Machine Learning 101a

Jan Peters
Gerhard Neumann

Purpose of this Lecture

- Statistics and Math Refresher
- Foundations of machine learning tools for robotics
  - We focus on regression methods and general principles
    - Often needed in robotics
  - More on machine learning in general: Machine Learning – Statistical Approaches 1

Content of this Lecture

- Math and Statistics Refresher
- What is Machine Learning?
- Model-Selection
- Linear Regression
  - Gauss’ Approach
    - Gauss’ Approach
    - Frequentist Approach
    - Bayesian Approach

Statistics Refresher:
Sweet memories from High School...

What is a random variable X ?

X is a variable whose value x is subject to variations due to chance

What is a distribution $p(X = x)$?

Describes the probability that the random variable will be equal to a certain value

What is an expectation?

$$\mathbb{E}_{p(X)}[f(x)] = \int p(x)f(x)dx$$

Statistics Refresher:
Sweet memories from High School...

- **What is a joint, a conditional and a marginal distribution?**

$$p(x, y) = p(y|x)p(x)$$

$$\text{joint} = \text{conditional} \times \text{marginal}$$

- **What is independence of random variables?**

$$p(x, y) = p(x)p(y)$$

- **What does marginalization mean?**

$$p(x) = \int p(x, y)dy$$

- **And finally… what is Bayes Theorem?**

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)}$$

Math Refresher:
Some more fancy math…

- From now on, matrices are your friends… derivatives too

$$\frac{dy}{d\boldsymbol{x}} = \left[\frac{dy}{dx_1}, \ldots, \frac{dy}{dx_n}\right] \qquad \frac{d\boldsymbol{y}}{d\boldsymbol{x}} = \begin{bmatrix} \frac{dy_1}{dx_1} & \cdots & \frac{dy_1}{dx_n} \\ \vdots & \vdots & \vdots \\ \frac{dy_m}{dx_1} & \cdots & \frac{dy_m}{dx_n} \end{bmatrix}$$

- Some more matrix calculus

$$\frac{d\boldsymbol{a}^T\boldsymbol{x}}{d\boldsymbol{x}} = \frac{d\boldsymbol{x}^T\boldsymbol{a}}{d\boldsymbol{x}} = \boldsymbol{a}^T \qquad \frac{d\boldsymbol{A}\boldsymbol{x}}{d\boldsymbol{x}} = \boldsymbol{A} \qquad \frac{d\boldsymbol{x}^T\boldsymbol{A}\boldsymbol{x}}{d\boldsymbol{x}} = \boldsymbol{x}^T(\boldsymbol{A} + \boldsymbol{A}^T)$$

- Need more ? Wikipedia on Matrix Calculus or The Matrix Cookbook

Math Refresher:
Inverse of matrices

How can we invert a matrix that is not a square matrix?

$$\boldsymbol{J} \in \mathbb{R}^{n \times m}$$

Left-Pseudo Inverse:

$$\boldsymbol{J}^{\dagger}\boldsymbol{J} = \underbrace{(\boldsymbol{J}^T\boldsymbol{J})^{-1}\boldsymbol{J}^T}_{\text{left multiplied}}\boldsymbol{J} = \boldsymbol{I}_m$$

