Part 1: Generalized domain representations and regularization

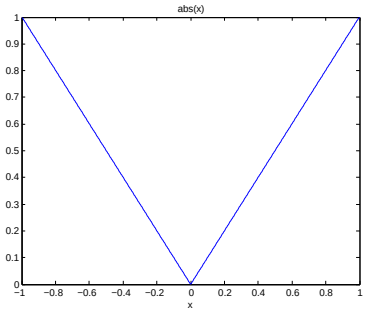
Dale Schuurmans

University of Alberta

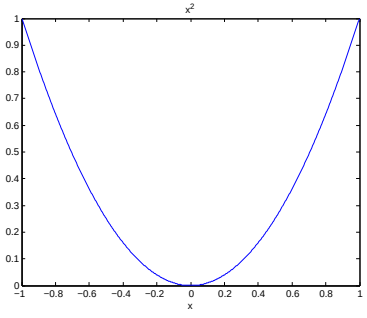
Warm up: loss functions

What prediction loss function $L(\hat{y}, y)$ to use?

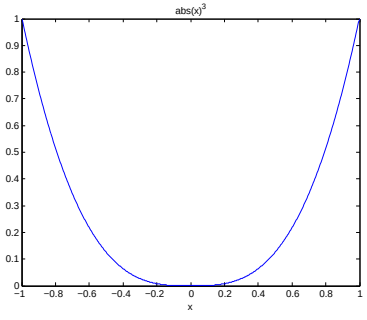
Absolute loss (L_1) $\quad |\hat{y} - y|$



Squared loss (L_2^2) $\quad (\hat{y} - y)^2$



L_p^p loss $\quad |\hat{y} - y|^p$



Loss functions

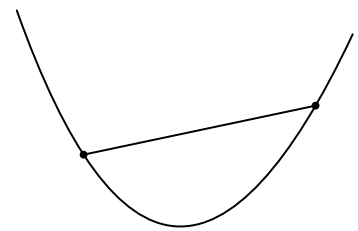
Convexity

ℓ convex if $\quad \ell(\rho \mathbf{w}_1 + (1-\rho)\mathbf{w}_2) \leq \rho\ell(\mathbf{w}_1) + (1-\rho)\ell(\mathbf{w}_2)$

for $0 \leq \rho \leq 1$ $\quad \ell$ at mean $\leq$ mean of ℓs

Properties

- ℓ convex, $\mathbf{w}$ local minima $\Rightarrow$ $\mathbf{w}$ global minima
- nonnegative weighted sum of convex is convex
- max of convex is convex
- ℓ convex $\Rightarrow$ $\ell(X\mathbf{w})$ convex in $\mathbf{w}$
- L_p^p loss convex if $p \geq 1$



Note

convex loss will generally result in tractable training problem

nonconvex loss will generally result in intractable training problem

(* we will see exceptions, but these will be somewhat special)

Loss functions

Smoothness

L_p^p loss differentiable for $p > 1$

L_1 loss not differentiable, but still convex

Properties

Nonsmooth optimization generally more expensive than smooth

But convexity still generally results in tractable training problems (as we saw for L_1 loss)

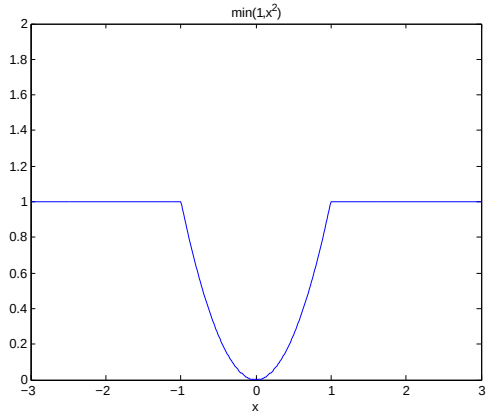
Other lecturers will explain algorithmic ideas behind efficient smooth/nonsmooth minimization.

Loss functions

Robust loss

$\min(1, (\hat{y} - y)^2)$

"gives up" on outliers



L_1 more robust than L_2^2

L_p^p more robust than L_q^q for $p \leq q$

These are iid losses

$L(\hat{\mathbf{y}}; \mathbf{y}) = \frac{1}{t} \sum_{i=1}^{t} L(\hat{y}_i; y_i)$

Non-iid losses

e.g., F-measure

Let us assume iid losses

Shorthand notation

$\hat{\ell}(\mathbf{w}) = L(X\mathbf{w}; \mathbf{y}) = \frac{1}{t} \sum_{i=1}^{t} L(X_{i:}\mathbf{w}; y_i)$

Loss functions

Today in Part 1

Just assume we've picked a convex loss $L(\hat{y}; y)$ (say L_1 or L_2^2)

Tomorrow in Part 2

Will show how loss function can be derived from other considerations

Wednesday in Part 3

Will show how robust loss can be expressed as a convex loss plus latent outlier indicators

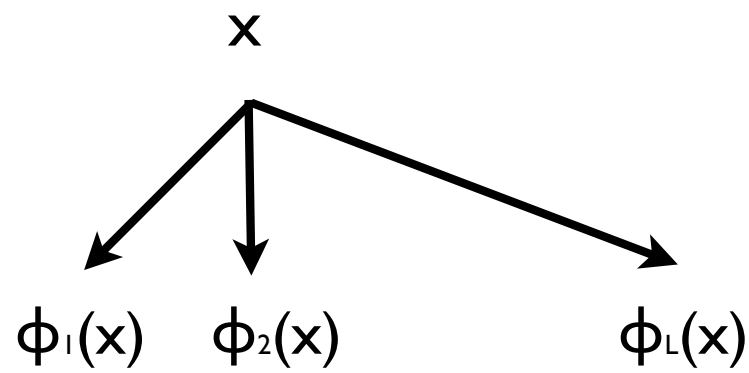
Generalizing the domain representation

Generalized domain representations

Simple idea: feature expansion

- expand representation $\mathbf{x} \mapsto \phi(\mathbf{x})$

New features are (nonlinear) function of original features



"basis functions", "features", "feature functions"

Feature expansion

Expand training set $X \mapsto \Phi$

$$\begin{bmatrix} X_{11} & \cdots & X_{1n} \\ \vdots & & \vdots \\ X_{t1} & \cdots & X_{tn} \end{bmatrix} \mapsto \begin{bmatrix} \phi_1(X_{1:}) & \cdots & \phi_L(X_{1:}) \\ \vdots & & \vdots \\ \phi_1(X_{t:}) & \cdots & \phi_L(X_{t:}) \end{bmatrix}$$

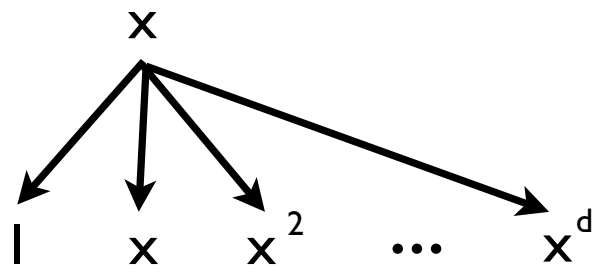
Learn a linear function over extended features
(a nonlinear function of the original features)

Generalized predictor

After learning an extended $L \times 1$ weight vector $\mathbf{w}$
get a nonlinear predictor

$$\mathbf{x} \mapsto \hat{y} = \sum_{j=1}^{L} w_j \phi_j(\mathbf{x}) = \mathbf{w}' \boldsymbol{\phi}(\mathbf{x})$$

Example: polynomial basis



Assume $x_i \in \mathbb{R}$ (scalar)

Training data expansion $\begin{bmatrix} x_1 \\ \vdots \\ x_t \end{bmatrix} \mapsto \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^d \\ \vdots & & & & \vdots \\ 1 & x_t & x_t^2 & \cdots & x_t^d \end{bmatrix}$

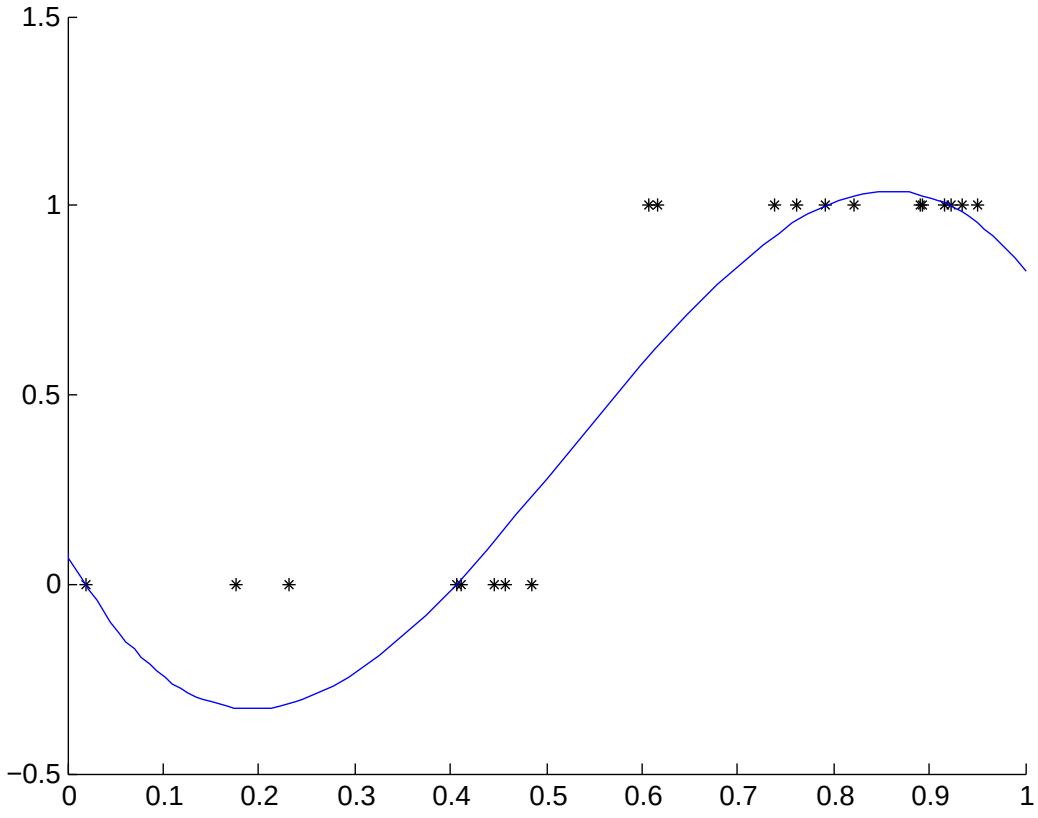
Training

Train $(d+1) \times 1$ vector of coefficients $\mathbf{w}$ using any desired loss

Learned predictor

$x \mapsto \hat{y} = \mathbf{w}'\phi(x) = \sum_{j=0}^{d} w_j x^j$

Example: polynomial basis



Example: trigonometric basis

Assume $x_i \in \mathbb{R}$ (scalar)

Training data expansion $\begin{bmatrix} x_1 \\ \vdots \\ x_t \end{bmatrix} \mapsto \begin{bmatrix} \phi_1(x_1) & \cdots & \phi_t(x_1) \\ \vdots & & \vdots \\ \phi_1(x_t) & \cdots & \phi_t(x_t) \end{bmatrix}$

Use t basis functions assuming $t = 2n + 1$ for some n

1 constant basis function	$\phi_1 = \frac{a_0}{2}$
n cosine basis functions	$\phi_{1+j} = \cos(jx)$ for $j = 1...n$
n sine basis functions	$\phi_{n+1+j} = \sin(jx)$ for $j = 1...n$

If data points happen to be evenly spaced

$x_i - x_{i-1} = \Delta$ constant

$$-\pi = -n\Delta = x_1 \quad \ldots \quad x_t = n\Delta = \pi$$

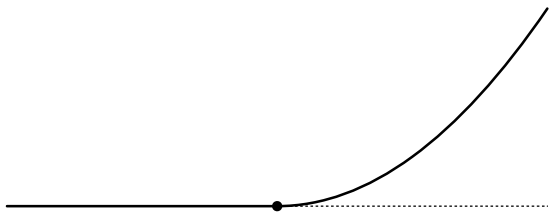
Then columns of Φ are orthonormal and Φ square

Hence exact fit of $\mathbf{y}$ given by $\mathbf{w}^* = \Phi'\mathbf{y}$ (discrete Fourier transform)

Example: basis splines

Assume $x_i \in \mathbb{R}$ (scalar)

$\phi_1(x) = 1$
$\phi_2(x) = x$
$\phi_3(x) = x^2$
$\phi_4(x) = x^3$
$\phi_5(x) = (x - x_1)_+^3 \cdots$
$\phi_j(x) = (x - x_{j-4})_+^3 \cdots$
$\phi_t(x) = (x - x_{t-4})_+^3$



