Modeling

Pontus Giselsson

Today's lecture

- signal reconstruction
- supervised learning - regression
  - least squares
  - ridge regression
  - lasso (group lasso)
  - elastic net
- supervised learning - classification
  - classification
  - support-vector machines
  - sparse classification
- control (model predictive control)

Functions

- in most examples, we use only the following functions

$$\|\cdot\|, \quad \tfrac{1}{2}\|\cdot\|^2, \quad \iota_V, V \text{ affine subspace}, \quad h(x) = \sum_i h_i(x_i)$$

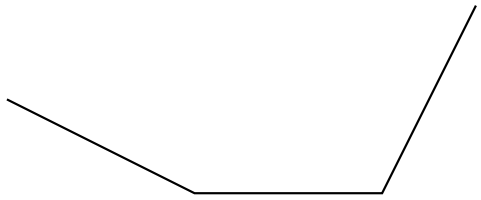
  where $h_i : \mathbb{R} \to \overline{\mathbb{R}}$ is

$$h_i(x) = \begin{cases} c_l(l-x) & \text{if } x \le l \\ 0 & \text{if } l \le x \le u \\ c_u(x-u) & \text{if } x \ge u \end{cases}$$

  where $c_l, c_u \in (0, \infty]$ (∞ included) and $l \le u$
- we compose these functions with affine operators $Lx - b$

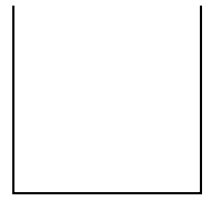
Graphical representations of h_i

- graphical representations of different h_i



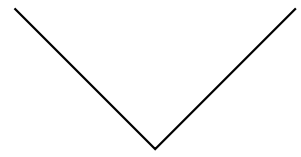
$c_l = \frac{1}{2},\ c_u = 2$

$l = -1,\ u = 1$



$c_l = c_u = \infty$

$l = -1,\ u = 1$



$c_l = c_u = 1$

$l = u = 0$

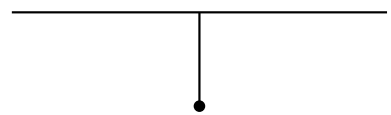
- special cases of h_i
  - upper and lower bounds
  - the 1-norm
  - skewed 1-norms
  - "soft" upper and lower bounds

0-norm

- in many applications we would ideally like to use the 0-norm
- the 0-norm $\|x\|_0$ counts the number of nonzero elements in x
- that is $\|x\|_0 = \sum_i h_i(x_i)$ where

$$h_i(x_i) = \begin{cases} 0 & \text{if } x_i = 0 \\ 1 & \text{else} \end{cases}$$

- graphical representation



- it is obviously nonconvex
- often the 1-norm is used as a convex proxy for this
- why? 1-norm is convex envelope of $\|x\|_0 + \iota_{\|x\| \leq 1}(x)$ for $\|x\| \leq 1$

Signal reconstruction

- in signal reconstruction, we have a noisy signal y
- assume that measurement from process with slow changes
- approximate with signal x that captures process behavior
- therefore: want neighboring time-steps to be close to each other
- we have two competing objectives, want $x \approx y$ and x vary slowly

Signal reconstruction

- introduce difference operator D

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

- then

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

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

Signal reconstruction

- introduce difference operator D

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

- then

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

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

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

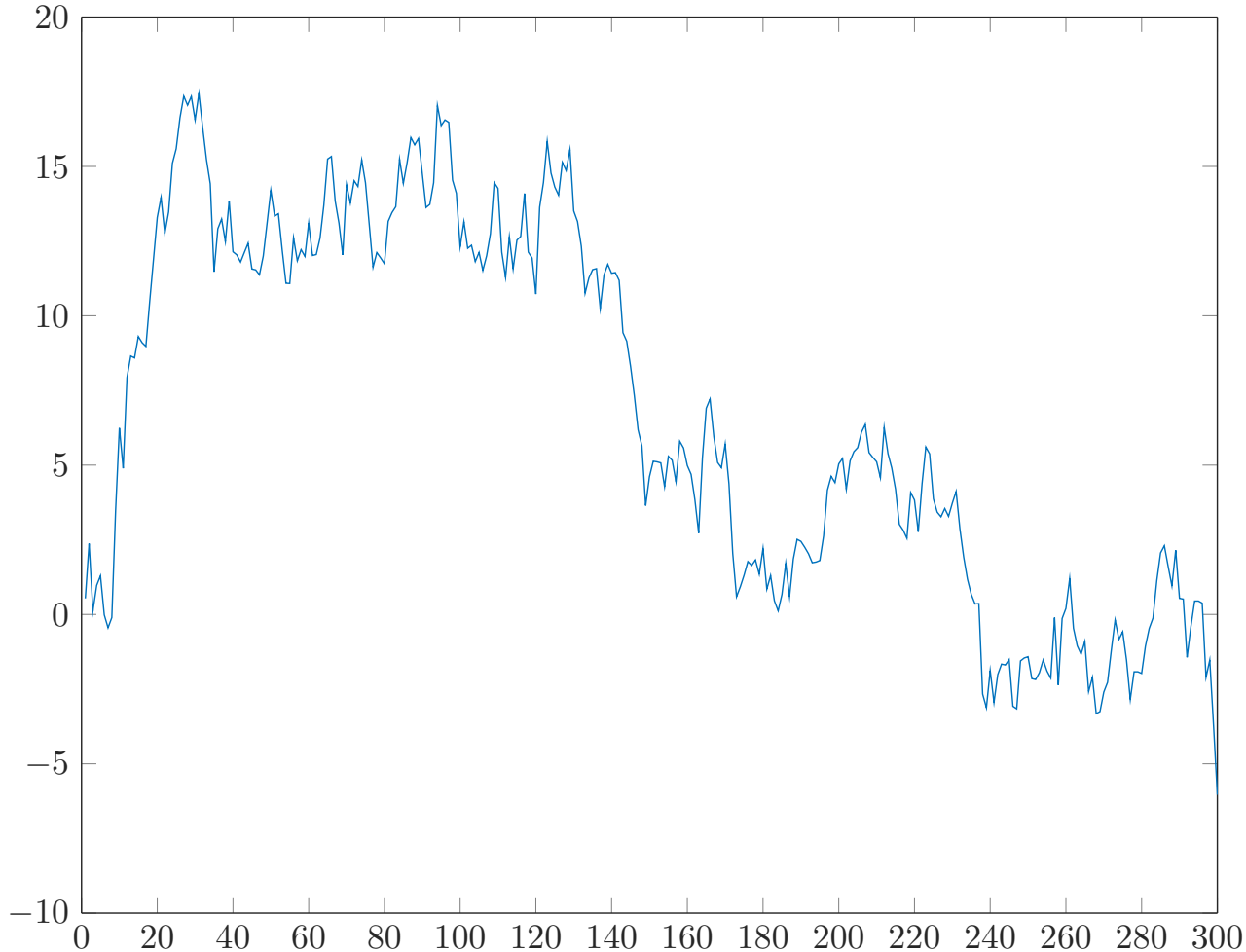
where y contains measurements and $\lambda > 0$ trades off objectives

Numerical example

- we have $y \in \mathbb{R}^{300}$
- y constructed by random walk in $\mathbb{R}$

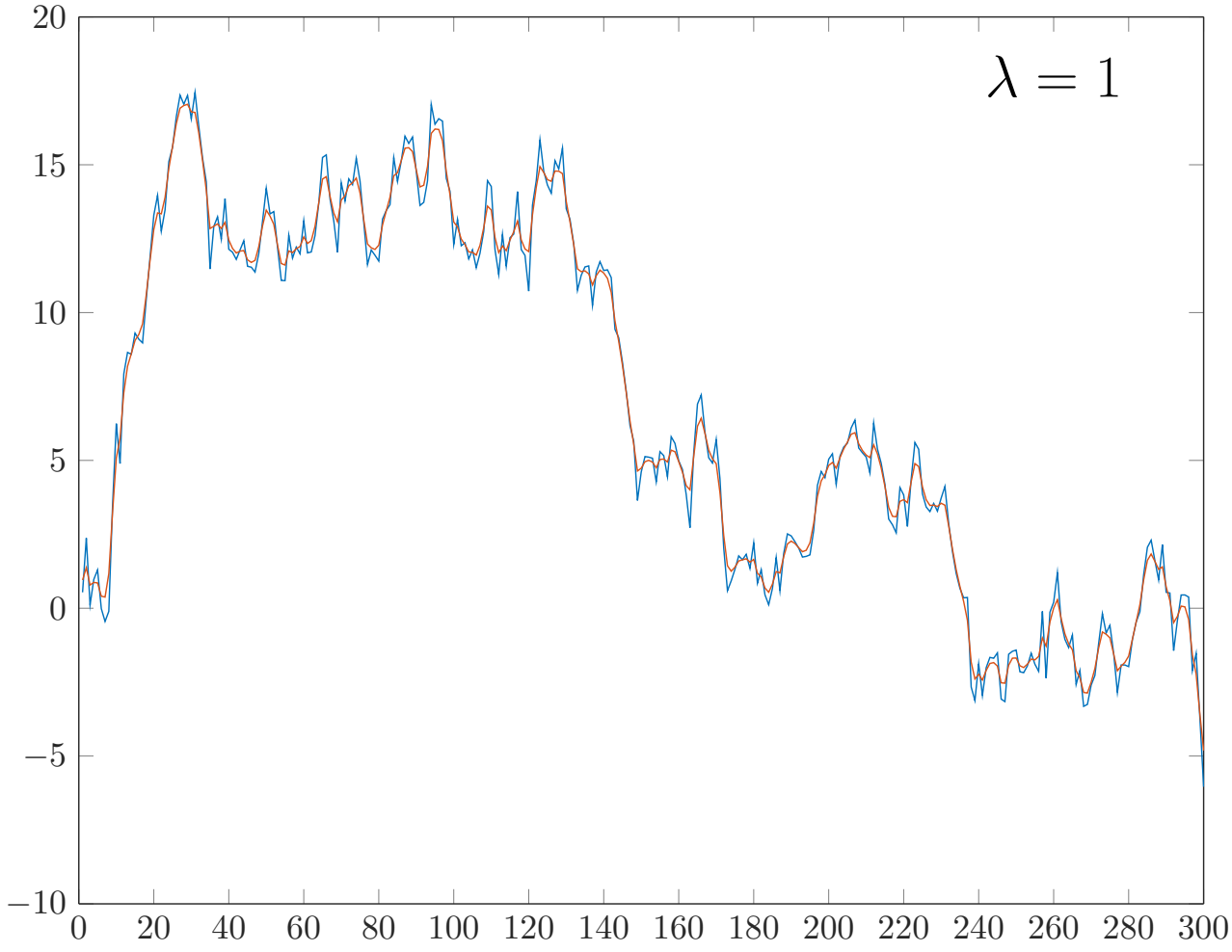
Result

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



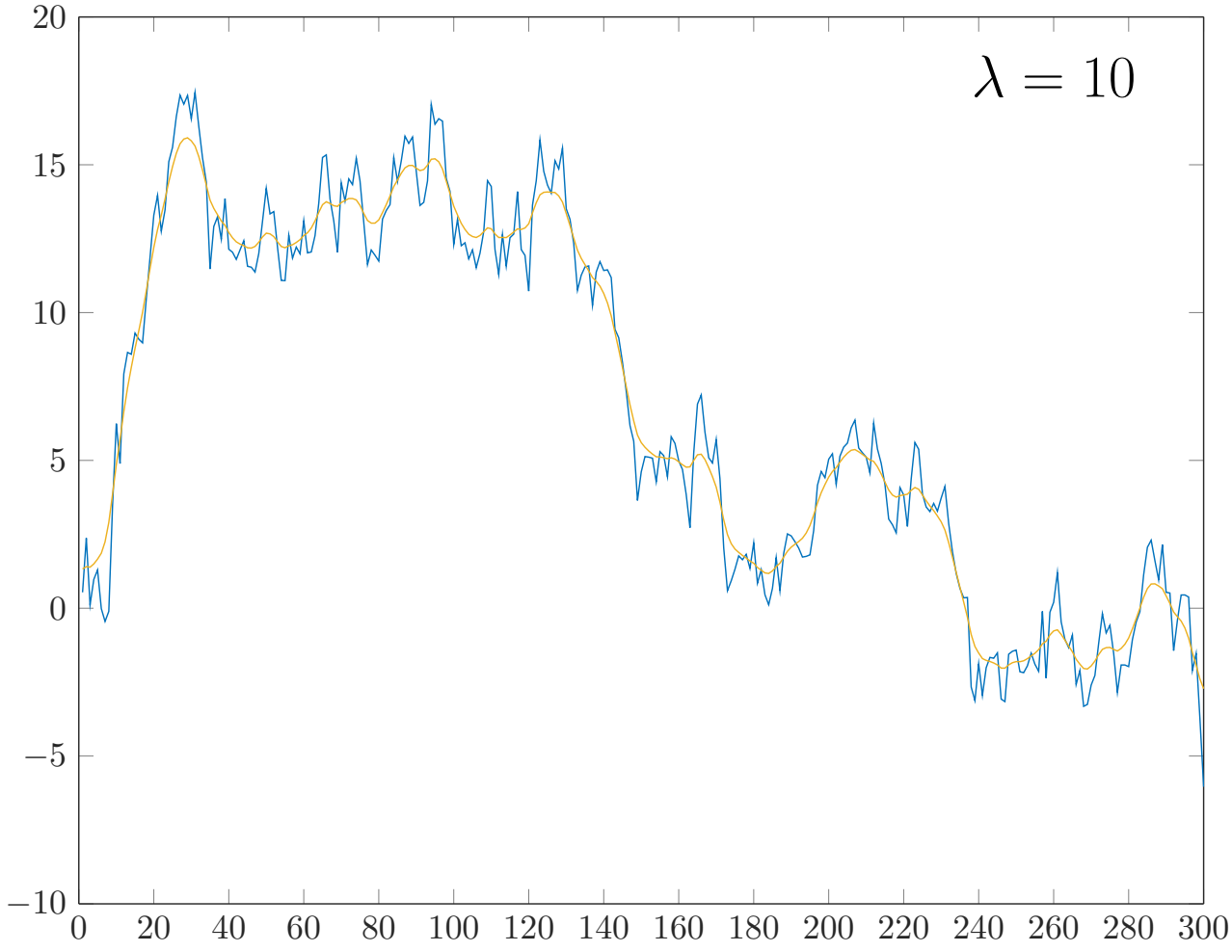
Result

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



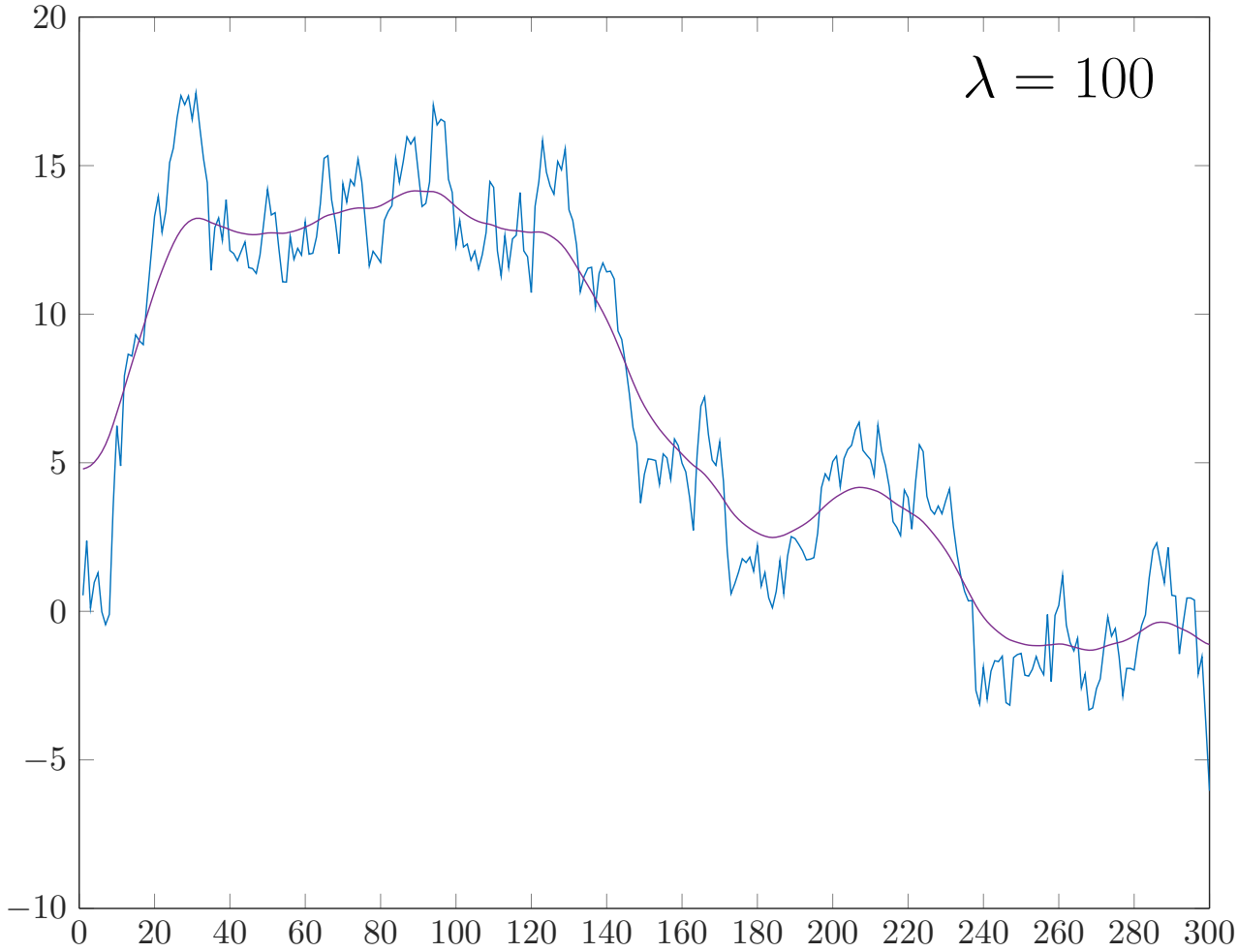
Result

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



Result

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



Example

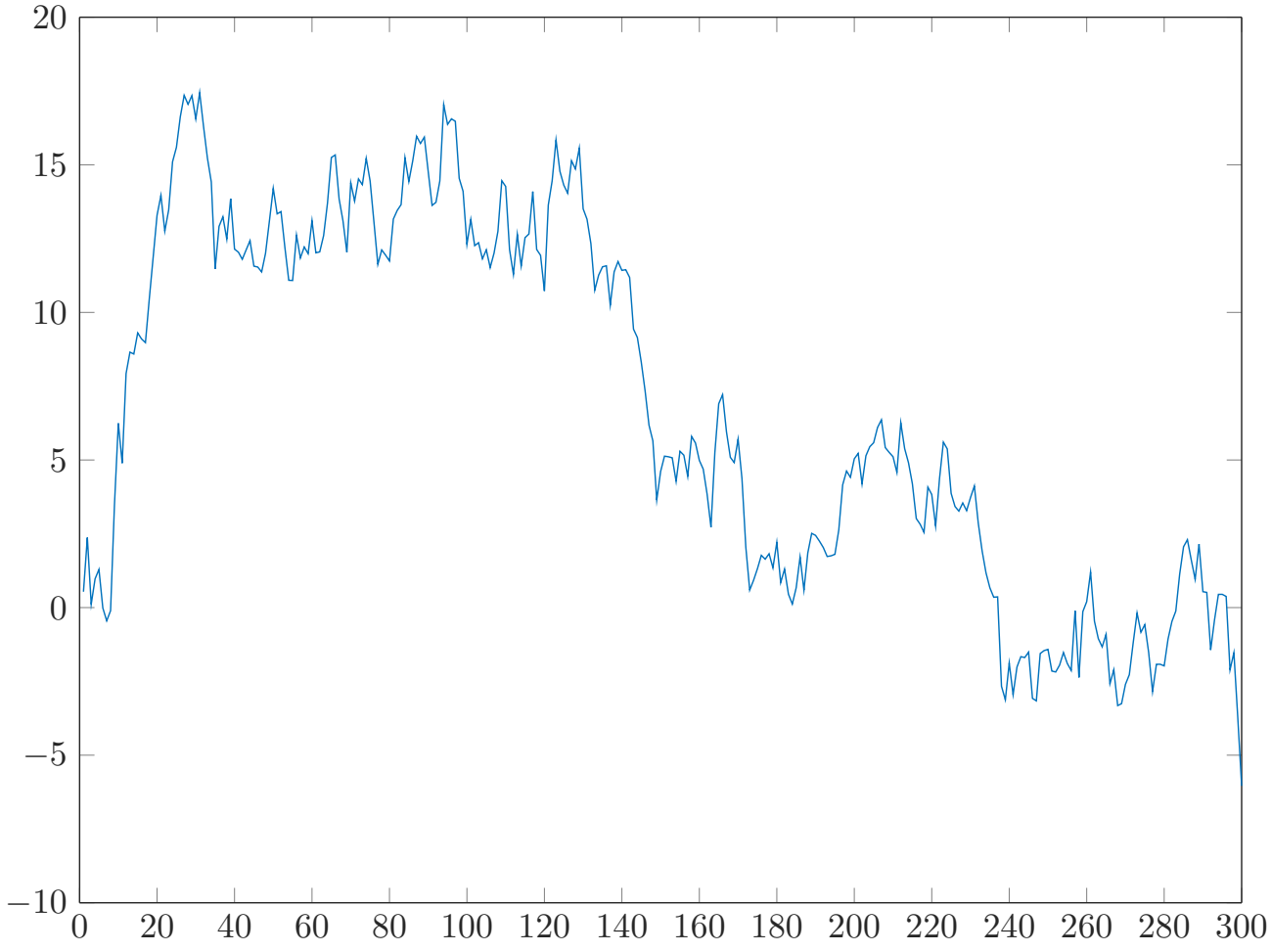
- what if we instead want piece-wise constant approximation?
- then we want Dx to be sparse
- how to model this?

Example

- what if we instead want piece-wise constant approximation?
- then we want Dx to be sparse
- how to model this?
- typically we want to minimize $\|Dx\|_0$
- nonconvex, use our convex proxy $\|Dx\|_1$

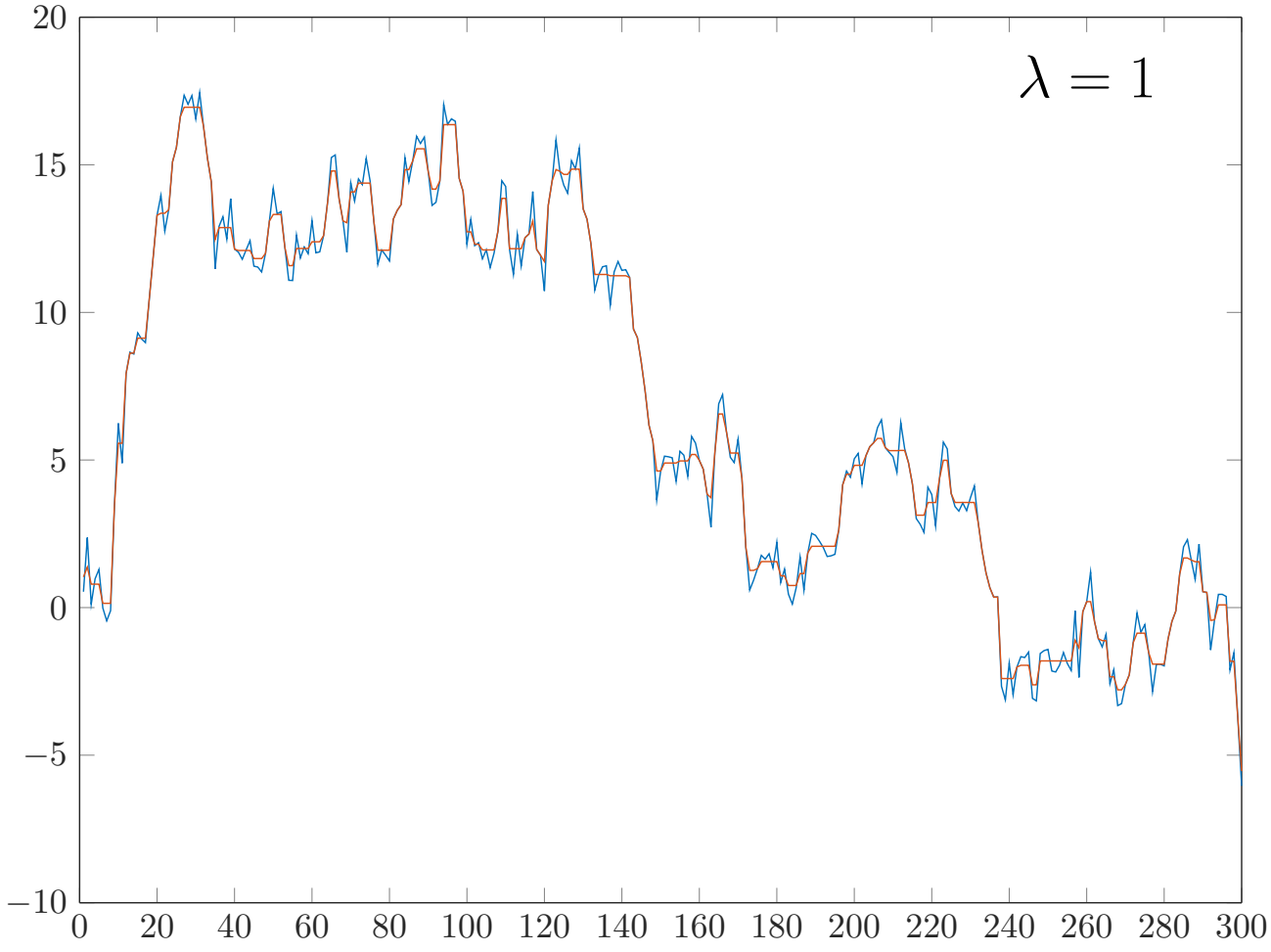
Result

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



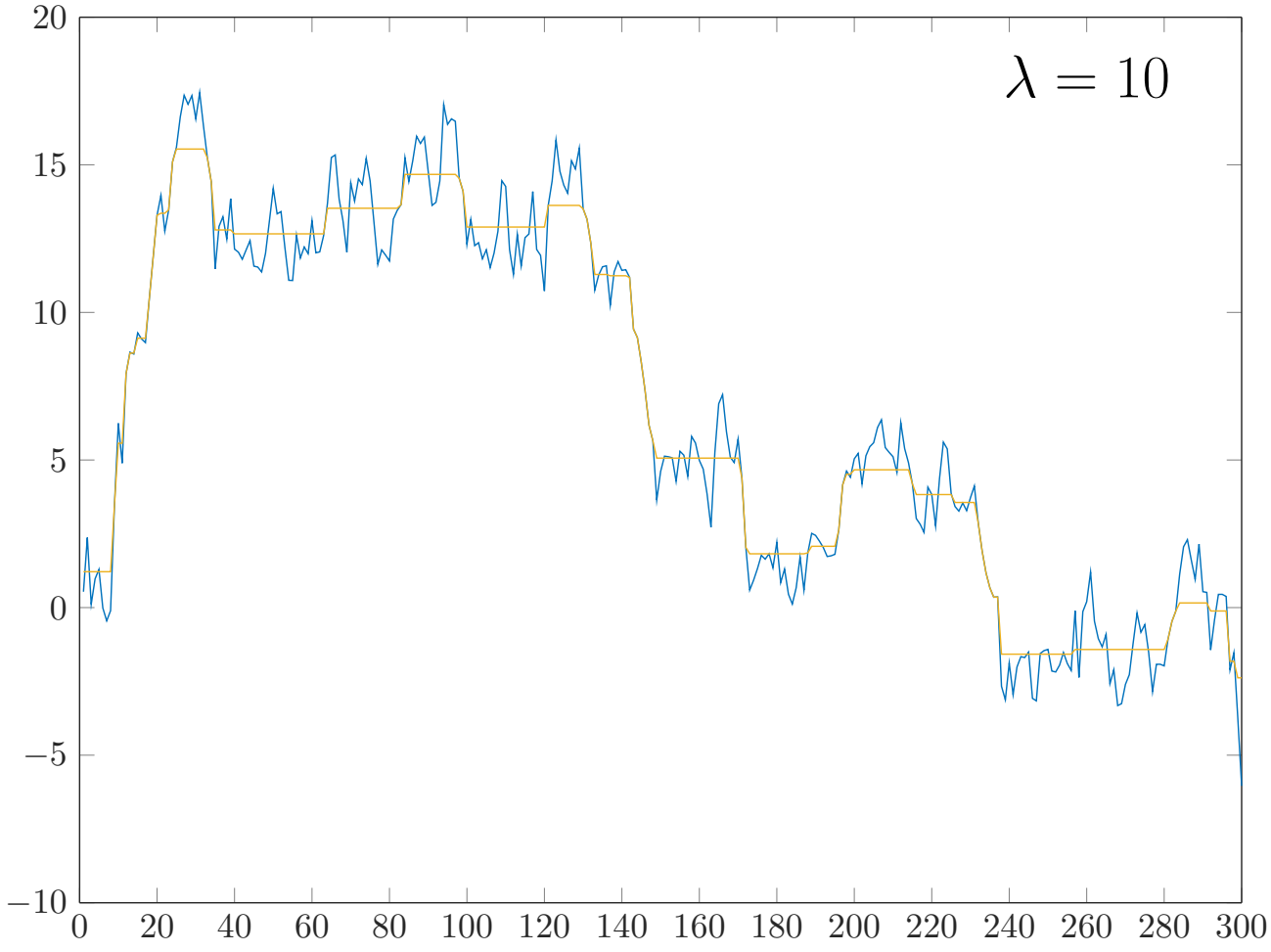
Result

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



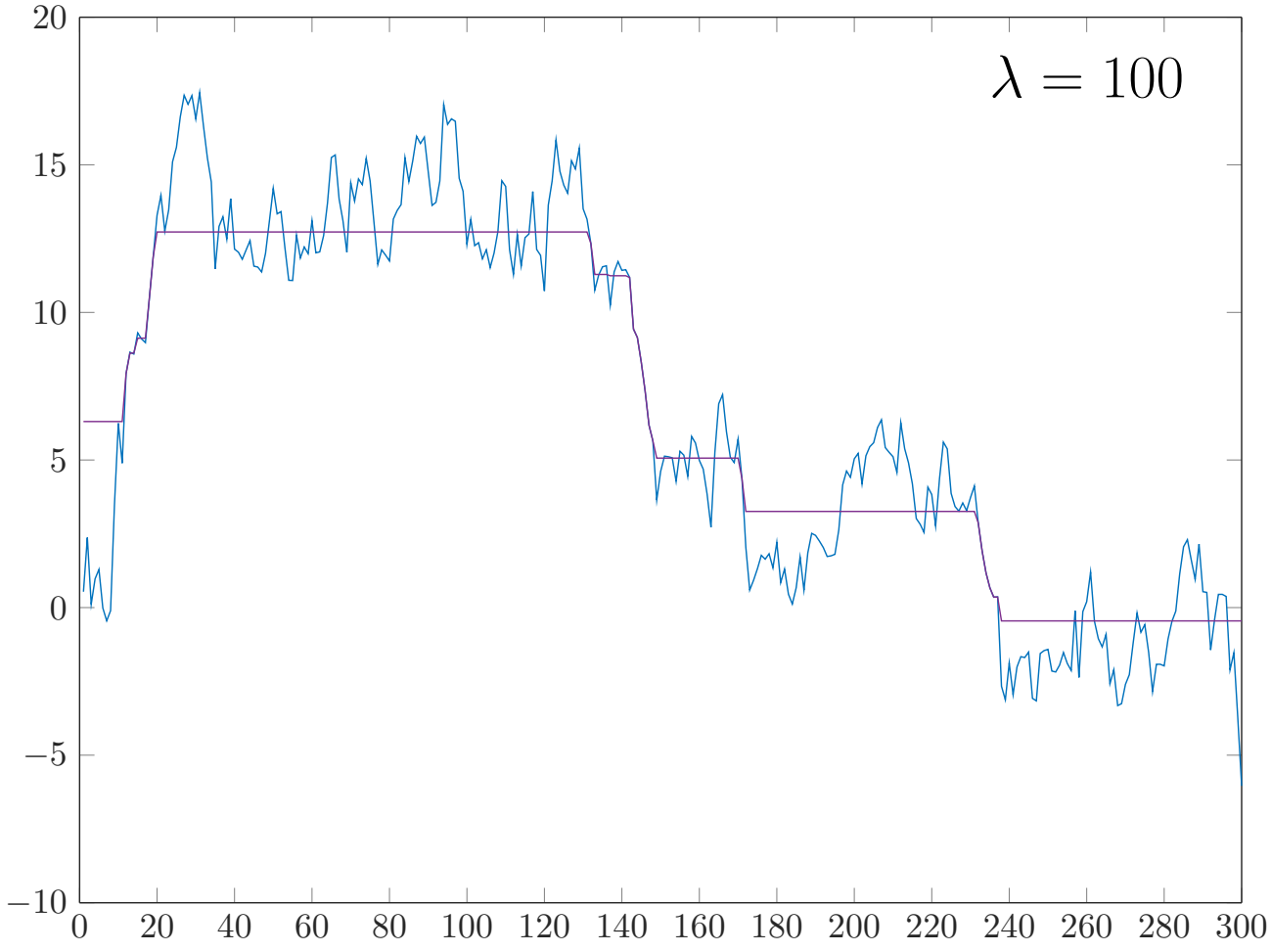
Result

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



Result

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



Piece-wise linear approximation

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

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

- this is zero on any line
- how to model piece-wise linear approximation?