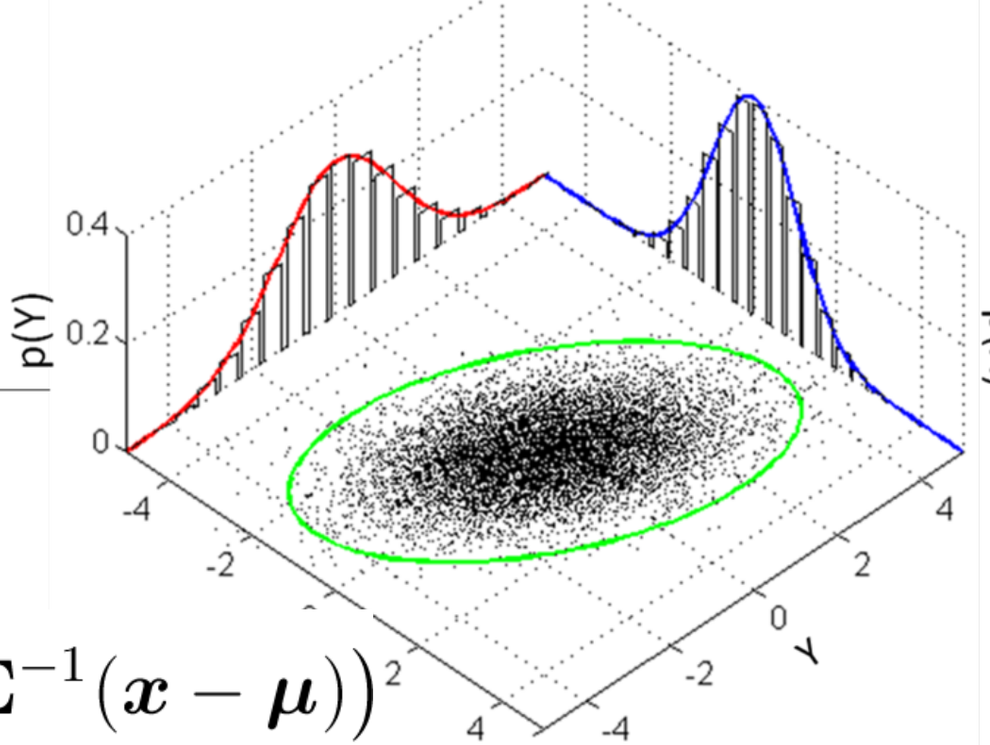
works, if J has full column rank

Right Pseudo Inverse:

$$\boldsymbol{J}\boldsymbol{J}^{\dagger} = \boldsymbol{J}\underbrace{\boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{J}^T)^{-1}}_{\text{right multiplied}} = \boldsymbol{I}_n$$

works, if J has full row rank

Statistics Refresher: Meet some old friends…



- **Gaussian Distribution**

$$\mathcal{N}\left(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}\right)=\frac{1}{\left|2\pi\boldsymbol{\Sigma}\right|^{\frac{1}{2}}}\exp\left(-0.5(\boldsymbol{x}-\boldsymbol{\mu})^{T}\boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})\right)^{2}$$

- Covariance matrix $\boldsymbol{\Sigma}$ captures linear correlation

- Product: Gaussian stays Gaussian

$$\mathcal{N}\left(\boldsymbol{x}|\boldsymbol{a},\boldsymbol{A}\right)\mathcal{N}\left(\boldsymbol{x}|\boldsymbol{b},\boldsymbol{B}\right)=\mathcal{N}(\boldsymbol{x}|\dots)$$

- Mean is also the mode

$$\boldsymbol{\mu}=\mathrm{argmax}_{\boldsymbol{x}}\mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu},\boldsymbol{\Sigma})$$

Statistics Refresher:
Meet some old friends…

- Joint from Marginal and Conditional

$$\mathcal{N}\left(\boldsymbol{x}|\boldsymbol{a},\boldsymbol{A}\right)\mathcal{N}\left(\boldsymbol{y}|\boldsymbol{b}+\boldsymbol{F}\boldsymbol{x},\boldsymbol{B}\right)=\mathcal{N}\left(\left[\begin{array}{c}\boldsymbol{x}\\ \boldsymbol{y}\end{array}\right]\middle|\left[\begin{array}{c}\boldsymbol{a}\\ \boldsymbol{b}+\boldsymbol{F}\boldsymbol{a}\end{array}\right],\left[\begin{array}{cc}\boldsymbol{A} & \boldsymbol{F}^T\boldsymbol{A}^T\\ \boldsymbol{F}^T\boldsymbol{A}^T & \boldsymbol{B}+\boldsymbol{F}\boldsymbol{A}^T\boldsymbol{F}^T\end{array}\right]\right)$$

$$p(\boldsymbol{x})p(\boldsymbol{y}|\boldsymbol{x}) \Rightarrow p(\boldsymbol{x},\boldsymbol{y})$$

- Marginal and Conditional Gaussian from Joint

$$\mathcal{N}\left(\left[\begin{array}{c}\boldsymbol{x}\\ \boldsymbol{y}\end{array}\right]\middle|\left[\begin{array}{c}\boldsymbol{a}\\ \boldsymbol{b}\end{array}\right],\left[\begin{array}{cc}\boldsymbol{A} & \boldsymbol{C}\\ \boldsymbol{C}^T & \boldsymbol{B}\end{array}\right]\right)=\mathcal{N}\left(\boldsymbol{x}|\boldsymbol{a},\boldsymbol{A}\right)\mathcal{N}\left(\boldsymbol{y}|\boldsymbol{b}+\boldsymbol{C}^T\boldsymbol{A}^{-1}(\boldsymbol{x}-\boldsymbol{a}),\boldsymbol{B}-\boldsymbol{C}^T\boldsymbol{A}^{-1}\boldsymbol{C}\right)$$

$$p(\boldsymbol{x},\boldsymbol{y}) \Rightarrow p(\boldsymbol{x}) \qquad p(\boldsymbol{y}|\boldsymbol{x})$$

