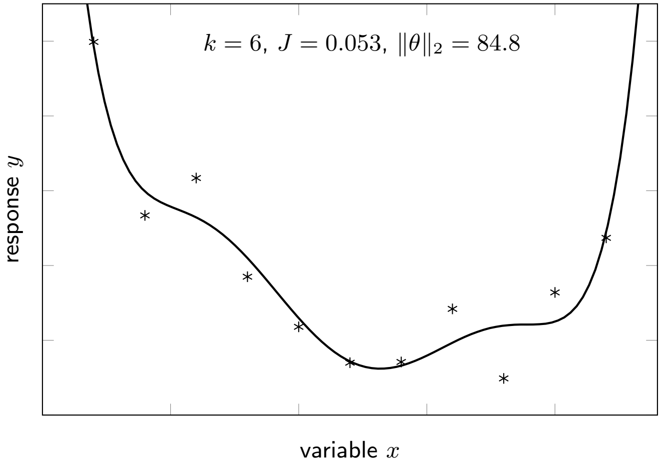
Polynomial models

- k: polynomial order; J: LS cost; $\|\theta\|_2$: norm of weights



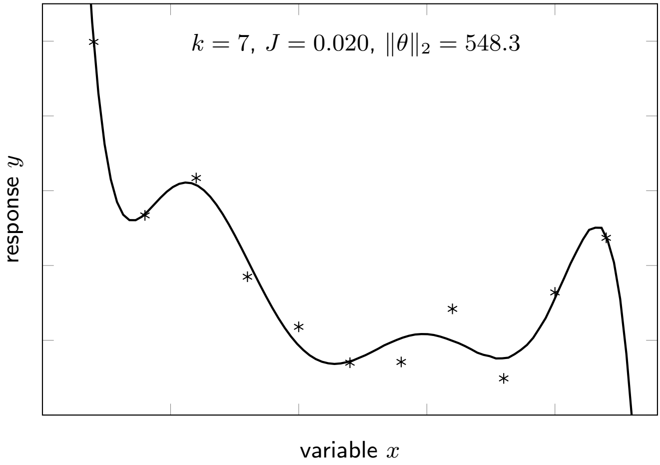
Polynomial models

- k: polynomial order; J: LS cost; $\|\theta\|_2$: norm of weights



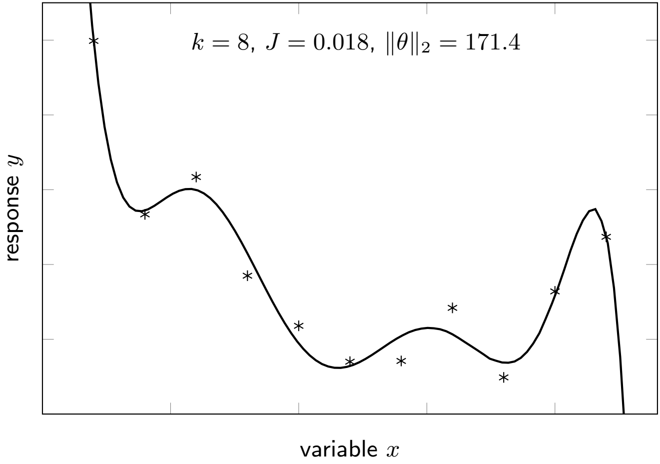
Polynomial models

- k: polynomial order; J: LS cost; $\|\theta\|_2$: norm of weights



Polynomial models

- k: polynomial order; J: LS cost; $\|\theta\|_2$: norm of weights



Generalization and overfitting

- *generalization*: how well does model perform on unseen data
- *overfitting*: model explains training data, but not unseen data
- which of the previous models would generalize best?
- how to reduce overfitting / improve generalization?