Piece-wise linear approximation

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

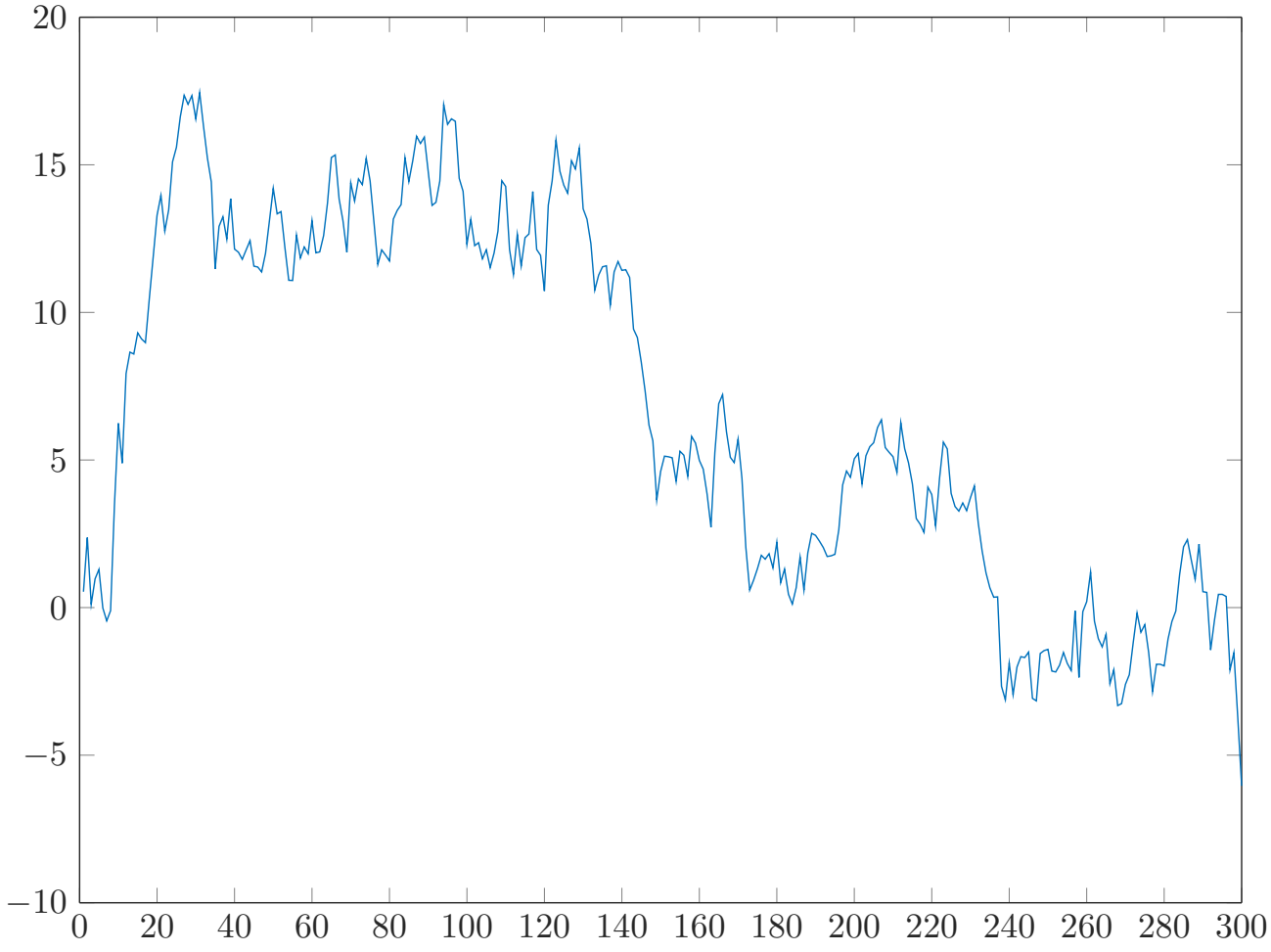
$$D_2 = \begin{bmatrix} 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -2 & 1 \end{bmatrix}$$

- this is zero on any line
- how to model piece-wise linear approximation?

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

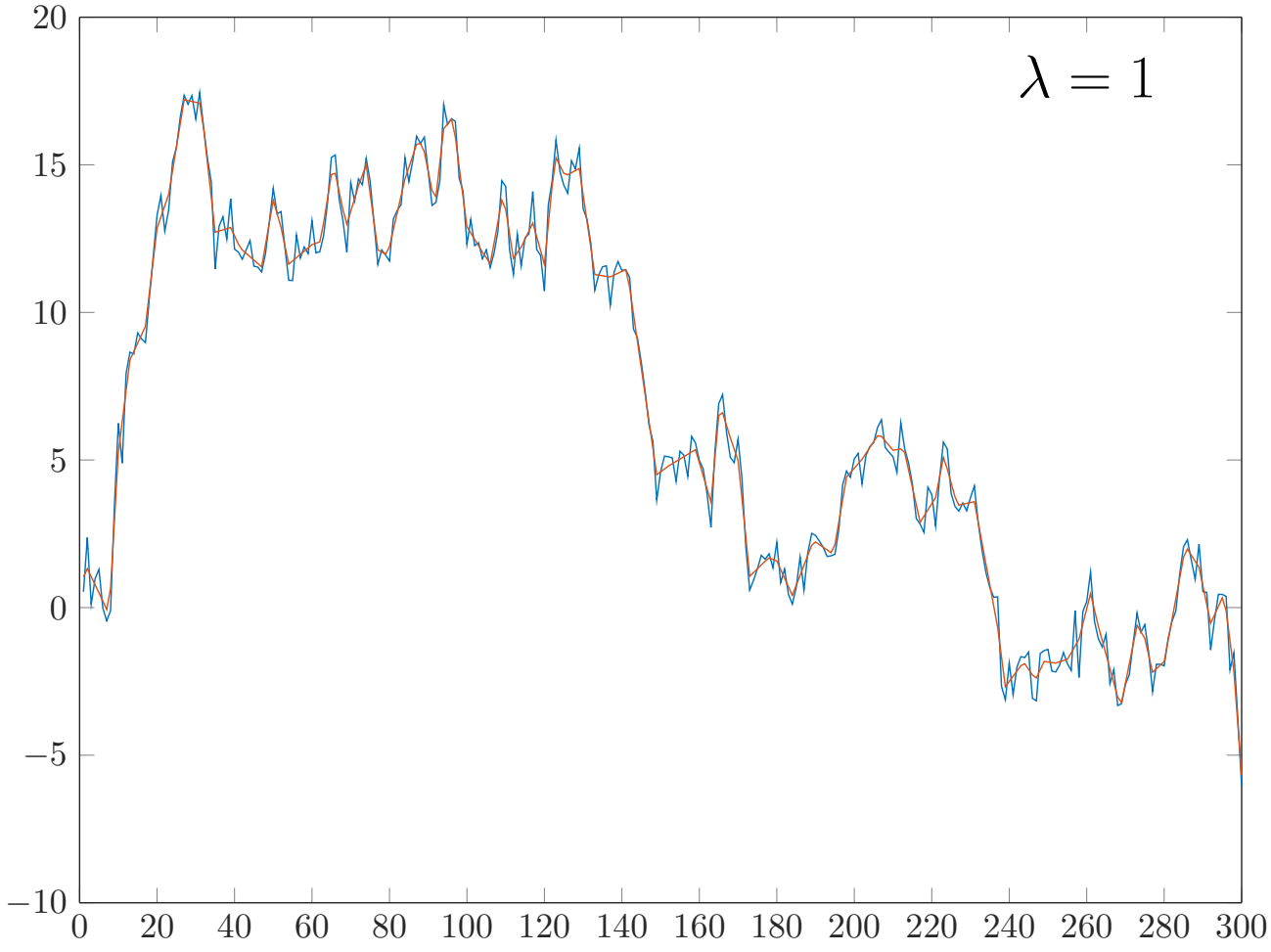
Result

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



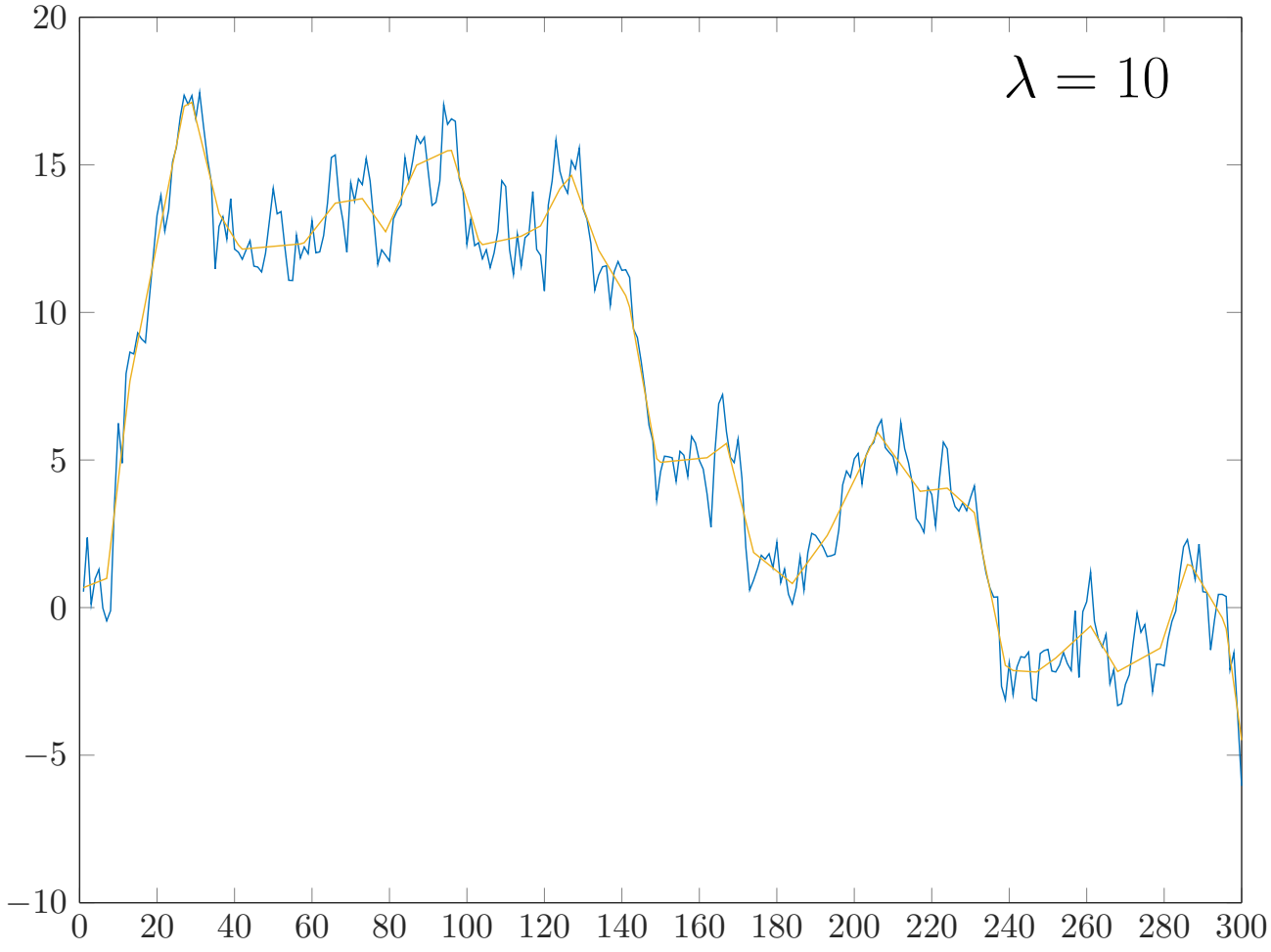
Result

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



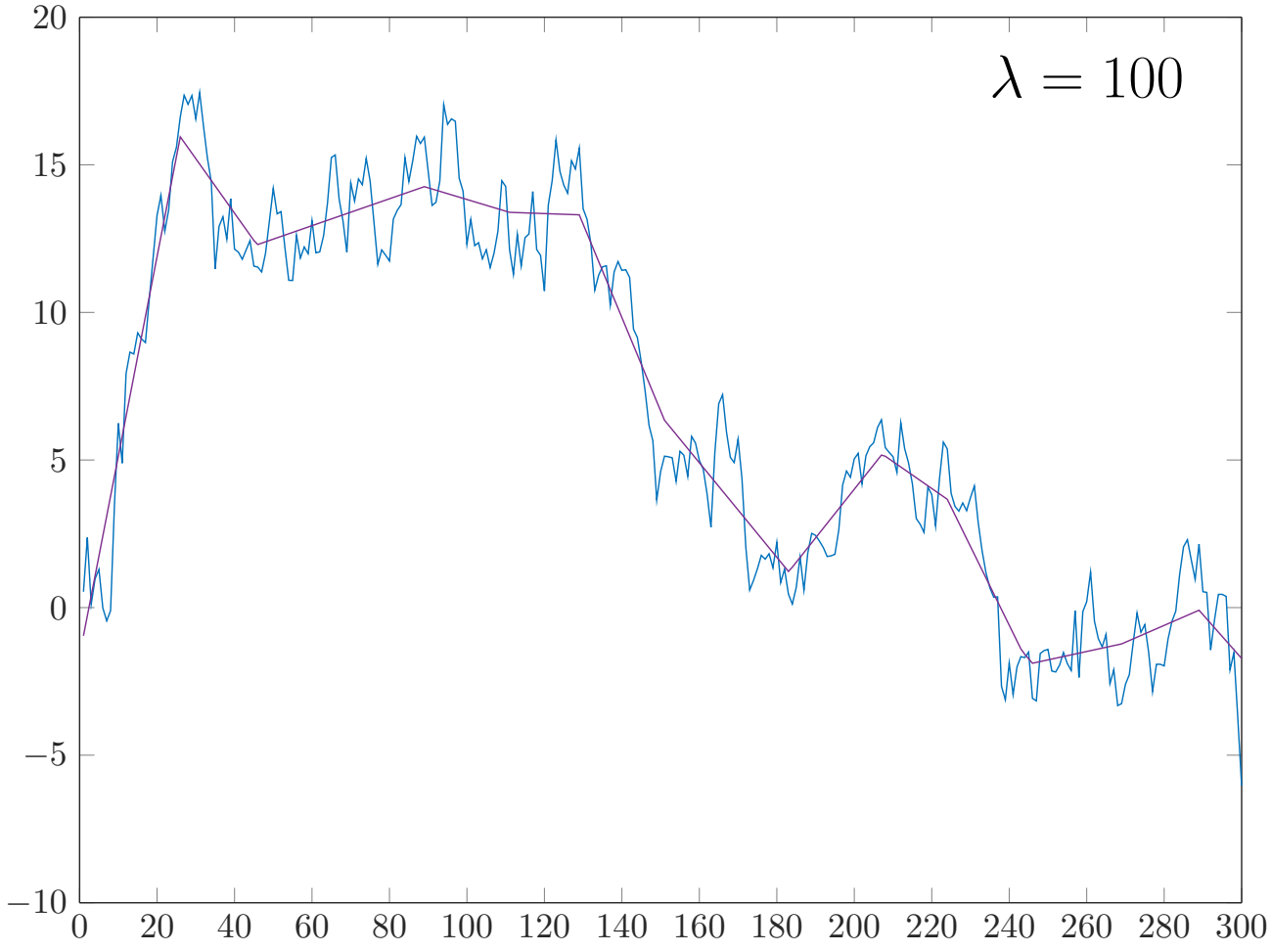
Result

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



Result

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



Smooth second derivative

- we might instead want a smooth second derivative
- how to model this?

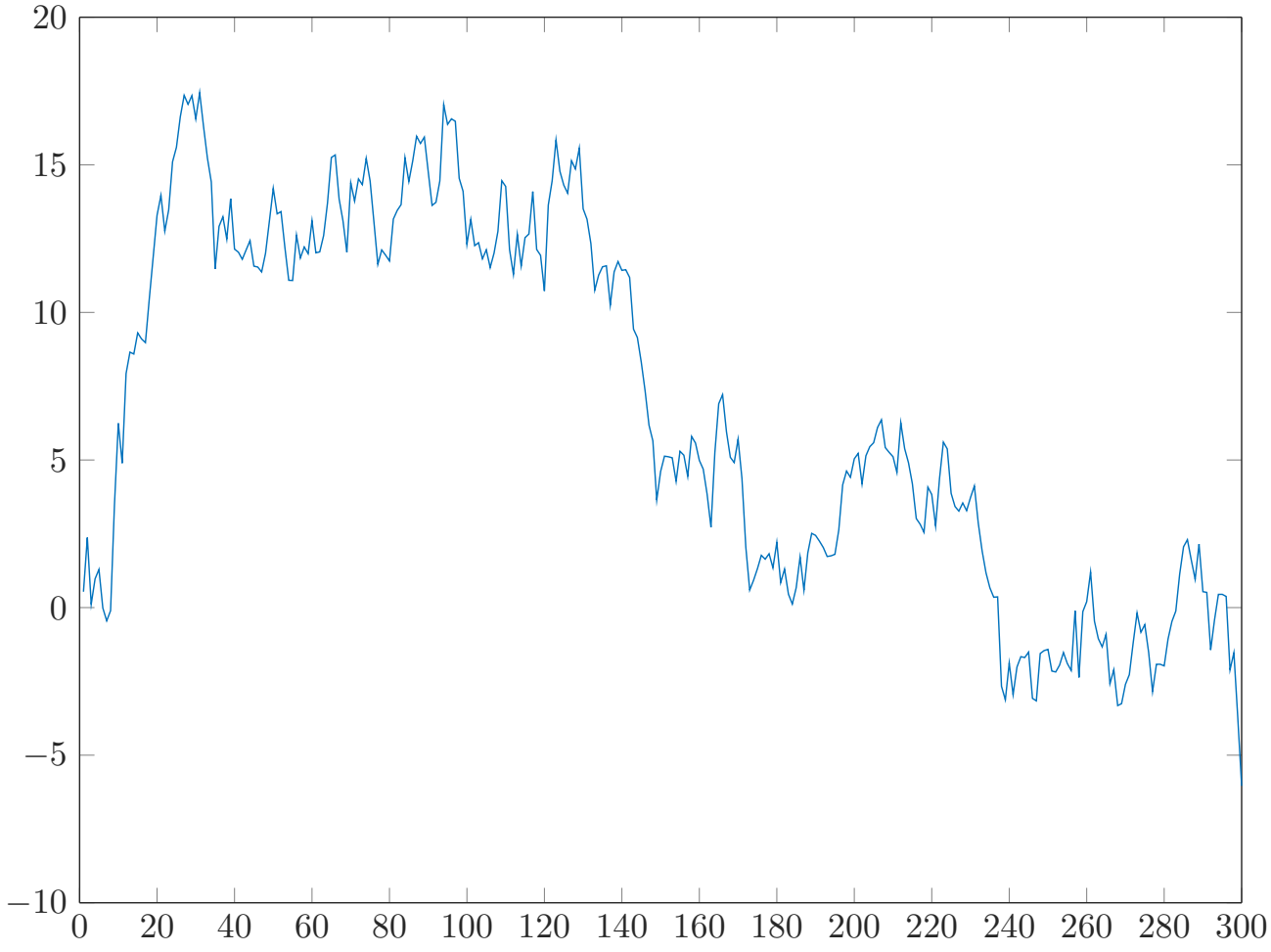
Smooth second derivative

- we might instead want a smooth second derivative
- how to model this?

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

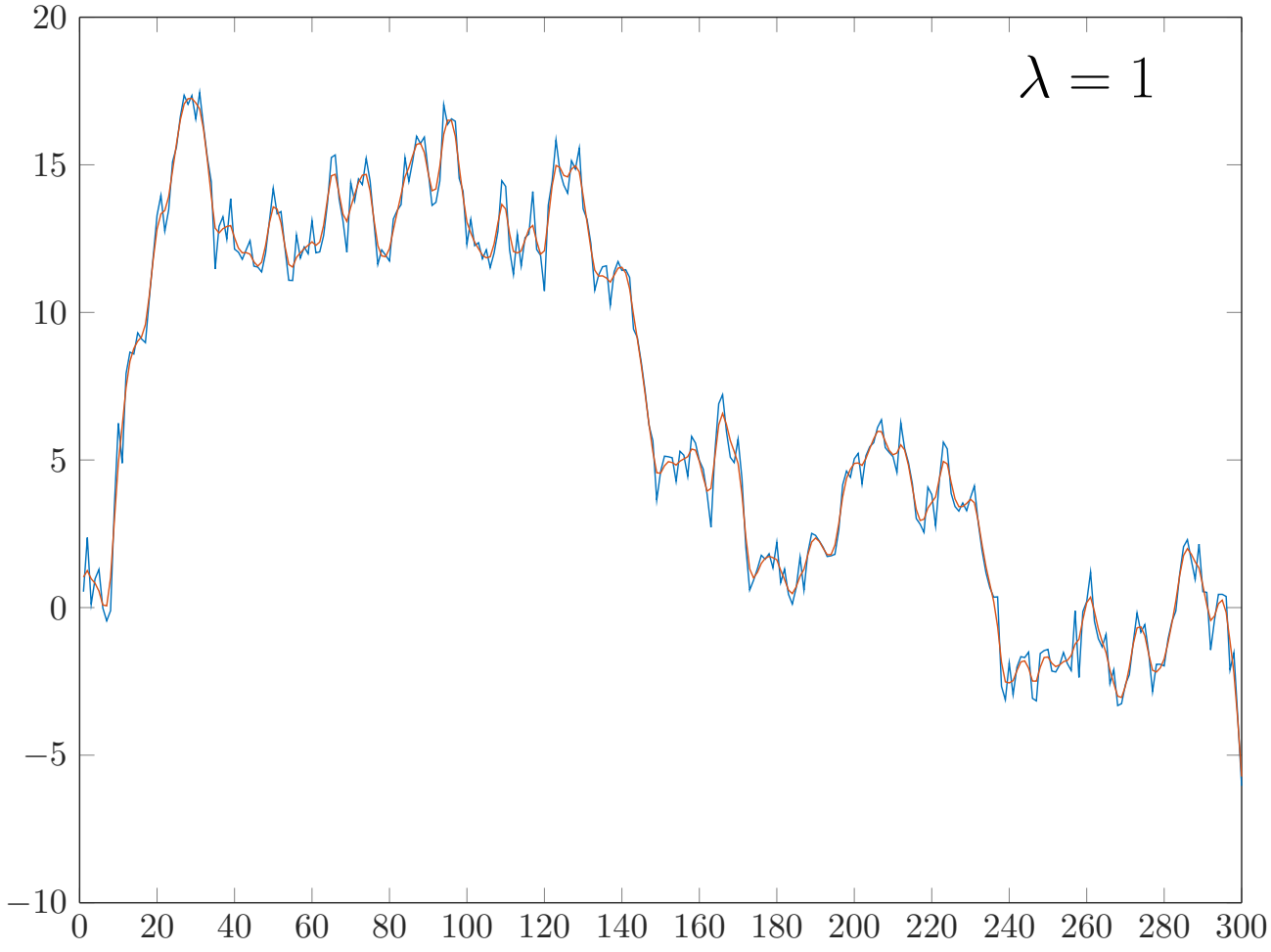
Result

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



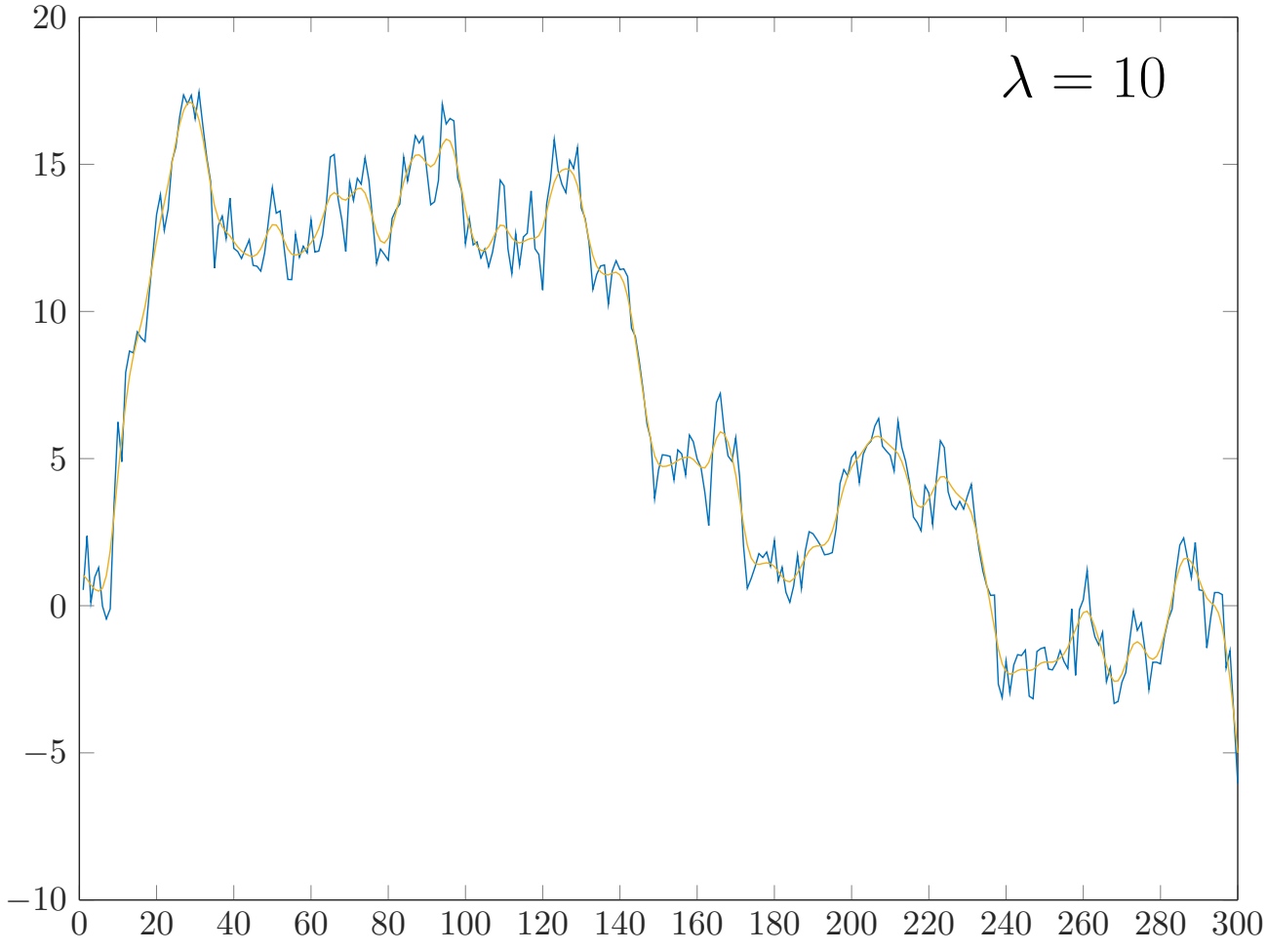
Result

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



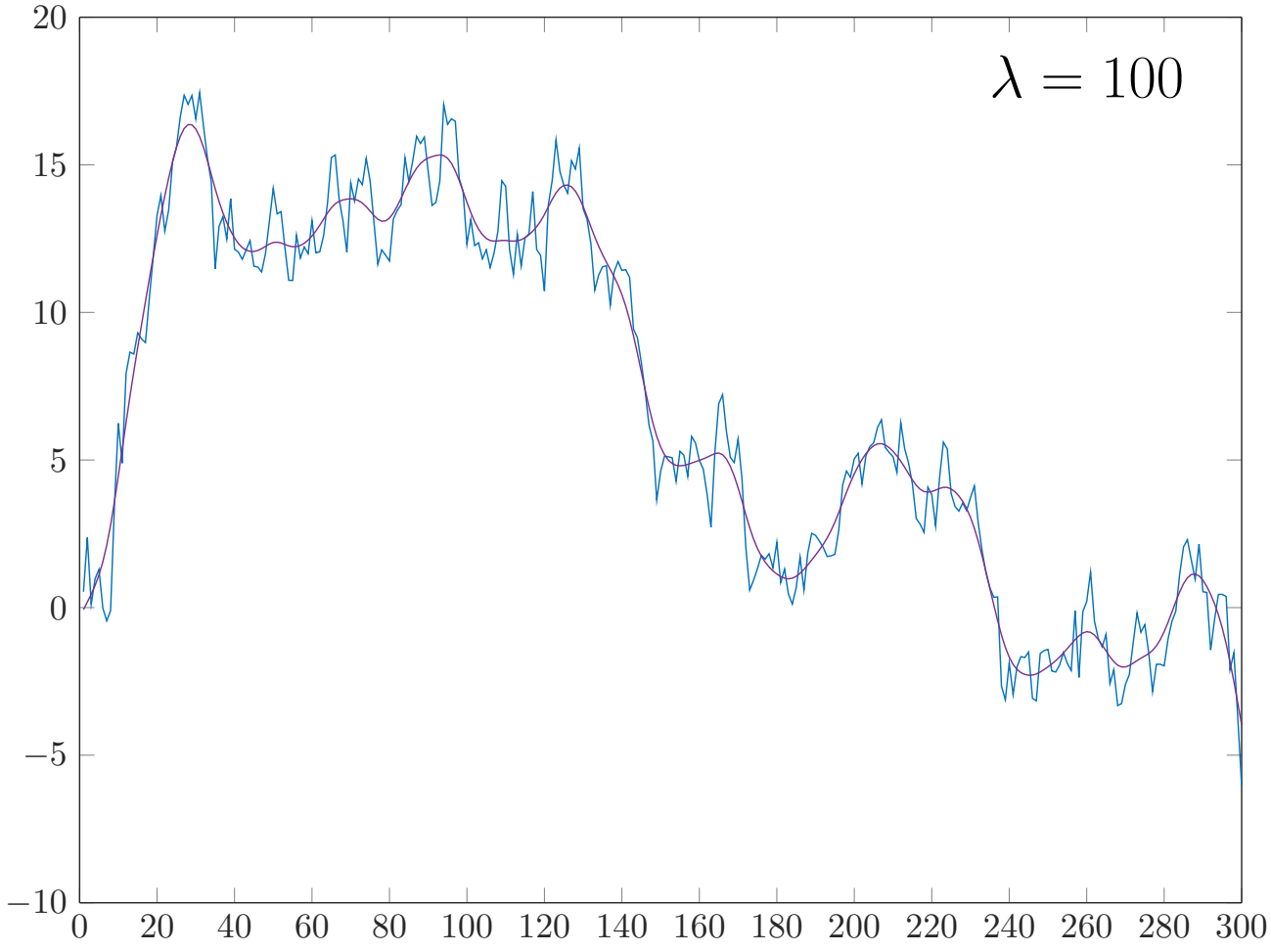
Result

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



Result

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



Periodic disturbances

- assume that our signal is disturbed by a periodic signal $p_d \in \mathbb{R}^n$
- p_d could model yearly/weekly/daily variations
- our measurement is still y
- we are interested in, say, a piece-wise linear estimation of $y - p$
- how to model this?