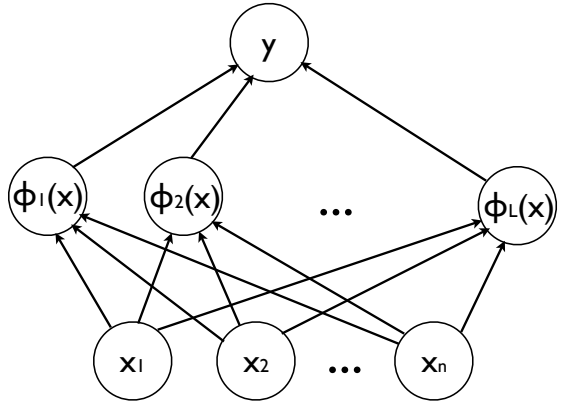
Linear combination is piecewise cubic

$x \mapsto \hat{y} = \sum_{j=1}^{t} w_j \phi_j(x)$

Predictor is continuous and has continuous 1st and 2nd derivatives

Example: feedforward neural network

Multilayer feedforward neural network
with a fixed preprocessing layer



E.g. $\phi_j(\mathbf{x}) = \mathrm{sign}(\mathbf{u}_j'\mathbf{w})$ for some $\mathbf{u}_j$

Given intermediate representation

Learn $\mathbf{w}$, get predictor $\mathbf{x} \mapsto \hat{y} = \sum_{j=1}^{L} w_j \phi_j(\mathbf{x})$

Local basis functions

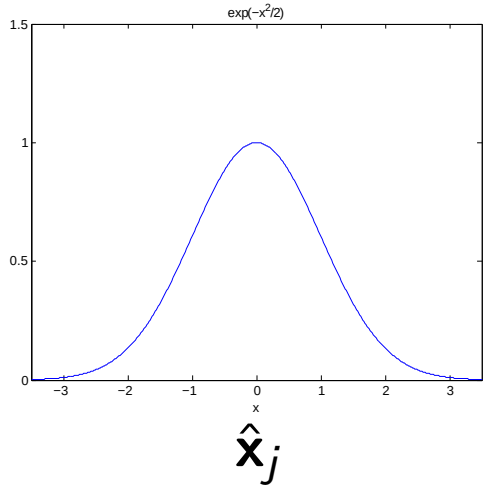
Local basis function

Choose a similarity function κ

$\phi_j(\mathbf{x}) = \kappa(\mathbf{x}, \hat{\mathbf{x}}_j)$ at $\hat{\mathbf{x}}_j$

$\kappa \geq 0$, maximized at $\mathbf{x} = \hat{\mathbf{x}}_j$

$\kappa(\mathbf{x}, \hat{\mathbf{x}}_j)$ decreasing in $\|\mathbf{x} - \hat{\mathbf{x}}_j\|$



Fixing prototype centers $\hat{\mathbf{x}}_1, ..., \hat{\mathbf{x}}_L$

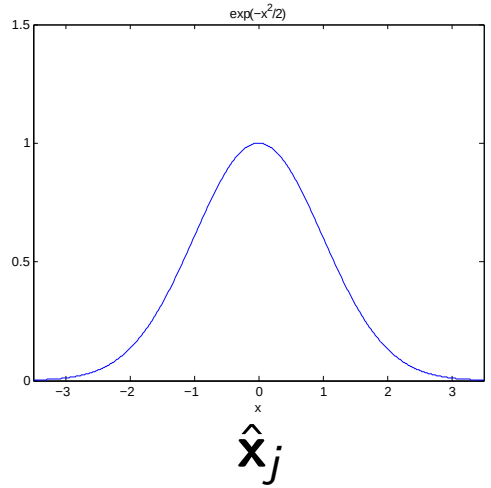
defines basis $\phi_1, ..., \phi_L$

Expand training set

$X \mapsto \Phi$

Learn weights $\mathbf{w}$ over expanded feature representation

Example: radial basis functions (rbfs)



$$\kappa(\mathbf{x}, \hat{\mathbf{x}}_j) = \exp(-\frac{1}{2\sigma^2}\|\mathbf{x} - \hat{\mathbf{x}}_j\|)$$

σ is a "width" parameter

Fully local methods

Locate a prototype center $\hat{\mathbf{x}}_i$ at *every* training point $X_{i:}$

$$X \mapsto \begin{bmatrix} \kappa(X_{1:}, X_{1:}) & \cdots & \kappa(X_{1:}, X_{t:}) \\ \vdots & & \vdots \\ \kappa(X_{t:}, X_{1:}) & \cdots & \kappa(X_{t:}, X_{t:}) \end{bmatrix} = K$$

Interpolation

For most local basis functions κ this enables interpolation

I.e. K is $t \times t$ square matrix

usually invertible (if training examples $X_{i:}$ not duplicated)

$\Rightarrow$ can solve $K\mathbf{w} = \mathbf{y}$ for $\mathbf{w}$

Example: for RBFs

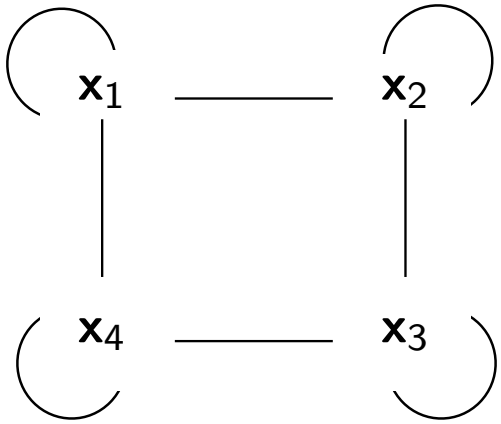
K is symmetric, diagonally dominant, invertible

Example: k nearest neighbors

k nearest neighbor basis function depends on entire training set X

$$\kappa(\mathbf{x}, X_{j:}) = \begin{cases} 1 & \text{if } X_{j:} \text{ among } k \text{ closest points to } \mathbf{x} \text{ in } X \\ 0 & \text{otherwise} \end{cases}$$

E.g. consider 3-nearest neighbors



$$K = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

(might not be invertible)

Fully local methods

- represent pairwise relationships of input training set X

General feature representations

Can construct feature representations for *arbitrary* objects

E.g. strings, graphs, documents

Each feature

- just computes some aspect of the object that hopefully is important for prediction
- map general objects into a feature vector representation

In practice

features are the main source of prior knowledge/constraints —carefully engineered

E.g.

image processing	— edge filters, line filters, SIFTs
document processing	— bag of words, TF-IDF, n-grams
network processing	— degree distribution, friend-of-friend dist'n

The elephant in the room

Dilemma

For a given problem, which features to use?

If $P_{\mathcal{Y}|\mathcal{X}}$ not known

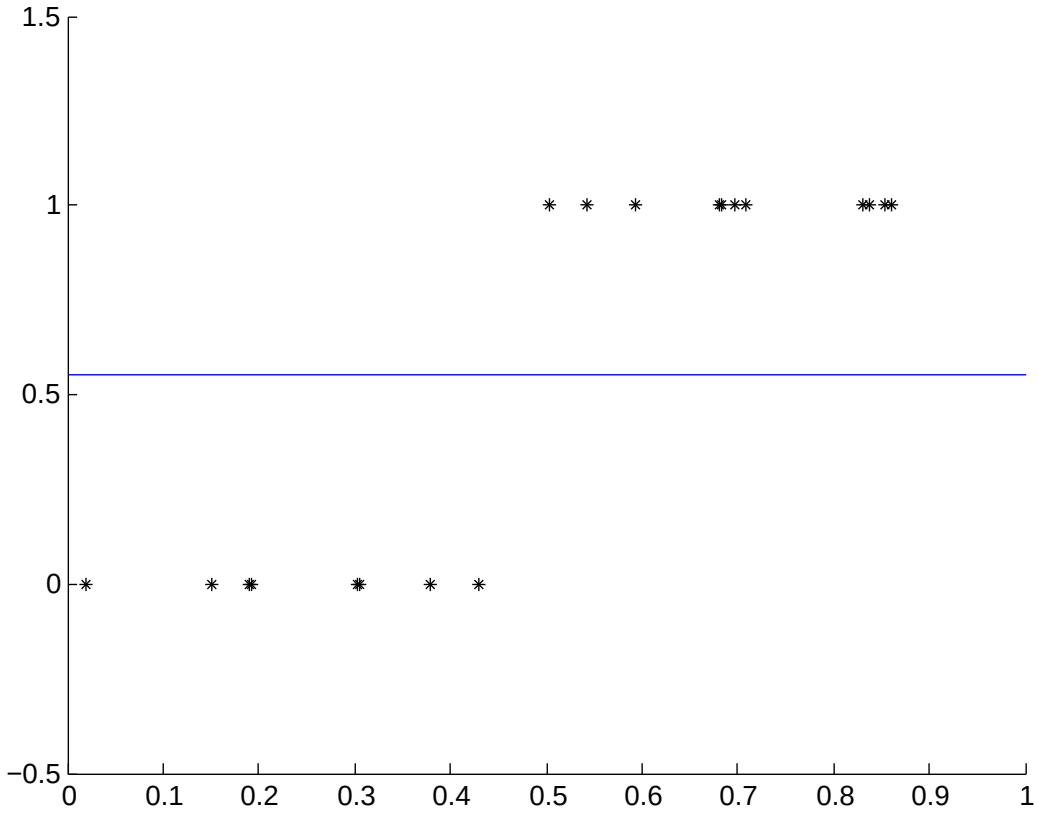
- why not try to be as expressive as possible?
- can represent any target function $f : \mathcal{X} \to \mathcal{Y}$ that way