Regularization

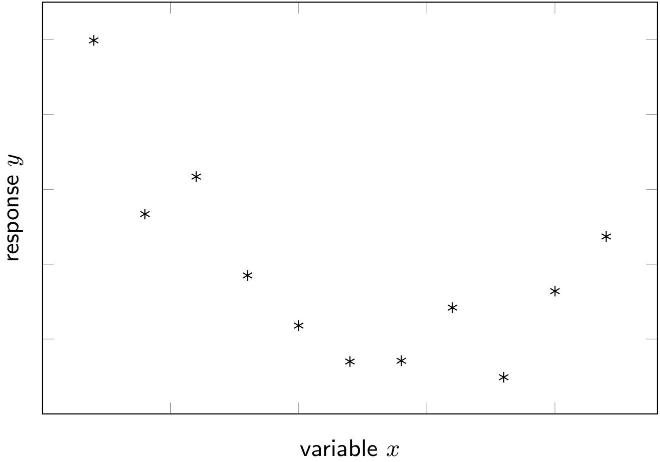
- reducing $\|\theta\|_2$ seems to reduce overfitting
- least squares with Tikhonov regularization

$$\text{minimize} \quad \|\Phi\theta - y\|_2^2 + \lambda\|\theta\|_2^2$$

- regularization parameter $\lambda \geq 0$ controls fit vs model expressivity
- optimization problem is also known as *ridge regression*

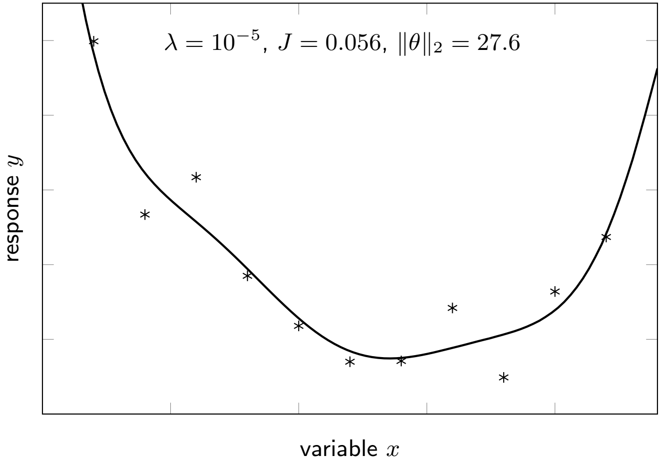
Ridge regression

- fit 8-degree polynomial with Tikhonov regularization
- λ: regularization parameter; J: LS cost; $\|\theta\|_2$: norm of weights



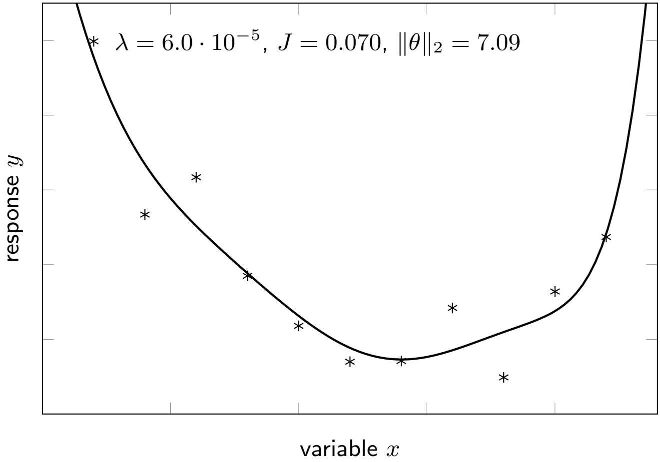
Ridge regression

- fit 8-degree polynomial with Tikhonov regularization
- λ: regularization parameter; J: LS cost; $\|\theta\|_2$: norm of weights



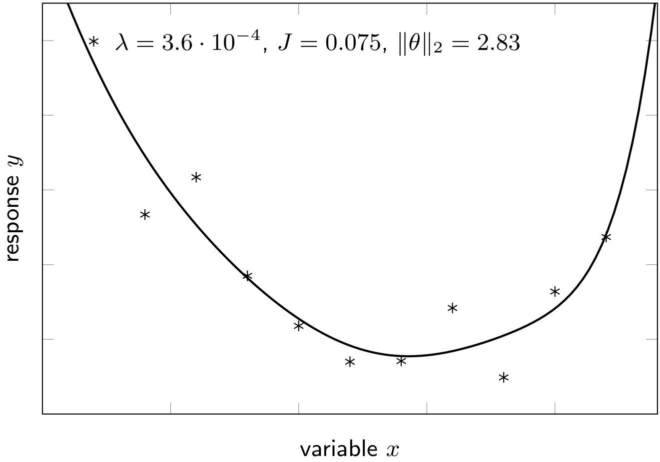
Ridge regression

- fit 8-degree polynomial with Tikhonov regularization
- λ: regularization parameter; J: LS cost; $\|\theta\|_2$: norm of weights