Statistics Refresher:
Meet some old friends…

- Bayes Theorem for Gaussians

$$p(\boldsymbol{x})p(\boldsymbol{y}|\boldsymbol{x}) \Rightarrow p(\boldsymbol{x}|\boldsymbol{y})$$

$$p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{0}, \boldsymbol{A})$$
$$p(\boldsymbol{y}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{F}\boldsymbol{x}, \sigma^2\boldsymbol{I})$$

$$\Rightarrow$$

$$p(\boldsymbol{y}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{F}\boldsymbol{a}, \sigma^2\boldsymbol{I} + \boldsymbol{F}\boldsymbol{A}\boldsymbol{F}^T)$$
$$p(\boldsymbol{x}|\boldsymbol{y}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\Sigma}\boldsymbol{F}^T\boldsymbol{y}, \sigma^2\boldsymbol{\Sigma})$$

$$\boldsymbol{\Sigma} = (\boldsymbol{F}^T\boldsymbol{F} + \sigma^2\boldsymbol{A}^{-1})^{-1}$$

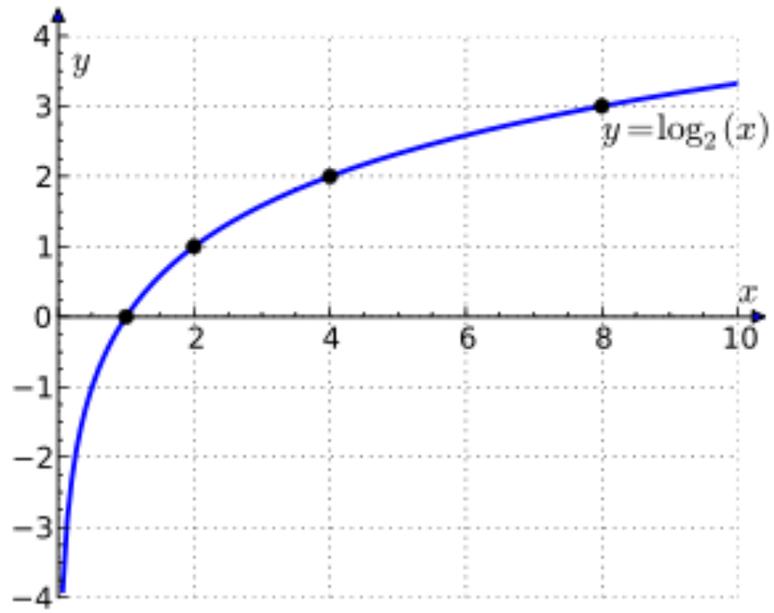
Damped Pseudo Inverse

- Not enough? Find more stuff here (credit to Marc Toussaint)

May I introduce you? The good old logarithm

It‘s monoton $a > b \Rightarrow \log(a) > \log(b)$

… but not boring, as:



- Product is easy… $\log \prod_{i=1}^{N} a_i = \sum_{i=1}^{N} \log a_i$
- Division a piece of cake… $\log \frac{1}{a} = -\log a$
- Exponents also… $\log\left(a^b\right) = b \log a$

Content of this Lecture

- Math and Statistics Refresher
- What is Machine Learning?
- Model-Selection
  - Linear Regression
    - Gauss’ Approach
    - Frequentist Approach
    - Bayesian Approach

Why Machine Learning

We are drowning in information and starving for knowledge.

-John Naisbitt.

Era of big data:

- In 2008 there are about 1 trillion web pages
- 20 hours of video are uploaded to YouTube every minute
- Walmart handles more than 1M transactions per hour and has databases containing more than 2.5 petabytes (2.5×10^{15}) of information.