Periodic disturbances

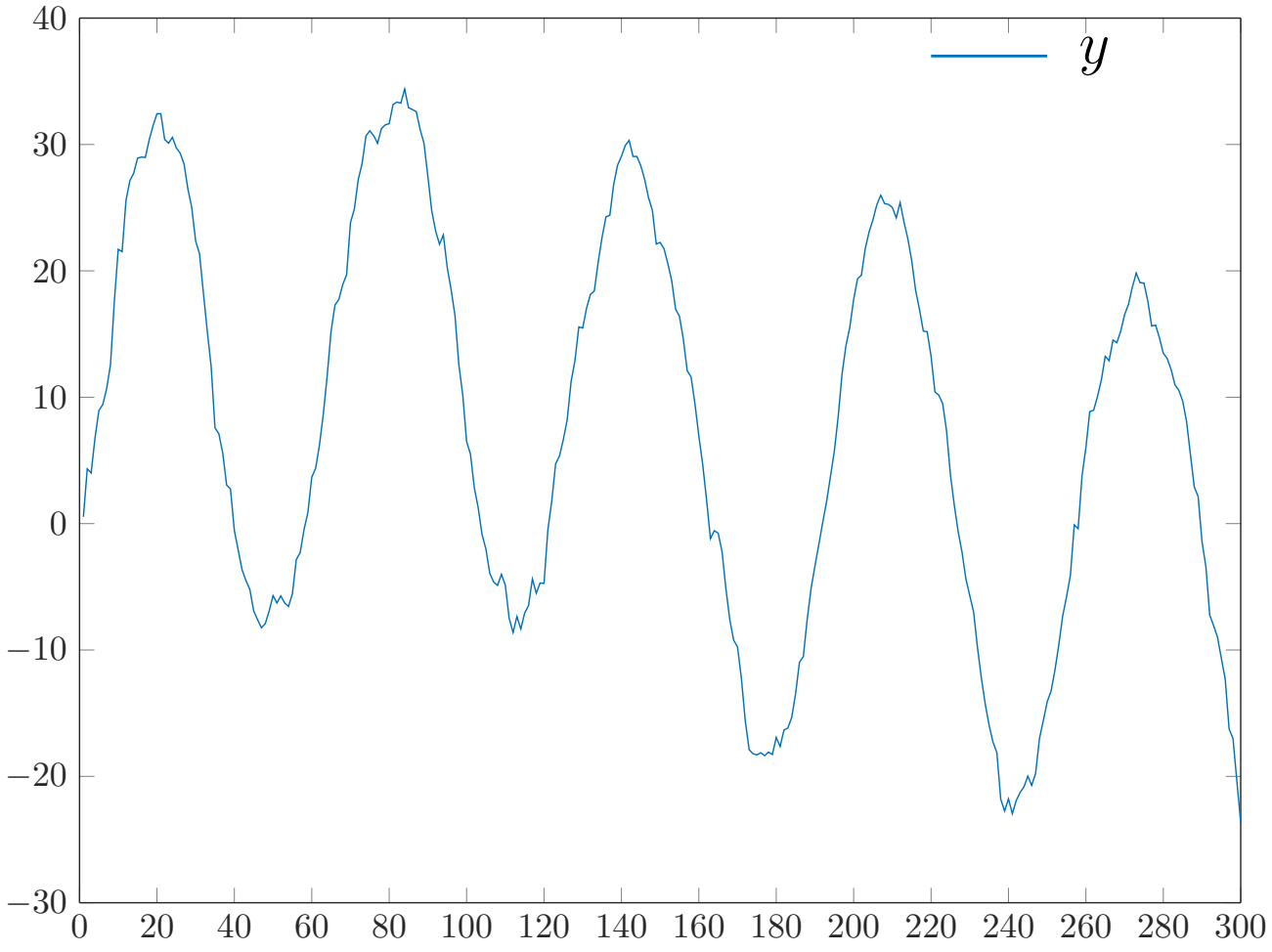
- assume that our signal is disturbed by a periodic signal $p_d \in \mathbb{R}^n$
- p_d could model yearly/weekly/daily variations
- our measurement is still y
- we are interested in, say, a piece-wise linear estimation of $y - p$
- how to model this? assume period is T

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\|x - (y-p)\|^2 + \lambda\|D_2 x\|_1 \\ \text{subject to} & p_i = p_{i+k_i T} \text{ for } i = 1, \ldots, T \text{ as long as } i + k_i T \leq n \end{array}$$

- x and p optimization variables! (p should estimate p_d)

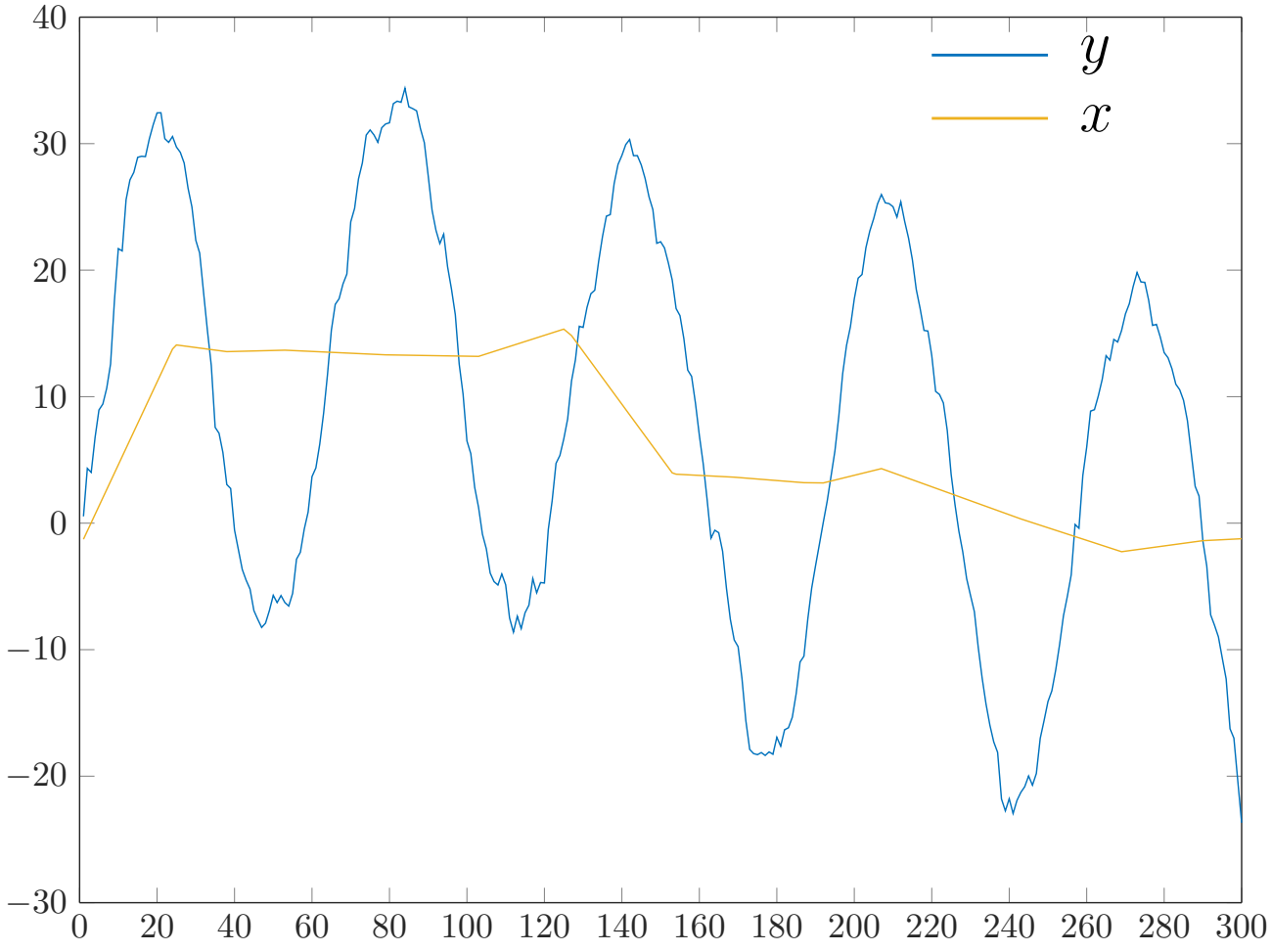
Result

minimize $\frac{1}{2}\|x-(y-p)\|^2+\lambda\|D_2x\|_1$
subject to $p_i = p_{i+k_iT}$ for $i=1,\ldots,T$ as long as $i+k_iT \le n$



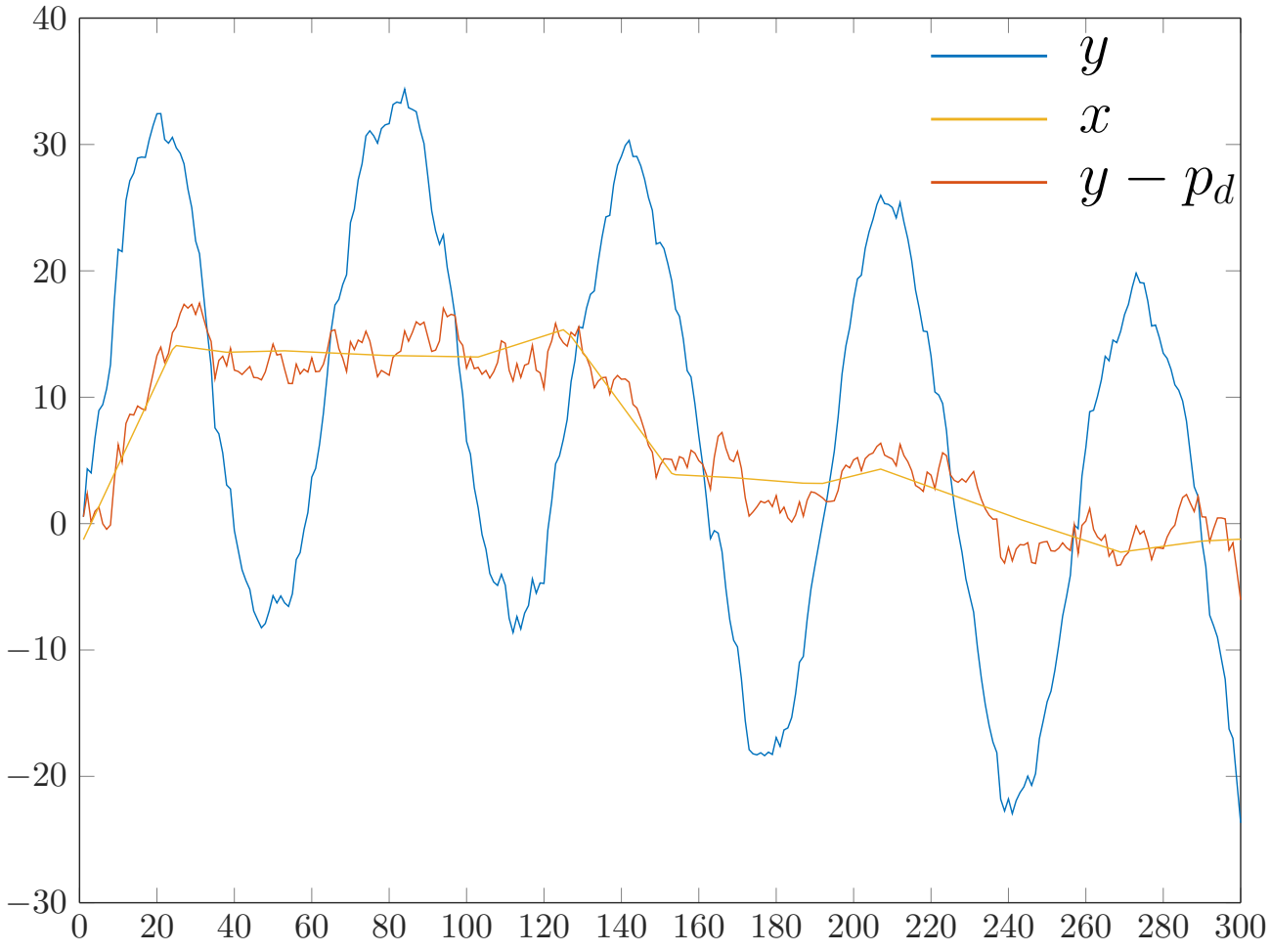
Result

minimize $\frac{1}{2}\|x-(y-p)\|^2+\lambda\|D_2x\|_1$
subject to $p_i = p_{i+k_iT}$ for $i=1,\ldots,T$ as long as $i+k_iT \le n$



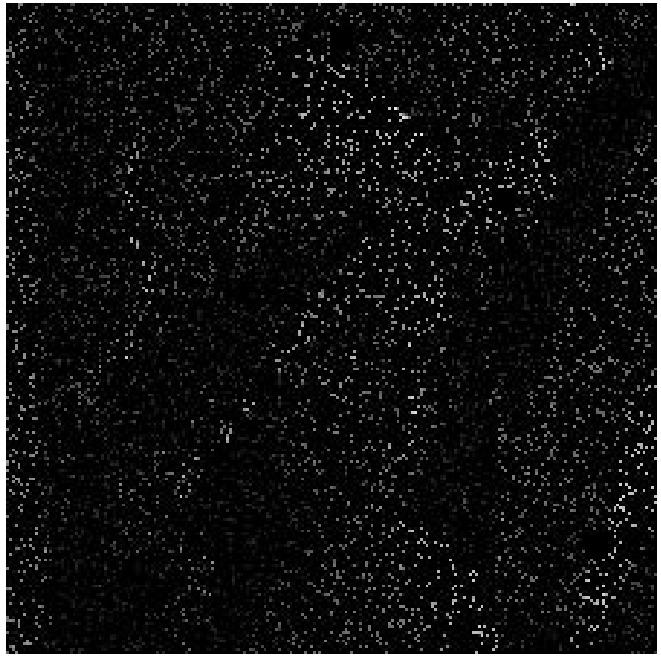
Result

minimize $\frac{1}{2}\|x-(y-p)\|^2+\lambda\|D_2x\|_1$
subject to $p_i = p_{i+k_iT}$ for $i=1,\dots,T$ as long as $i+k_iT \le n$



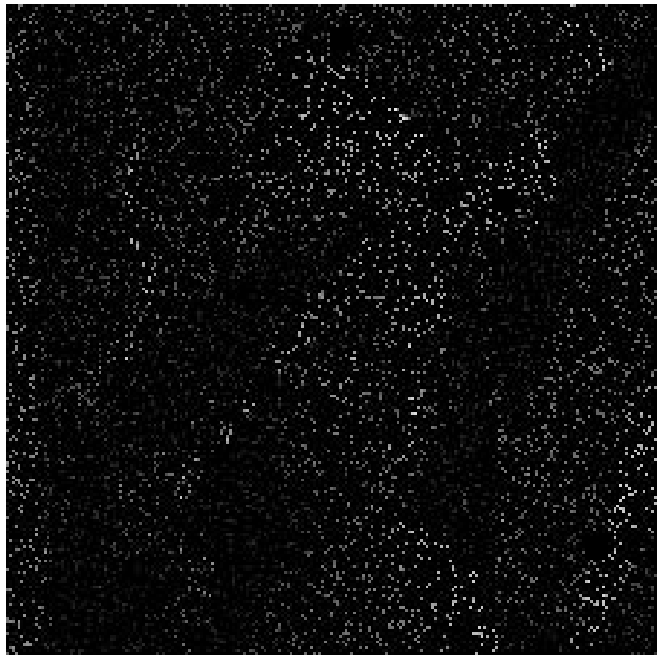
Two-dimensional reconstruction

- can also reconstruct images (2D-signals)
- example: 90% of pixels in image lost



Two-dimensional reconstruction

- can also reconstruct images (2D-signals)
- example: 90% of pixels in image lost



- reconstruct using difference in 2D (TV-norm)

$$\text{minimize} \sum_{i=1}^{n-1}\sum_{j=1}^{m} |x_{i,j} - x_{i+1,j}| + \sum_{i=1}^{n}\sum_{j=1}^{m-1} |x_{i,j} - x_{i,j+1}|$$

- known pixels are set to correct value

Two-dimensional reconstruction

- example: 70% of pixels in image lost

Two-dimensional reconstruction

- example: 70% of pixels in image lost

Two-dimensional reconstruction

- example: 50% of pixels in image lost

Two-dimensional reconstruction

- example: 50% of pixels in image lost

Two-dimensional reconstruction

- example: 30% of pixels in image lost

Two-dimensional reconstruction

- example: 30% of pixels in image lost

Comparison to ground truth

Modeling idea

- if you want to enforce something approximately, use $\|\cdot\|_2^2$
- if you want to be likely to enforce something (sparsity), 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 is represented as points x_i in an 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*

Examples

- features might be:
  1. frequencies of words in a dictionary
  2. boolean variables, e.g., is actor in movie m?
  3. numerical values of blood pressure, temperature, price
  4. rating of a movie, music, etc
- examples might be:
  1. emails
  2. different actors
  3. patients
  4. streaming service customers

Supervised and unsupervised learning

- we can roughly divide learning tasks into
  - supervised learning
  - unsupervised learning
- supervised learning:
  - also have response variables y_i for each example
  - response variables can be real-valued (regression)
  - response variables can be integer-valued (classification)
  - objective: create model of unknown function $x \mapsto y(x)$ (x data-vector and y response variable)
- unsupervised learning:
  - no response variables
  - objective: learn information or structure about data
- we will talk about supervised learning

Linear model

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

$$\hat{y}(x) = b + \langle s, x \rangle$$

- objective: minimize prediction error on data:

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

- let $\beta = (s, b)$, then the problem becomes:

$$\text{minimize} \sum_{i=1}^{n} (\langle s, x_i \rangle + b - y_i)^2 = \|\Phi\beta - y\|^2$$

- least squares problem

Removing constant term

- what is optimal b?

$$\text{minimize } \sum_{i=1}^{n}(\langle s, x_i\rangle + b - y_i)^2 = \|\Phi\beta - y\|^2$$

- optimality condition w.r.t. b:

$$bn + \sum_{i=1}^{n}(\langle s, x_i\rangle - y_i) = 0 \quad \Leftrightarrow \quad b = \tfrac{1}{n}(\sum_{i=1}^{n} y_i - \langle s, x_i\rangle) = \bar{y} - \langle s, \bar{x}\rangle$$

  where $\bar{x}$ and $\bar{y}$ are mean values
- let $\tilde{x}_i = x_i - \bar{x}$ and $\tilde{y}_i = y_i - \bar{y}$, then it is equivalent to solve

$$\text{minimize } \sum_{i=1}^{n}(\langle s, \tilde{x}_i\rangle - \tilde{y}_i)^2 = \|\tilde{X}s - \tilde{y}\|^2$$

  where $\tilde{X}$ contains all $\tilde{x}_i$
- we assume from now that average subtracted from data

Scaling response variables

- what happens if we scale our responses y with a nonzero scalar γ?
- the problem becomes

$$\text{minimize } \|Xs - \gamma y\|^2 = \|\gamma(X\tfrac{s}{\gamma} - y)\|^2 = \gamma^2\|X\tfrac{s}{\gamma} - y\|^2$$

- the solution gets scaled with γ^{-1}
- convention: scale y with norm of y

Scaling features

- consider the least squares problem ($X = [x_1 \ldots, x_n]$)

$$\text{minimize } \|Xs - y\|^2 = \left\| \sum_{i=1}^{n} s_i x_i - y \right\|^2$$

- "select linear combination of features that best approximates y"
- big value of s_i means feature i important in describing y
- for any i replace x_i with $\hat{x}_i = 2x_i$, what happens with solution s?

Scaling features

- consider the least squares problem ($X = [x_1 \ldots, x_n]$)

$$\text{minimize } \|Xs - y\|^2 = \left\| \sum_{i=1}^{n} s_i x_i - y \right\|^2$$

- "select linear combination of features that best approximates y"
- big value of s_i means feature i important in describing y
- for any i replace x_i with $\hat{x}_i = 2x_i$, what happens with solution s?
- we get $\hat{s}_i = \frac{1}{2} s_i$, the rest the same
- is feature i now less important in prediction?

Scaling features

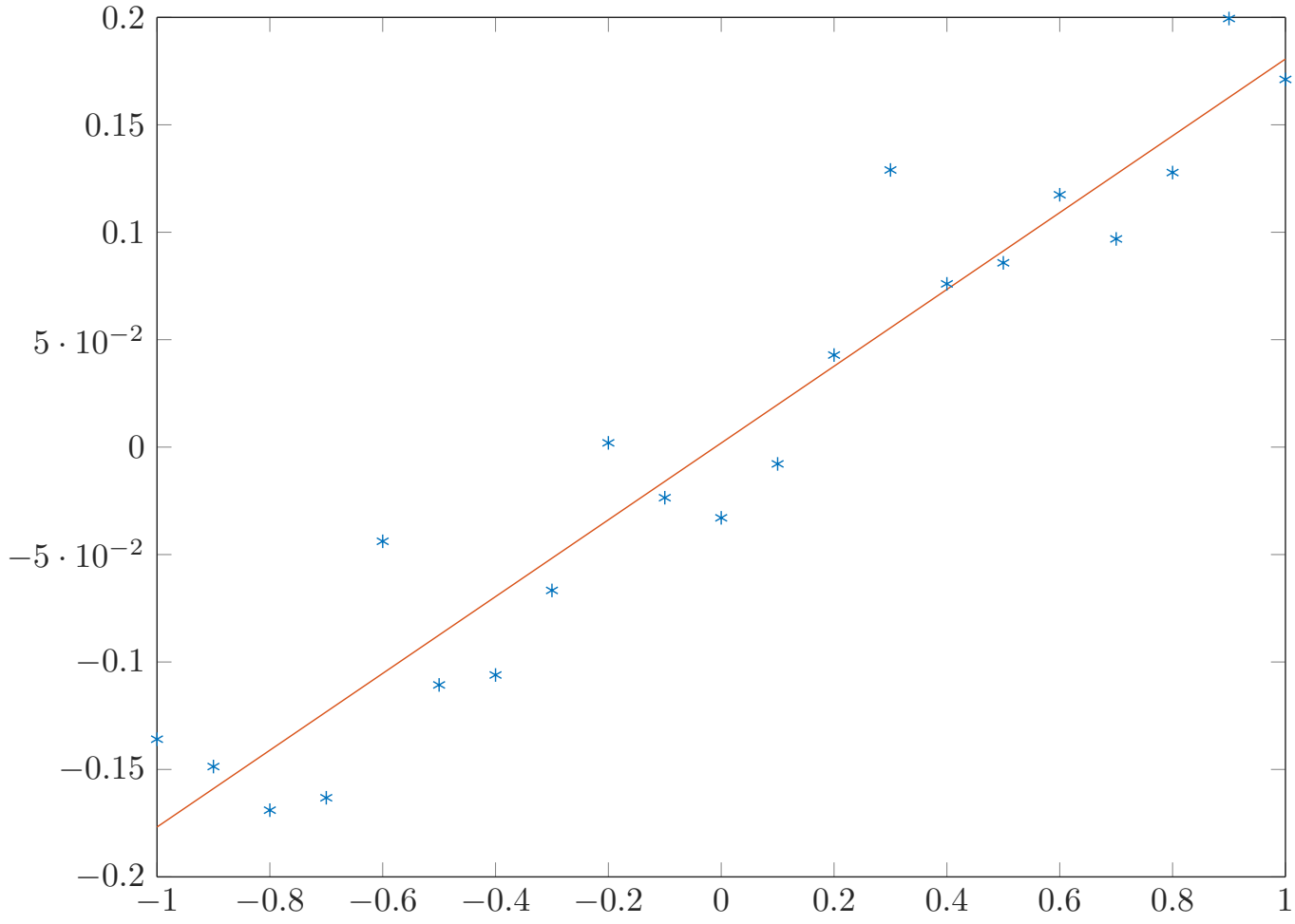
- consider the least squares problem ($X = [x_1 \ldots, x_n]$)

$$\text{minimize } \|Xs - y\|^2 = \left\| \sum_{i=1}^{n} s_i x_i - y \right\|^2$$

- "select linear combination of features that best approximates y"
- big value of s_i means feature i important in describing y
- for any i replace x_i with $\hat{x}_i = 2x_i$, what happens with solution s?
- we get $\hat{s}_i = \frac{1}{2} s_i$, the rest the same
- is feature i now less important in prediction?
- of course not, to avoid this, scale all features to unit norm
- (diagonal elements of $X^T X$ become 1 $\Rightarrow$ Jacobi scaling)

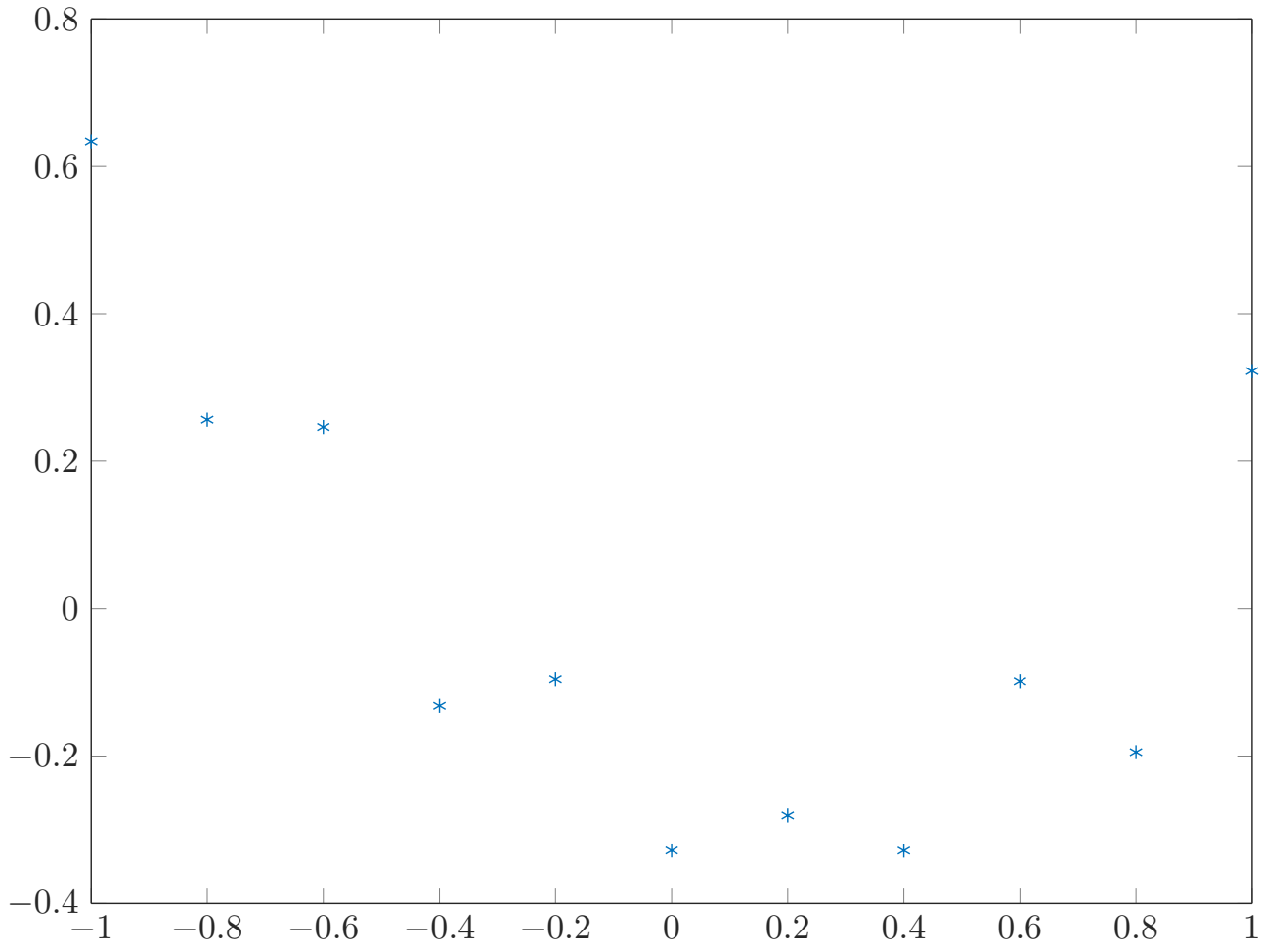
Example

- fit affine line to data using LS:



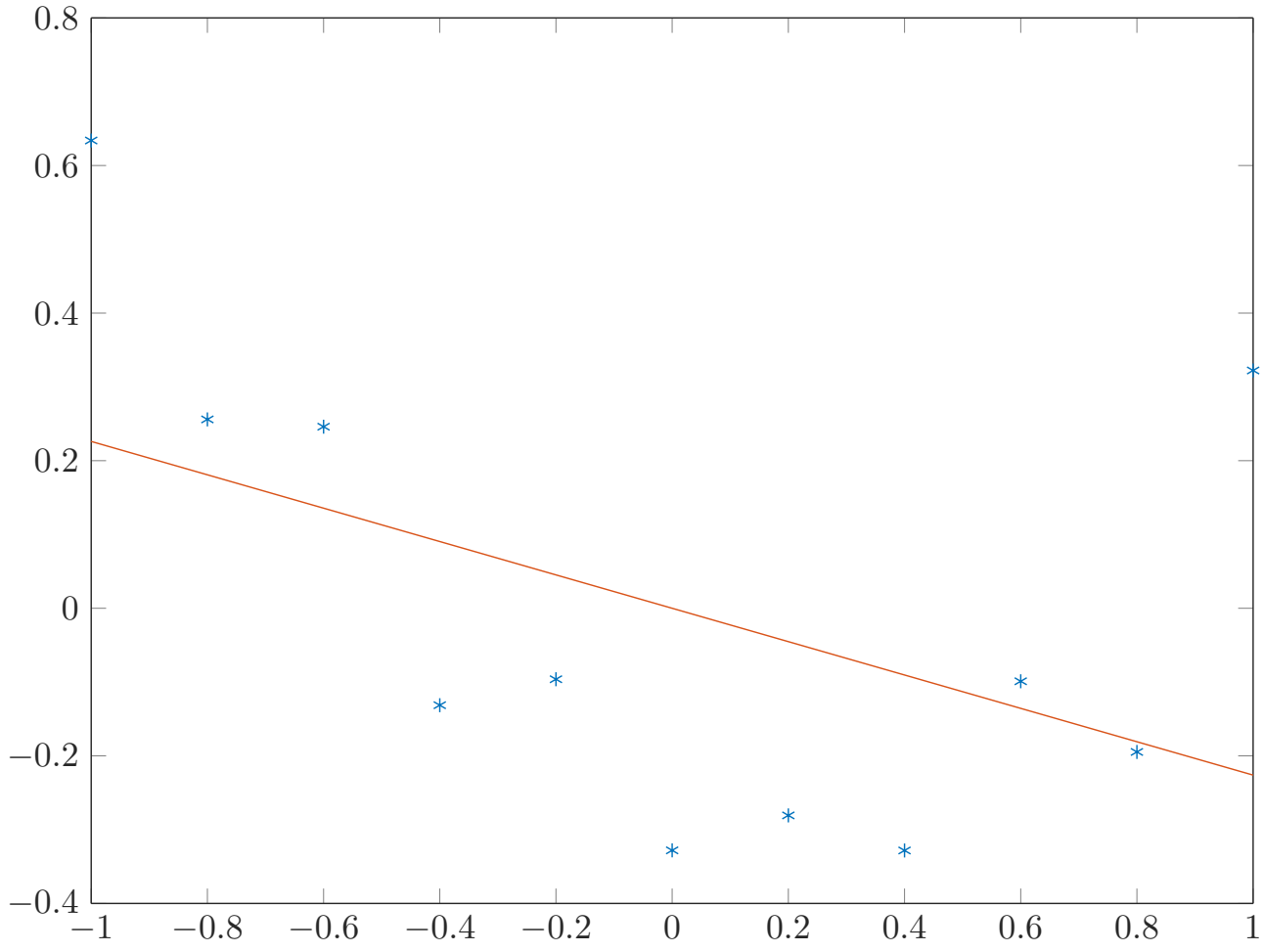
Nonaffine example

- fit affine line to data using LS:



Nonaffine example

- fit affine line to data using LS:



Polynomial models

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

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

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

$$\hat{y}(x) = b + sx + wx^2 = \langle \beta, \phi(x) \rangle$$

  where $\beta = (b, s, w)$ and $\phi(x) = (1, x, x^2)$
- for $x \in \mathbb{R}^2$, this becomes

$$\hat{y}(x) = b + s_1 x_1 + s_2 x_2 + w_{11} x_i^2 + w_{12} x_i x_j + w_{22} x_j^2 = \langle \beta, \phi(x) \rangle$$

- where

$$\beta = (b, s_1, s_2, w_{11}, w_{12}, w_{22})$$
$$\phi(x) = (1, x_1, x_2, x_1^2, x_1 x_2, x_2^2)$$

- we add new features to problem, still linear in parameters β_i

Least squares estimate

- data model example:

$$\hat{y}(x) = b + s_1 x_1 + s_2 x_2 + w_{11} x_i^2 + w_{12} x_i x_j + w_{22} x_j^2 = \langle \beta, \phi(x) \rangle$$

- least squares estimate

$$\text{minimize } (\hat{y}(x_i) - y_i)^2 = (\langle \beta, \phi(x_i) \rangle - y_i)^2$$

- build new data matrix

$$X = \begin{bmatrix} 1 & [x_1]_1 & [x_1]_2 & [x_1]_1[x_1]_1 & [x_1]_1[x_1]_2 & [x_1]_2[x_1]_2 \\ \vdots & & & & & \\ 1 & [x_m]_1 & [x_m]_2 & [x_m]_1[x_m]_1 & [x_m]_1[x_m]_2 & [x_m]_2[x_m]_2 \end{bmatrix}$$