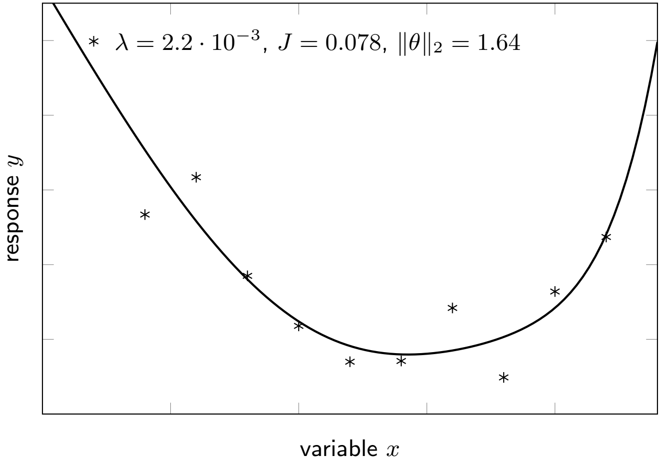
Ridge regression

- fit 8-degree polynomial with Tikhonov regularization
- λ: regularization parameter; J: LS cost; $\|\theta\|_2$: norm of weights



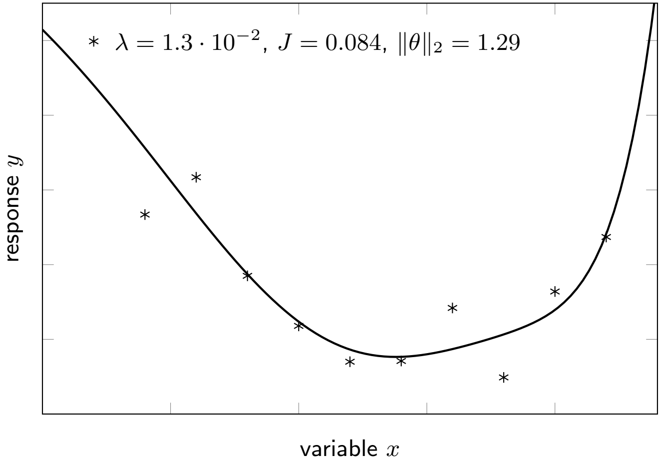
Ridge regression

- fit 8-degree polynomial with Tikhonov regularization
- λ: regularization parameter; J: LS cost; $\|\theta\|_2$: norm of weights



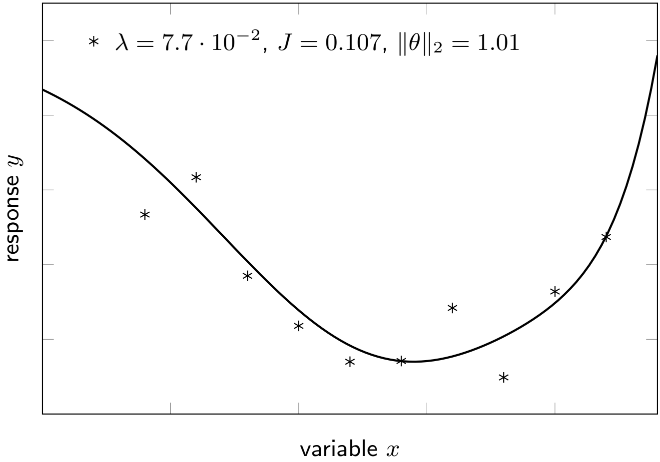
Ridge regression

- fit 8-degree polynomial with Tikhonov regularization
- λ: regularization parameter; J: LS cost; $\|\theta\|_2$: norm of weights



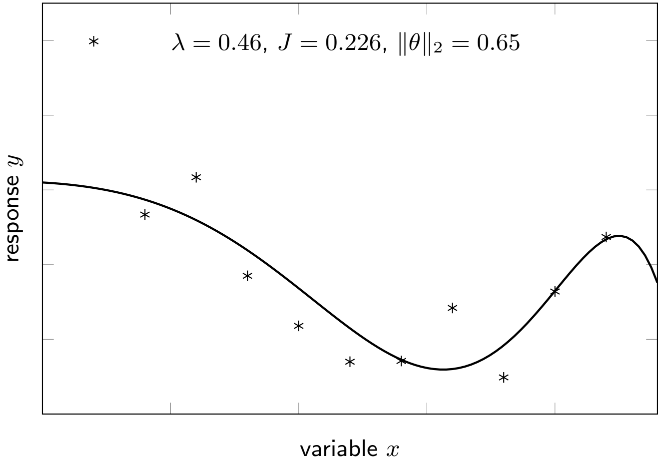
Ridge regression

- fit 8-degree polynomial with Tikhonov regularization
- λ: regularization parameter; J: LS cost; $\|\theta\|_2$: norm of weights