➡ **No human being can deal with the data avalanche!**

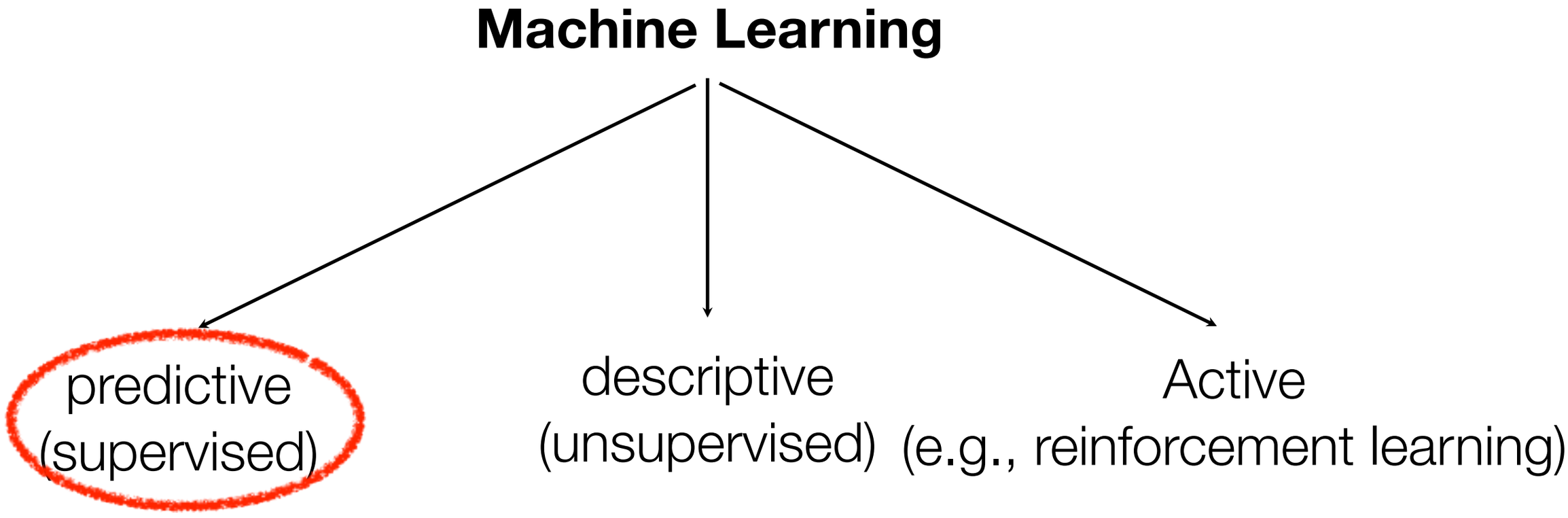
Why Machine Learning?

I keep saying the sexy job in the next ten years will be statisticians and machine learners. People think I'm joking, but who would've guessed that computer engineers would've been the sexy job of the 1990s? The ability to take data — to be able to understand it, to process it, to extract value from it, to visualize it, to communicate it — that's going to be a hugely important skill in the next decades.