Fundamental dilemma

underfitting versus overfitting

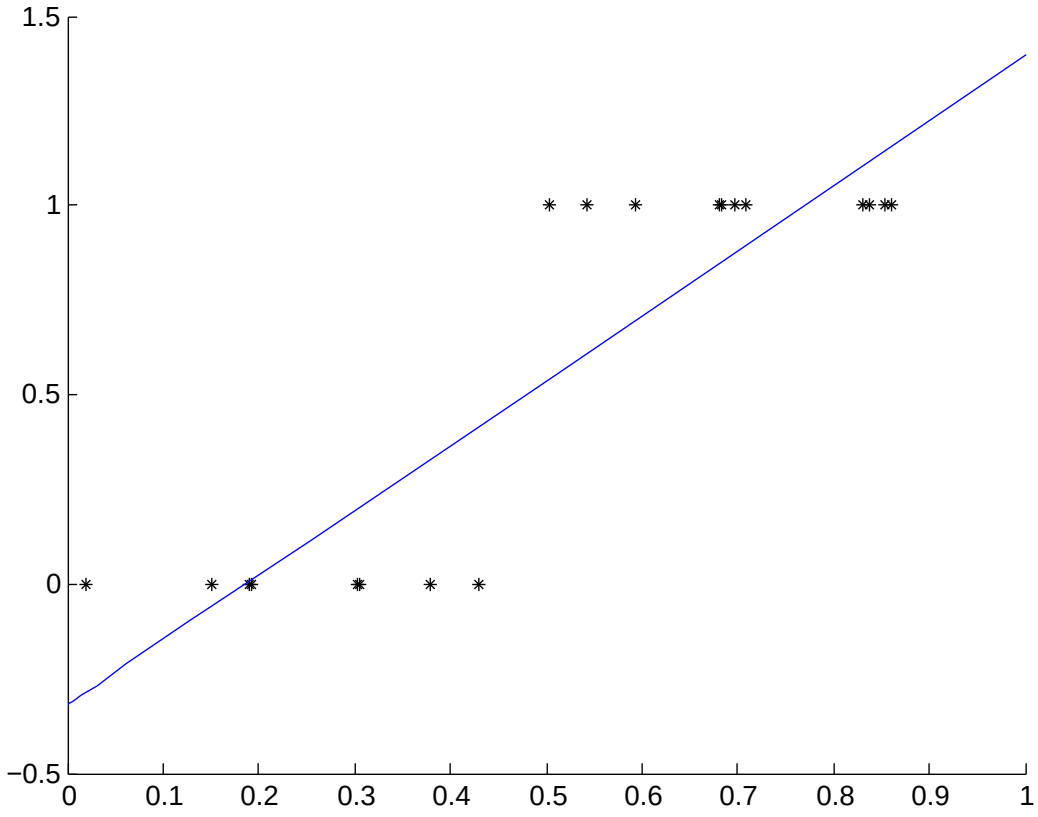
Dilemma

Example: polynomial fitting



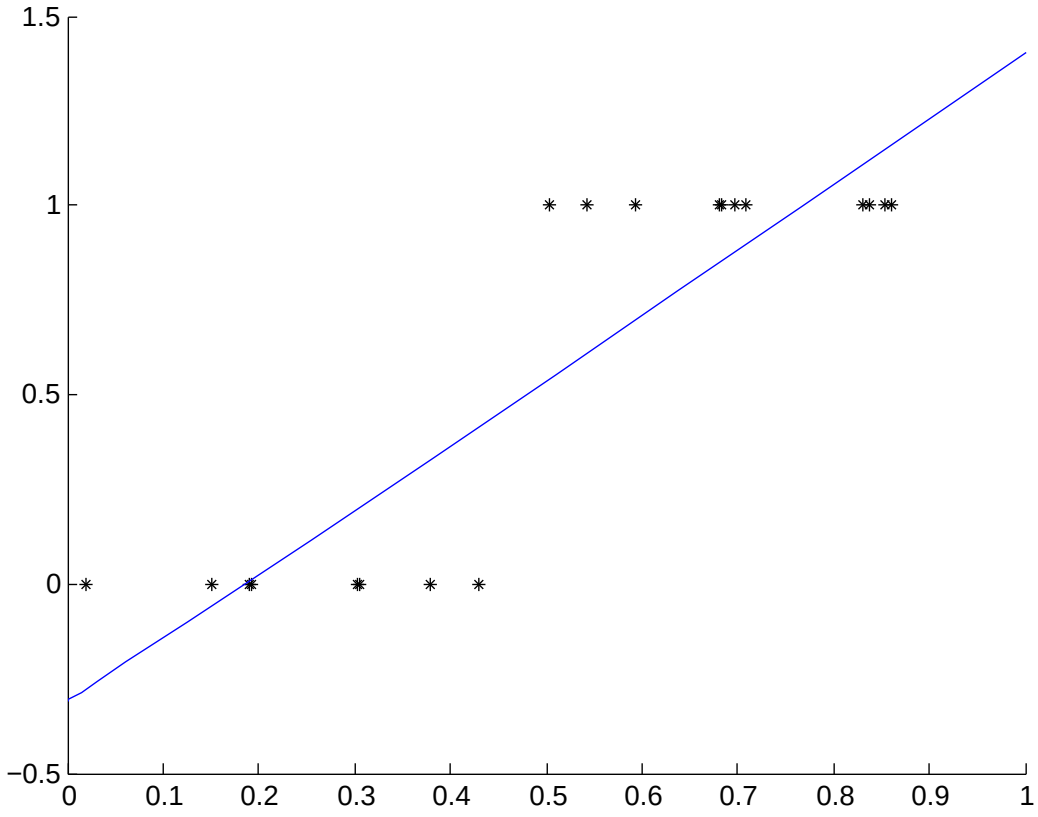
Dilemma

Example: polynomial fitting



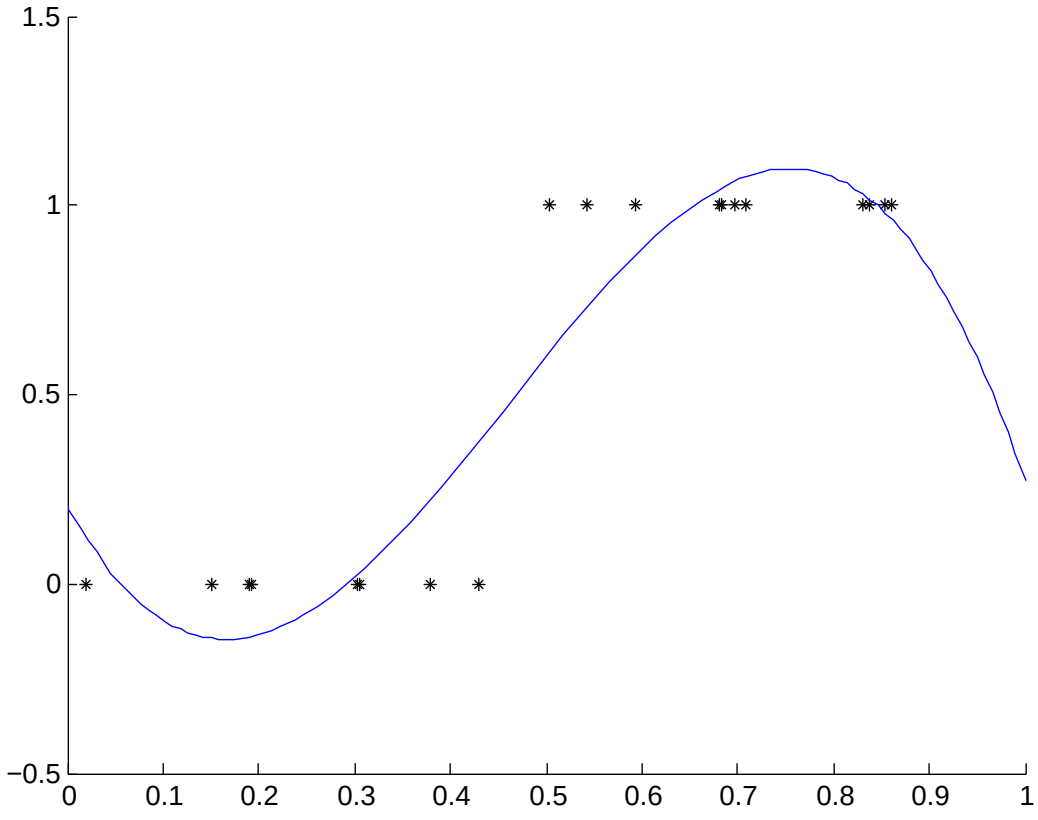
Dilemma

Example: polynomial fitting



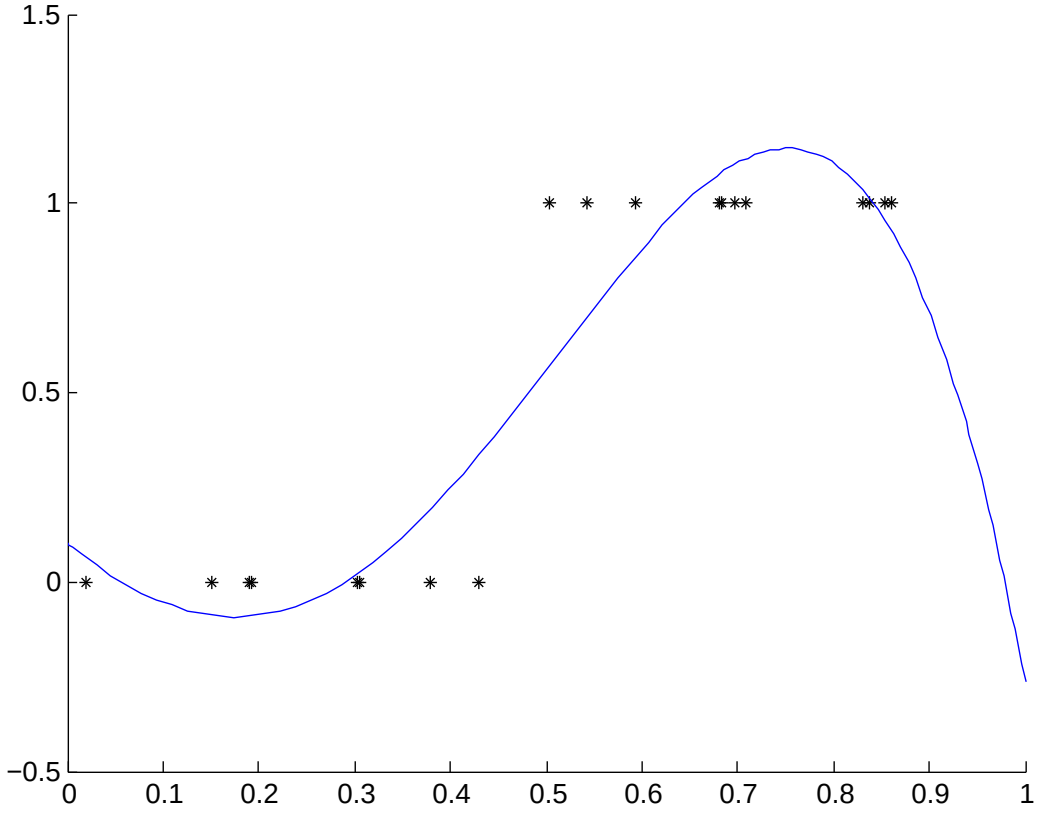
Dilemma

Example: polynomial fitting



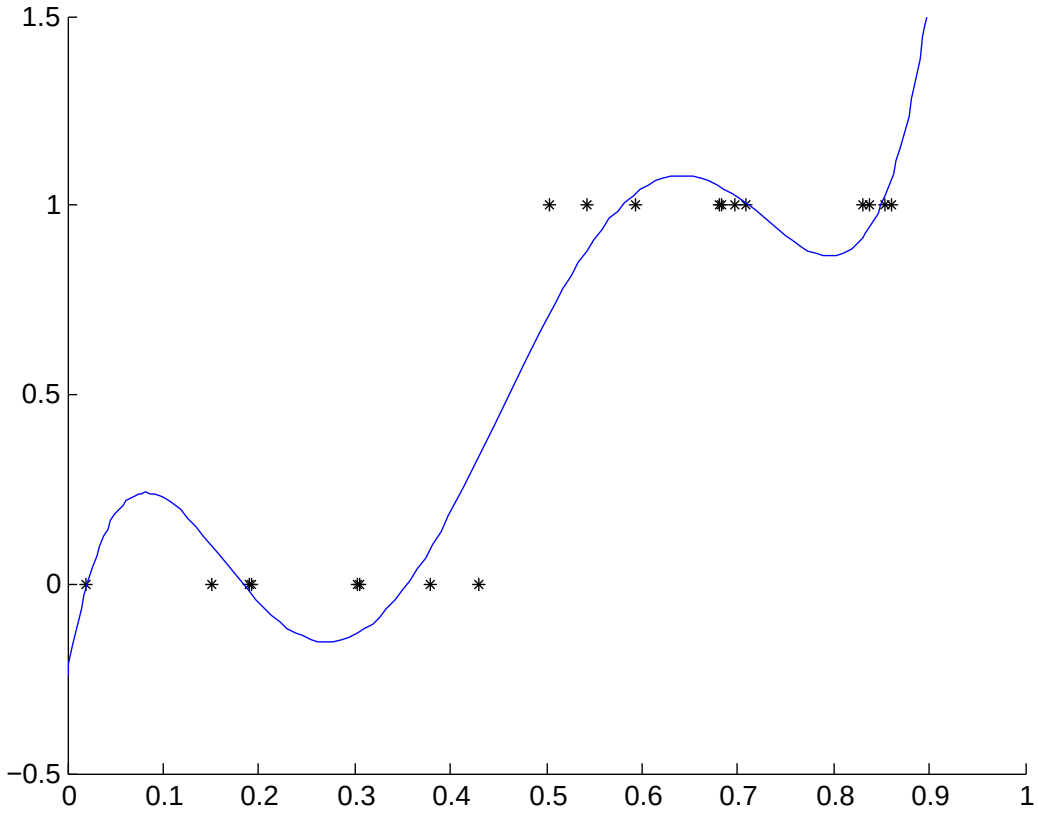
Dilemma

Example: polynomial fitting



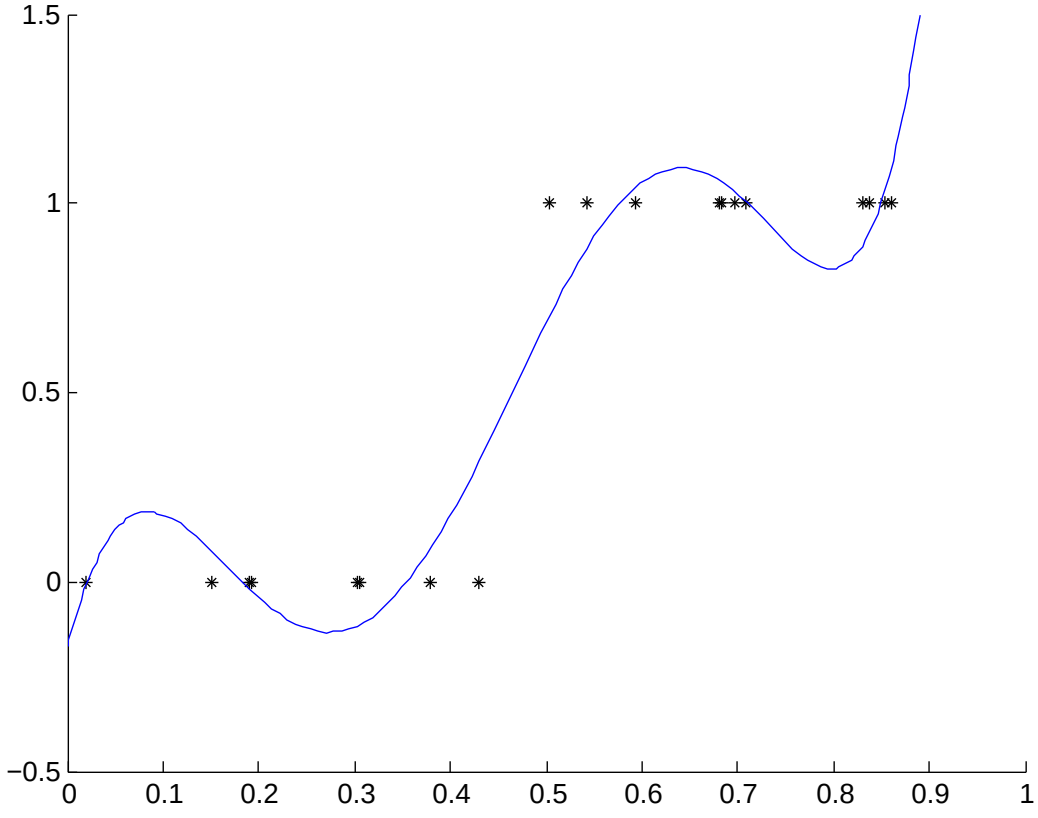
Dilemma

Example: polynomial fitting



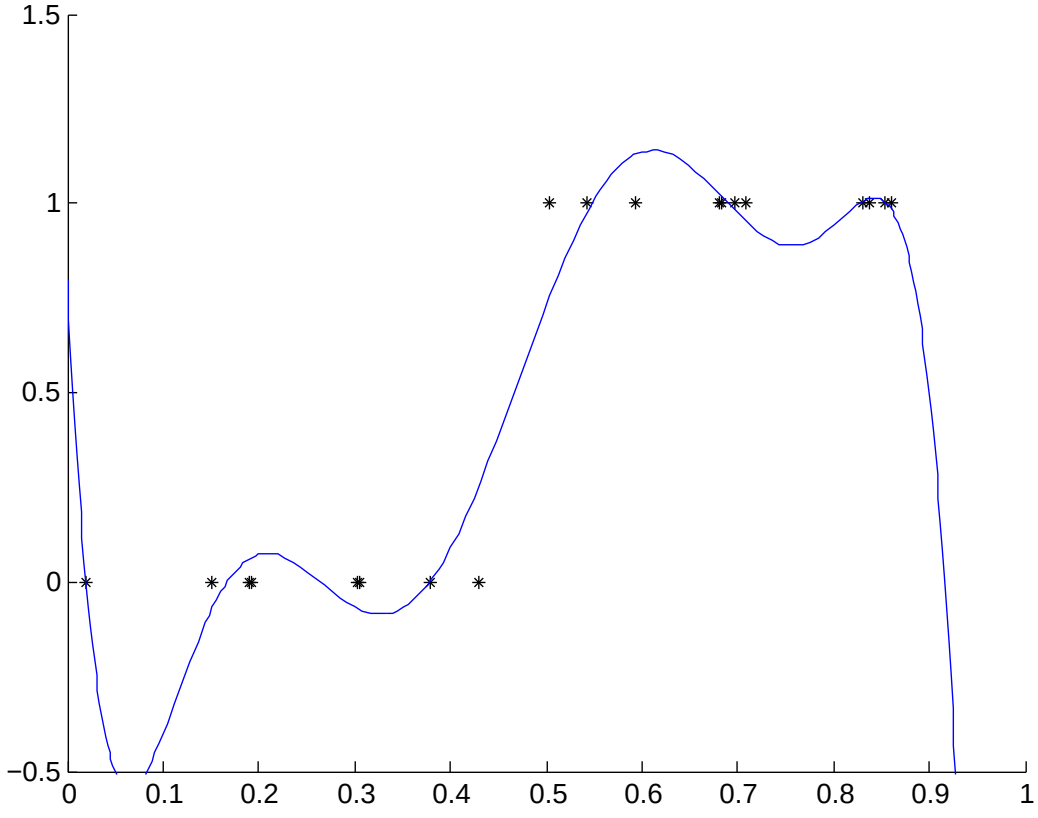
Dilemma

Example: polynomial fitting



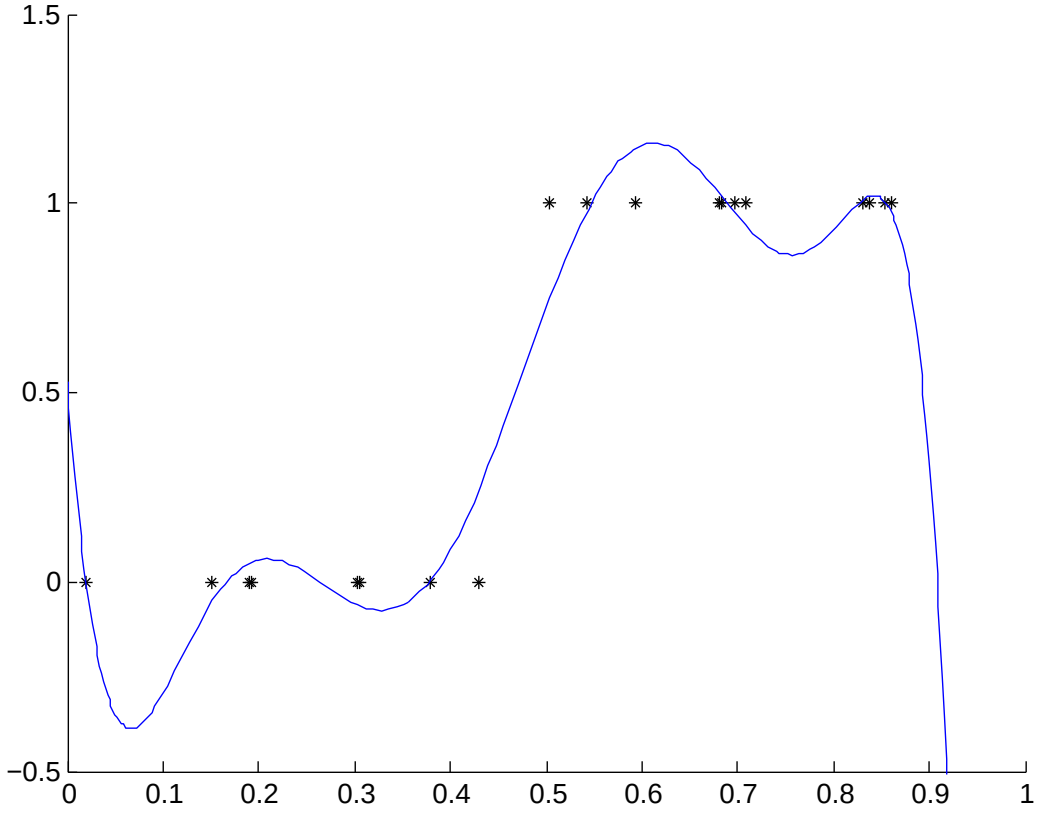
Dilemma