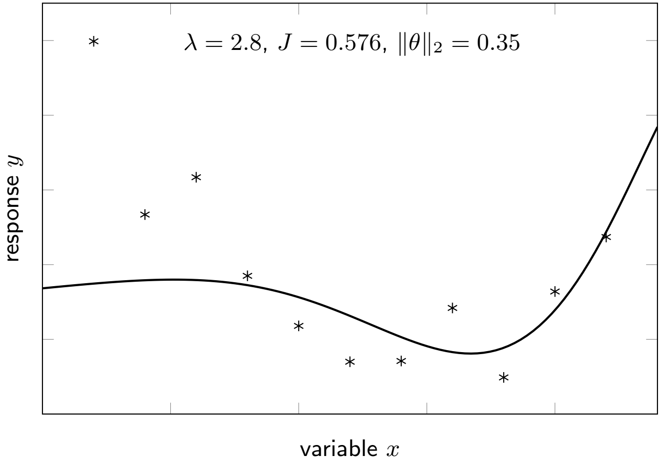
Ridge regression

- fit 8-degree polynomial with Tikhonov regularization
- λ: regularization parameter; J: LS cost; $\|\theta\|_2$: norm of weights



Ridge regression

- fit 8-degree polynomial with Tikhonov regularization
- λ: regularization parameter; J: LS cost; $\|\theta\|_2$: norm of weights



Selecting model parameters

- parameters in machine learning models are called *hyperparameters*
- ridge model has polynomial order and λ as hyperparameters
- how to select hyperparameters?

- **training set**
  - solve training problems with different hyperparameters
- **validation set**
  - estimate generalization performance of all trained models
  - use this to select model that seems to generalize best
- **test set**
  - final assessment on how chosen model generalizes to unseen data
  - not used for model selection

Feature selection

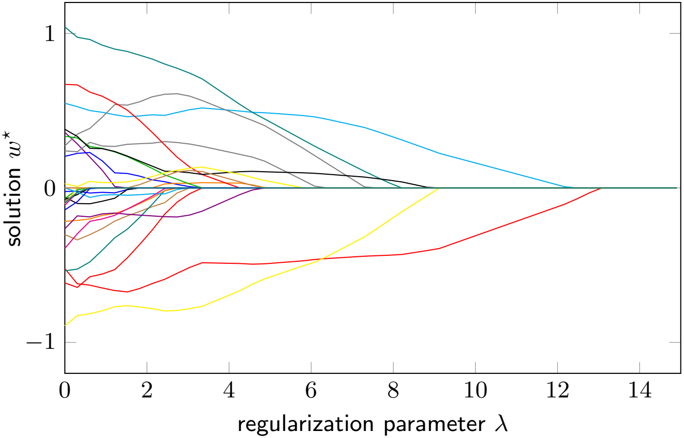
- assume that we have $X \in \mathbb{R}^{m \times n}$ with $m < n$ (or $m \ll n$)
- we would like to select a subset of features to explain data
- easier to interpret solutions
- we typically want subset to have cardinality (much) smaller than m
- since cardinality function is nonconvex, we use instead the ℓ_1 norm

$$\text{minimize} \quad \|Xw - y\|_2^2 + \lambda \|w\|_1$$

- optimization problem is also known as *lasso*
- typically gives sparse solutions
- λ decided by cross validation and desired sparsity

Lasso

- lasso problem with $X \in \mathbb{R}^{30 \times 200}$ for different λ



References

- these lecture notes are based to a large extent on the following courses developed by Pontus Giselsson at Lund:
  - Large-Scale Convex Optimization
  - Optimization for Learning

- the original slides can be downloaded from

`https://archive.control.lth.se/ls-convex-2015/`

`http://www.control.lth.se/education/engineering-program/frtn50-optimization-for-learning/`