Hal Varian, 2009

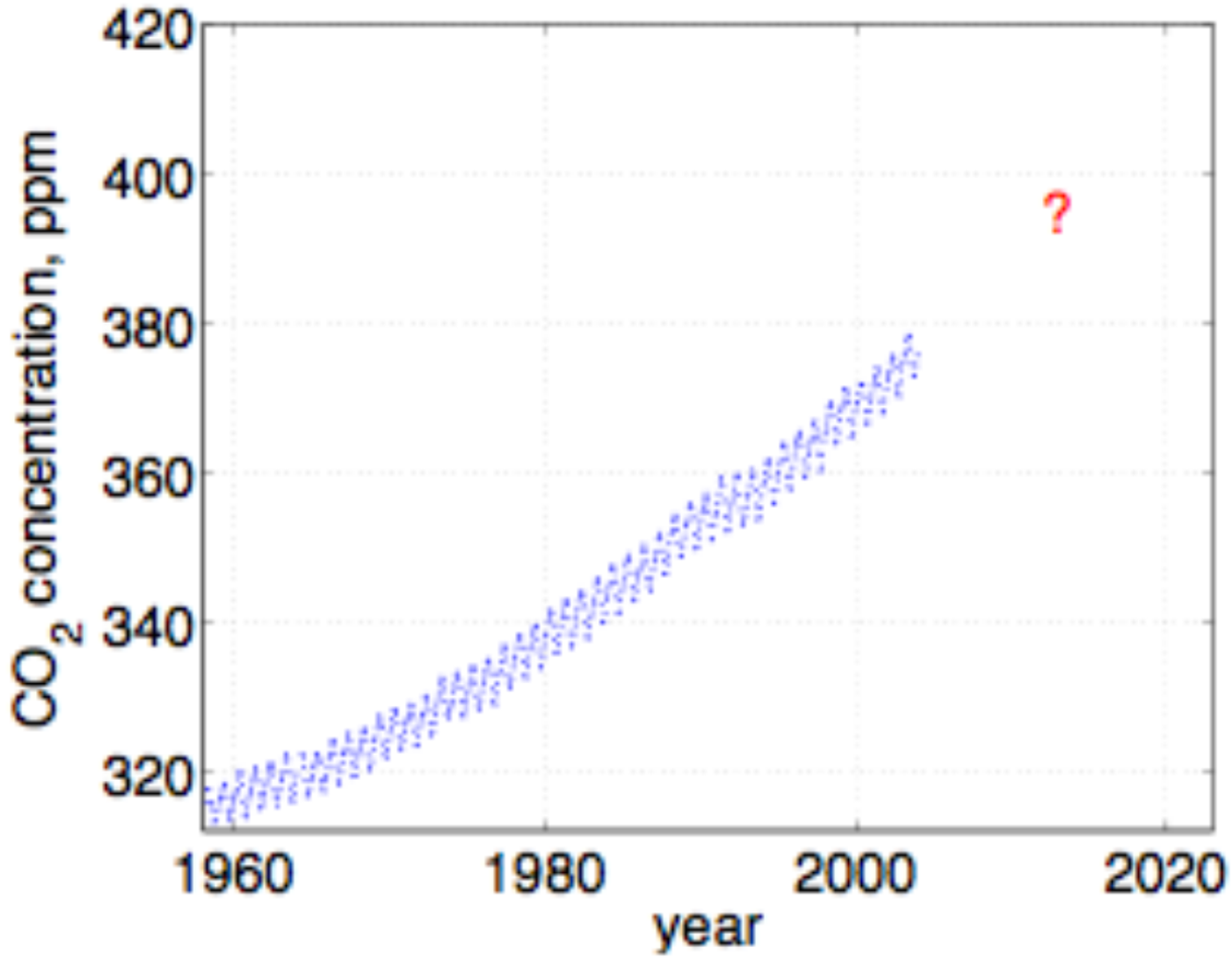
Chief Engineer of Google

Types of Machine Learning



Prediction Problem (=Supervised Learning)

What will be the CO^2 concentration in the future?