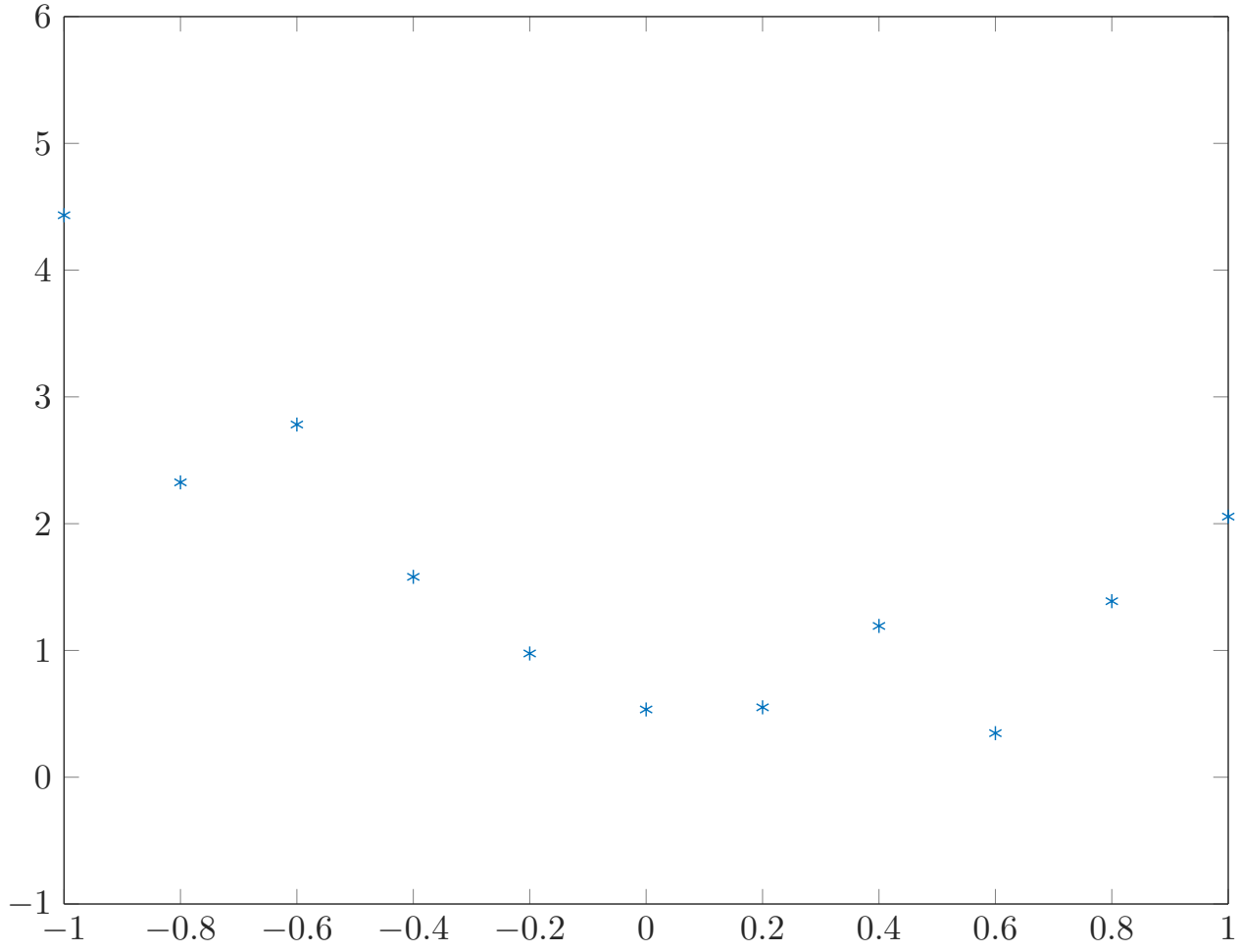
- then LS problem can be written as

$$\text{minimize } \|X\beta - y\|^2$$

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

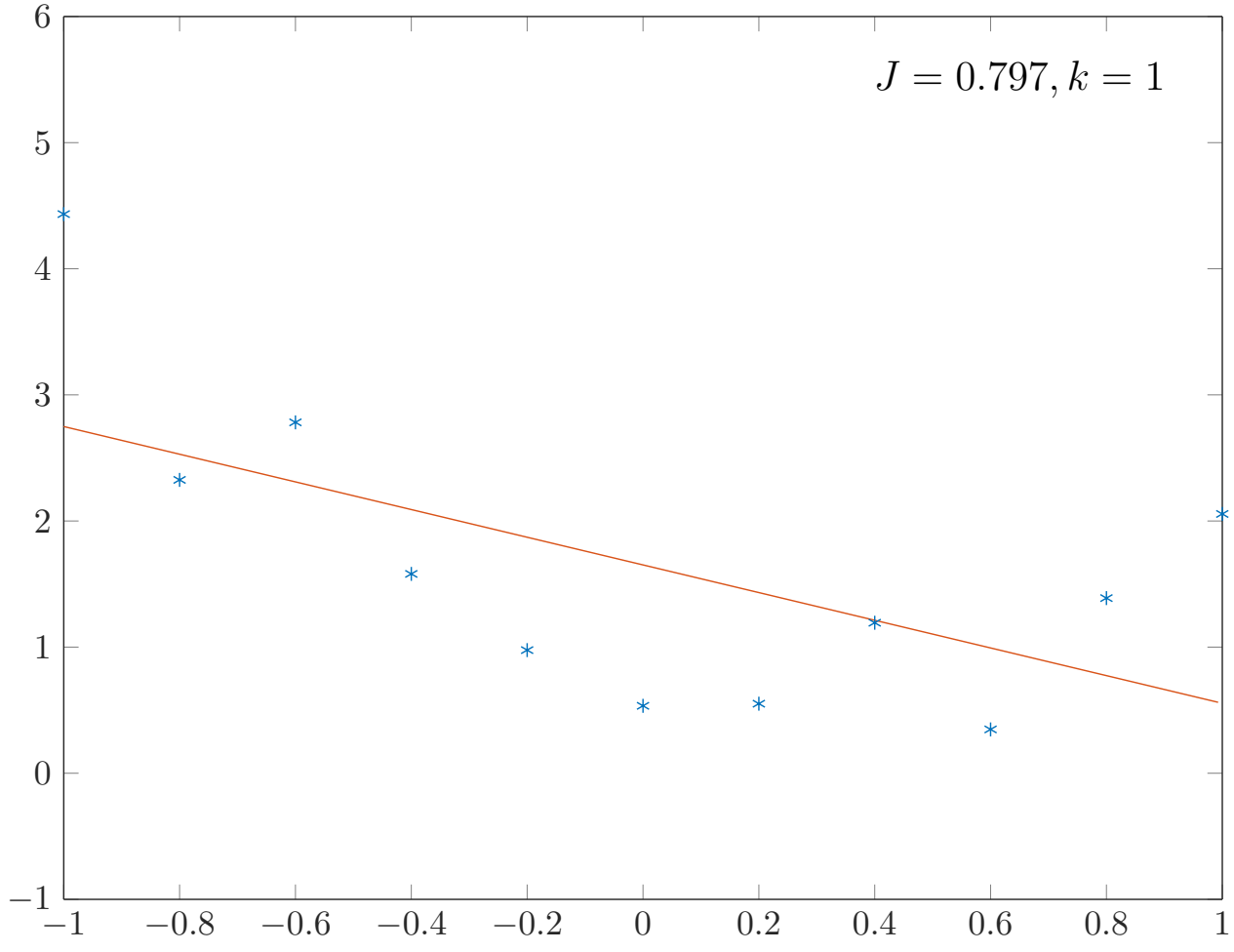
Nonaffine example

- fit polynomial of degree k to data using LS (J is cost):



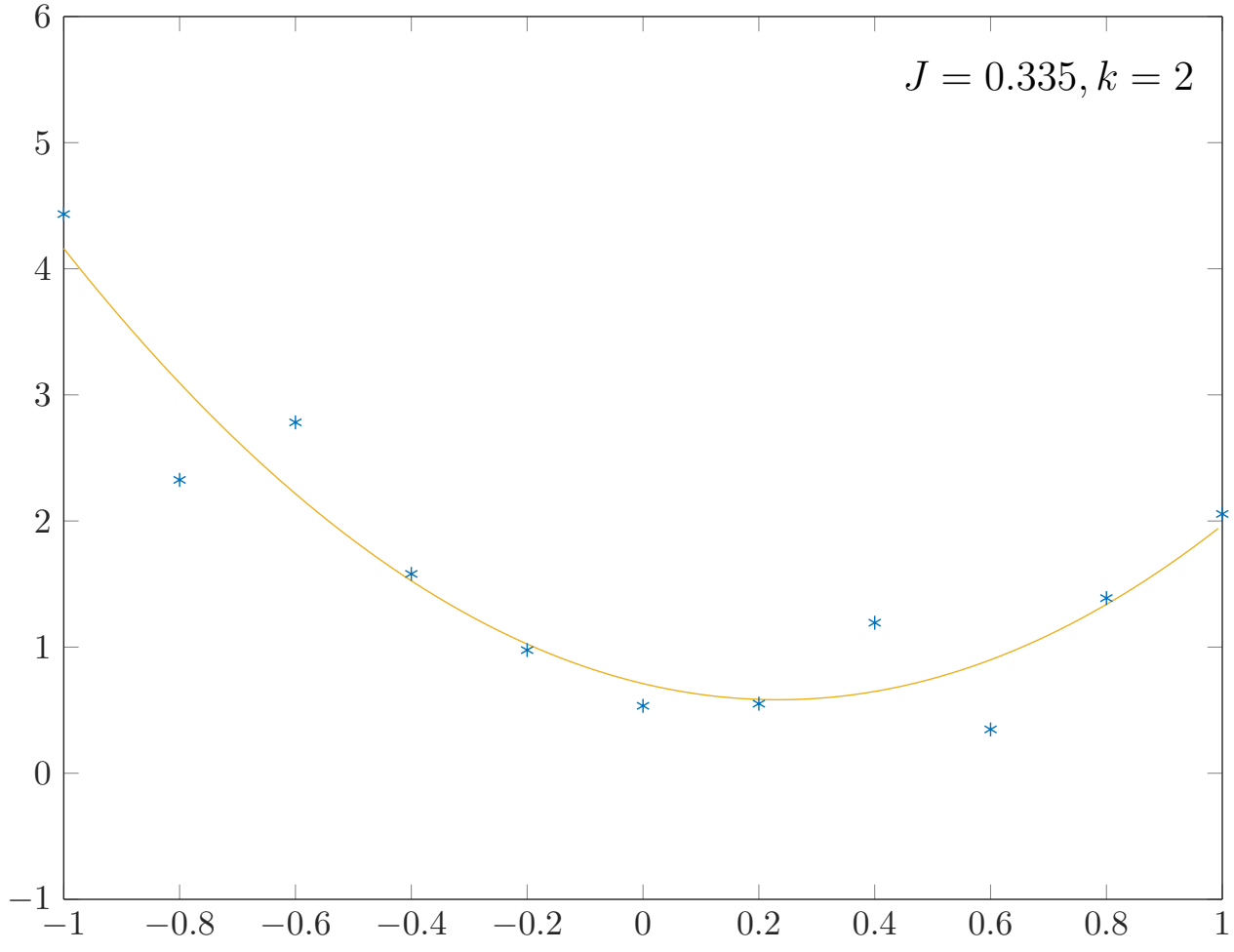
Nonaffine example

- fit polynomial of degree k to data using LS (J is cost):



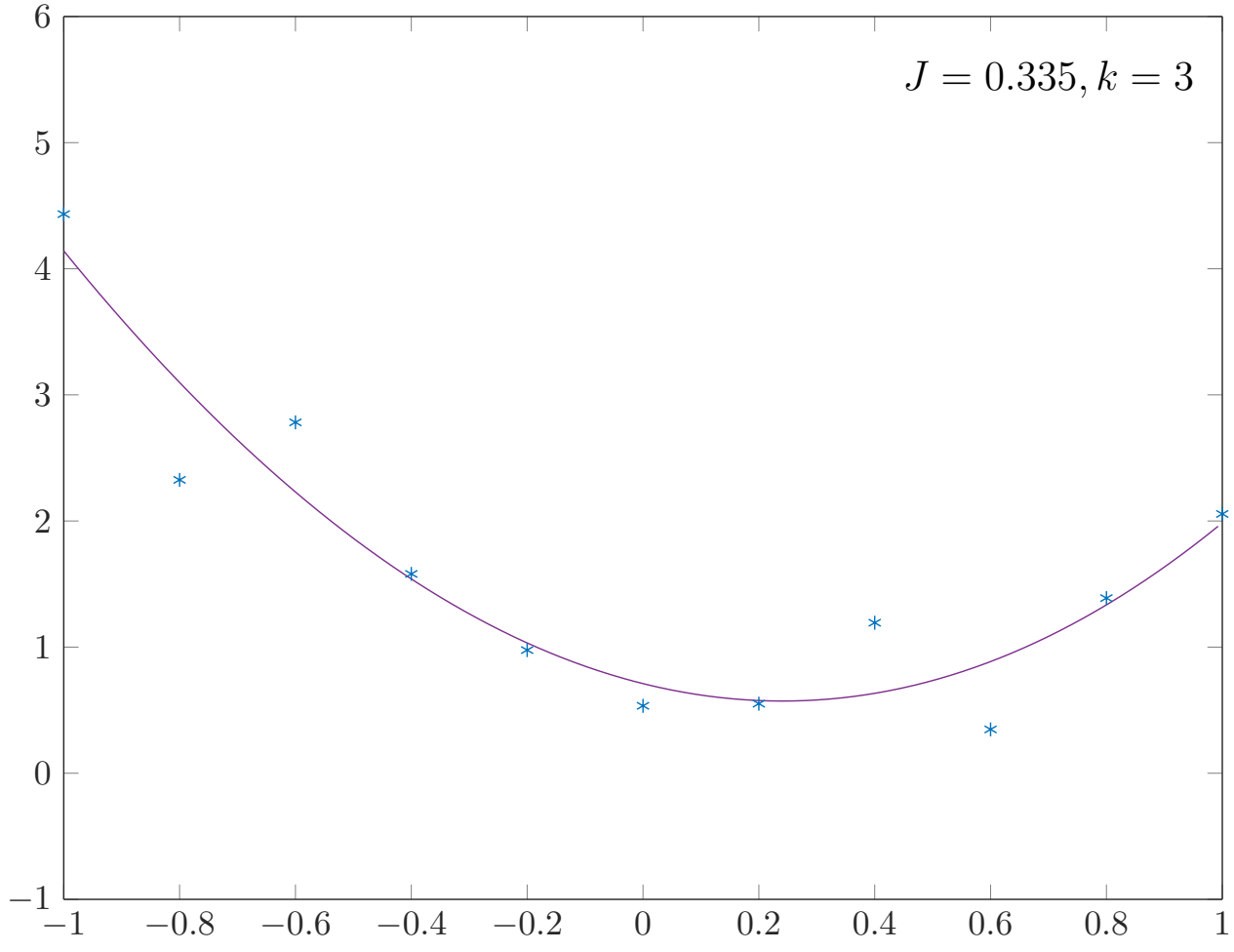
Nonaffine example

- fit polynomial of degree k to data using LS (J is cost):



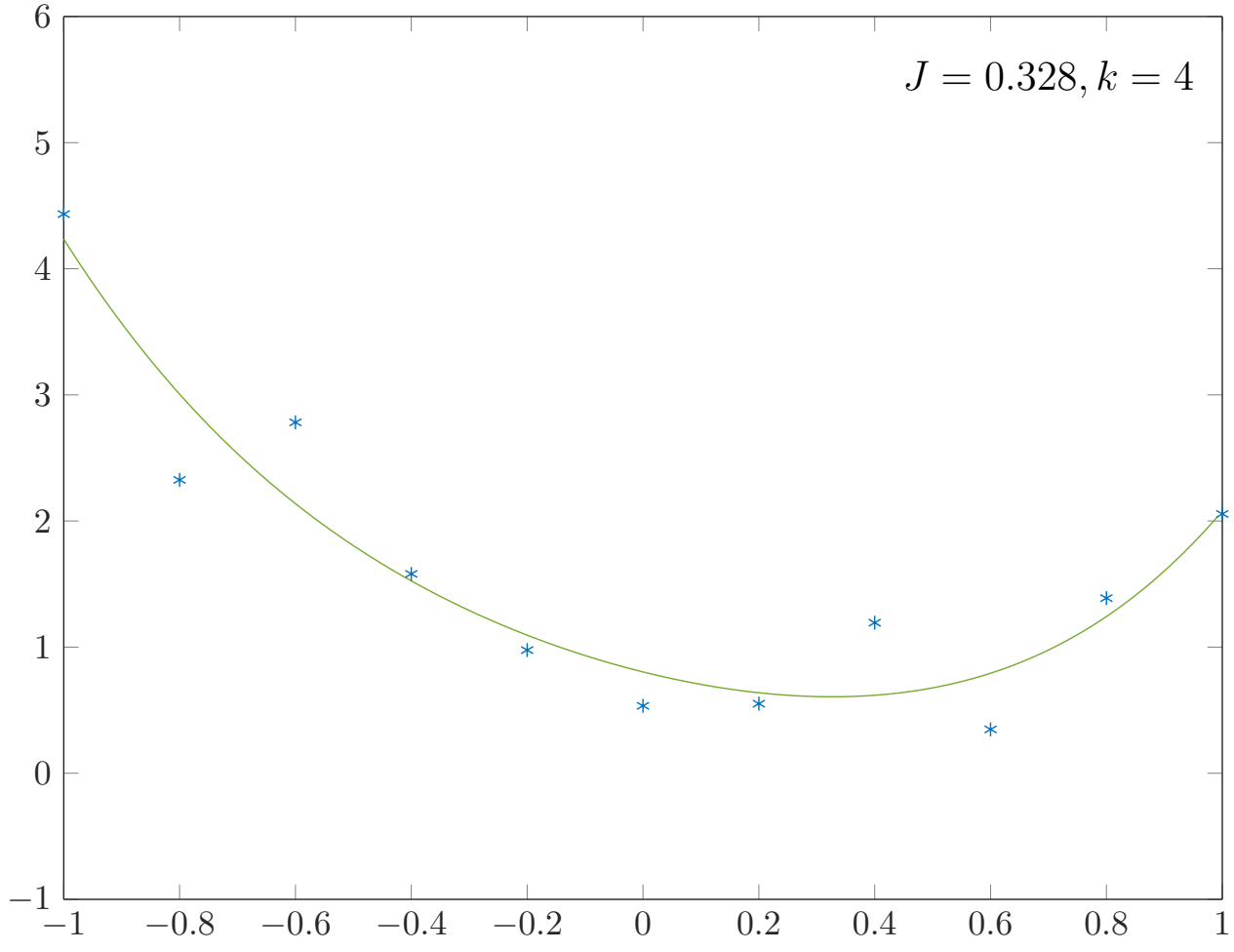
Nonaffine example

- fit polynomial of degree k to data using LS (J is cost):



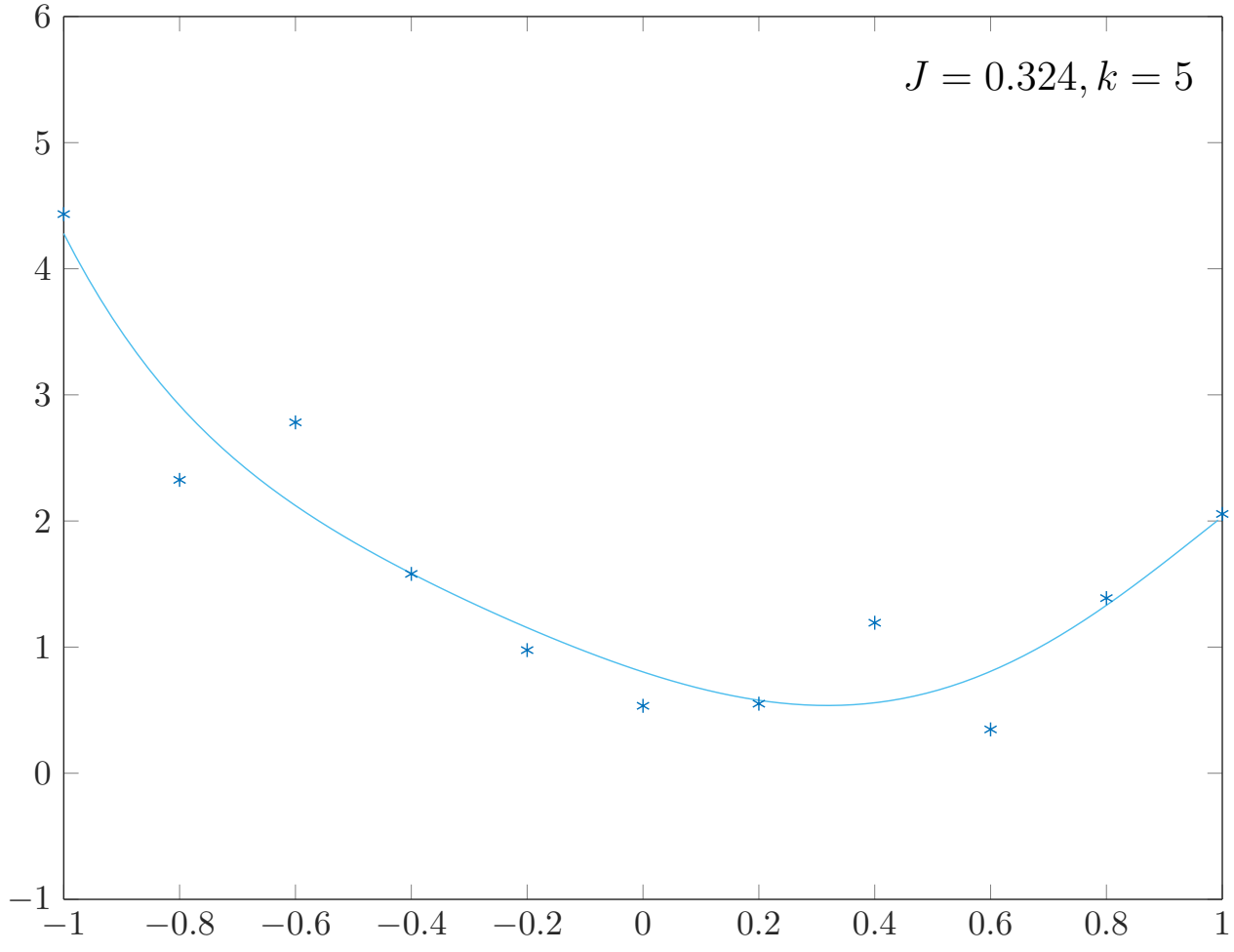
Nonaffine example

- fit polynomial of degree k to data using LS (J is cost):



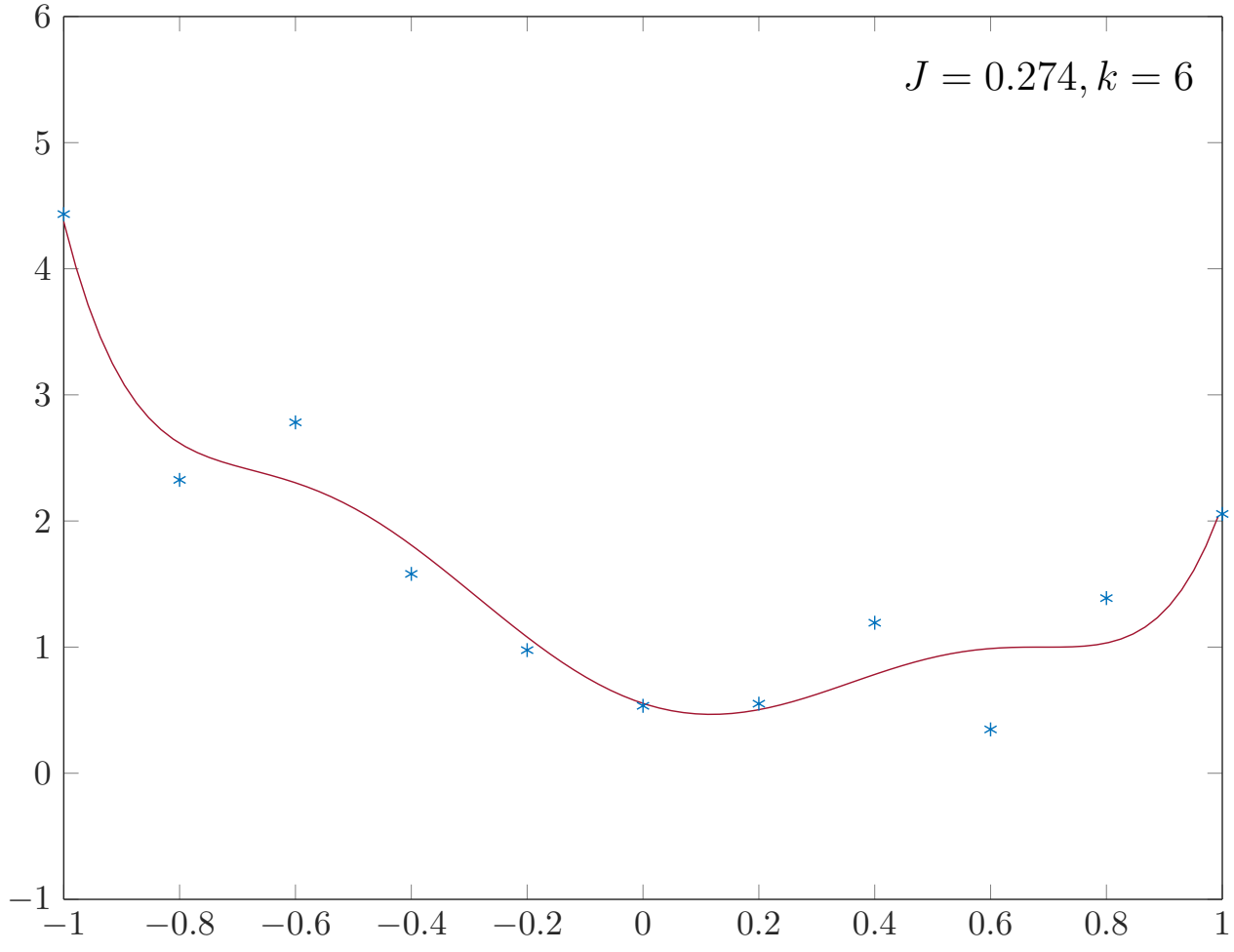
Nonaffine example

- fit polynomial of degree k to data using LS (J is cost):



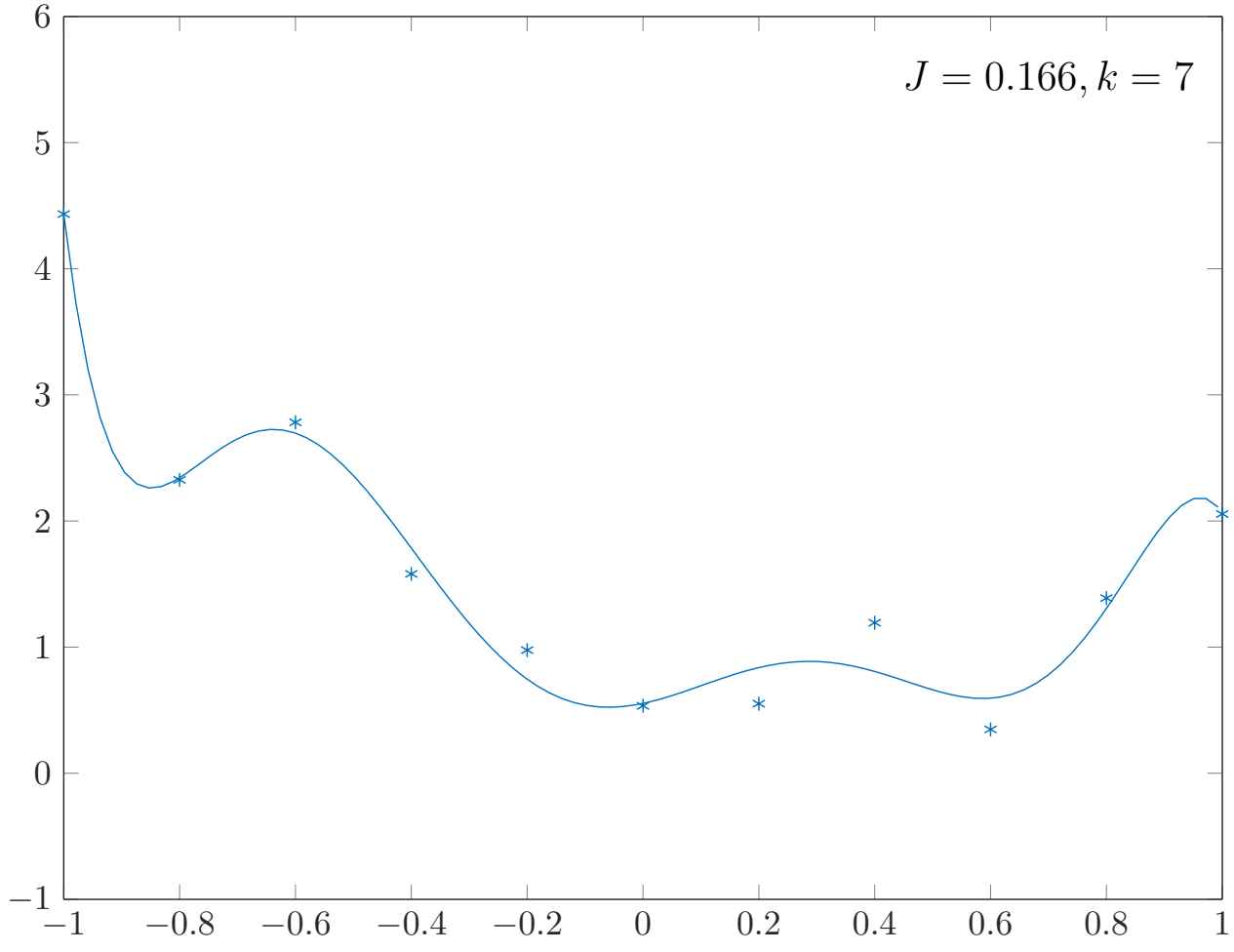
Nonaffine example

- fit polynomial of degree k to data using LS (J is cost):



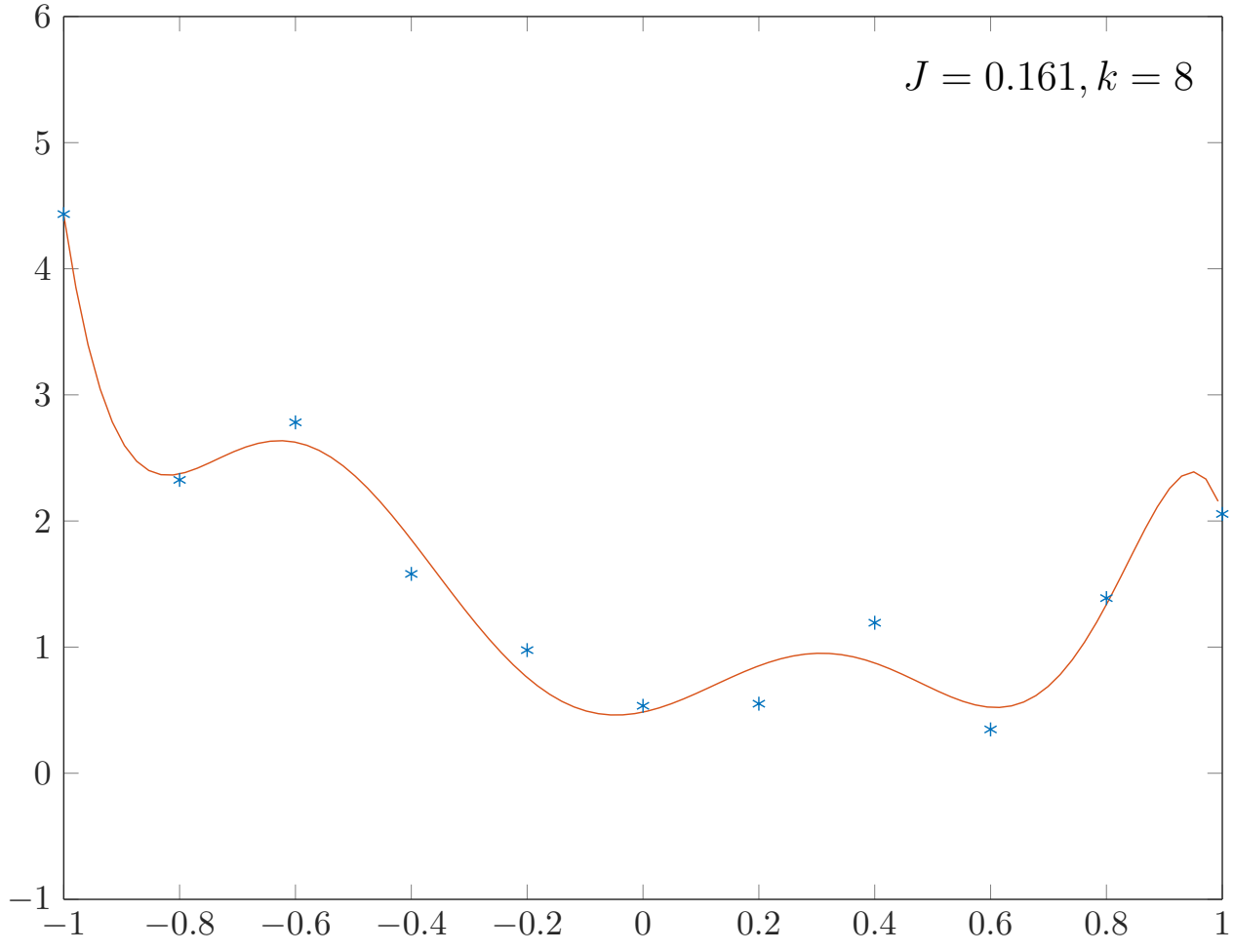
Nonaffine example

- fit polynomial of degree k to data using LS (J is cost):