Example: polynomial fitting



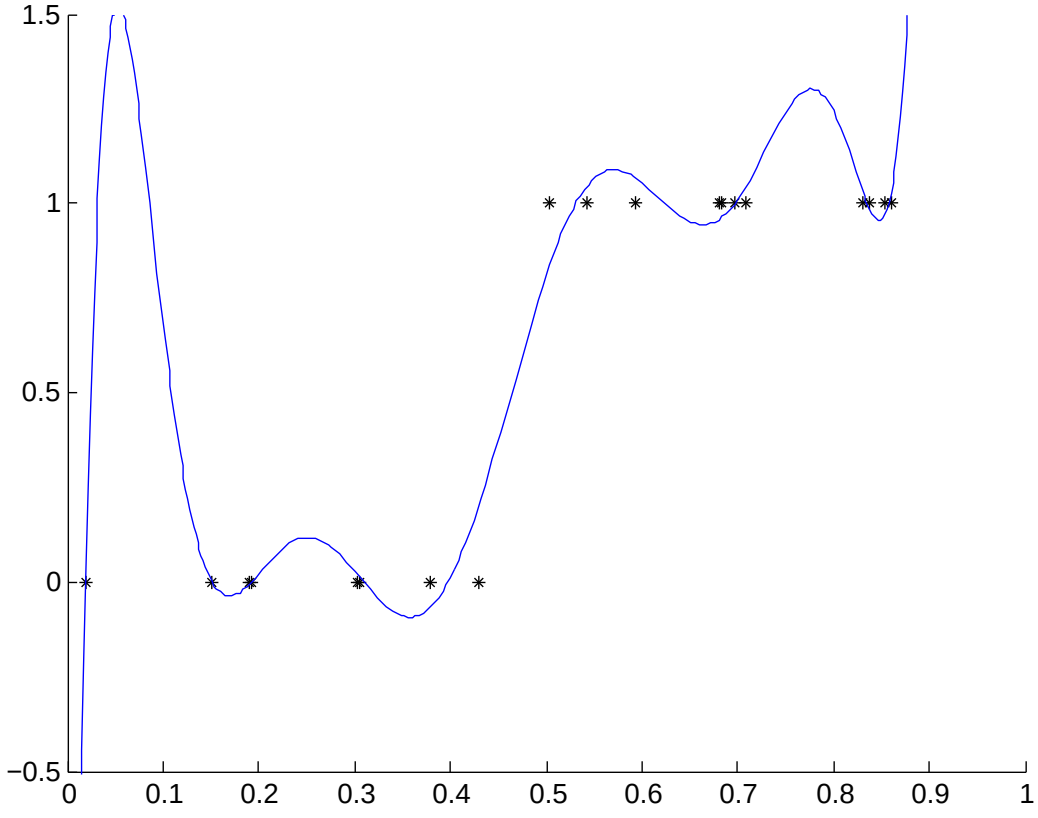
Dilemma

Example: polynomial fitting



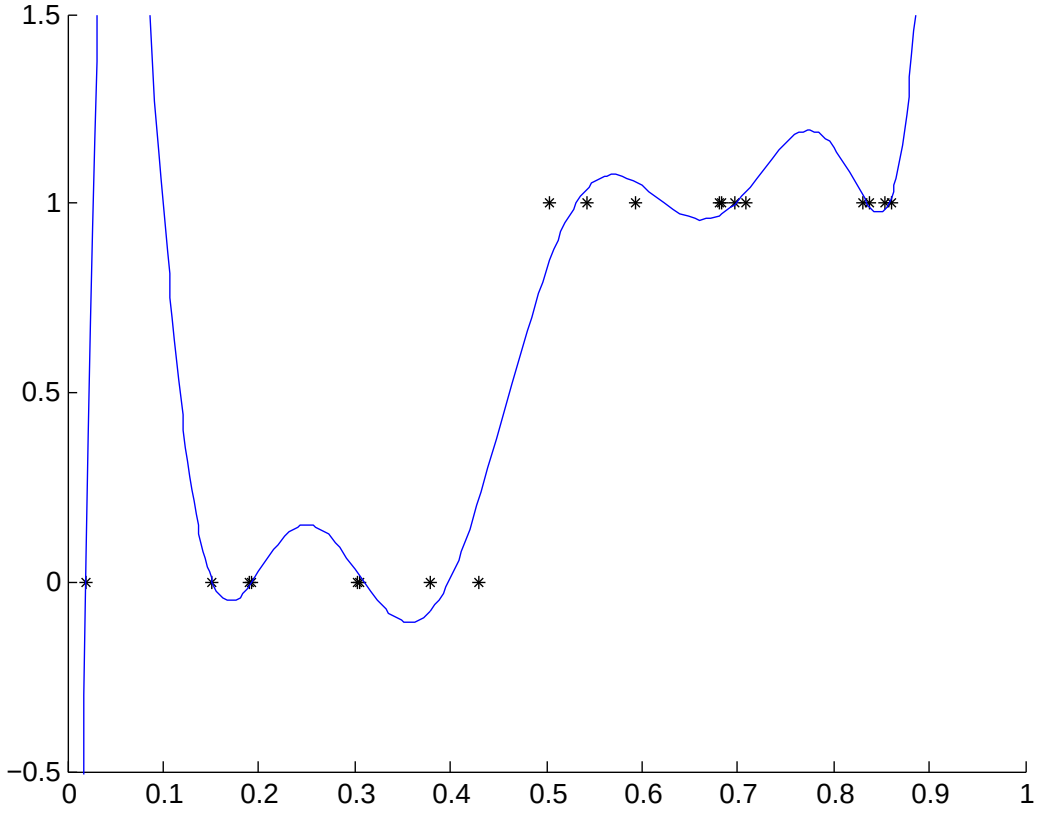
Dilemma

Example: polynomial fitting



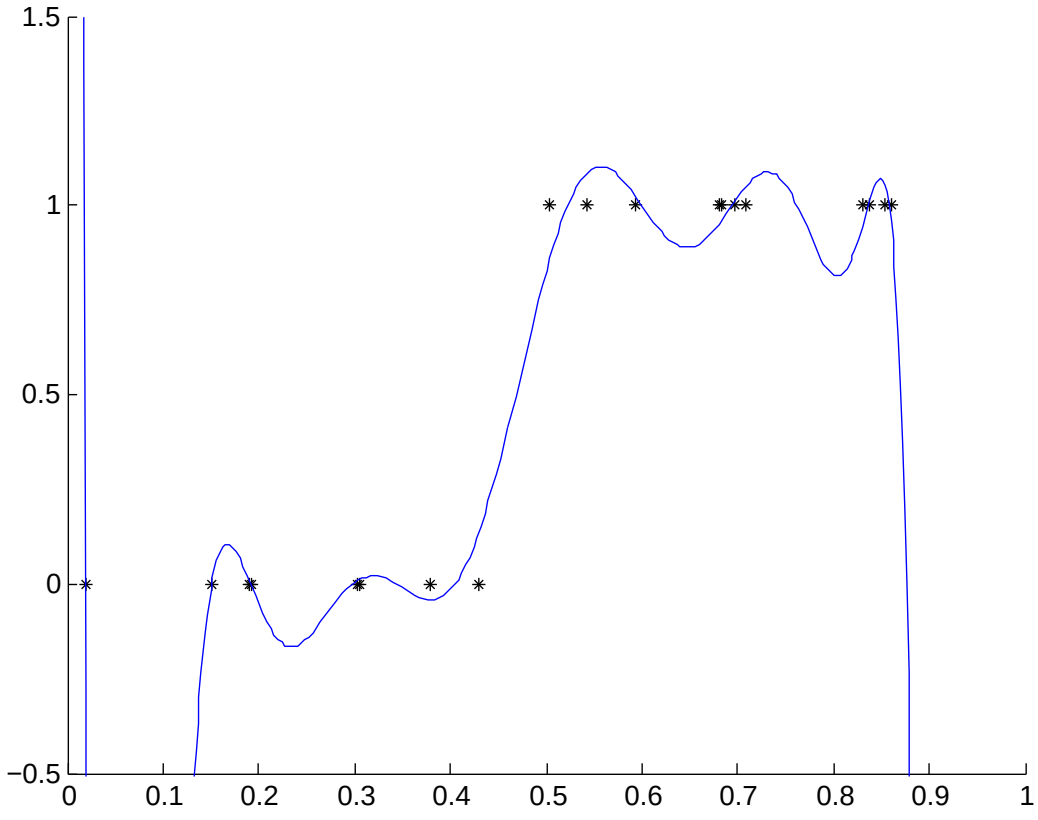
Dilemma

Example: polynomial fitting



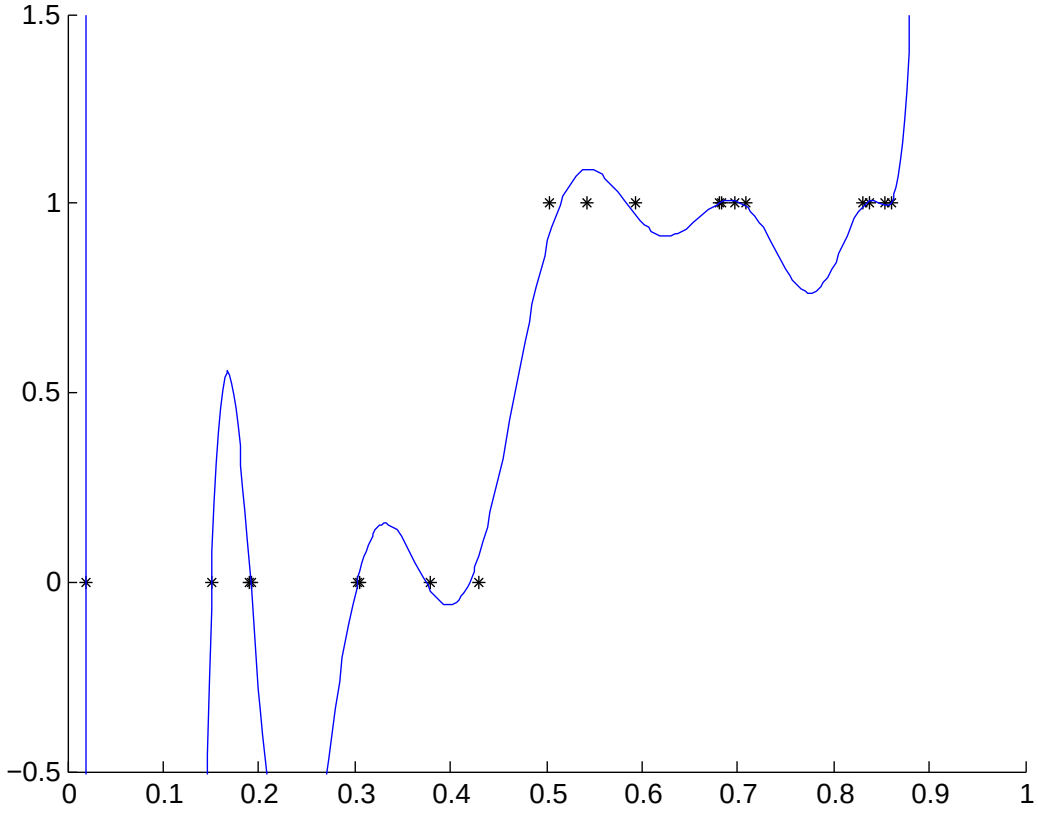
Dilemma

Example: polynomial fitting



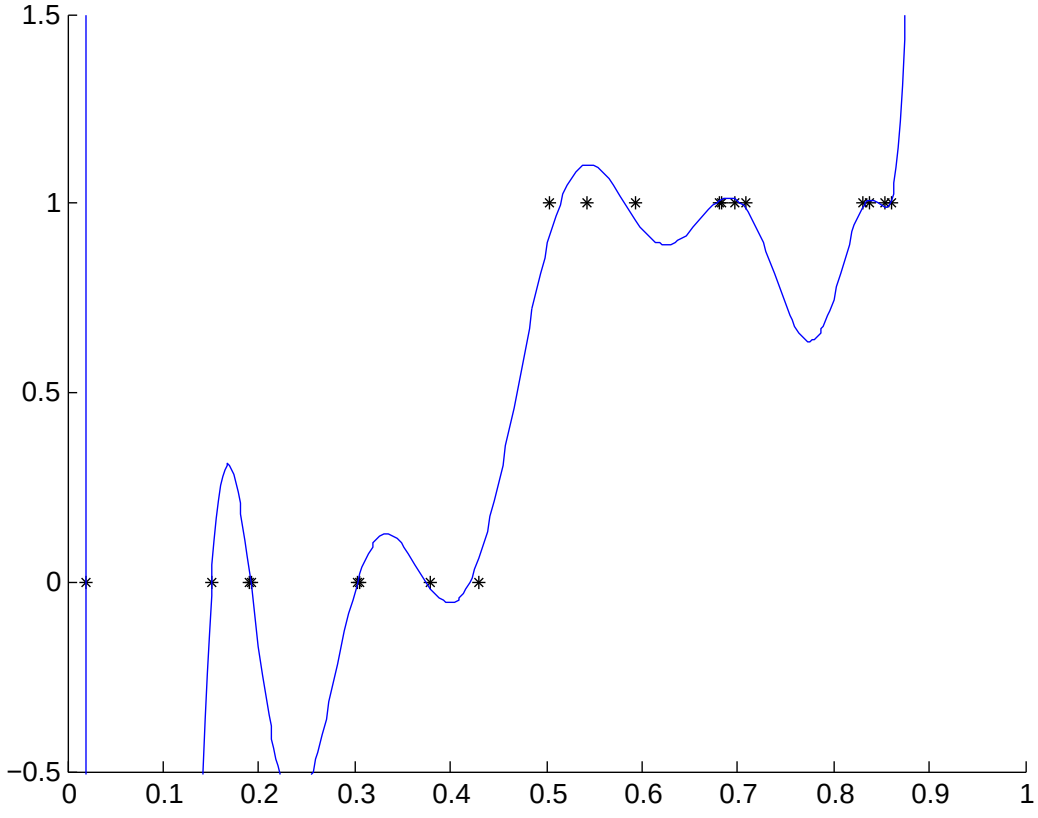
Dilemma

Example: polynomial fitting



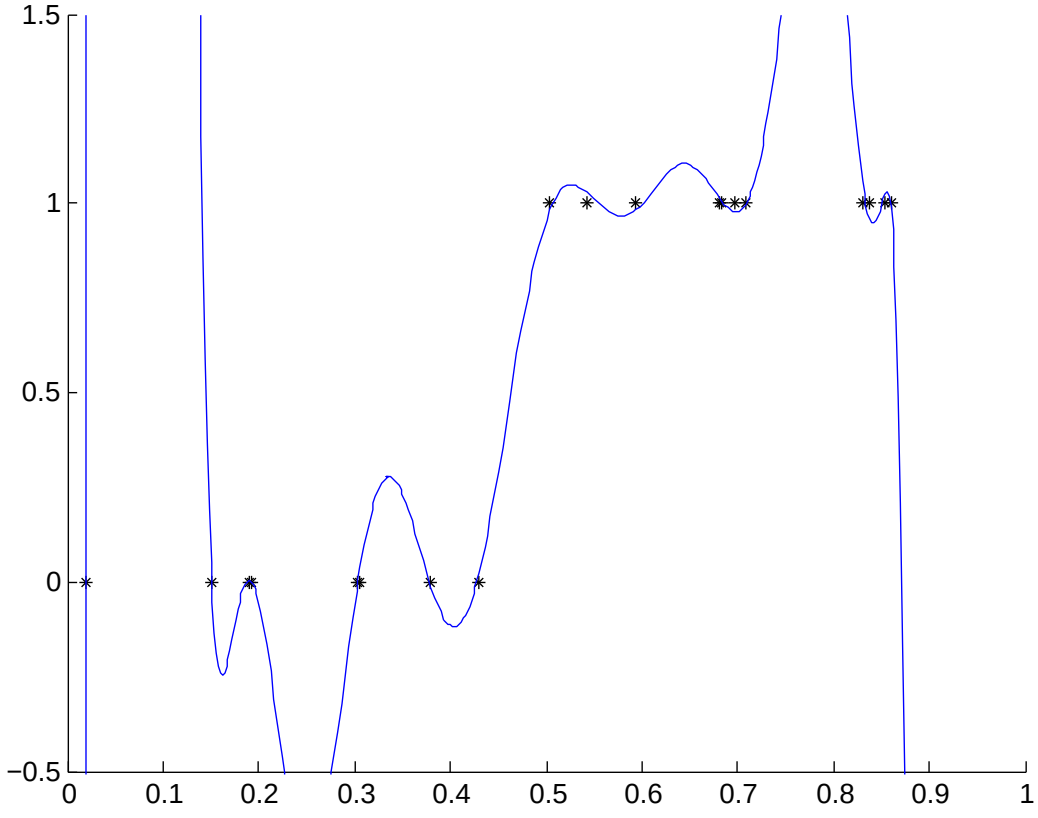
Dilemma

Example: polynomial fitting



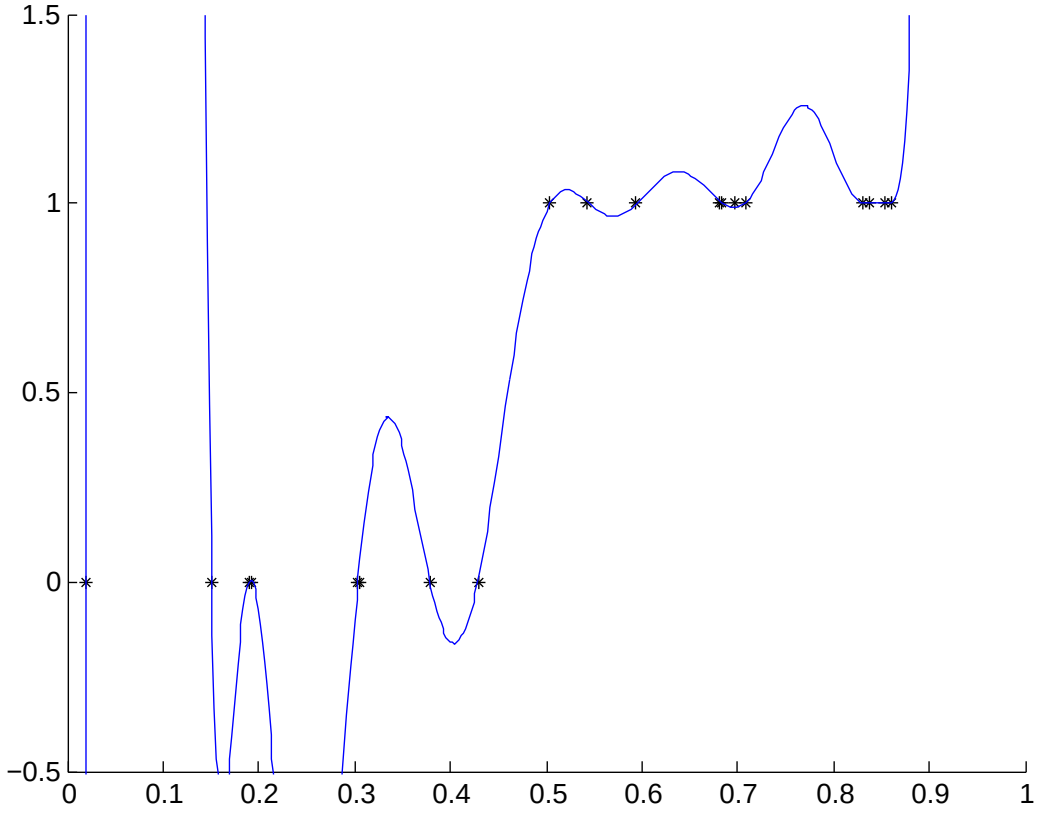
Dilemma

Example: polynomial fitting



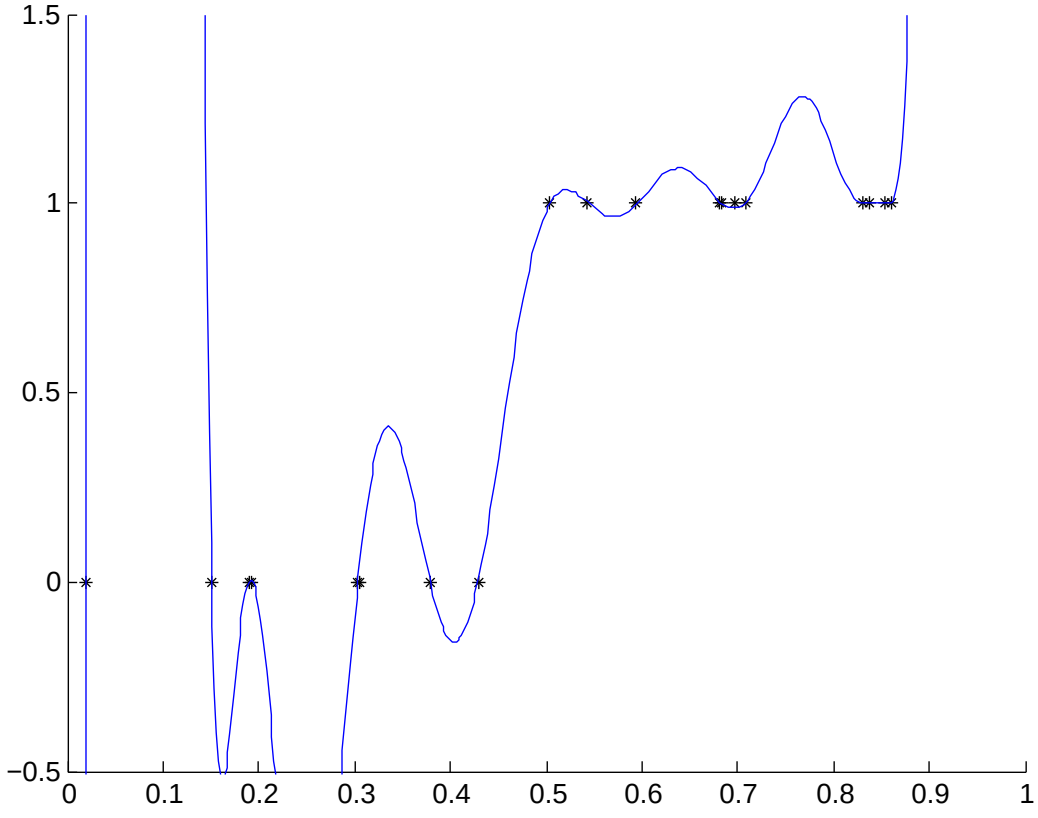
Dilemma

Example: polynomial fitting