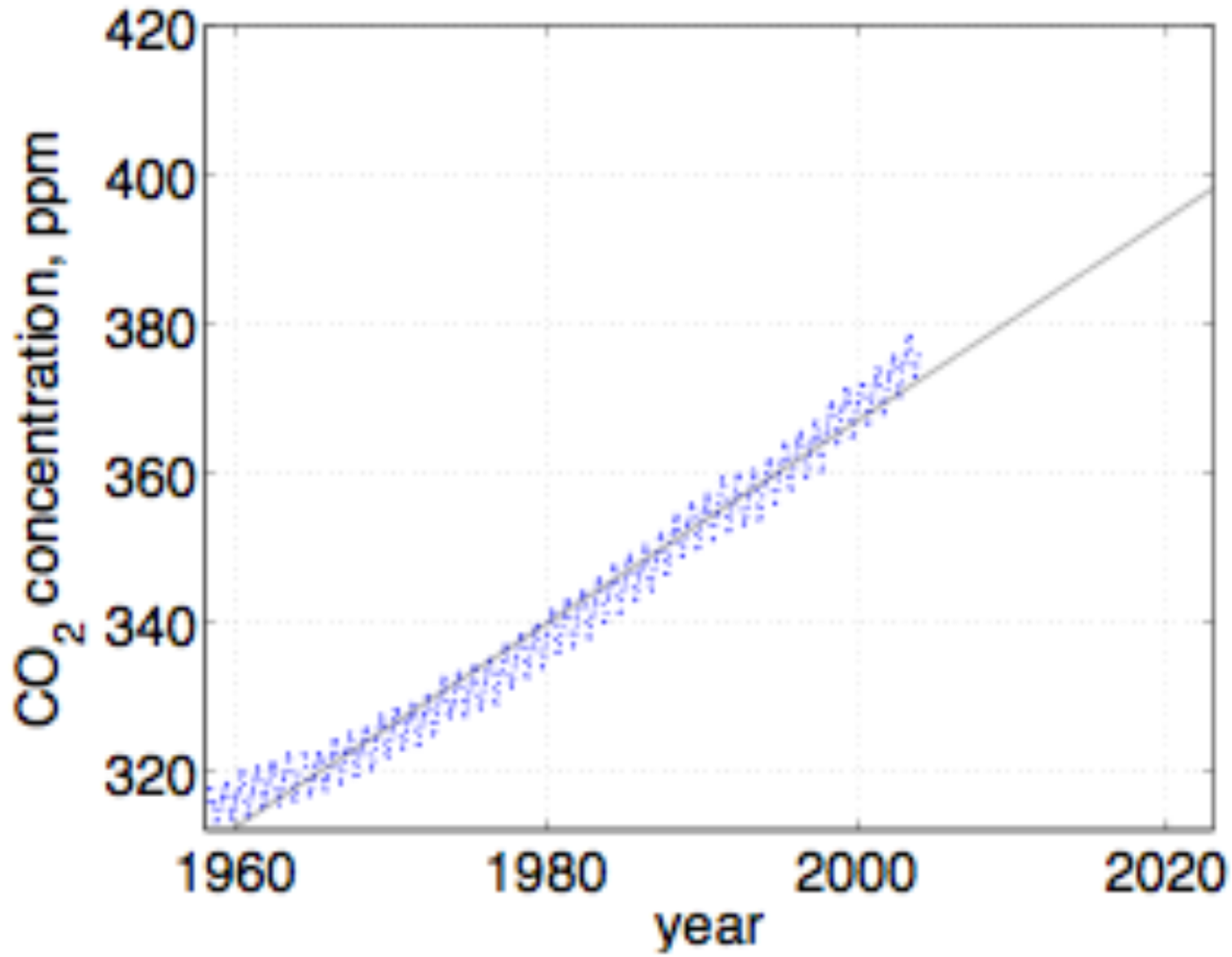


Prediction Problem (=Supervised Learning)

What will be the CO^2 concentration in the future?

Different prediction models possible

- Linear

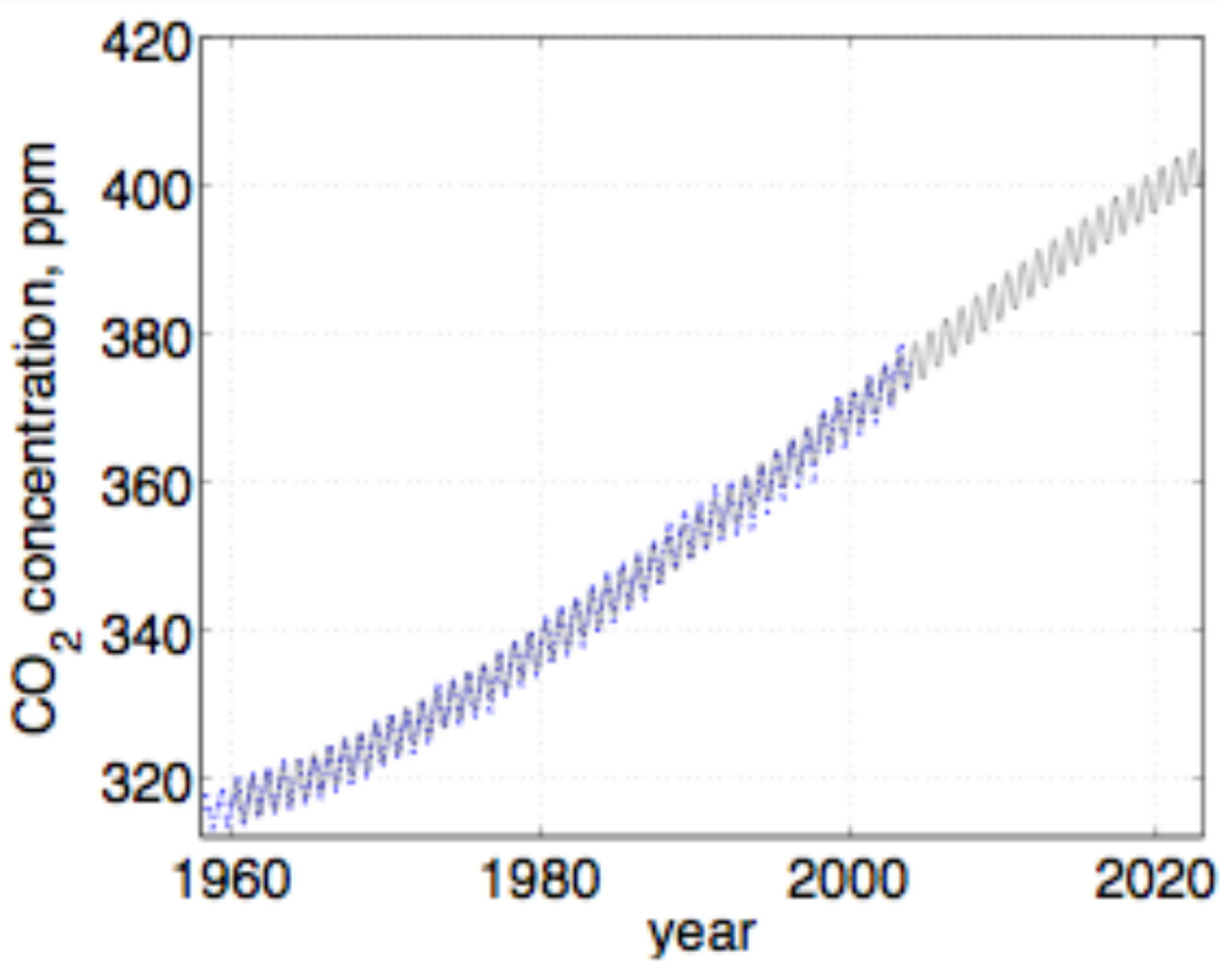


Prediction Problem (=Supervised Learning)