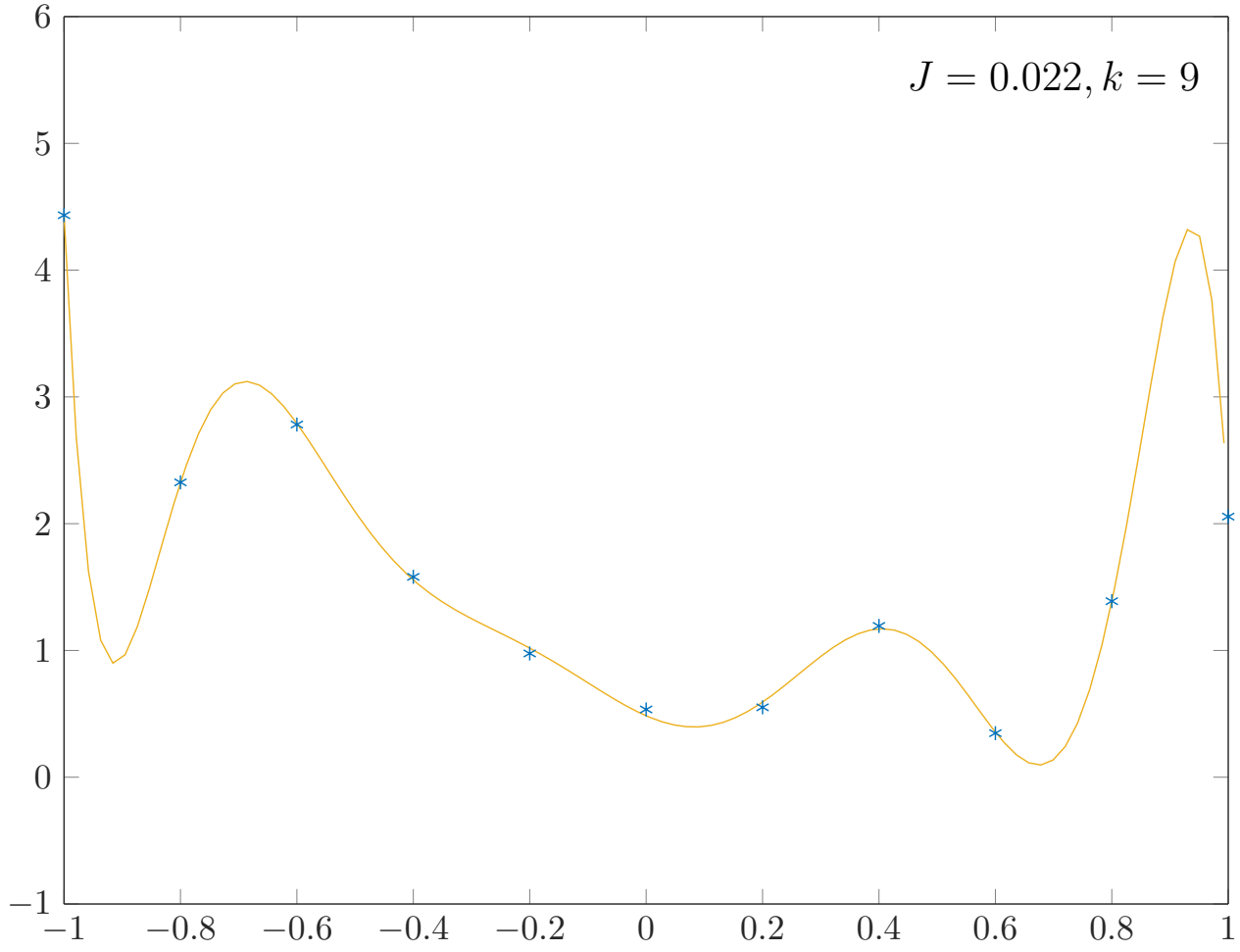
Nonaffine example

- fit polynomial of degree k to data using LS (J is cost):



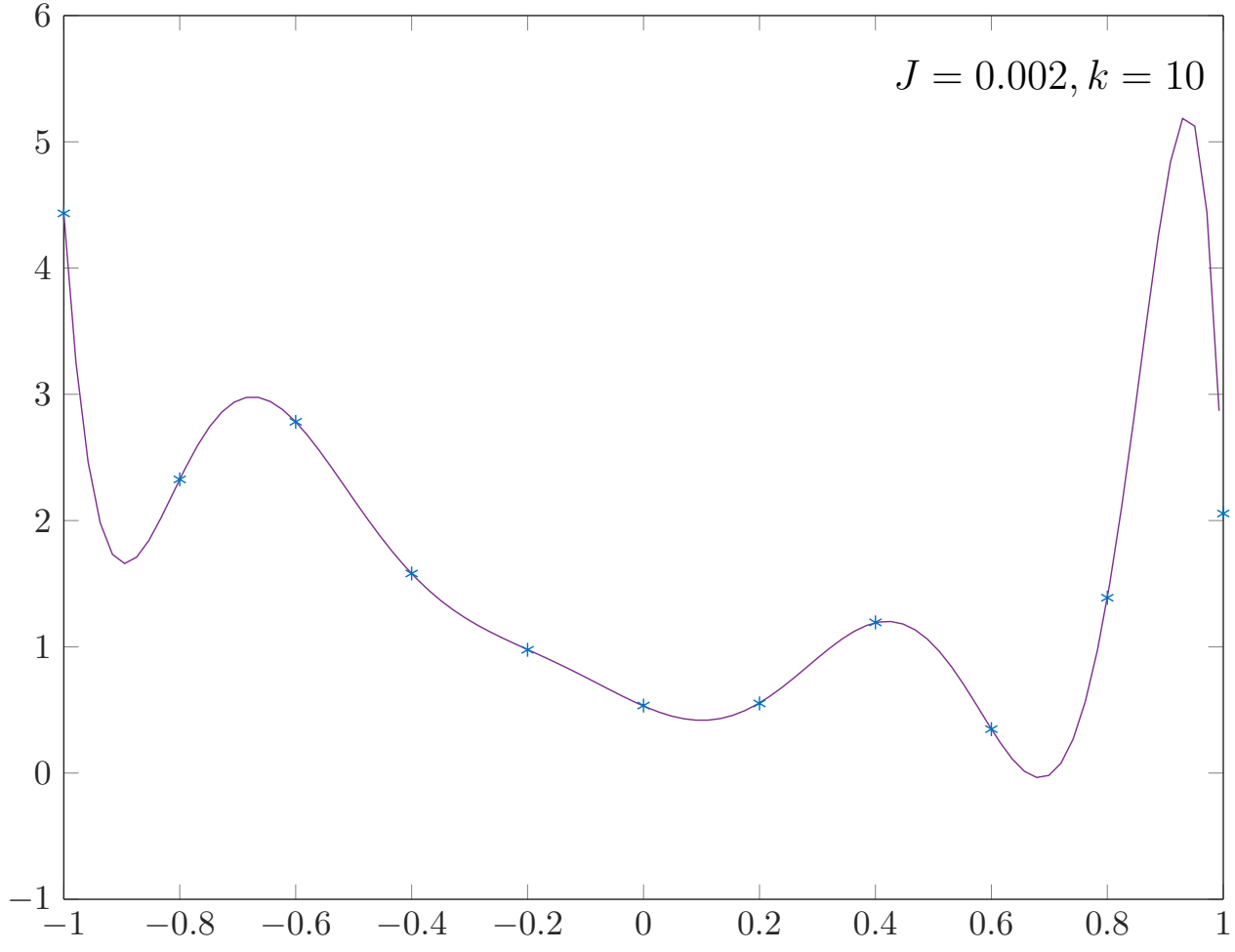
Nonaffine example

- fit polynomial of degree k to data using LS (J is cost):



Nonaffine example

- fit polynomial of degree k to data using LS (J is cost):

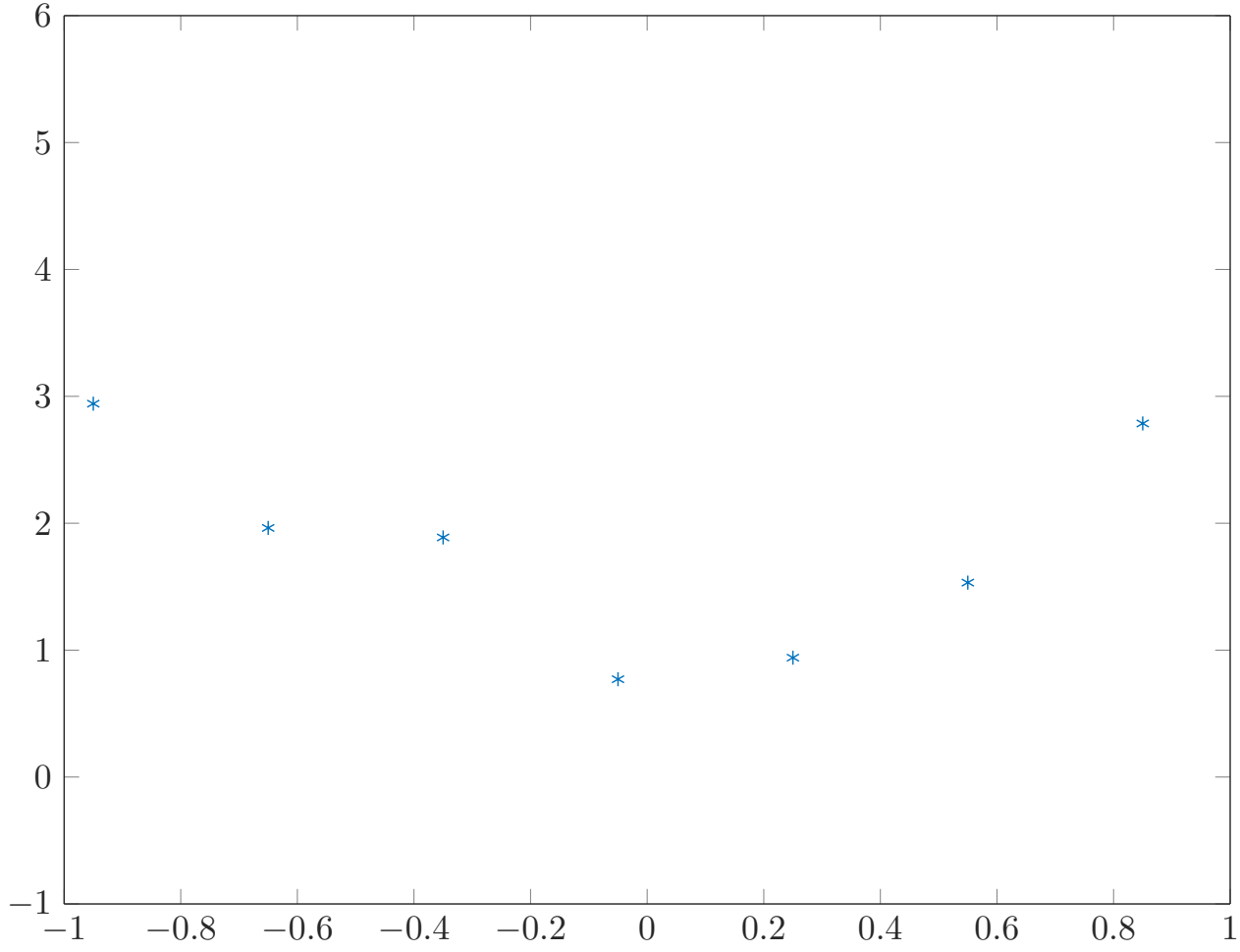


Evaluating model

- how to evaluate what model order to select?
- cross validation, train polynomial on subset of full data
- keep rest of data (30%) to validate the model

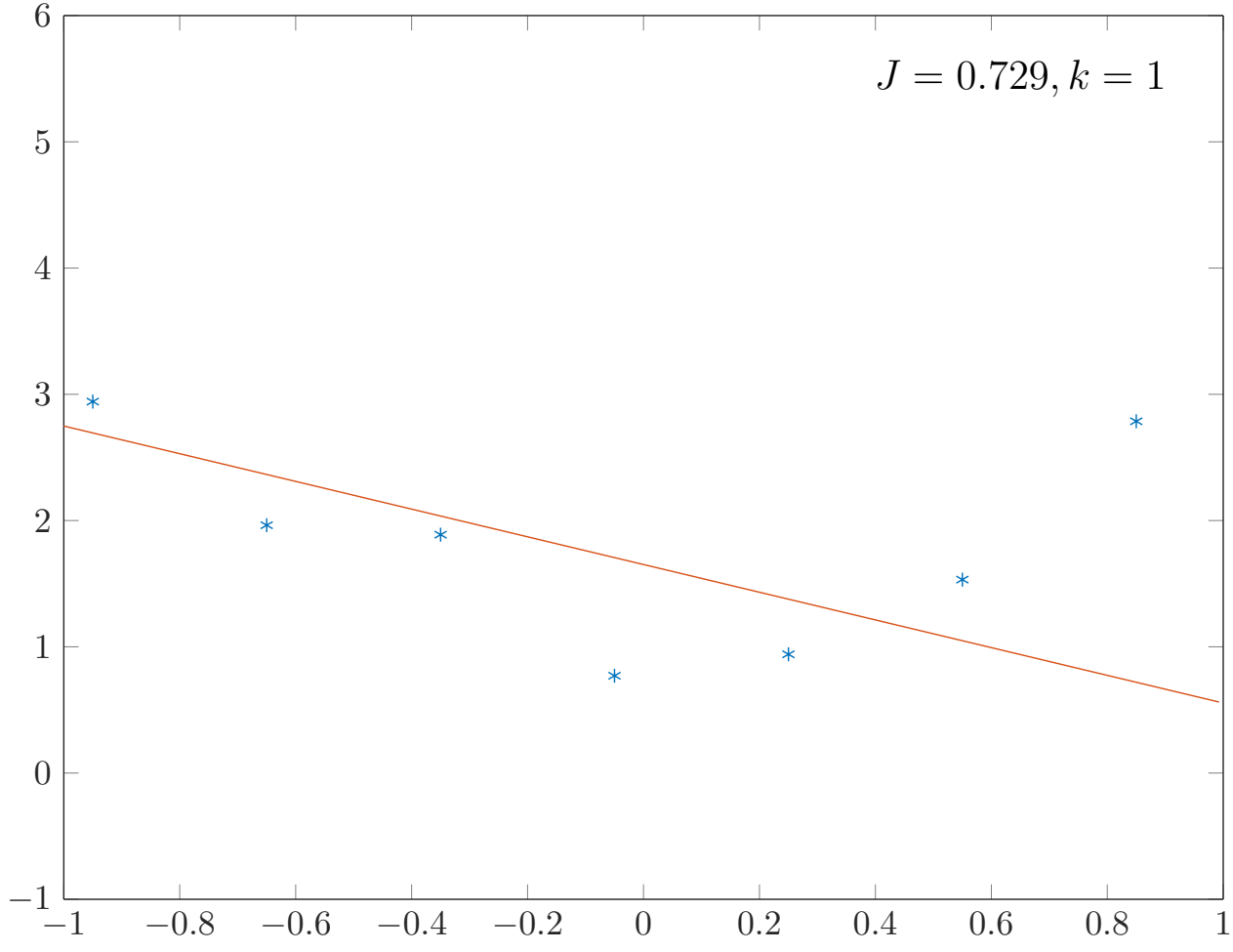
Validate example

- how does fitted polynomial explain test data (J validation cost)?



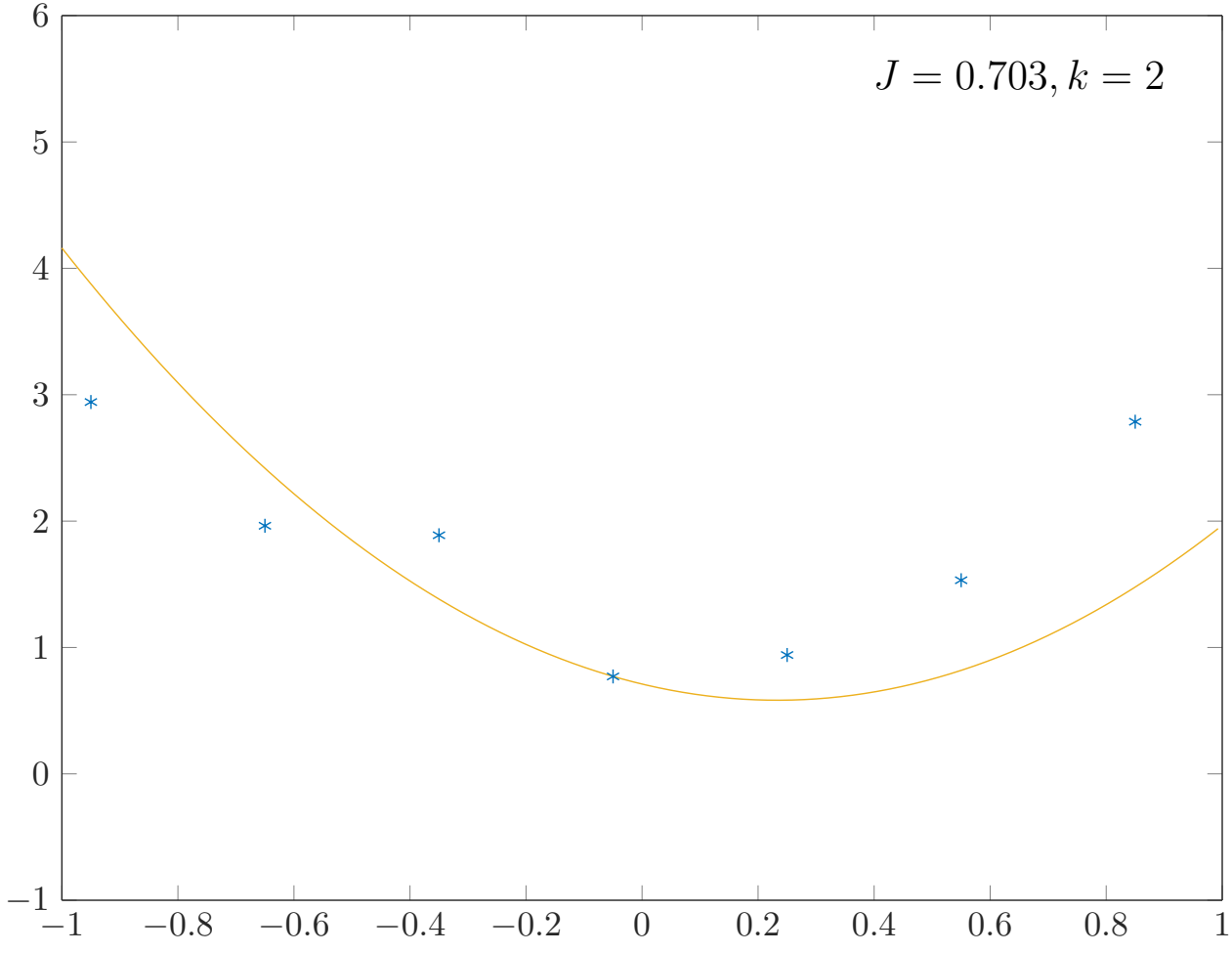
Validate example

- how does fitted polynomial explain test data (J validation cost)?



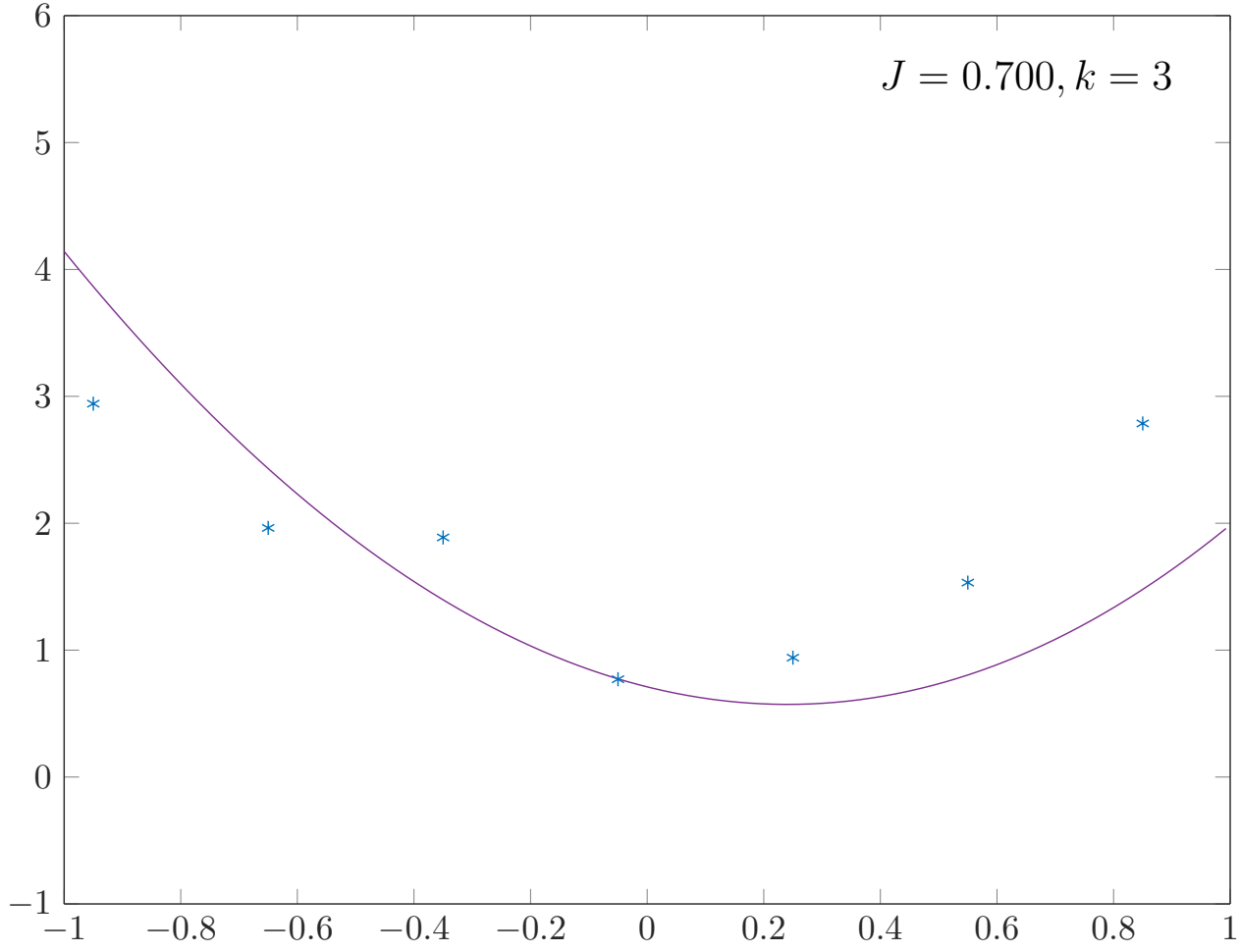
Validate example

- how does fitted polynomial explain test data (J validation cost)?



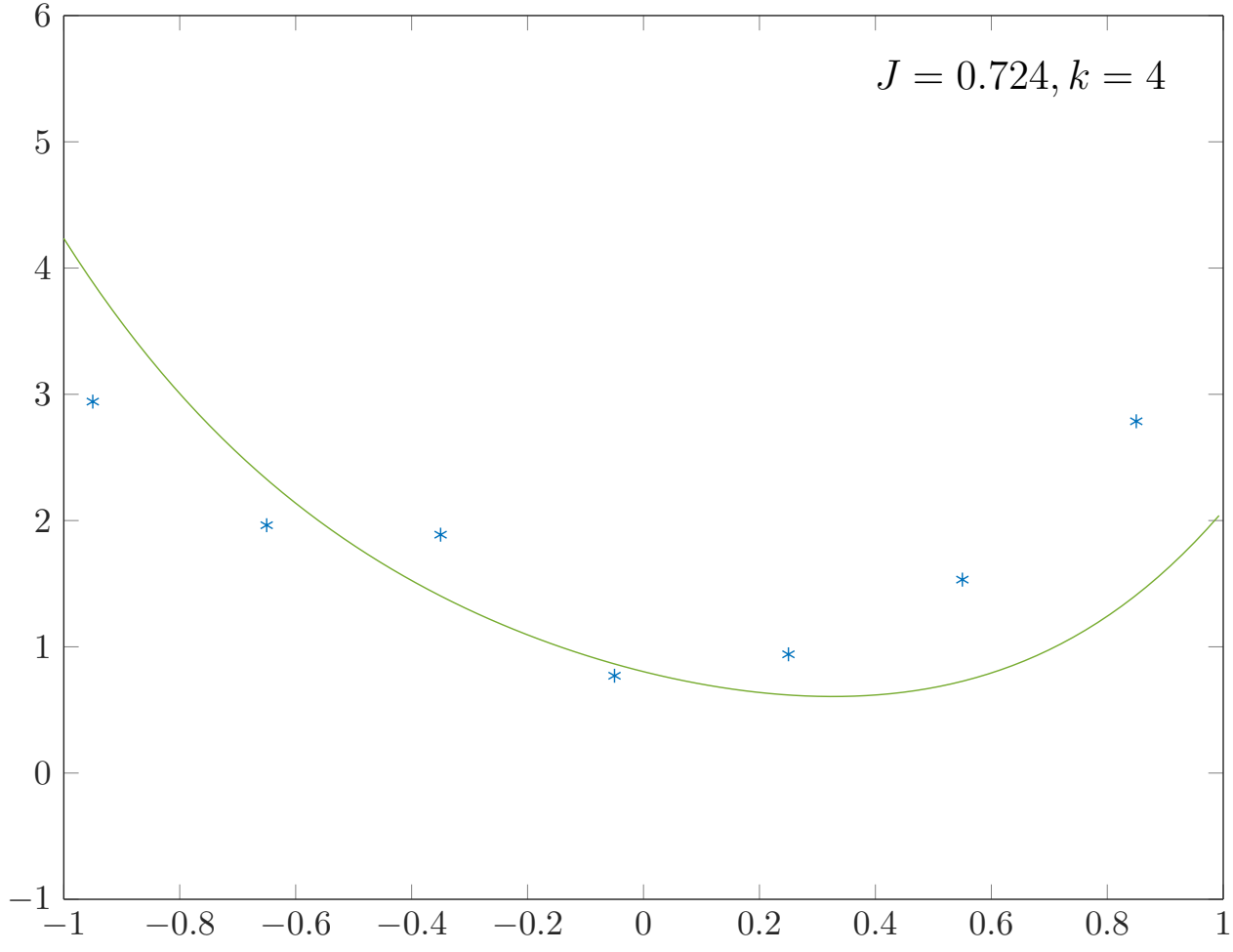
Validate example

- how does fitted polynomial explain test data (J validation cost)?



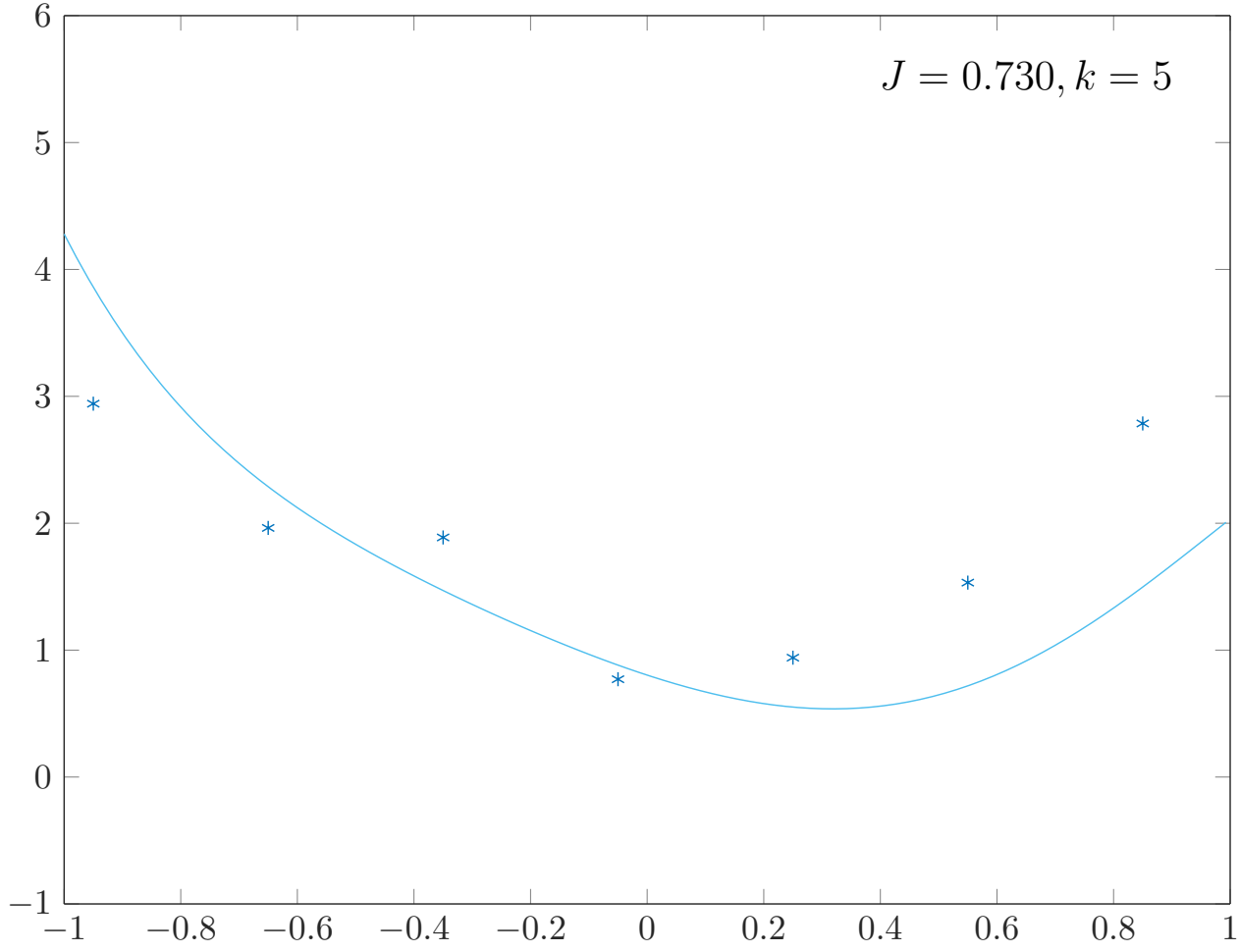
Validate example

- how does fitted polynomial explain test data (J validation cost)?



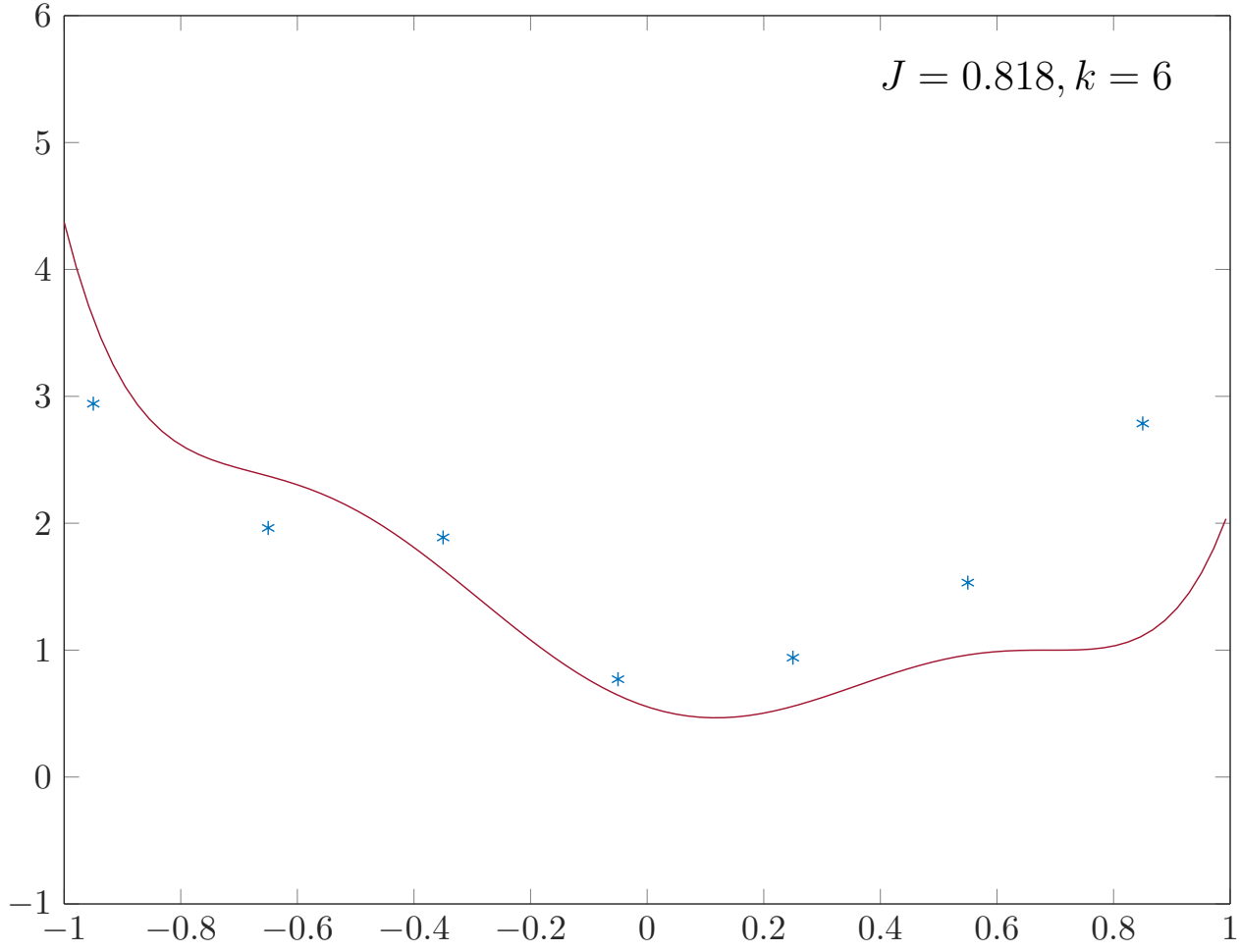
Validate example

- how does fitted polynomial explain test data (J validation cost)?