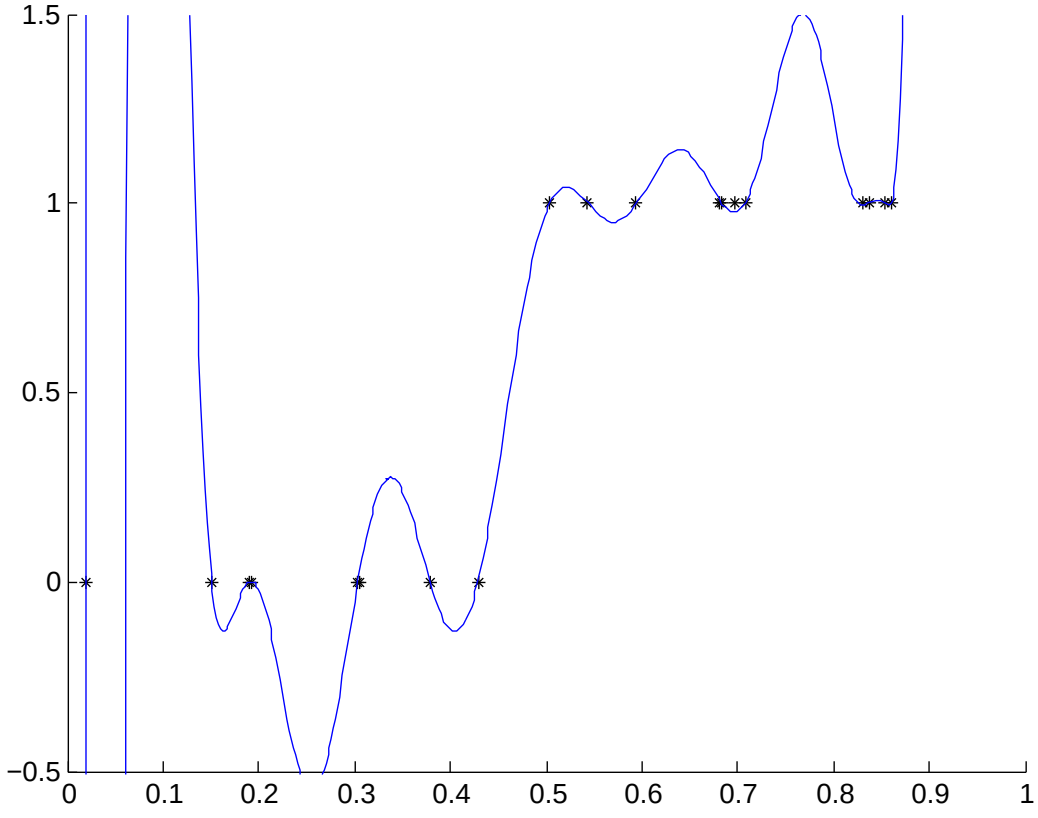
Dilemma

Example: polynomial fitting



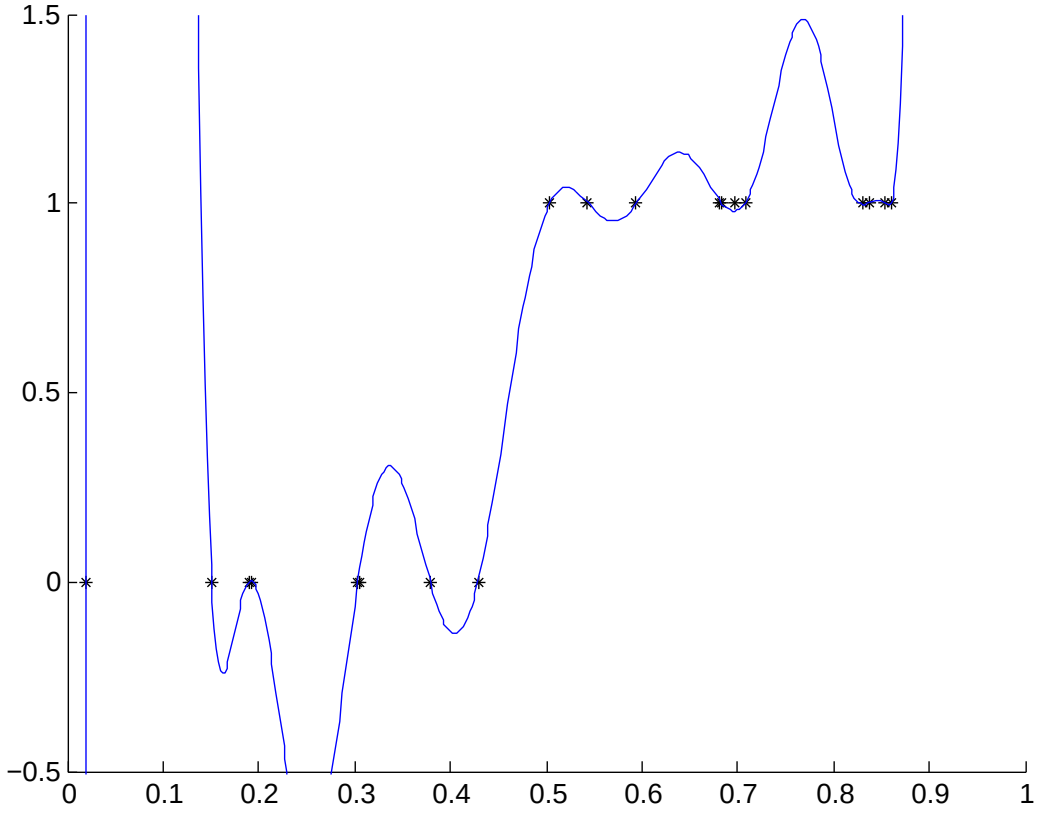
Dilemma

Example: polynomial fitting



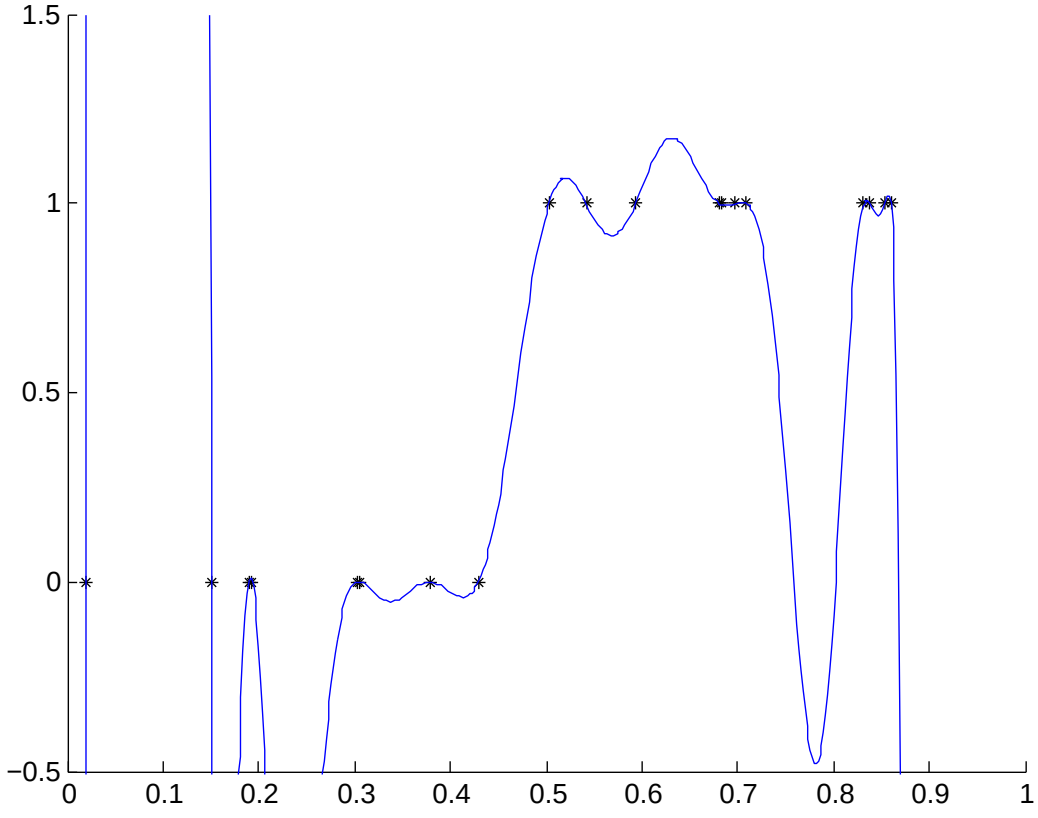
Dilemma

Example: polynomial fitting



Dilemma

Example: polynomial fitting



Overfitting versus underfitting

too many features	risks	overfitting
too few features	risks	underfitting

Strategies

Feature selection
- choose "right" set of basis functions

Regularization
- "smooth" functions by limiting size of weights

Overfitting versus underfitting

Regularization

"smoothing"

Limit slope of hypothesis function

$\min_{\mathbf{w}} L(\Phi\mathbf{w}; \mathbf{y}) + \beta\|\mathbf{w}\|$ where $\beta \geq 0$ is a regularization parameter