What will be the CO^2 concentration in the future?

Different prediction models possible

- Linear
- Exponential with seasonal trends



Formalization of Predictive Problems

In predictive problems, we have the following data-set

$$\mathscr{D} = \left\{ (\mathbf{x}_i, y_i) \middle| \, i = 1, 2, 3, \ldots, n \right\}$$

$$\boldsymbol{x} \ldots \text{inputs}, \; y \ldots \text{output} \; / \; \text{target}$$

Two most prominent examples are:

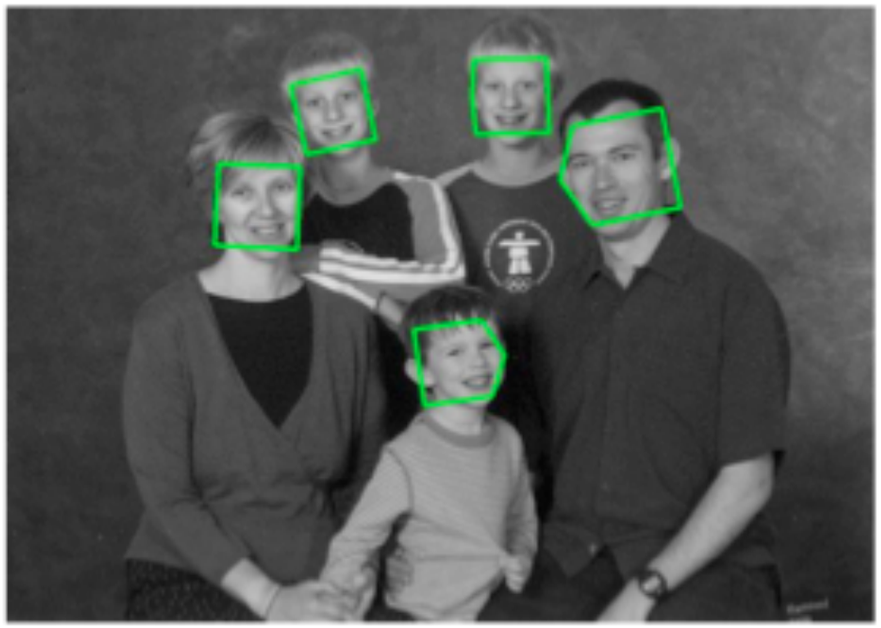
1. *Classification*: Discrete outputs or labels $y_q \in \{0, 1\}$.

   Most likely class: $y_q = \mathbf{argmax}_y p(y | \mathbf{x} = \mathbf{x}_q)$

2. *Regression*: Continuous outputs or labels $y_q \in \mathbb{R}$.