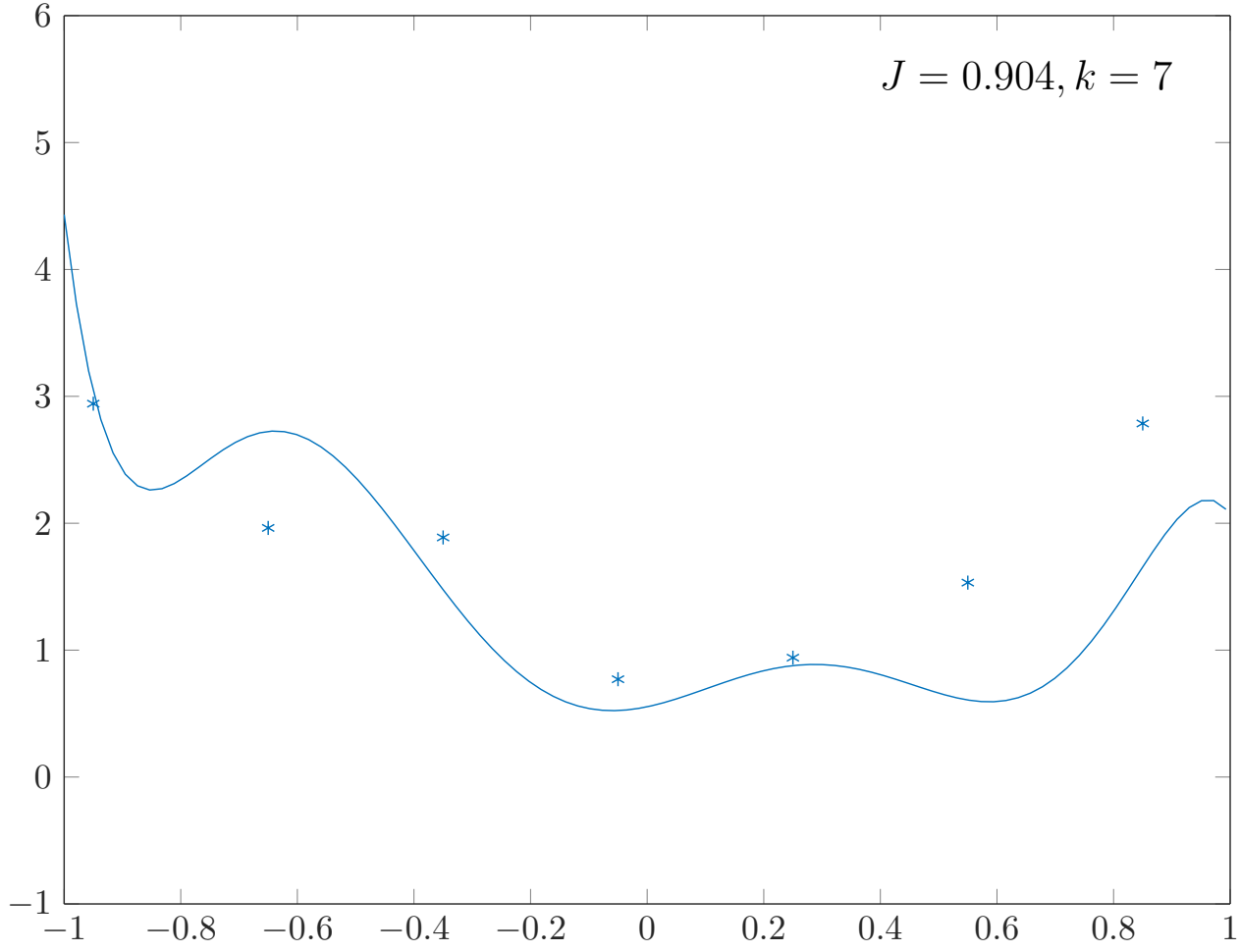
Validate example

- how does fitted polynomial explain test data (J validation cost)?



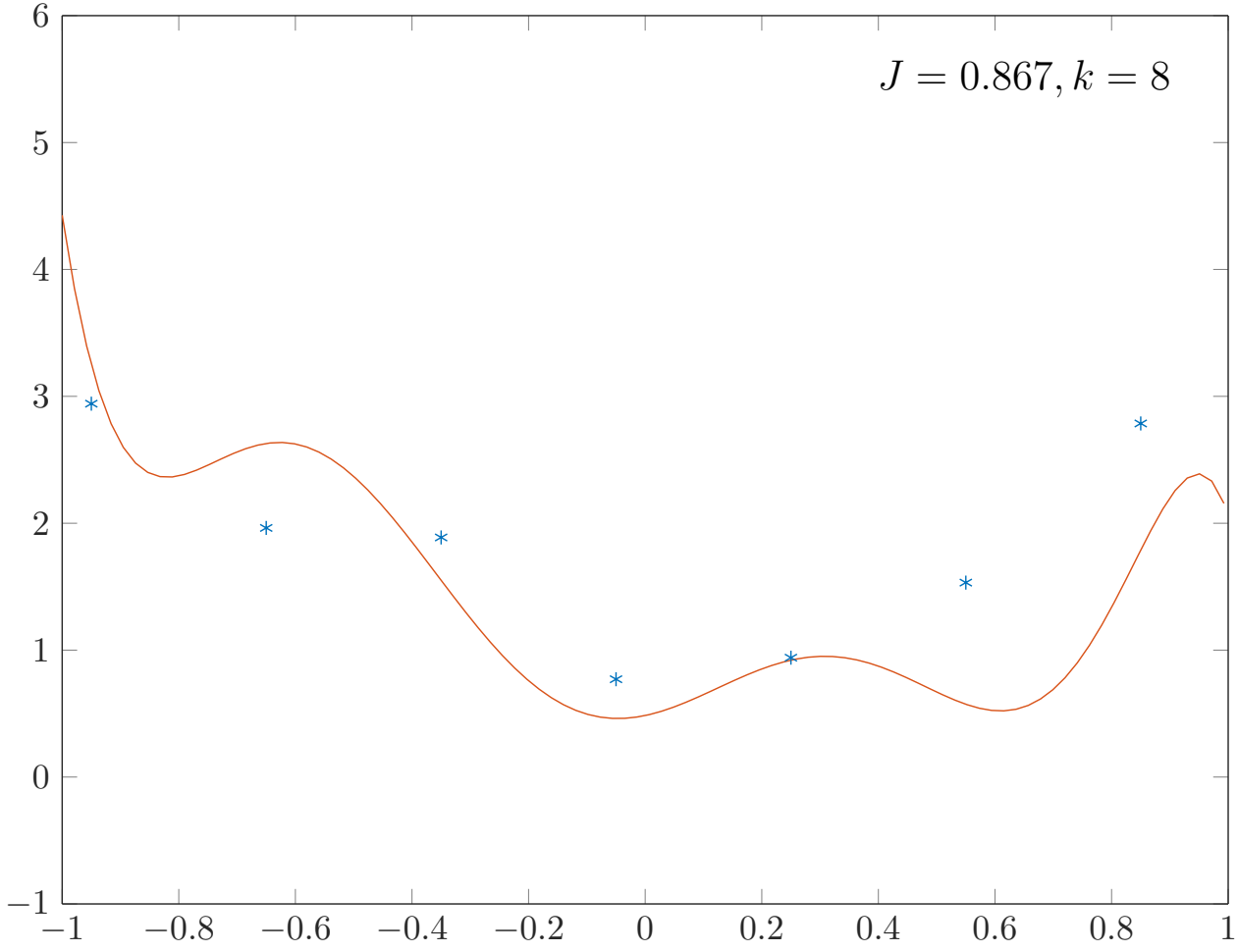
Validate example

- how does fitted polynomial explain test data (J validation cost)?



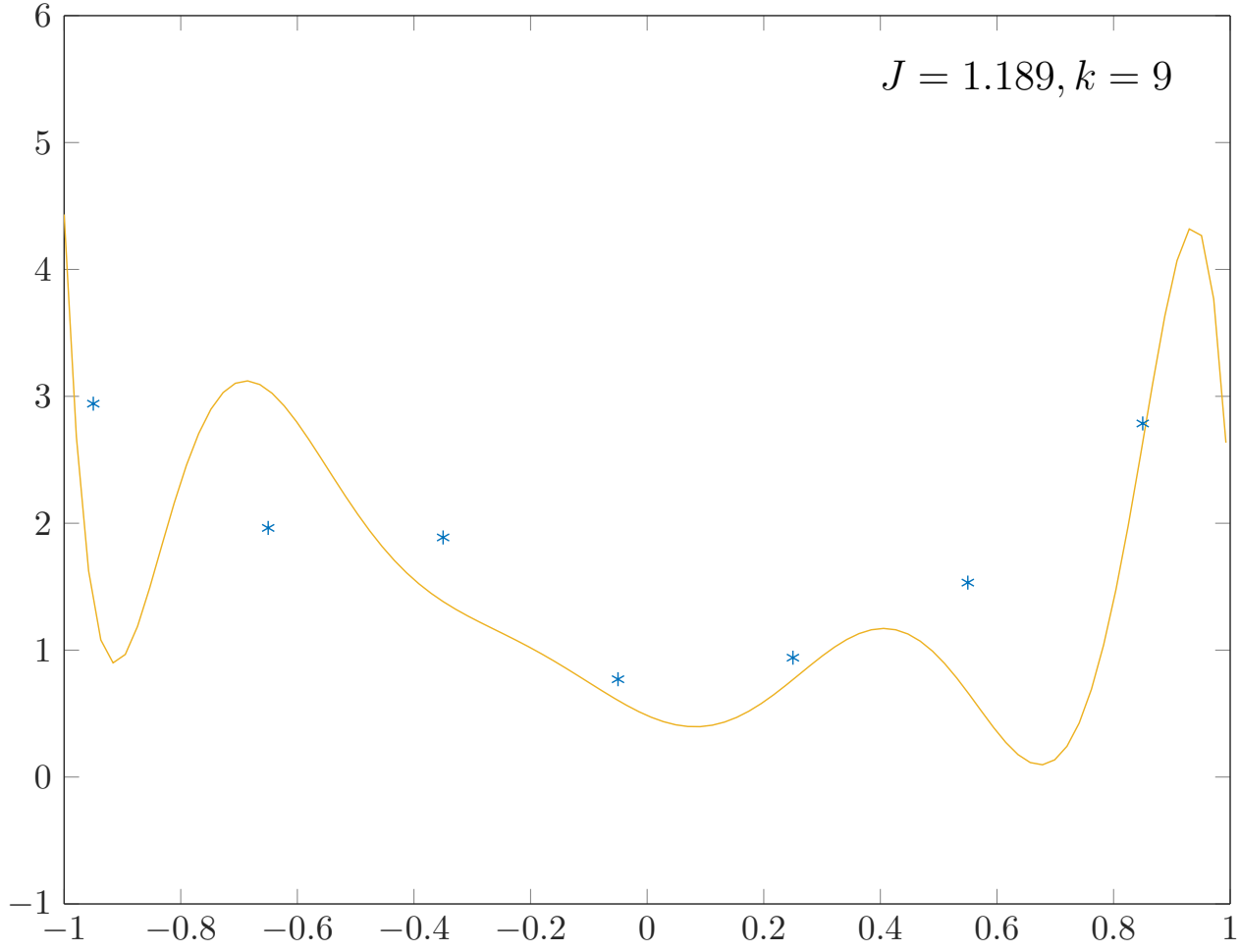
Validate example

- how does fitted polynomial explain test data (J validation cost)?



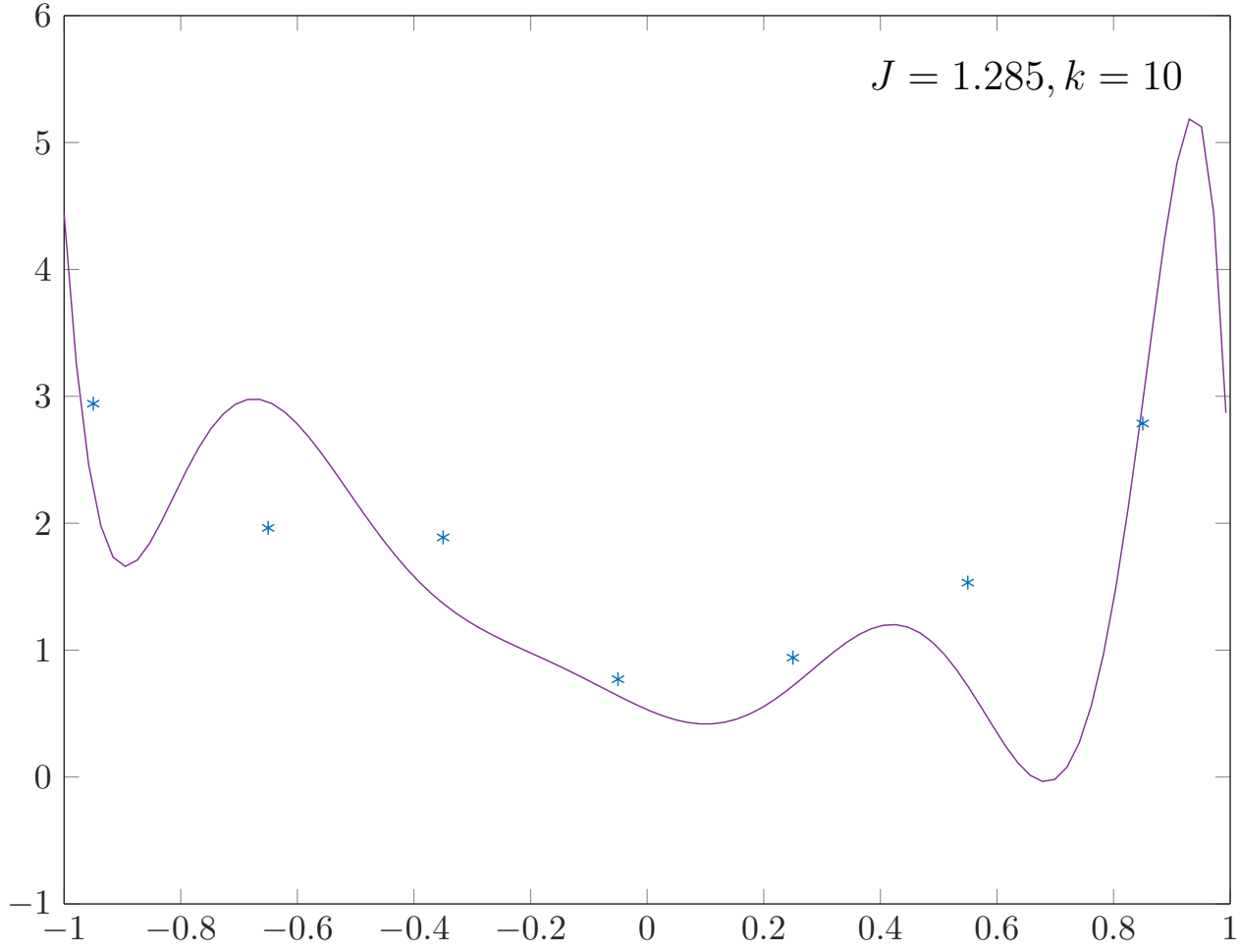
Validate example

- how does fitted polynomial explain test data (J validation cost)?



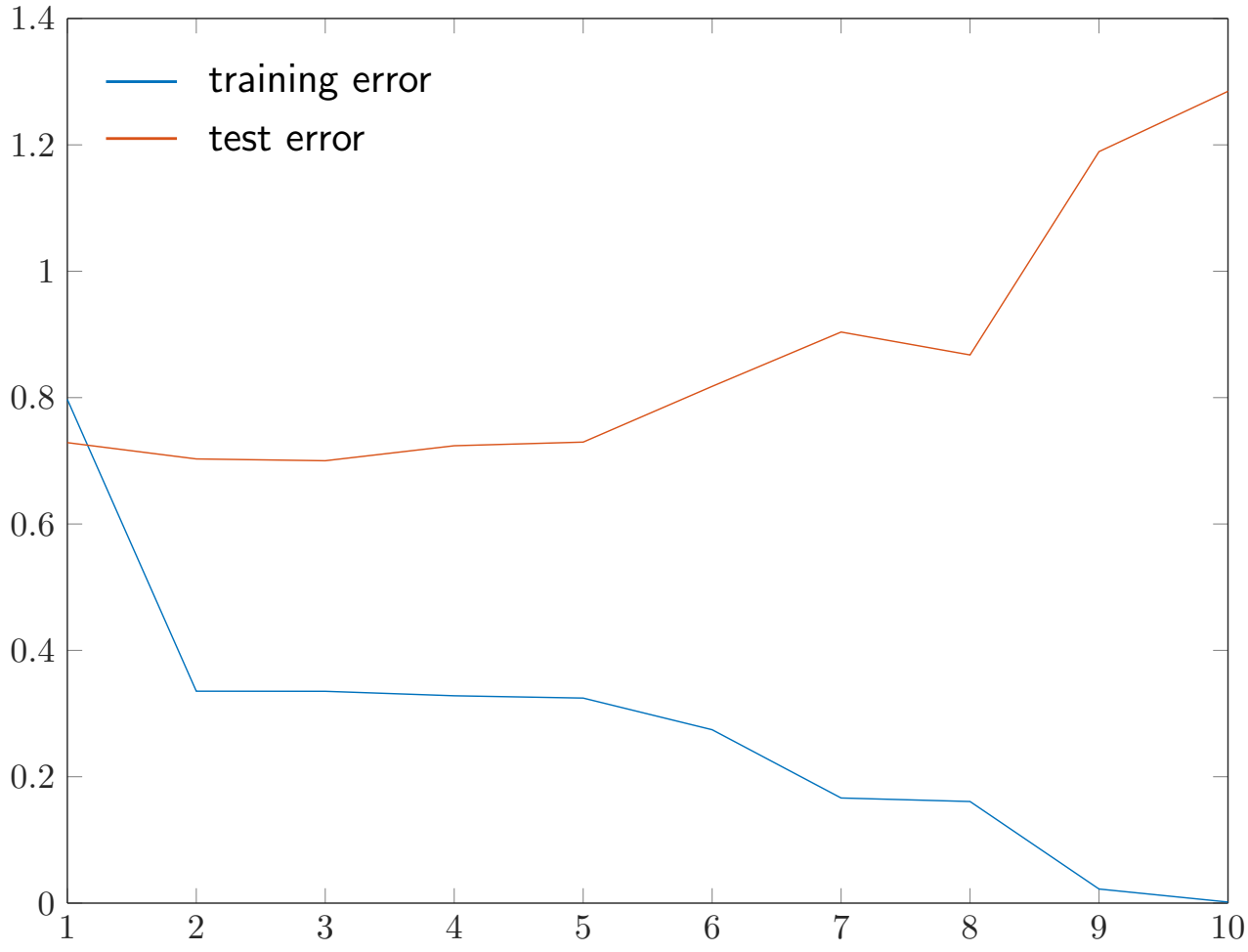
Validate example

- how does fitted polynomial explain test data (J validation cost)?



Tradeoff

- training error and test error vs polynomial order:



Characterizing the LS solution

- by assumption, data matrix X has full column rank
- optimal s satisfies (let gradient be zero)

$$X^T X s - X^T y = 0 \quad \Leftrightarrow \quad s = (X^T X)^{-1} X^T y$$

- let $X^T X = U \Sigma U^T$, with $\Sigma = \mathrm{diag}(\sigma_1, \ldots, \sigma_n)$ and U unitary $U^T = U^{-1}$
- then solution to problem is

$$\begin{aligned} s &= (U \Sigma U^T)^{-1} X^T y = U \Sigma^{-1} U^T X^T y \\ &= \sum_{i=1}^{n} \frac{1}{\sigma_i} \langle u_i, X^T y \rangle u_i \end{aligned}$$

- elements with small singular values in $X^T X$ amplified
- might amplify noise in those directions
- can give rise to overfitting if model of too high complexity

Tikhonov regularization

- what if we instead solve regularized Least squares problem

$$\text{minimize } \|Xs - y\|^2 + \lambda \|s\|^2$$

- optimal s satisfies (let gradient be zero)

$$(X^T X + \lambda I)s - X^T y = 0 \quad \Leftrightarrow \quad s = (X^T X + \lambda I)^{-1} X^T y$$

- let $X^T X = U \Sigma U^T$, where $\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_n)$ and $U^T = U^{-1}$
- then $I = U(\lambda I)U^T$ and solution to problem is

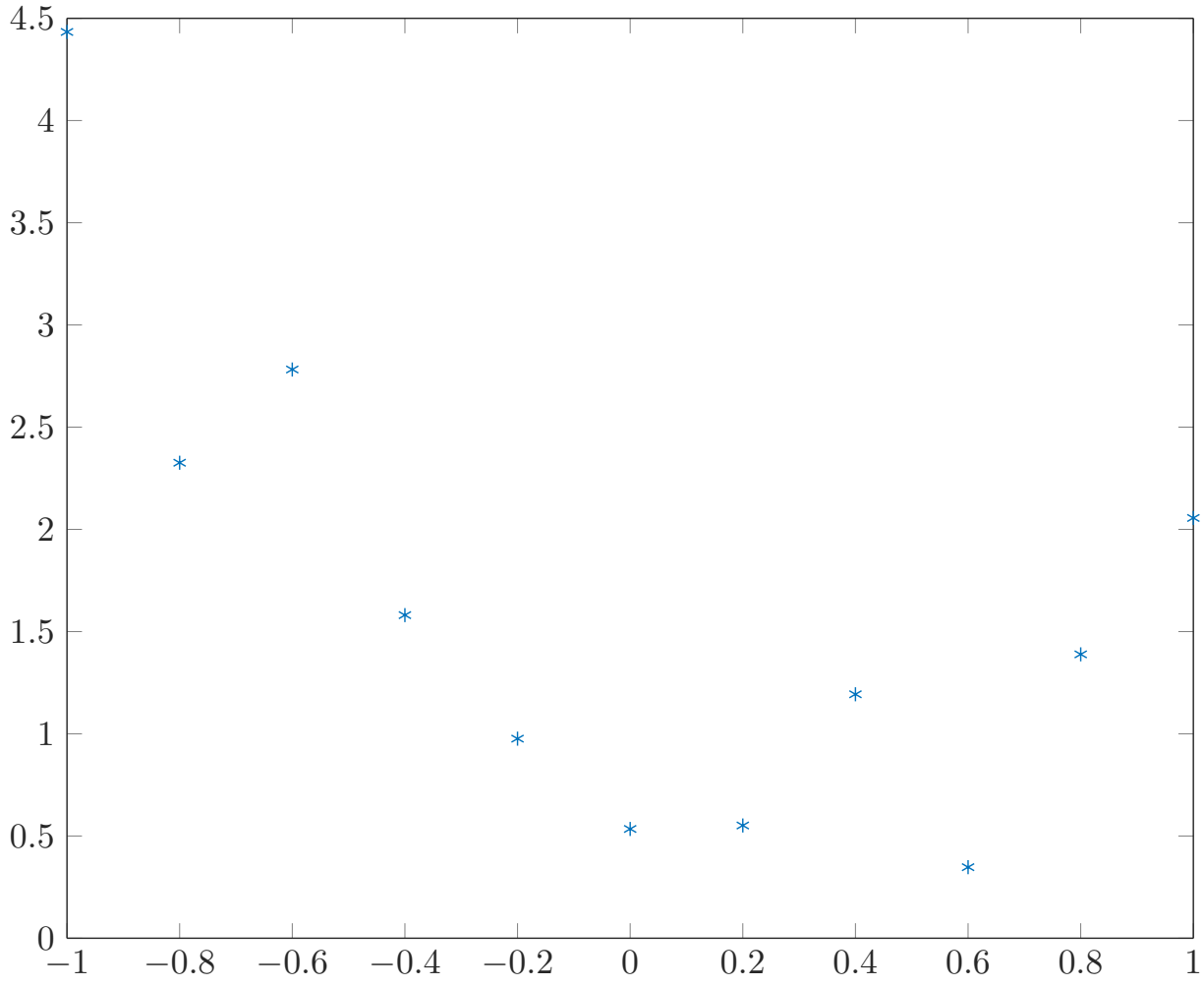
$$s = (U(\Sigma + \lambda I)U^T)^{-1} X^T y = U(\Sigma + \lambda I)^{-1} U^T X^T y$$

$$= \sum_{i=1}^{n} \tfrac{1}{\sigma_i + \lambda} \langle u_i, X^T y \rangle u_i$$

- for small σ_i, factor $\approx \frac{1}{\lambda}$, for large σ_i, factor $\approx \frac{1}{\sigma_i}$
- reduced influence from small singular values in $X^T X$
  - $\Rightarrow$ reduces noise amplification in those directions
  - $\Rightarrow$ reduces overfitting when using too complex models
- (choose lambda using cross validation)

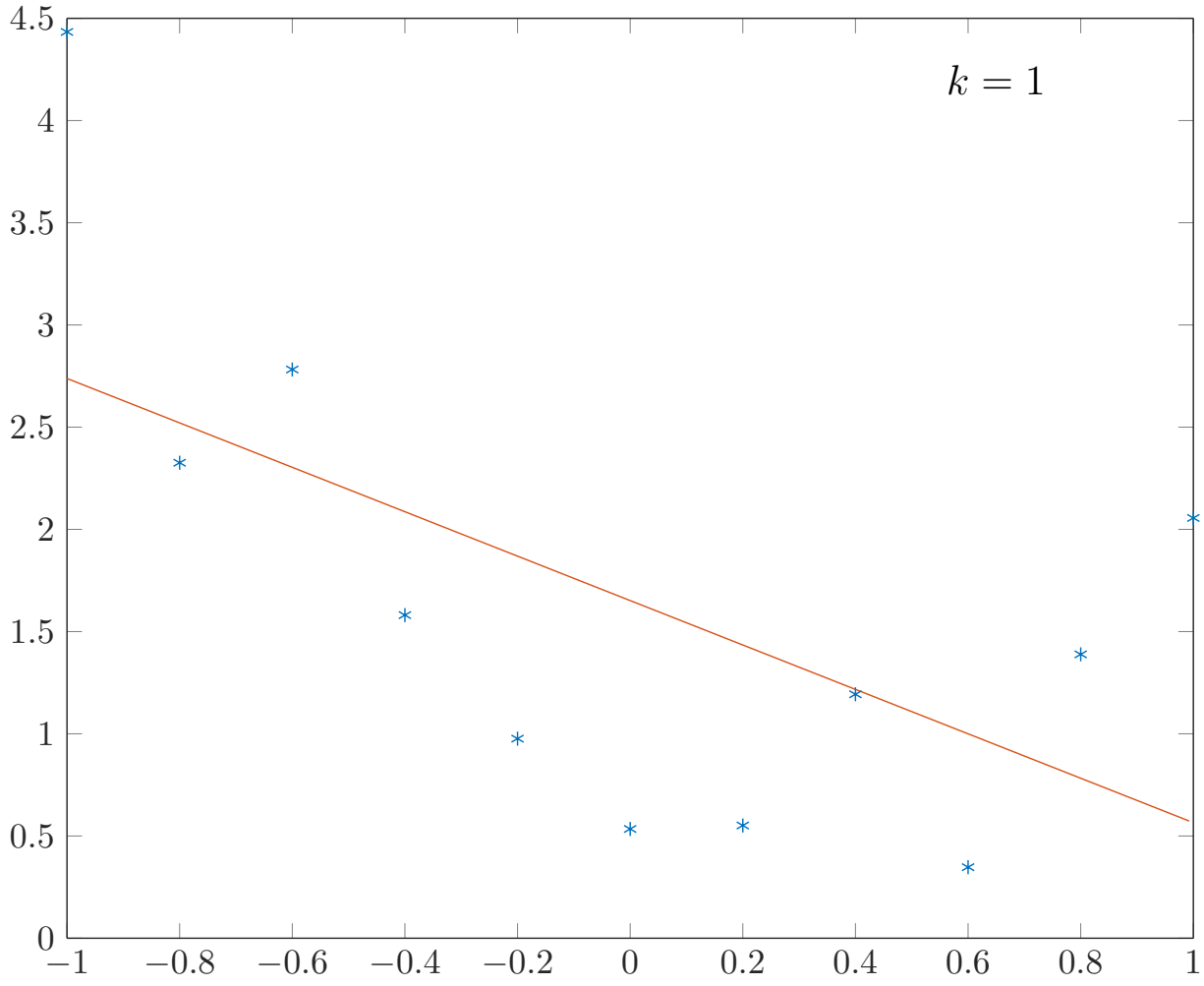
Example

- same example as before
- overfitting is reduced for (too) high complexity models



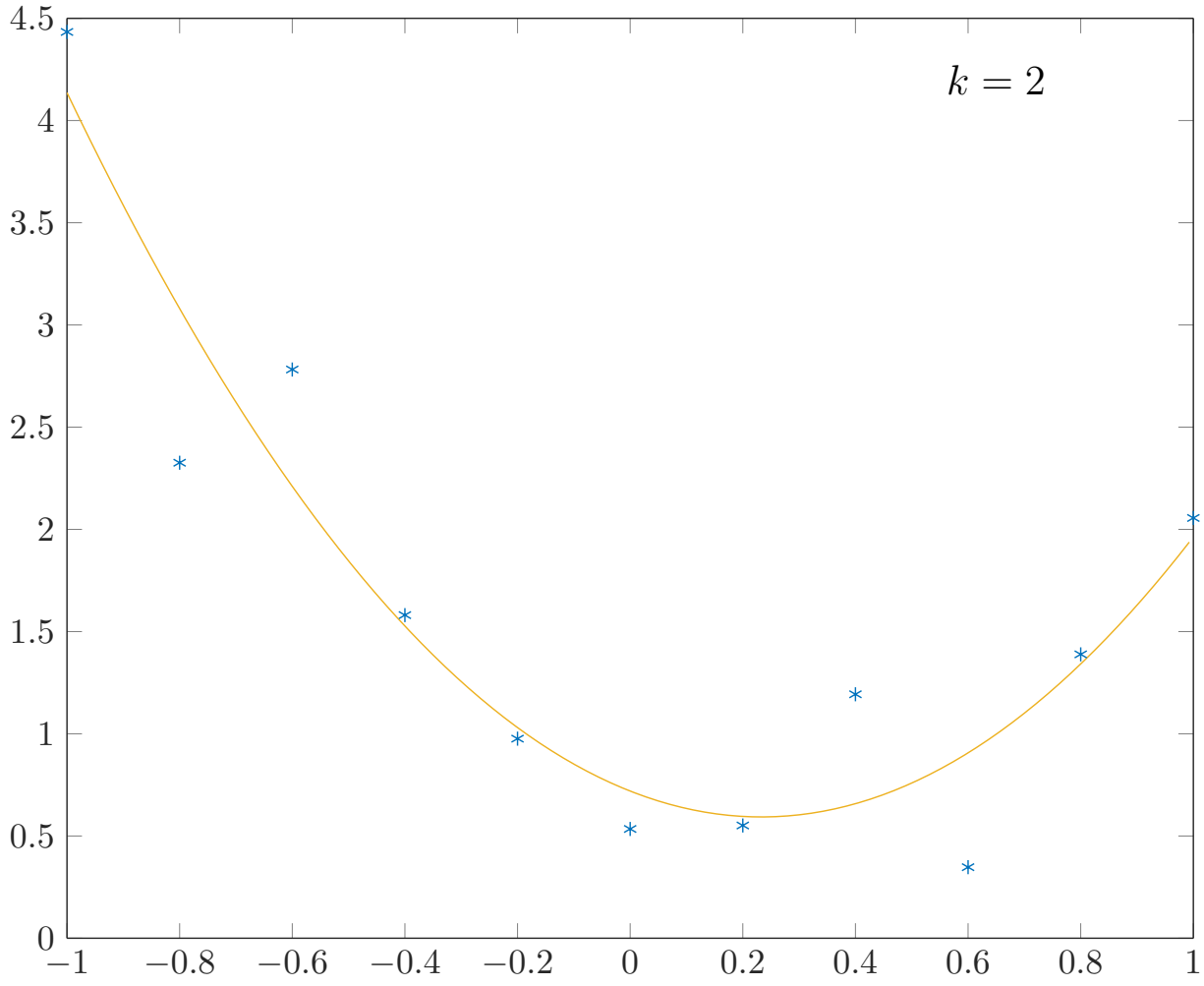
Example

- same example as before
- overfitting is reduced for (too) high complexity models