Tradeoff between minimizing error and size of $\mathbf{w}$

How to measure size of $\mathbf{w}$?

L_2 norm $\rightarrow$ leads to kernels

L_1 norm $\rightarrow$ leads to sparsity

Euclidean regularization

Euclidean regularization

Penalize $\mathbf{w}$ by its (squared) Euclidean norm

$$\min_{\mathbf{w}} L(\Phi\mathbf{w}; \mathbf{y}) + \frac{\beta}{2}\|\mathbf{w}\|_2^2$$

$\beta > 0$ a regularization parameter

The L_2^2 regularizer is a convenient choice because

$\frac{1}{2}\|\mathbf{w}\|_2^2$ is

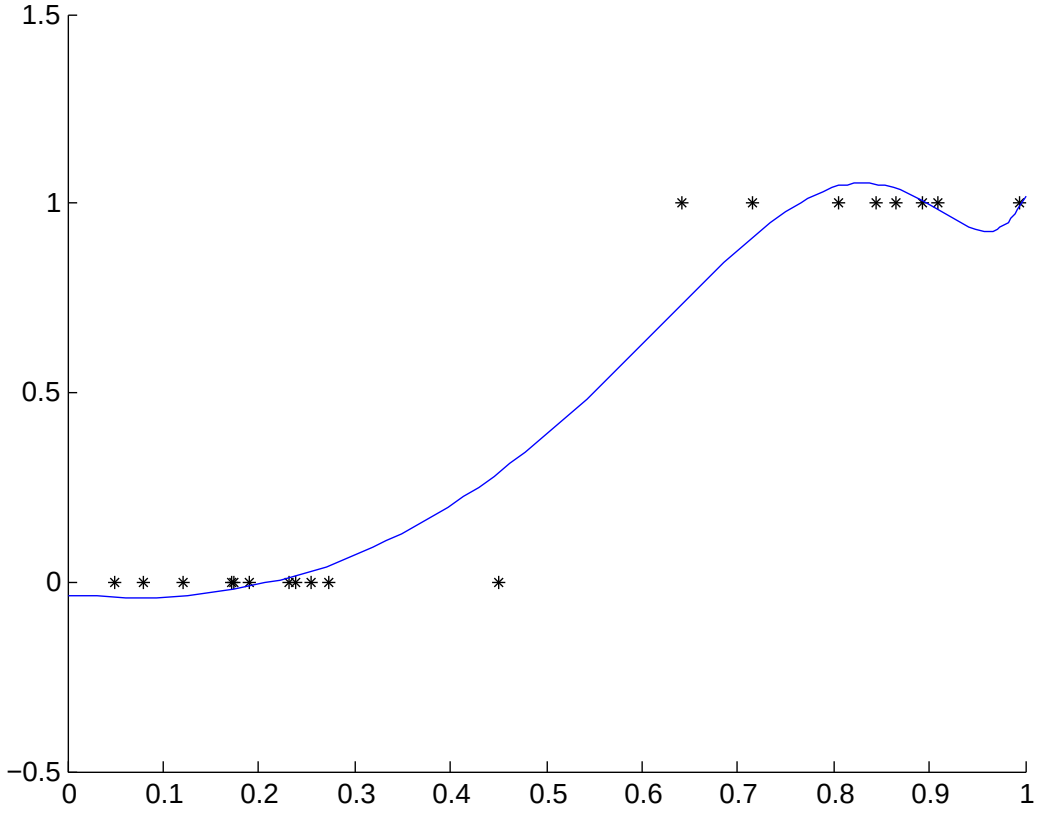
- convex
- smooth
- simple (e.g. $\nabla_{\mathbf{w}} = \mathbf{w}$)

More importantly

Euclidean regularization leads to an amazing generalization beyond finite dimensional feature vectors

Euclidean regularization

Example: polynomial fitting



Important property of Euclidean regularization

Simple representer theorem

For any L and any increasing R, if

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} L(\Phi\mathbf{w}; \mathbf{y}) + R(\|\mathbf{w}\|_2^2)$$

exists, then $\mathbf{w}^* = \Phi'\mathbf{a}^*$ for some $\mathbf{a}^*$

Proof

Since $\mathbf{w}^* = \mathbf{w}_0^* + \mathbf{w}_1^*$ for $\mathbf{w}_1^* \in \text{rowspan}(\Phi)$ and $\mathbf{w}_0^* \perp \text{rowspan}(\Phi)$

$$\mathbf{w}_1^* = \Phi'\mathbf{a}^* \text{ for some } \mathbf{a}^*$$
$$\Phi\mathbf{w}^* = \Phi\mathbf{w}_0^* + \Phi\mathbf{w}_1^* = \Phi\mathbf{w}_1^*$$
$$\|\mathbf{w}^*\|_2^2 = \|\mathbf{w}_0^* + \mathbf{w}_1^*\|_2^2 = \|\mathbf{w}_0^*\|_2^2 + \|\mathbf{w}_1^*\|_2^2$$

If $\mathbf{w}_0^* \neq 0$ then

$$L(\Phi\mathbf{w}^*; \mathbf{y}) + R(\|\mathbf{w}^*\|_2^2) = L(\Phi\mathbf{w}_1^*; \mathbf{y}) + R(\|\mathbf{w}_0^*\|_2^2 + \|\mathbf{w}_1^*\|_2^2)$$
$$> L(\Phi\mathbf{w}_1^*; \mathbf{y}) + R(\|\mathbf{w}_1^*\|_2^2) \text{ contradiction } \blacksquare$$

Important property of Euclidean regularization

Equivalent adjoint formulation

Can instead solve for example weights $\mathbf{a}$, where $\mathbf{w} = \Phi'\mathbf{a}$

Original training	$\min_{\mathbf{w}} L(\Phi\mathbf{w}; \mathbf{y}) + \frac{\beta}{2}\mathbf{w}'\mathbf{w}$
Original prediction	$\mathbf{x} \mapsto \hat{y} = \mathbf{w}^{*\prime}\phi(\mathbf{x})$

Adjoint training	$\min_{\mathbf{a}} L(\Phi\Phi'\mathbf{a}; \mathbf{y}) + \frac{\beta}{2}\mathbf{a}'\Phi\Phi'\mathbf{a}$
Adjoint prediction	$\mathbf{x} \mapsto \hat{y} = \mathbf{a}^{*\prime}\Phi\phi(\mathbf{x})$

Equivalent! by simple representer theorem

Key observation

Adjoint formulation does not require feature vectors
only inner products between feature vectors

Important property of Euclidean regularization

Equivalent kernel formulation

Assume a function $\kappa(\cdot,\cdot)$ that computes inner products $\kappa(\Phi_{i:},\Phi_{j:}) = \Phi_{i:}\Phi'_{j:}$

Kernel training	$\min_{\mathbf{a}} L(K\mathbf{a};\mathbf{y}) + \frac{\beta}{2}\mathbf{a}'K\mathbf{a}$
	where $K_{ij} = \kappa(X_{i:},X_{j:})$
Kernel prediction	$\mathbf{x} \mapsto \hat{y} = \mathbf{a}^{*\prime}\mathbf{k}$
	where $\mathbf{k}_i = \kappa(X_{i:},\mathbf{x}')$

Example

Polynomial feature vector

$$\phi(x) = \left(\sqrt{\binom{d}{0}}, \sqrt{\binom{d}{1}}x, \sqrt{\binom{d}{2}}x^2, ..., \sqrt{\binom{d}{d}}x^d\right)$$

Corresponding kernel

$$\kappa(x_1,x_2) = (x_1x_2+1)^d = \sum_{i=0}^{d}\binom{d}{i}x_1^i x_2^i = \phi(x_1)'\phi(x_2)$$

Direct computation can be arbitrarily more efficient

Kernels

Pairwise similarity measure on a set of objects $\mathcal{X}$

vectors, strings, sentences, documents, trees, graphs

Instead of features, choose a kernel $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$

symmetric: $\kappa(\mathbf{x}_1, \mathbf{x}_2) = \kappa(\mathbf{x}_2, \mathbf{x}_1)$

semidefinite:

for any finite set $\{\mathbf{x}_1, ..., \mathbf{x}_t\} \subset \mathcal{X}$

$$\begin{bmatrix} \kappa(\mathbf{x}_1, \mathbf{x}_1) & \cdots & \kappa(\mathbf{x}_1, \mathbf{x}_t) \\ \vdots & & \vdots \\ \kappa(\mathbf{x}_t, \mathbf{x}_1) & \cdots & \kappa(\mathbf{x}_t, \mathbf{x}_t) \end{bmatrix} \succeq 0$$

Strictly generalizes finite dimensional feature vectors

Example

$\kappa(\mathbf{x}_1, \mathbf{x}_2) = \exp(-\frac{1}{2\sigma^2}\|\mathbf{x}_1 - \mathbf{x}_2\|_2^2)$

does not have finite dimensional feature representation

Kernels

Reproducing kernel Hilbert space

Given a symmetric, semidefinite operator $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$
coherently defines a Hilbert space

- Basis given by features $\phi_{\hat{\mathbf{x}}}$ for all $\hat{\mathbf{x}} \in \mathcal{X}$
- $\mathcal{H}_0 =$ finite linear combinations of $\phi_{\hat{\mathbf{x}}}$
- Define $\langle \sum_{i=1}^n a_i \phi_{\hat{\mathbf{x}}_i}, \sum_{j=1}^m b_i \phi_{\hat{\mathbf{x}}_j} \rangle_{\mathcal{H}} = \sum_{i=1}^n \sum_{j=1}^m a_i b_j \kappa(\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j)$
- Define $\| \sum_{i=1}^n a_i \phi_{\hat{\mathbf{x}}_i} \|_{\mathcal{H}} = \langle \sum_{i=1}^n a_i \phi_{\hat{\mathbf{x}}_i}, \sum_{i=1}^n a_i \phi_{\hat{\mathbf{x}}_i} \rangle_{\mathcal{H}}^{1/2}$
- $\mathcal{H} =$ completion of $\mathcal{H}_0$ under $\| \cdot \|_{\mathcal{H}}$

Representer theorem still holds

For any L and any increasing R

$$h^* = \arg\min_{h \in \mathcal{H}} L(h(X); \mathbf{y}) + R(\|h\|_{\mathcal{H}}^2)$$

can be written $h^*(\cdot) = \sum_{i=1}^t \mathbf{a}_i^* \phi_{X_{i:}}(\cdot) = \sum_{i=1}^t \mathbf{a}_i^* \kappa(X_{i:}, \cdot)$ for some $\mathbf{a}^*$

Feature selection

Feature selection

Problem

Choose a subset of feature functions to use

- i.e. choose a subset of $\{\phi_1, ..., \phi_L\}$

Difficulty

- 2^L subsets
- Intractable to enumerate
- Finding a feature subset of size $\leq d$ that minimizes training error
  NP-hard in general

Idea

Use a convex relaxation of feature selection

- L_1 regularization

L_1 regularization

L_1 regularized training problem

$$\min_{\mathbf{w}} L(\Phi\mathbf{w}; \mathbf{y}) + \beta\|\mathbf{w}\|_1$$

$\beta \geq 0$ regularization parameter

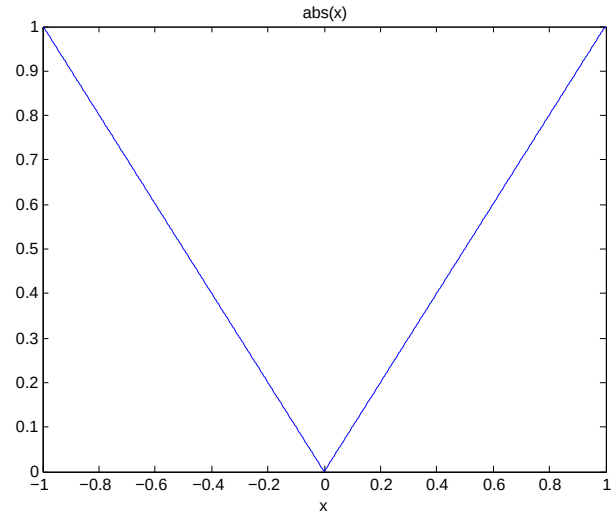
Properties

- Convex in $\mathbf{w}$
- Nonsmooth
- Implicitly encourages sparsity (i.e. $w_j = 0$ for some j)
- Provides a tractable relaxation of feature selection

L_1 regularization

Why does L_1 regularization yield sparse solutions?

$\|\mathbf{w}\|_1 = \sum_{j=1}^{L} |w_j|$



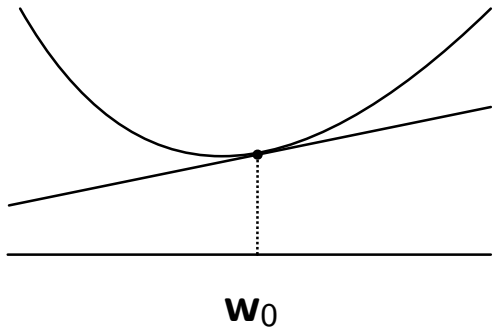
$$\frac{\partial}{\partial w_j} = \begin{cases} 1 & \text{if } w_j > 0 \\ -1 & \text{if } w_j < 0 \\ \text{undef} & \text{if } w_j = 0 \end{cases}$$

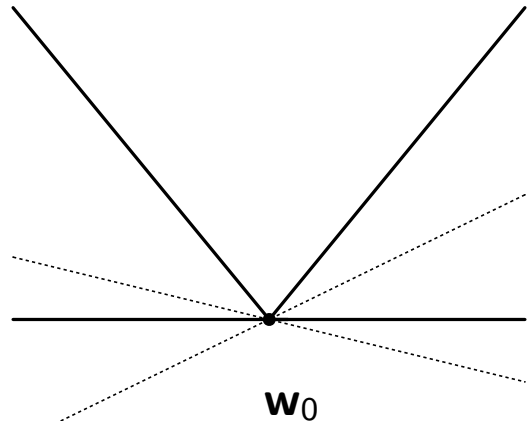
L_1 regularization

Subgradients

For a differentiable convex function ℓ, always have

$$\ell(\mathbf{w}) \geq \ell(\mathbf{w}_0) + (\mathbf{w} - \mathbf{w}_0)'\nabla\ell(\mathbf{w}_0)$$





A **subgradient** at $\mathbf{w}_0$

is any $\mathbf{d}_0$ such that $\ell(\mathbf{w}) \geq \ell(\mathbf{w}_0) + (\mathbf{w} - \mathbf{w}_0)'\mathbf{d}_0$

Theorem

If ℓ differentiable at $\mathbf{w}_0$ then $\mathbf{d}_0$ is unique and $\mathbf{d}_0 = \nabla\ell(\mathbf{w}_0)$.

What if ℓ not differentiable at $\mathbf{w}_0$?

Then $\mathbf{d}_0$ is not unique

L_1 regularization

Implicit feature selection

Consider a descent step from a current $\mathbf{w}$

$$\frac{\partial}{\partial w_j} = \beta \text{sign}(w_j) + \sum_{i=1}^{t} L'(\Phi_{i:}\mathbf{w}; y_i)\Phi_{ij} \quad \text{if } w_j \neq 0$$

What if current value of $w_j = 0$?

if $\quad |\sum_{i=1}^{t} L'(\Phi_{i:}\mathbf{w}; y_i)\Phi_{ij}| < \beta$

w_j stays at 0; that is, no local descent from $w_j = 0$

else

can reduce the objective
by moving w_j in direction of $-\sum_{i=1}^{t} L'(\Phi_{i:}\mathbf{w}; y_i)\Phi_{ij}$

L_1 regularization

Efficiently solvable if L convex

$$\min_{\mathbf{w}} L(\Phi\mathbf{w}; \mathbf{y}) + \beta\|\mathbf{w}\|_1$$
$$= \min_{\mathbf{w},\boldsymbol{\xi}} L(\Phi\mathbf{w}; \mathbf{y}) + \beta\mathbf{1}'\boldsymbol{\xi} \text{ subject to } \boldsymbol{\xi} \geq \mathbf{w},\ \boldsymbol{\xi} \geq -\mathbf{w}$$

- convex objective (if L convex)
- linear constraints

E.g.

If $L(\hat{y}; y) = (\hat{y} - y)^2$ get a quadratic program
If $L(\hat{y}; y) = |\hat{y} - y|$ get a linear program

Problem

L_1 regularization blocks the representer theorem!

How to combine kernels and feature selection?