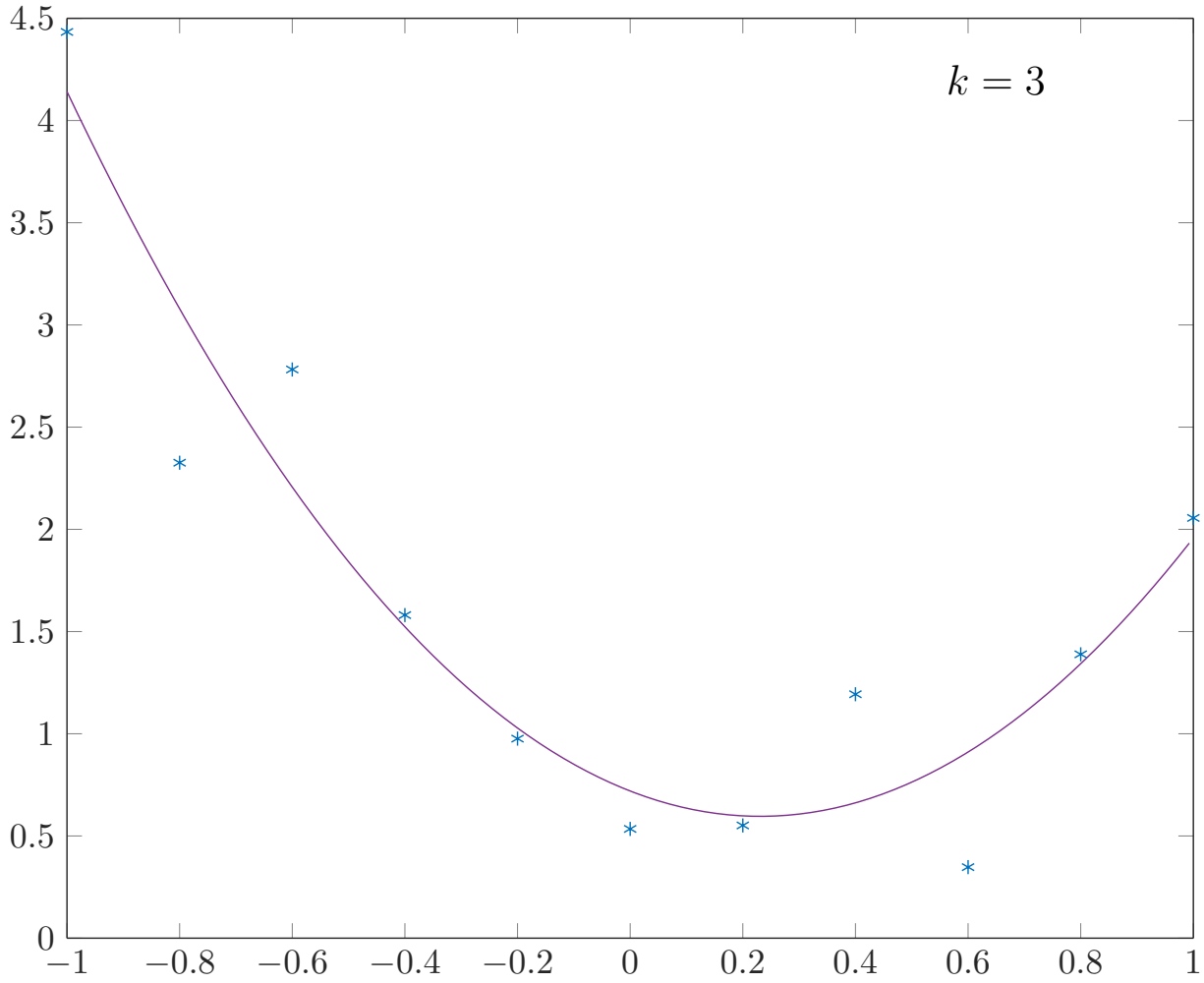
Example

- same example as before
- overfitting is reduced for (too) high complexity models



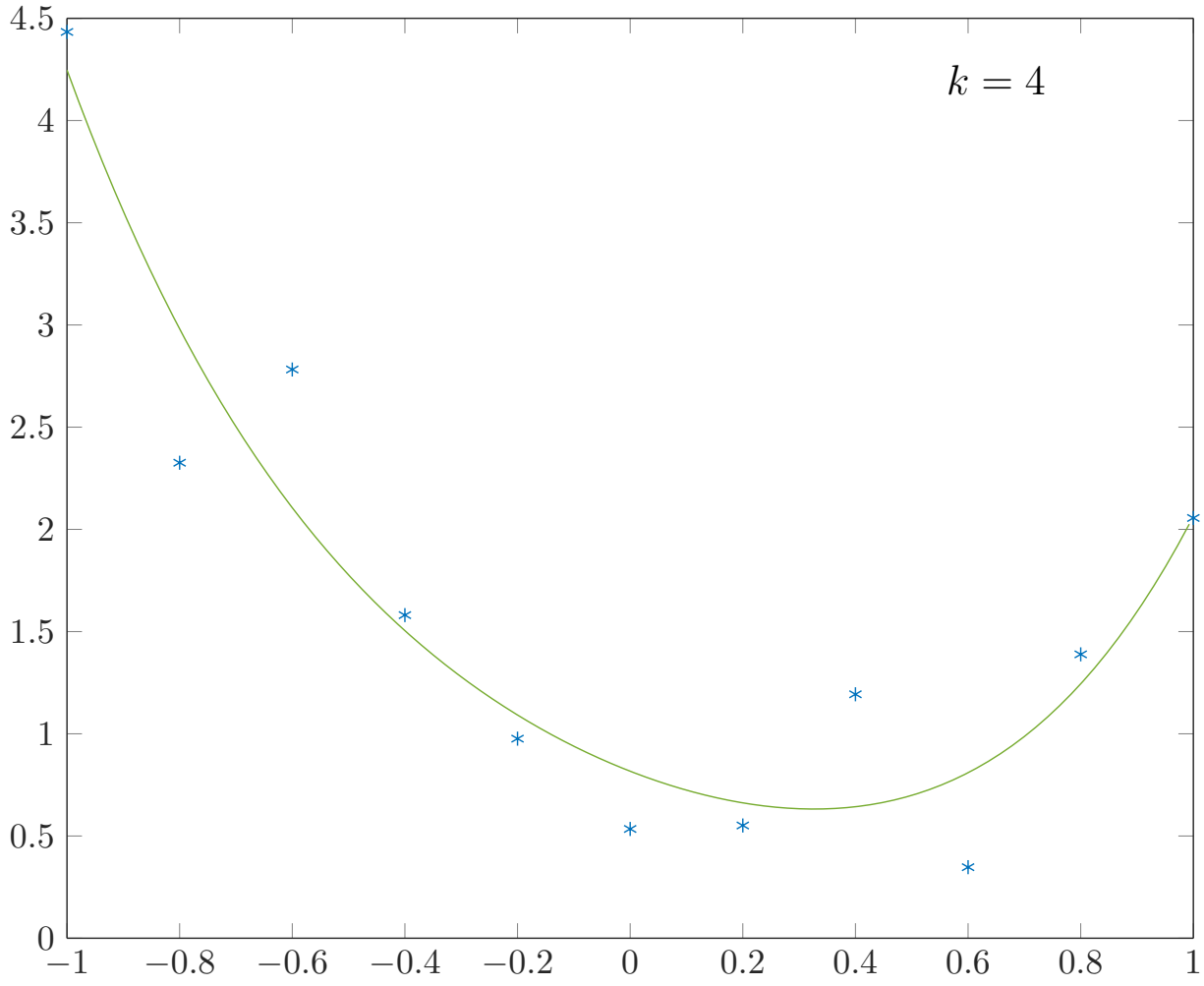
Example

- same example as before
- overfitting is reduced for (too) high complexity models



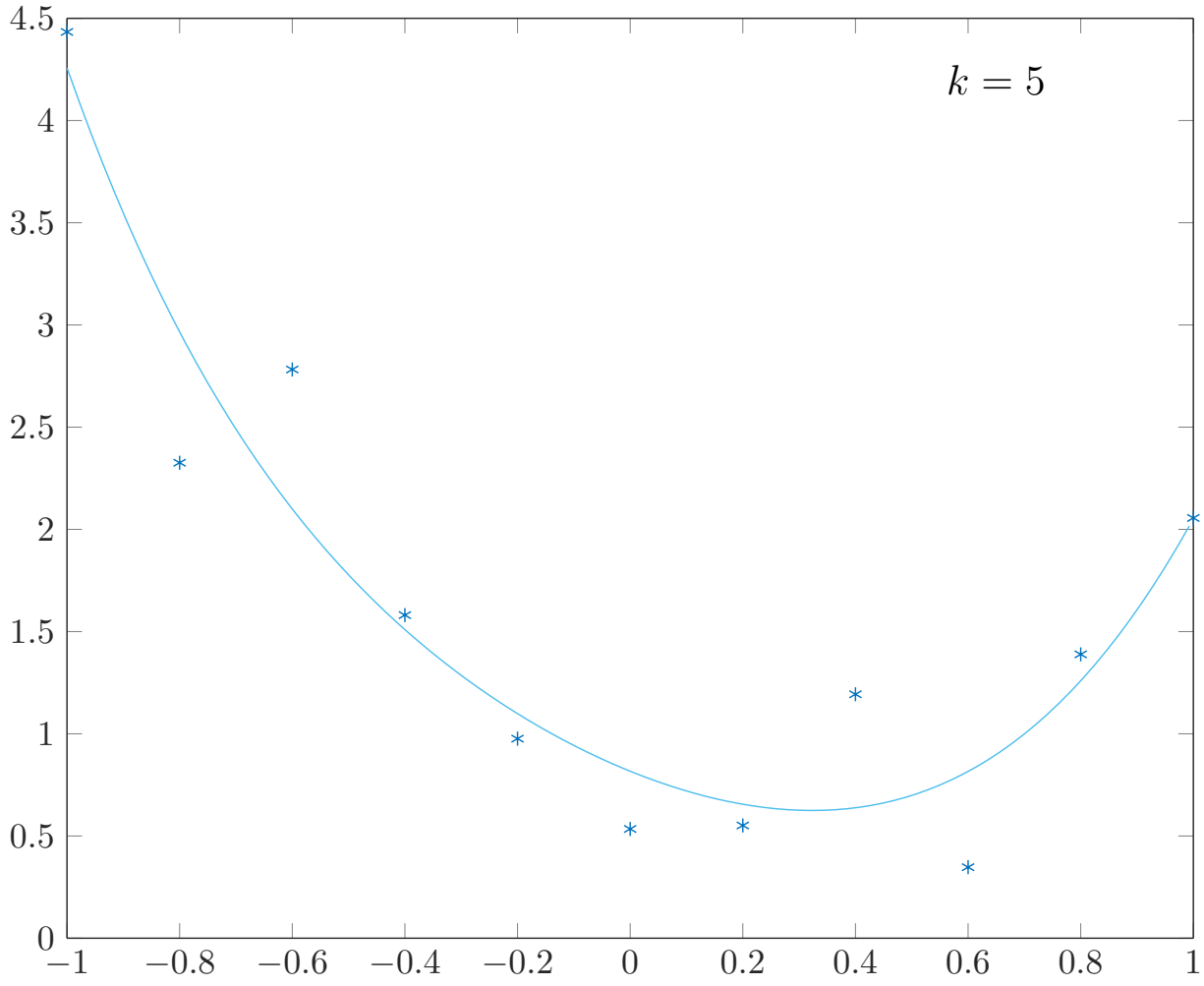
Example

- same example as before
- overfitting is reduced for (too) high complexity models



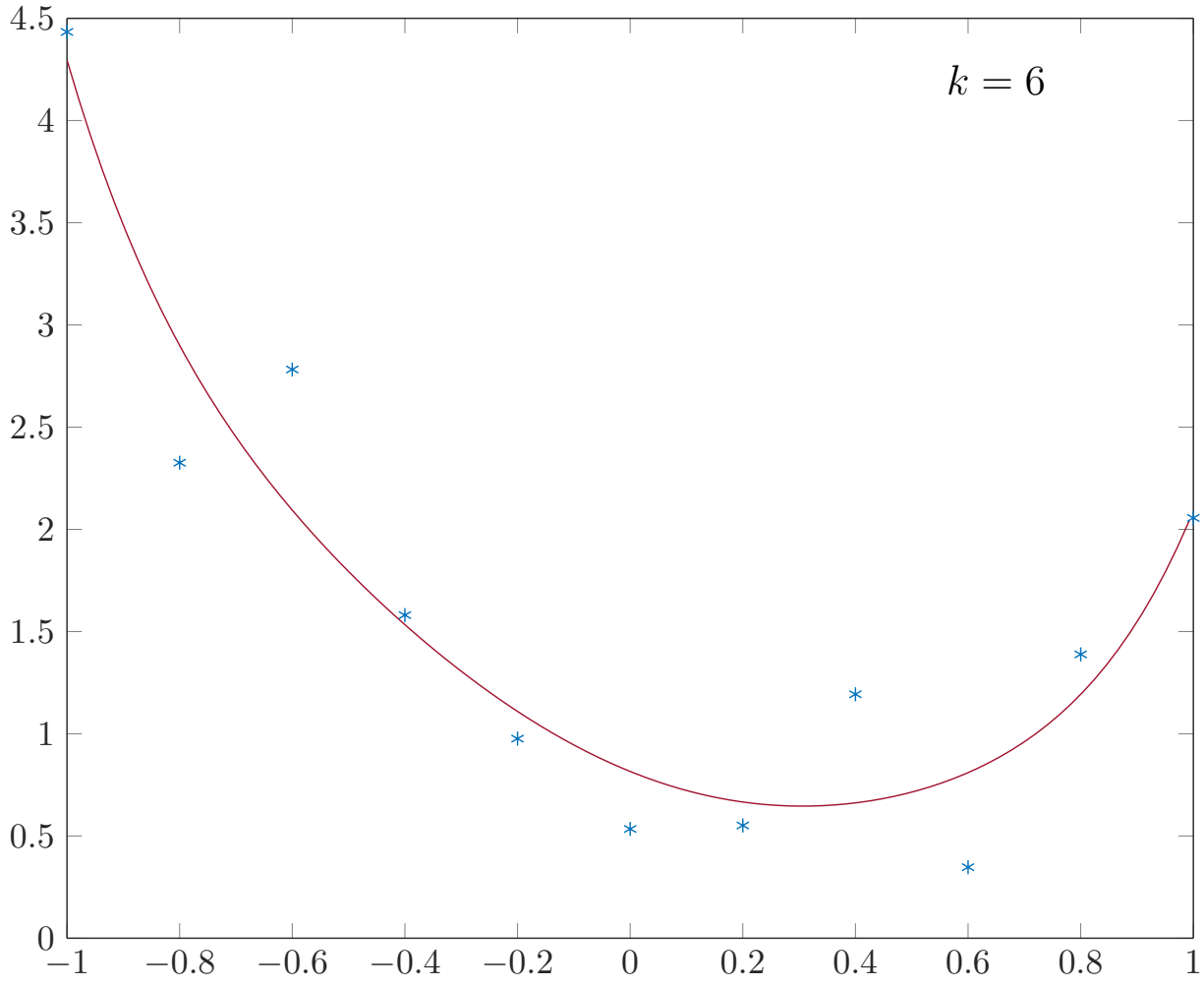
Example

- same example as before
- overfitting is reduced for (too) high complexity models



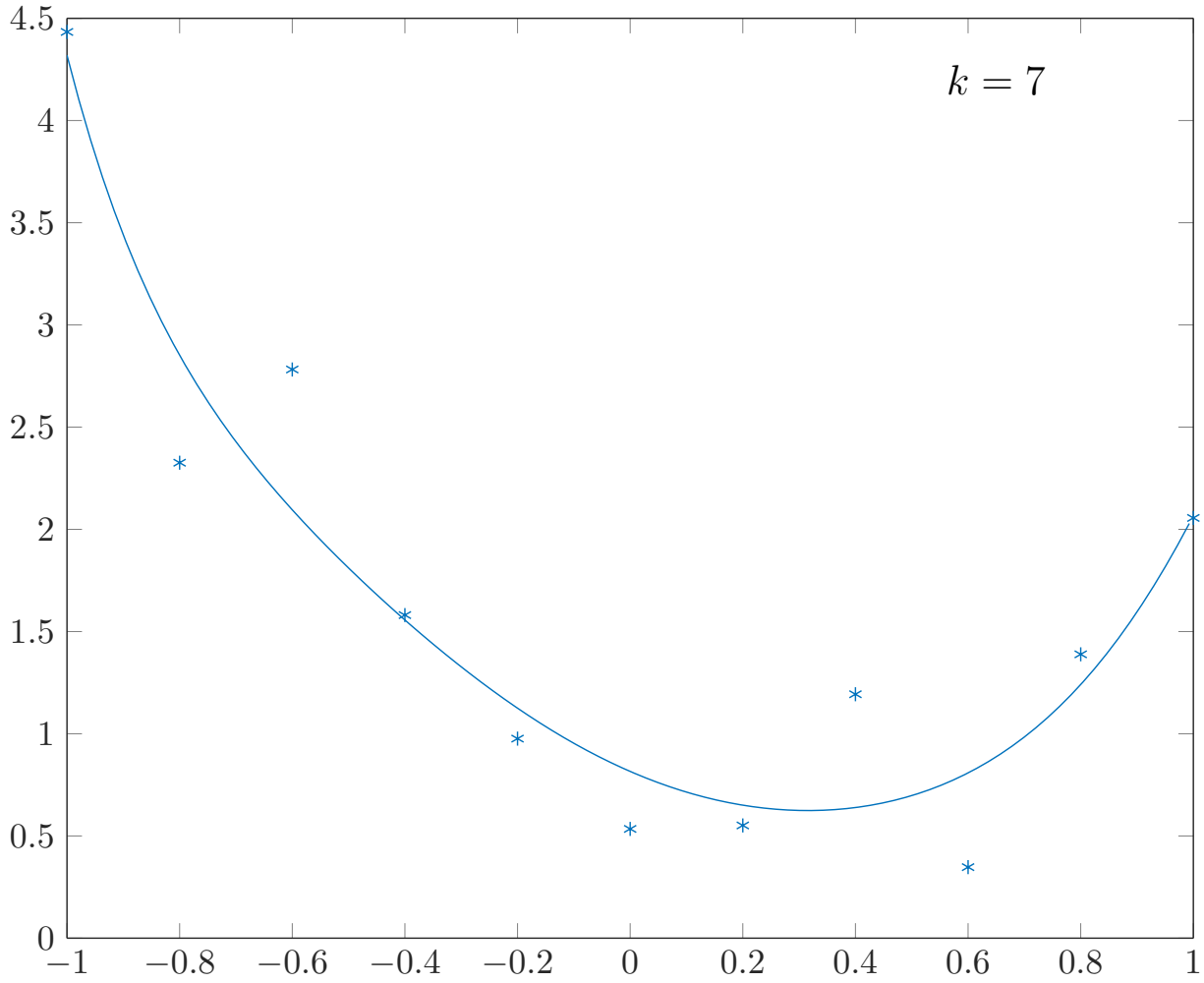
Example

- same example as before
- overfitting is reduced for (too) high complexity models



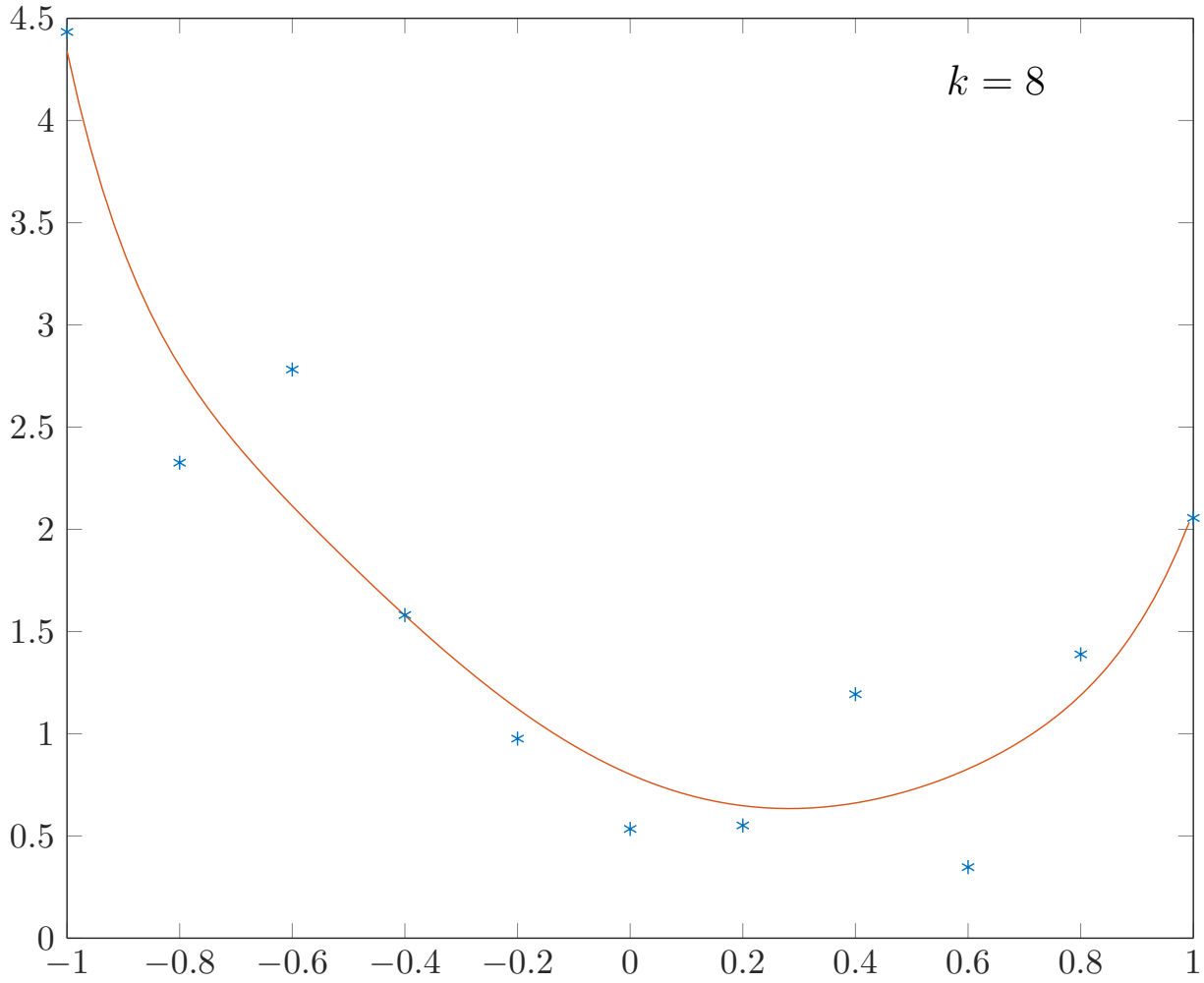
Example

- same example as before
- overfitting is reduced for (too) high complexity models



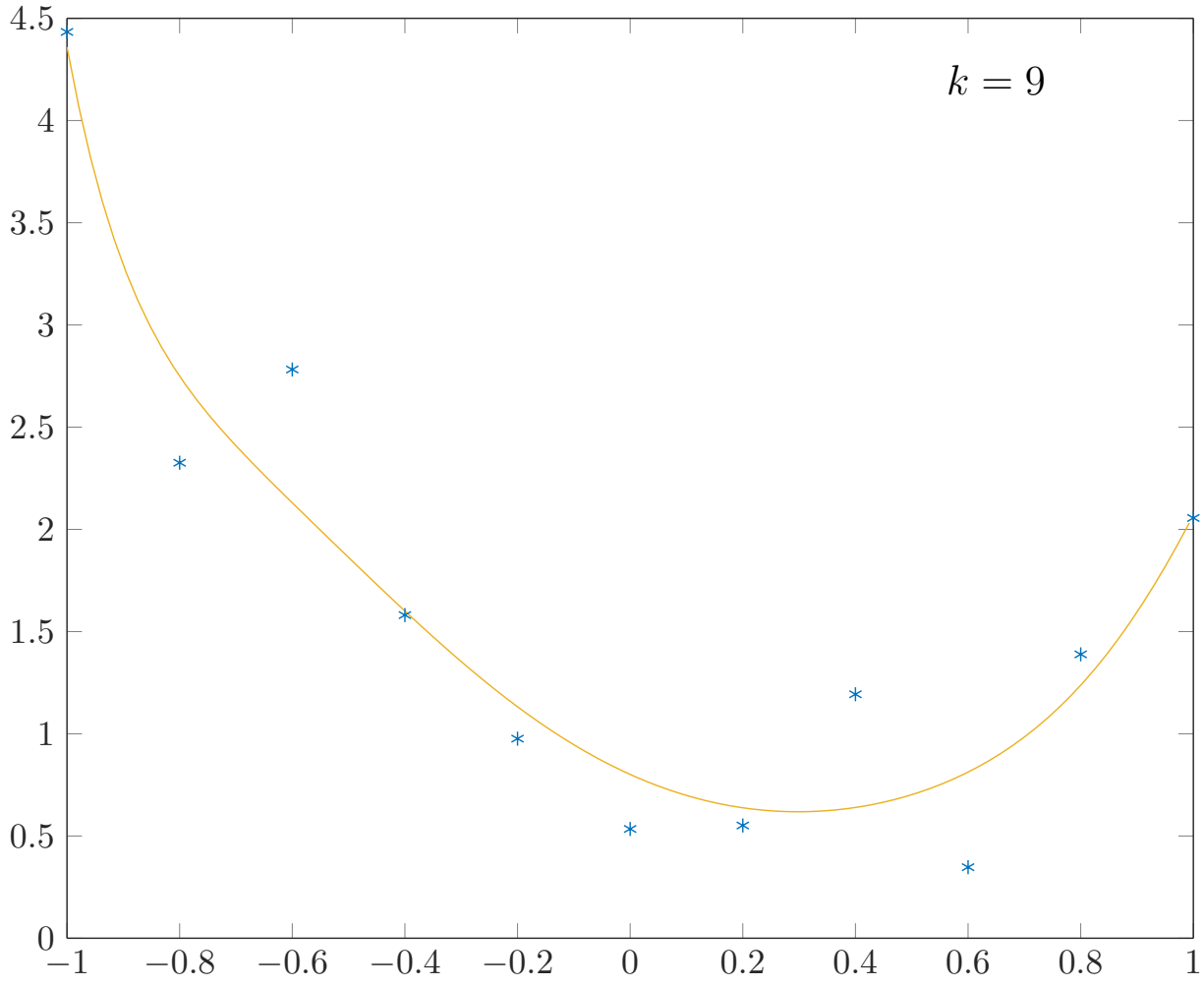
Example

- same example as before
- overfitting is reduced for (too) high complexity models



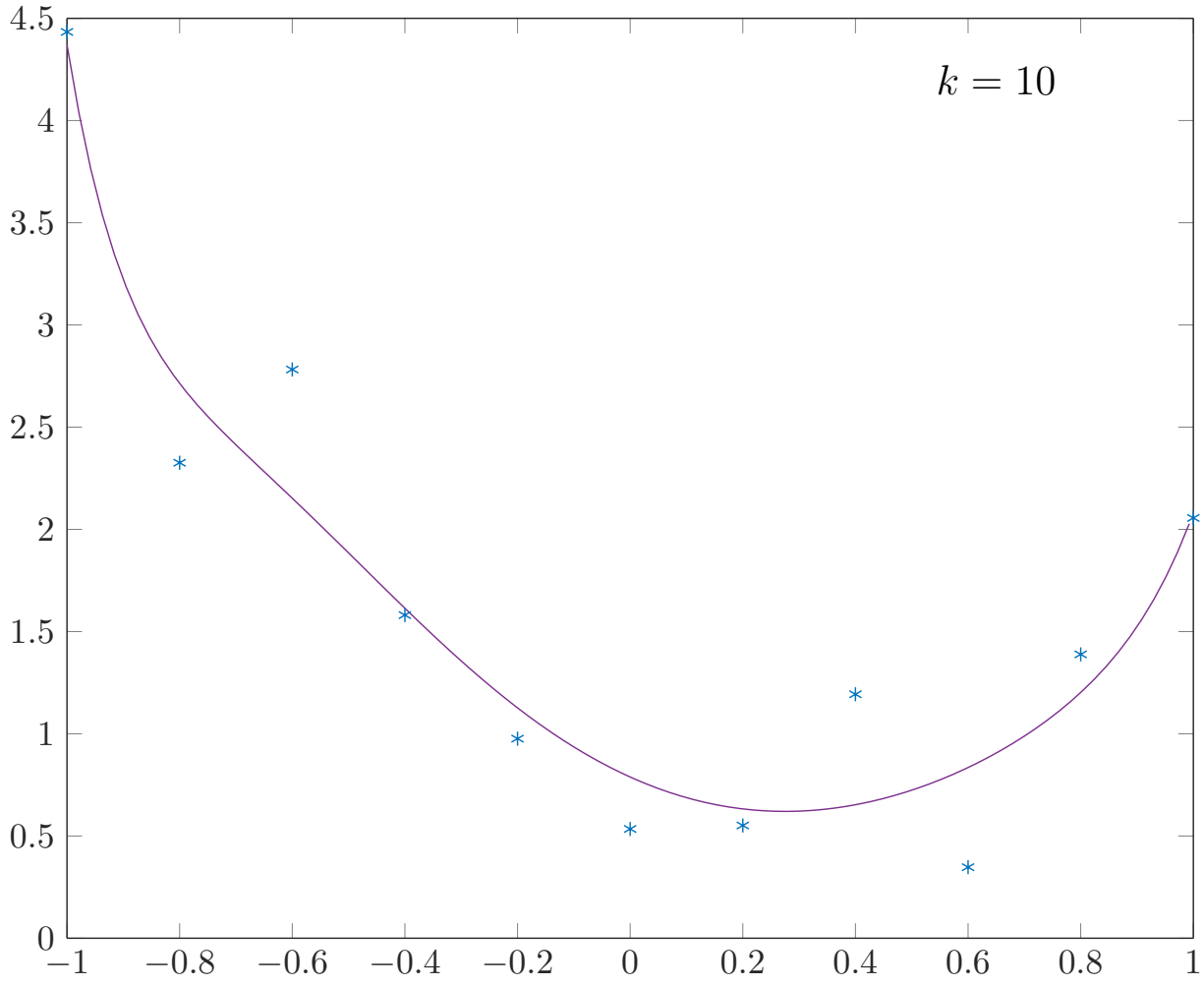
Example

- same example as before
- overfitting is reduced for (too) high complexity models



Example

- same example as before
- overfitting is reduced for (too) high complexity models



Underdetermined systems

- so far, we have discussed overdetermined linear systems
- what if we have an underdetermined linear system?
- that is, X is does not have full column rank
- that is, there is an infinite amount of solutions to

$$\text{minimize } \|Xs - y\|^2$$

- using Tikhonov regularization
  - we get unique solution
  - avoid solutions with large norms (that run away in nullspace of X)
  - we can reduce overfitting when using too complex models

Feature selection

- assume that we have $X \in \mathbb{R}^{m \times n}$ with $m < n$ (or $m \ll n$)
- that is, (far) fewer examples than features
- LS solution not unique (but typically nonzero in all elements)
- we would like to select a subset of features to explain data
- easier to interpret solution
- typically want subset to have cardinality (much) less than m
- this leads us to pose the following 1-norm problem

$$\begin{array}{ll} \text{minimize} & \|Xs - y\|^2 \\ \text{subject to} & \|s\|_0 \leq k \end{array}$$

where k is a positive integer that decides size of subset

Convex relaxation

- $\|s\|_0$ is nonconvex, instead use the convex proxy $\|s\|_1$:

$$\begin{array}{ll} \text{minimize} & \|Xs - y\|^2 \\ \text{subject to} & \|s\|_1 \leq t \end{array}$$

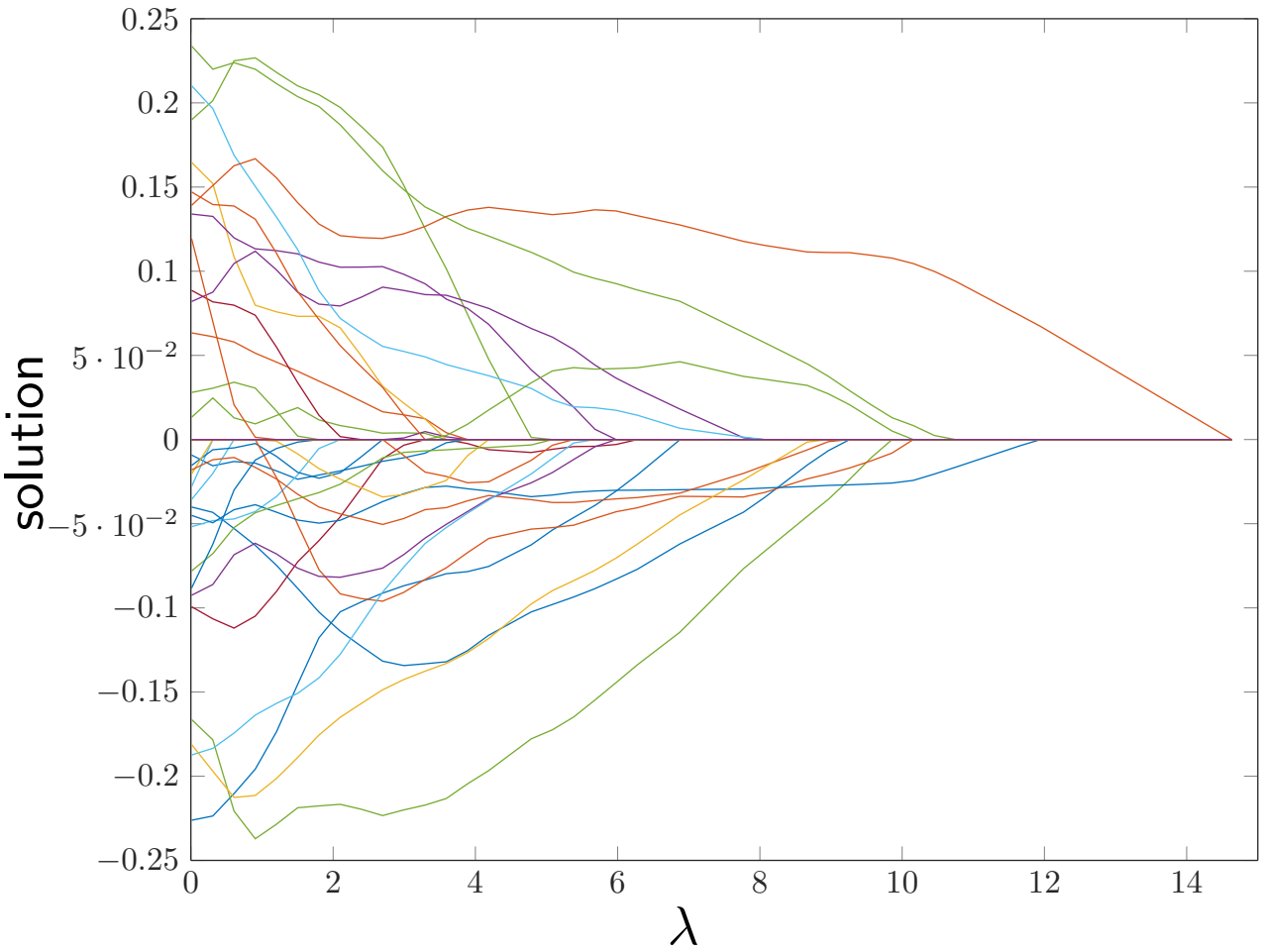
- this is equivalent, for some λ:

$$\text{minimize } \|Xs - y\|^2 + \lambda \|s\|_1$$

- this problem is called the *lasso* problem
- typically gives sparse solutions
- λ decided by cross validation and desired sparsity

Example

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



- more nonzero elements in solution as λ decreases
- note that $\|x\|_0 = 30$ for small λ (i.e., 170 $x_i = 0$)

Solving the lasso

- coordinate descent is most commonly used algorithm
- let $i = 0$, $\lambda_i = \|X^T b\|_\infty$, which gives $s^*_{\lambda_i} = 0$, proceed as:
  1. set $\lambda_{i+1} < \lambda_i$
  2. discard variables that will be zero in $s^*_{\lambda_{i+1}}$ (screening)
  3. use $s^*_{\lambda_i}$ as warm-start for problem with λ_{i+1}
  4. solve lasso with λ_{i+1} (using, e.g., coordinate descent)
  5. cross-validate to decide if solution with λ_{i+1} good $\Rightarrow$ return $s^*_{\lambda_{i+1}}$
  6. increase i by 1 and goto 1.

Lasso and correlation

- assume that two equal features exist, e.g., $x_i = x_{i+1}$ for some i
- let w.l.o.g. $i = 1$, and let s^* solve lasso problem for given λ

$$\text{minimize } \|Xs - y\|^2 + \lambda \|s\|_1$$

- assume that there exist solution with $\Delta = s_1^* + s_2^*$
- assume w.l.o.g. that $\Delta > 0$, then $s_1^*, s_2^* \in [0, \Delta]$
- further any $s_1 \in [0, \Delta]$, $s_2 = \Delta - s_1$ are optimal

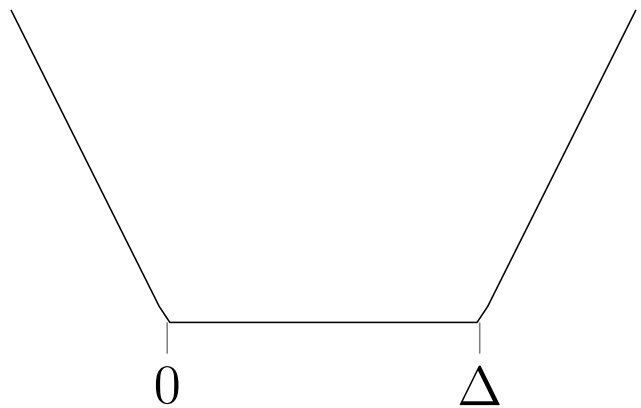
Lasso and correlation

- the problem of selecting s_1 and s_2 reduces to

$$\text{minimize } \|x_1(s_1 + s_2) + \hat{b}\|^2 + \lambda|s_1| + \lambda|s_2|$$

  for an appropriate $\hat{b}$
- if $s_2 = \Delta - s_1 \Rightarrow$ quadratic cost unchanged
- rest reduces to

$$\min_{s_1} \lambda(|s_1| + |\Delta - s_1|)$$



- optimal if $s_1 \in [0, \Delta]$ (and $s_2 = \Delta - s_1 \in [0, \Delta]$)

Lasso and correlation

- if instead x_1 and x_2 are almost linearly dependent
- the problem is of selecting s_1 and s_2 reduces to

$$\text{minimize } \|x_1 s_1 + x_2 s_2 + \hat{b}\|^2 + \lambda|s_1| + \lambda|s_2|$$

- if x_1 (slightly) better explains data, s_2 will be set to 0
- want to be sure that both features are selected

Tikhonov regularization

- add tikhonov regularization to the lasso

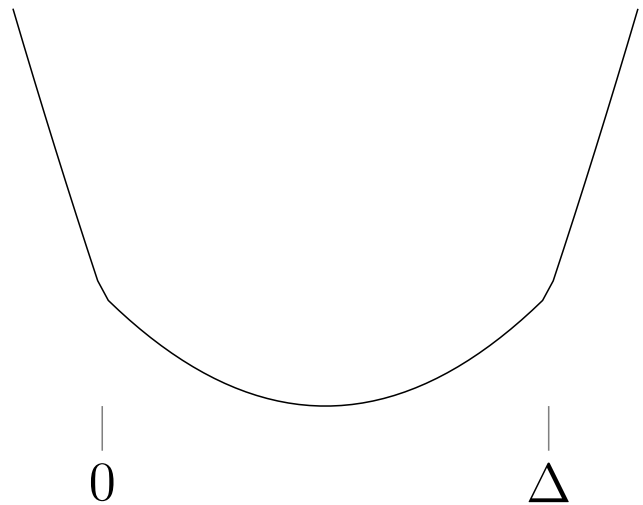
$$\text{minimize } \|Xs - y\|^2 + \lambda_1 \|s\|_1 + \lambda_2 \|s\|^2$$

- this problem is called *elastic net*
- assume that $x_1 = x_2$ and that s^* solves the elastic net
- assume that there exist solution $\Delta = s_1^* + s_2^*$
- then $s_1^* = s_2^* = \frac{\Delta}{2}$

Tikhonov regularization

- proof: as before, quadratic cost unchanged for $s_2 = \Delta - s_1$
- remaining (regularization) part is

$$\min_{s_1} \lambda_1(|s_1| + |\Delta - s_1|) + \lambda_2(s_1^2 + (\Delta - s_1)^2)$$



- that is $s_1 = \Delta/2$ and $s_2 = \Delta - s_1 = \Delta/2$
- for almost correlated features, both (or none) probably selected

Group lasso

- sometimes we want groups of variables to be 0 or nonzero
- introduce $s = (s_1, \ldots, s_p)$ where $s_i \in \mathbb{R}^{n_i}$
- the group lasso problem is

$$\text{minimize } \tfrac{1}{2}\|Xs - b\|^2 + \lambda \sum_{i=1}^{p} \|s_i\|$$

- in 1D, case (i.e., $n_i = 1$) it reduces to the lasso
- prox of each part of group norm is

$$\operatorname*{argmin}_s \{\|s\| + \tfrac{1}{2}\|s - z\|^2\} = \begin{cases} 0 & \text{if } \|z\| \leq 1 \\ z - z/\|z\| & \text{else} \end{cases}$$

- there are no kinks saying that individual components should be 0

Classification

- you have a set of labeled data $(x_i, y_i) \in \mathbb{R}^n \to \mathbb{R}$
- the y_i are binary, i.e., $y_i \in \{-1, 1\}$
- the objective is to find a hyperplane that separates points
- that is, we want to find s and r such that

$$\langle s, x_i \rangle \geq 0 \text{ for all } i \text{ with } y_i = 1$$
$$\langle s, x_i \rangle \leq 0 \text{ for all } i \text{ with } y_i = 0$$

- this can model, e.g., a spam filter
- x_i are number of encountered words or phrases
- y_i is label for spam or no spam
- want to train spam filter (decide s) for future predictions

Optimization formulation

- ideally, we want a function $h(Xs) = \sum h_i(\langle x_i, s\rangle, y_i)$ with

$$h_i(z_i, y_i) = \begin{cases} 0 & \text{if } z_i y_i \geq 0 \\ 1 & \text{else} \end{cases}$$

- this counts x_i that are mis-classified for a certain s
- optimizing this minimizes number of mis-classifications on data

Convex proxy

- we settle for a convex proxy $h(Xs) = \sum h_i(\langle x_i, s\rangle, y_i)$ with

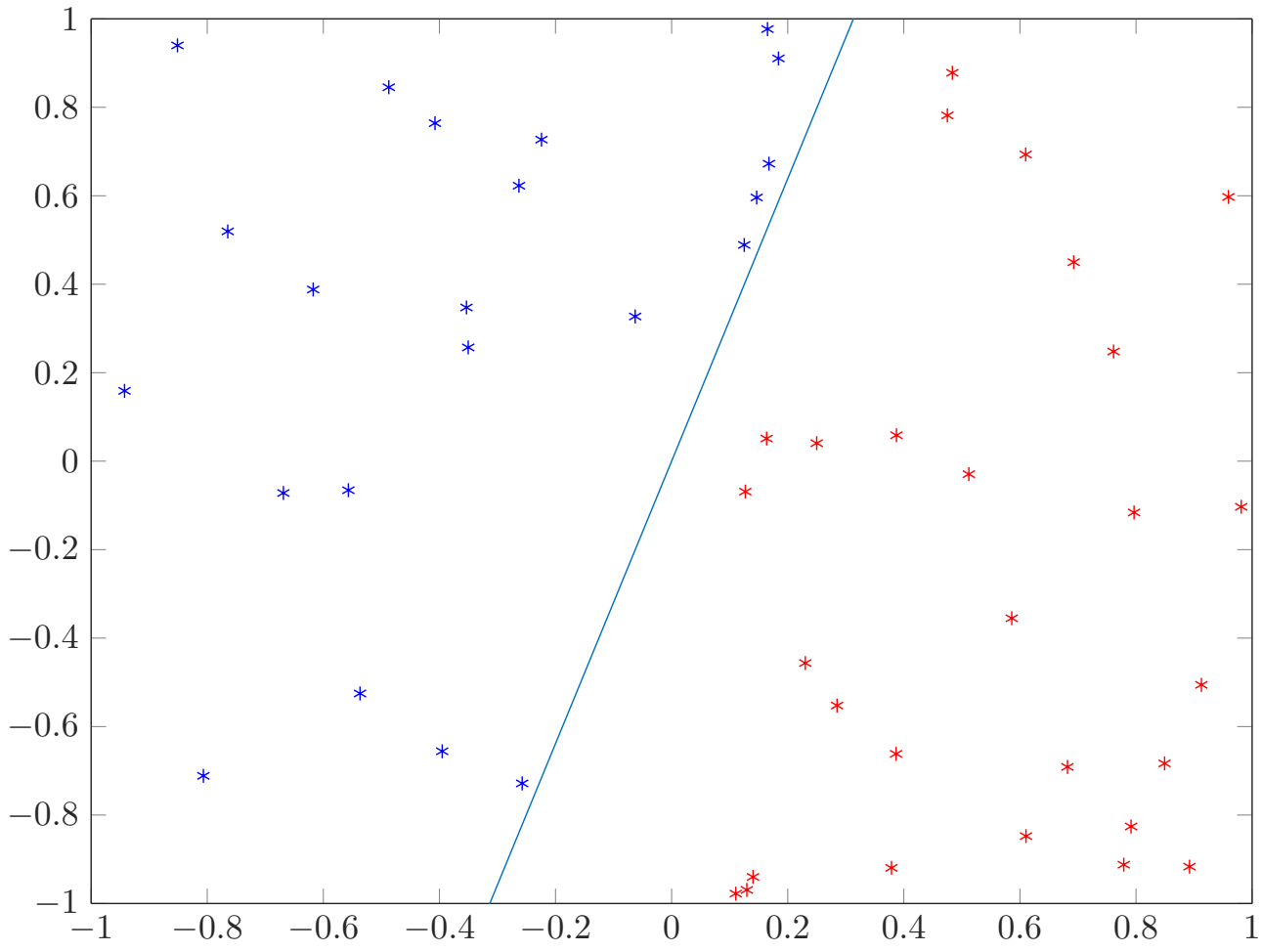
$$h_i(z_i, y_i) = \begin{cases} 0 & \text{if } z_i y_i \geq 1 \\ 1 - y_i z_i & \text{else} \end{cases}$$

- this is called hinge loss
- can be written as $\max(0, 1 - z_i y_i)$
- the classification problem becomes

$$\text{minimze } h(Xs)$$

Example

- classification problem with $X \in \mathbb{R}^{50\times 2}$



Regularization

- add squared 2-norm regularization $\Rightarrow$ support vector machine

$$\text{minimize } h(Xs) + \|s\|^2$$

  - Tikhonov regularization, reduces overfitting for complex models
  - often solved via the dual

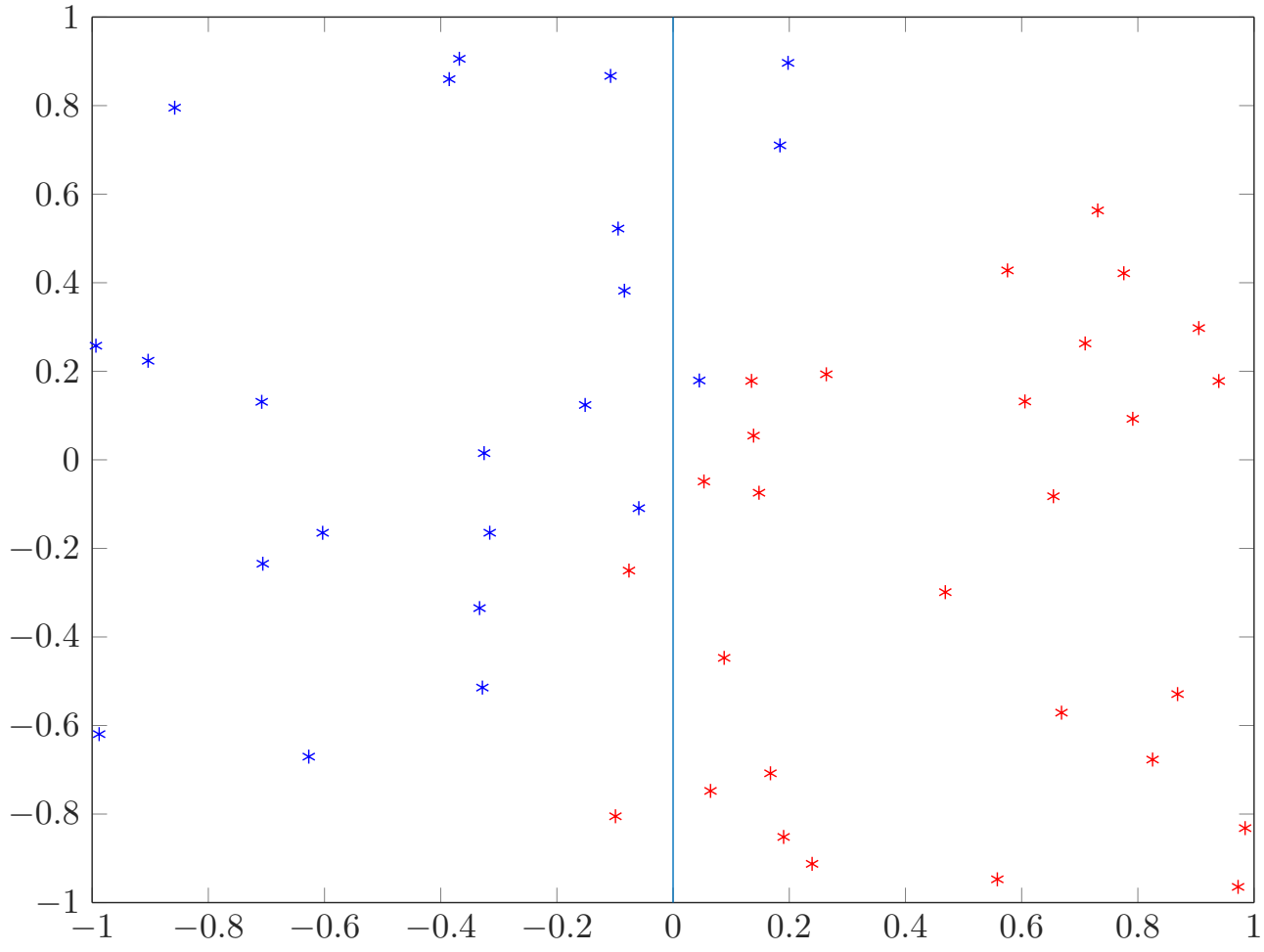
- add 1-norm regularization $\Rightarrow$ sparse classification

$$\text{minimize } h(Xs) + \|s\|_1$$

  - find subset of parameters to split the data

Sparse example

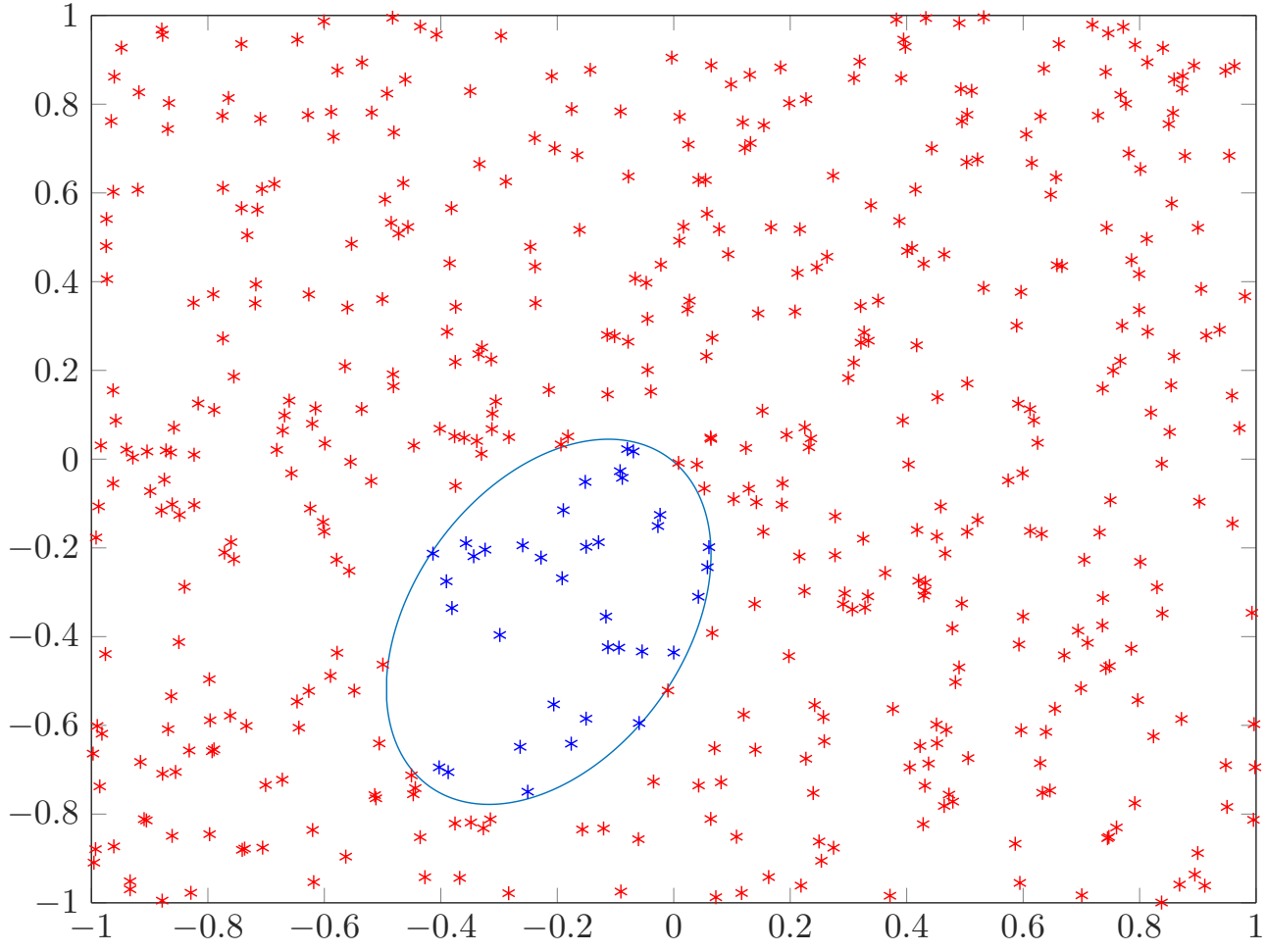
- sparse classification problem with $X \in \mathbb{R}^{50\times 2}$



- gives sparse classifier despite some misclassifications
- (more useful in higher dimensions)

Nonlinear classification

- can do nonlinear classification by adding features
- done as in normal regression case already covered
- example: features for second order (x_1x_2, x_1^2, x_2^2) included



Model predictive control

- we want to control a linear dynamical system described by

$$x_{t+1} = Ax_t + Bu_t$$

- u_t is the input at time t, x_t is system state at time t
- state at time $t + 1$ depends linearly on state and input at t
- we have limitations on inputs

$$u_t \in \mathcal{U}$$

- we have desired region for (linear combination of) states

$$Cx_t \in \mathcal{X}$$

- define desired operating point x^r
- let u^r be consistent input, i.e., such that $x^r = Ax^r + Bu^r$
- objective: choose $\{u_t\}_{t=0}^{\infty}$ such that $x_{t+1} \to x_r$ as $t \to \infty$

Feedback

- we typically have disturbances v_t in the model, e.g.,:

$$x_{t+1} = Ax_t + Bu_t + v_t$$

- therefore we cannot precompute whole sequence $\{u_t\}_{t=0}^{\infty}$
- use feedback: in each sample t compute u_t
- many ways to compute feedback, e.g., using optimization

Formulate optimization cost

- solve problem as if disturbance nonexistent
- want x_t to stay close to x^r for all future t
- not necessary on x^r because will anyway drift due to disturbance
- want u_t to stay close to consistent u^r for all future t
- suggested cost (using future estimated states,inputs) $\hat{x}, \hat{u}$:

$$\sum_{\tau=1}^{\infty} \|x - x^r\|^2 + \|u_t - u^t\|^2$$

- might want to penalize individual states and inputs differently:

$$\sum_{i=1}^{\infty} \|x_t - x^r\|_Q^2 + \|u_t - u^t\|_R^2$$

Formulate optimization constraints

- add (estimated) dynamics relation as constraints

$$\hat{x}_{t+1} = A\hat{x}_t + B\hat{u}_t, \qquad t = 0, \ldots, \infty$$

  with initial condition $\hat{x}_0 = x_t$ (measurement of current state)
- add constraints on inputs and states

$$u_t \in \mathcal{U}, \quad Cx_t \in \mathcal{X}, \qquad t = 0, \ldots, \infty$$

Tractable formulation

- the full problem is on the form

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{\infty} \|\hat{x}_t - x^r\|_Q^2 + \|\hat{u}_t - u^t\|_R^2 \\ \text{subject to} \quad & \hat{x}_{t+1} = A\hat{x}_t + B\hat{u}_t, \quad t = 0, \ldots, \infty \\ & \hat{u}_t \in \mathcal{U}, \quad C\hat{x}_t \in \mathcal{X}, \quad t = 0, \ldots, \infty \\ & \hat{x}_0 = x_t \end{aligned}$$

- this is not tractable due to infinite horizon $\Rightarrow$ truncate

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{N} \|\hat{x}_t - x^r\|_Q^2 + \|\hat{u}_t - u^t\|_R^2 \\ \text{subject to} \quad & \hat{x}_{t+1} = A\hat{x}_t + B\hat{u}_t, \quad t = 0, \ldots, N \\ & \hat{u}_t \in \mathcal{U}, \quad C\hat{x}_t \in \mathcal{X}, \quad t = 0, \ldots, N \\ & \hat{x}_0 = x_t \end{aligned}$$

- if Q, R positive semidefinite and $\mathcal{U}, \mathcal{X}$ convex $\Rightarrow$ problem convex
- if R positive definite and system controllable $\Rightarrow$ unique solution
- choose N large enough to see all dynamics (and a bit longer)
- MPC: iteratively solve this problem and apply $\hat{u}_0$

Other behavior

- if we want piece-wise constant input, add term:

$$\sum_{t=1}^{N-1} \|\hat{u}_{t+1} - \hat{u}_t\|_1$$

- if we want to maximize, e.g., profit, change objective to

$$\sum_{t=1}^{N} \langle p, x_t \rangle$$

(gives *economic MPC*)

Infeasibility

- for some initial condition $\hat{x}_0$, the problem might be infeasible
- then have no input to send to system
- avoid this by having soft constraints on states
- then can be shown that problem is never infeasible

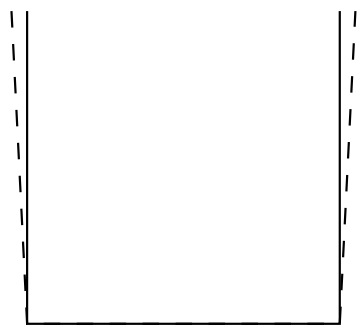
Upper and lower bounds

- if, e.g., $\mathcal{X}$ model upper and lower bounds, i.e.

$$\iota_{\mathcal{X}}(z) = \begin{cases} 0 & \text{if } z \in [l, u] \\ \infty & \text{else} \end{cases}$$

- replace with cost for some large $c > 0$

$$h(z) = \begin{cases} 0 & \text{if } z \in [l, u] \\ c(l - z) & \text{if } z \leq l \\ c(z - u) & \text{if } z \geq u \end{cases}$$



Desired closed-loop behavior?

- want to guarantee stability, feasibility, and performance
- usually feasibility is hardest
  - can formulate tube-based MPC
  - then state in tube independent on disturbance in compact set
  - require full tube to be feasible
  - conservative!
- stability and performance
  - consider abstract formulation of MPC-problem

$$V_N(x_0) = \min_{(\{x_t\},\{u_t\})\in\mathcal{D}(x_0)} \sum_{t=1}^{N} \ell(x_t u_t)$$

  - under some assumptions and for some $\alpha \in (0,1)$, we have:

$$V_N(x_{t+1}) \leq V_N(x_t) + \alpha\ell(x_t, u_t)$$

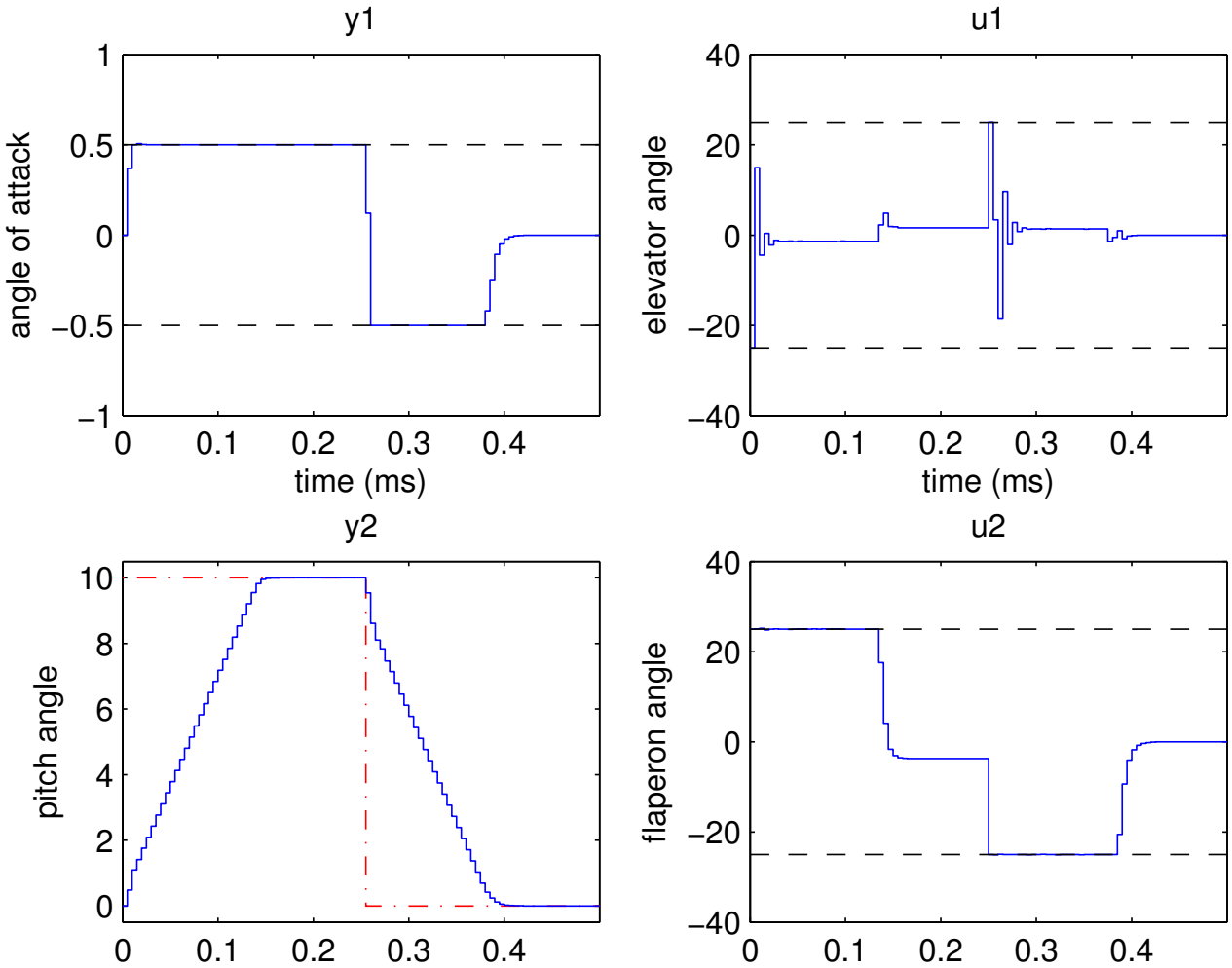
  - then telescope summation gives

$$\alpha \sum_{t=0}^{\infty} \ell(x_t, u_t) \leq V_N(x_0)$$

  that is $\ell(x_t, u_t) \to 0$ as $t \to \infty$

Example

- use MPC to control control pitch angle and angle in aircraft
- upper and lower bounds on input
- soft upper and lower bounds on output
- objective: follow references and satisfy constraints



Other applications

- unsupervised learning (just data, no response variables)
- portfolio optimization
- circuit design
- antenna array design
- digital filter design
- optimal advertising
- supply chain management
- ...