Naive approach does not work:

$$\beta_1 \|\mathbf{w}\|_1 + \frac{\beta_2}{2} \|\mathbf{w}\|_2^2$$

blocks representer theorem—no equivalent adjoint form

—hence no equivalent kernel form

Fortunately

It is possible to combine L_1 and L_2 keeping kernels,
but requires an indirect approach:

- introduce separate feature selection variables $\boldsymbol{\mu}$
- exploit Fenchel conjugate of L

Kernel selection

Kernel selection

Relating feature and kernel selection

Consider a feature representation Φ, a $t \times L$ matrix

Get kernel matrix

$$K = \Phi\Phi' = \sum_{j=1}^{L} \Phi_{:j}\Phi'_{:j} = \sum_{j=1}^{L} K_j$$

I.e. each basis feature $\Phi_{:j}$ corresponds to a rank 1 kernel matrix

$$K_j = \Phi_{:j}\Phi'_{:j}$$

Kernel selection

Introduce auxiliary feature/kernel selection variables

Let $1 \geq \boldsymbol{\mu} \geq 0$ be a vector of selection weights

Consider $\tilde{\Phi} = \Phi\Delta(\boldsymbol{\mu})^{1/2}$
($\Delta(\boldsymbol{\mu})$ denotes putting $\boldsymbol{\mu}$ on main diagonal of square matrix)

Get

$$\tilde{K} = \tilde{\Phi}\tilde{\Phi}' = \Phi\Delta(\boldsymbol{\mu})\Phi' = \sum_{j=1}^{L} \mu_j \Phi_{:j}\Phi'_{:j} = \sum_{j=1}^{L} \mu_j K_j$$

Will use $\boldsymbol{\mu}$ to select features/kernels

Kernel selection

Get training problem

$$\min_{0 \leq \boldsymbol{\mu} \leq 1} \min_{\mathbf{w}} L(\Phi \Delta(\boldsymbol{\mu})^{1/2} \mathbf{w}; \mathbf{y}) + \frac{\beta_2}{2} \|\mathbf{w}\|_2^2 + \beta_1 \|\boldsymbol{\mu}\|_1$$

Challenge

- convex in $\mathbf{w}$ given $\boldsymbol{\mu}$
- convex in $\boldsymbol{\mu}$ given $\mathbf{w}$
- But not jointly convex in $\mathbf{w}$ and $\boldsymbol{\mu}$

Can a global solution be computed efficiently?

Yes

Aside: Fenchel duality

Given a function $\ell(\mathbf{w})$

Define its Fenchel conjugate as

$$\ell^*(\boldsymbol{\alpha}) = \sup_{\mathbf{w}} \boldsymbol{\alpha}'\mathbf{w} - \ell(\mathbf{w})$$

Guaranteed to be convex in $\boldsymbol{\alpha}$ (since max of linear is convex)

Strong duality property

If $\ell(\mathbf{w})$ is a closed, convex function then $\ell^{**}(\mathbf{w}) = \ell(\mathbf{w})$

That is

$$\ell(\mathbf{w}) = \sup_{\boldsymbol{\alpha}} \boldsymbol{\alpha}'\mathbf{w} - \ell^*(\boldsymbol{\alpha})$$

Fenchel duality

Equivalent dual problem

Can get an equivalent reformulation of L_2 regularized training

$$\min_{\mathbf{w}} L(\Phi\mathbf{w}; \mathbf{y}) + \frac{\beta}{2}\|\mathbf{w}\|_2^2 \qquad \text{primal problem}$$

$$= \max_{\boldsymbol{\alpha}} -L^*(\boldsymbol{\alpha}; \mathbf{y}) - \frac{1}{2\beta}\boldsymbol{\alpha}' K \boldsymbol{\alpha} \qquad \text{dual problem}$$

where

$$L^*(\boldsymbol{\alpha}; \mathbf{y}) = \sup_{\hat{\mathbf{y}}} \boldsymbol{\alpha}'\hat{\mathbf{y}} - L(\hat{\mathbf{y}}; \mathbf{y}) \quad \text{and} \quad K = \Phi\Phi'$$

Important

The representer theorem holds so expressible in terms of a kernel

(* some weak technical conditions apply on L)

Kernel selection

Putting the pieces together

Add an L_1 regularizer on $\boldsymbol{\mu}$ and jointly optimize

$$\min_{0 \leq \boldsymbol{\mu} \leq 1} \min_{\mathbf{w}} L(\Phi \Delta(\boldsymbol{\mu})^{1/2} \mathbf{w}; \mathbf{y}) + \frac{\beta_2}{2} \|\mathbf{w}\|_2^2 + \beta_1 \mathbf{1}' \boldsymbol{\mu}$$

$$= \min_{0 \leq \boldsymbol{\mu} \leq 1} \max_{\boldsymbol{\alpha}} -L^*(\boldsymbol{\alpha}; \mathbf{y}) - \frac{1}{2\beta_2} \sum_{j=1}^{t} \mu_j \boldsymbol{\alpha}' K_j \boldsymbol{\alpha} + \beta_1 \mathbf{1}' \boldsymbol{\mu}$$

The latter form is a concave-convex program—no local minima

Various computational strategies exist

equivalent convex reformulation of latter form above

Boosting

Boosting

Incremental training for large or infinite bases

Saw previously that kernels could be used to implicitly train with large or infinite bases

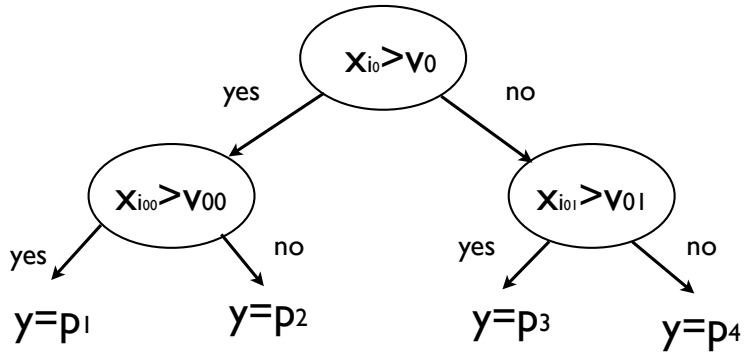
But what if you have a large basis but not corresponding efficient kernel?

Two classical examples of training with infinite bases:

- decision trees
- feedforward neural networks

Decision trees

Special generalized linear function



Each path corresponds to a single feature

$$\phi_1(\mathbf{x}) = 1_{(x_{i_0} > v_0 \text{ and } x_{i_{00}} > v_{00})} \quad \in \{0, 1\}$$

$$\phi_2(\mathbf{x}) = 1_{(x_{i_0} > v_0 \text{ and } x_{i_{00}} \leq v_{00})} \quad \in \{0, 1\}$$

$$\phi_3(\mathbf{x}) = 1_{(x_{i_0} \leq v_0 \text{ and } x_{i_{01}} > v_{01})} \quad \in \{0, 1\}$$

$$\phi_3(\mathbf{x}) = 1_{(x_{i_0} \leq v_0 \text{ and } x_{i_{01}} \leq v_{01})} \quad \in \{0, 1\}$$

Same predictor can be represented by a generalized linear function

$$\hat{y} = \begin{bmatrix} \phi_1(\mathbf{x}) \\ \phi_2(\mathbf{x}) \\ \phi_3(\mathbf{x}) \\ \phi_4(\mathbf{x}) \end{bmatrix}' \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ p_4 \end{bmatrix}$$

Decision trees

Linear predictor plus tree constraint over bases

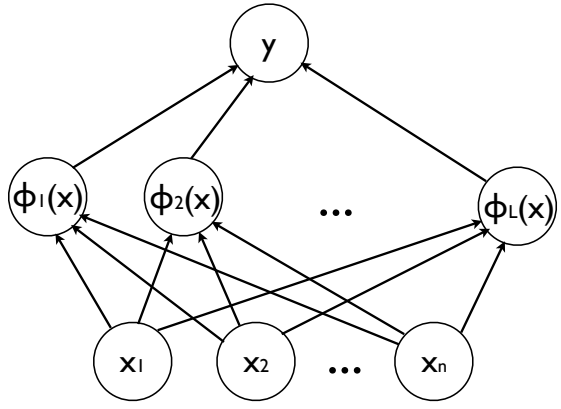
- NP-hard to find best tree of bounded size
- Standard training algorithms are heuristics
- Linear predictor = weighted forest of decision trees
- Could just learn a linear predictor over same basis

Difficulty

- Set of bases is infinite
- How to learn a linear model in such a case?
  (don't have an equivalent kernel)

Multilayer neural network

Two-layer feedforward neural networks



Difficulty

- Optimally training a 2-layer neural network is NP-hard
- Fixing # bases creates intractable feature selection problem
- Backpropagation training = local optimization heuristic

Can 2-layer neural network be trained efficiently?

- Idea: use L_1 regularization instead of feature selection
- Set of bases is infinite

Boosting

Incremental strategy for training a linear model

over a large or even infinite feature set

Strategy

- Do not enumerate basis
- Grow basis one function at a time by greedy procedure

Maintain a sparse model

At stage k have selected $k-1$ bases

$$h_{k-1}(\mathbf{x}) = \sum_{j=1}^{k-1} w_j \phi_j(\mathbf{x})$$

Boosting

Greedy coordinate descent

Let

$$\ell(\mathbf{w}^{(k-1)}) := \sum_{i=1}^{t} L(h_{k-1}(\mathbf{x}_i); y_i)$$
$$= \sum_{i=1}^{t} L\Big(\sum_{j=1}^{k-1} w_j \phi_j(\mathbf{x}_i); y_i\Big)$$

Score of a new candidate feature ϕ_k

$$\left.\frac{\partial \ell}{\partial w_k}\right|_{\mathbf{w}=\mathbf{w}^{(k-1)}} = \sum_{i=1}^{t} L'\Big(\sum_{j=1}^{k-1} w_j \phi_j(\mathbf{x}_i); y_i\Big)\phi_k(\mathbf{x}_i)$$
$$= \mathbf{L}'_{k-1}\boldsymbol{\phi}_k$$

Boosting

Weak learning problem

Find steepest coordinate descent direction

$$\min_{\phi_k \in \Phi} \mathbf{L}'_{k-1} \phi_k$$

Behaves like weighted misclassification error

If $\mathbf{L}'_{k-1} \phi_k \geq 0$, halt

Once ϕ_k selected, solve for w_k by line search

$$\min_{w_k} \sum_{i=1}^{t} L(h_{k-1}(\mathbf{x}_i) + w_k \phi_k(\mathbf{x}_i); y_i)$$

Boosting

Convergence theorem

If

- L convex
- L' is b-Lipschitz continuous:

$$\|\mathbf{L}'(h) - \mathbf{L}'(g)\| \leq b\|h - g\| \quad \text{for some } b < \infty$$

- $\|\phi_k\| \leq B$ for some $B < \infty$
- Φ negation closed: $\phi \in \Phi \Rightarrow -\phi \in \Phi$
- Weak learner is approximately optimal: $\exists 0 < \gamma \leq 1$ such that

$$\mathbf{L}'_{k-1}\phi_k \leq \gamma \mathbf{L}'_{k-1}\phi^*_k \quad \text{for all } k$$

Then

- h_k converges to a global minimizer of L

(Mason et al. 2000)

Boosting

Adding regularization

Can use same strategy to converge to minimizer of

$$\min_{h \in \text{span}(\Phi)} L(h(X); \mathbf{y}) + \beta \|\mathbf{w}\|_1$$

or

$$\min_{h \in \text{span}(\Phi)} L(h(X); \mathbf{y}) + \frac{\beta}{2} \|\mathbf{w}\|_2^2$$

provided totally corrective weight update used.
That is, given ϕ_k, solve

$$\min_{w_1,\ldots,w_k} \sum_{i=1}^{t} L\Big(\sum_{j=1}^{k-1} w_j \phi_j(\mathbf{x}_i); y_i \Big) + R([w_1, \ldots, w_k])$$

I.e. jointly re-optimize w_k with all previous weights

Boosting

Catch

Requires approximately optimal weak learner
(Warning: many papers sweep this little detail under the rug)

Good news

Tractable for some cases

- E.g. $\Phi =$ "decision stumps", $\quad \phi(\mathbf{x}) = 1_{(x_j < c)}$ or $1_{(x_j \geq c)}$

Bad news

Intractable for almost all interesting bases

- E.g. $\Phi =$ "perceptrons" (linear threshold classifiers)
  NP-hard, even to approximate (Höffgen & Simon 1992)

Boosting

Crazy idea: sample bases randomly!

Can still guarantee a near optimal hypothesis with high probability

Set up

Let $H = \{h(\mathbf{x}) : \int w(\theta)\phi_\theta(\mathbf{x})p(\theta)d\theta \text{ such that } \|w(\theta)\| \leq c \ \forall\theta\}$

Assume $\|\phi\| \leq 1$ for all $\phi \in \Phi$ and L is b-Lipschitz

Sample $\theta_1, ..., \theta_s \sim p(\theta)$

Let $\hat{H} = \{h(\mathbf{x}) = \sum_j w_j\phi_{\theta_j}(\mathbf{x}) : |w_j| \leq c \ \forall j\}$

Theorem

For any $h \in H$ with probability at least $1 - \delta$
there exists some $\hat{h} \in \hat{H}$ such that

$$L(\hat{h}(X); \mathbf{y}) \leq L(h(X); \mathbf{y}) + bc\sqrt{\frac{t}{s}}\left(1 + \sqrt{8\log\frac{1}{\delta}}\right)$$