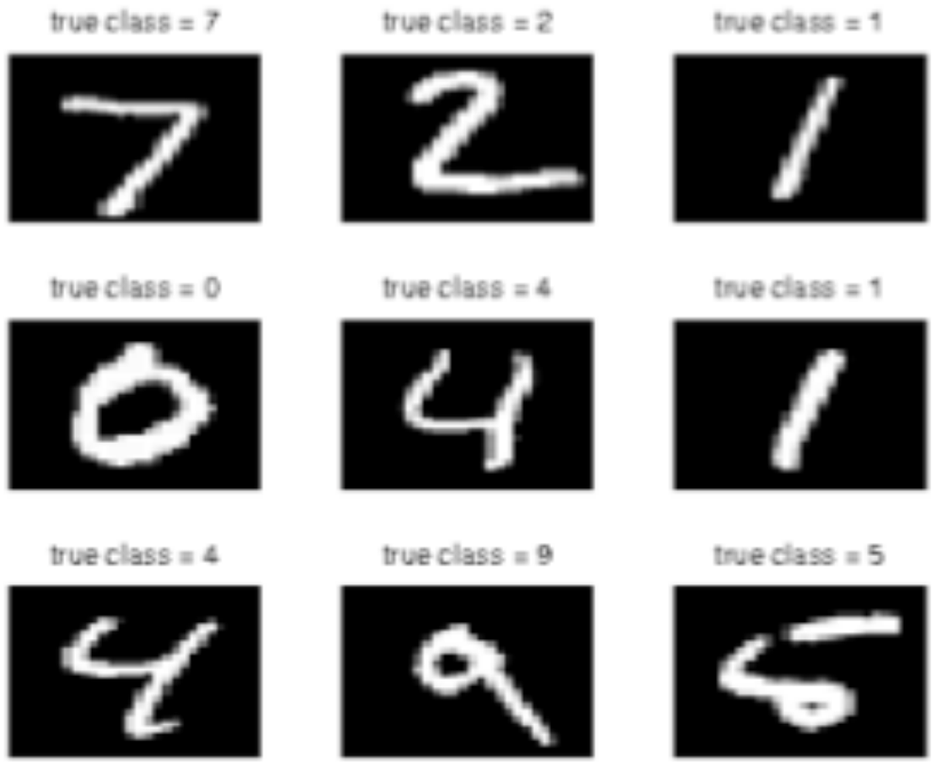
   Expected output: $y_q = E\{y | \mathbf{x} = \mathbf{x}_q\} = \int y p(y | \mathbf{x} = \mathbf{x}_q) d y.$

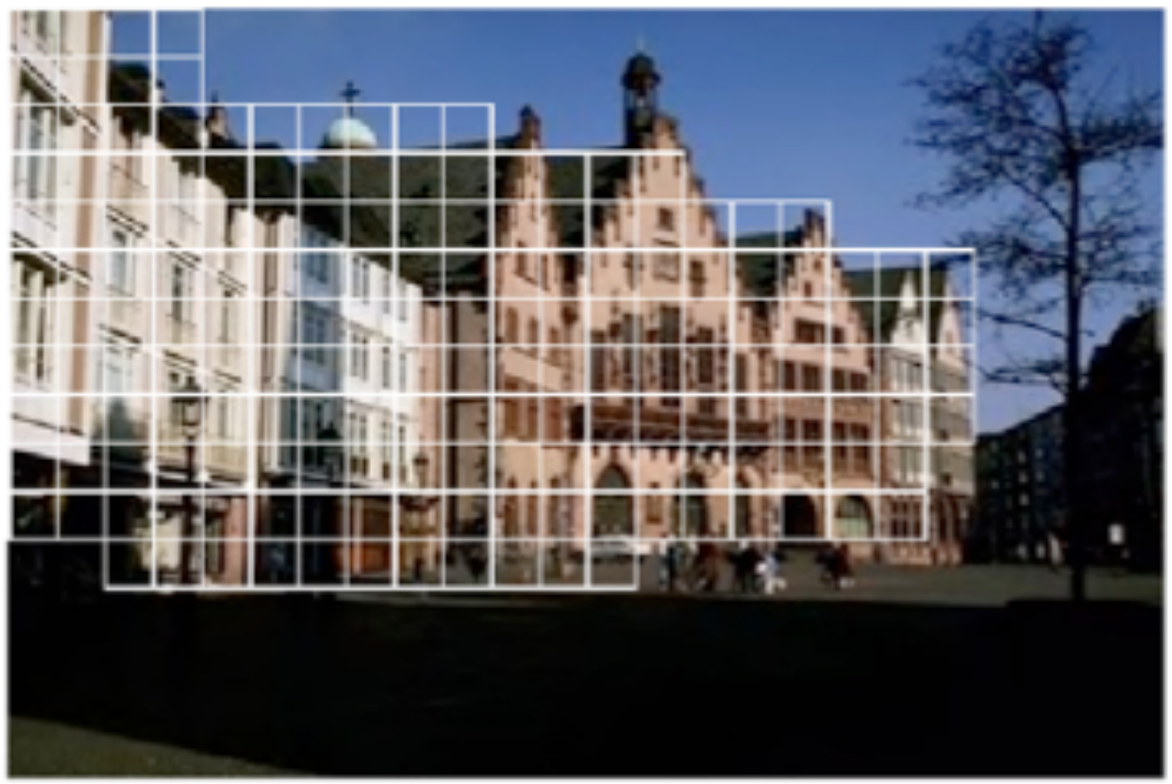
Examples of Classification



- Document classification, e.g., Spam Filtering

- Image classification: Classifying flowers, face detection, face recognition, handwriting recognition, ...





Examples of Regression

- Predict tomorrow's stock market price given current market conditions and other possible side information.

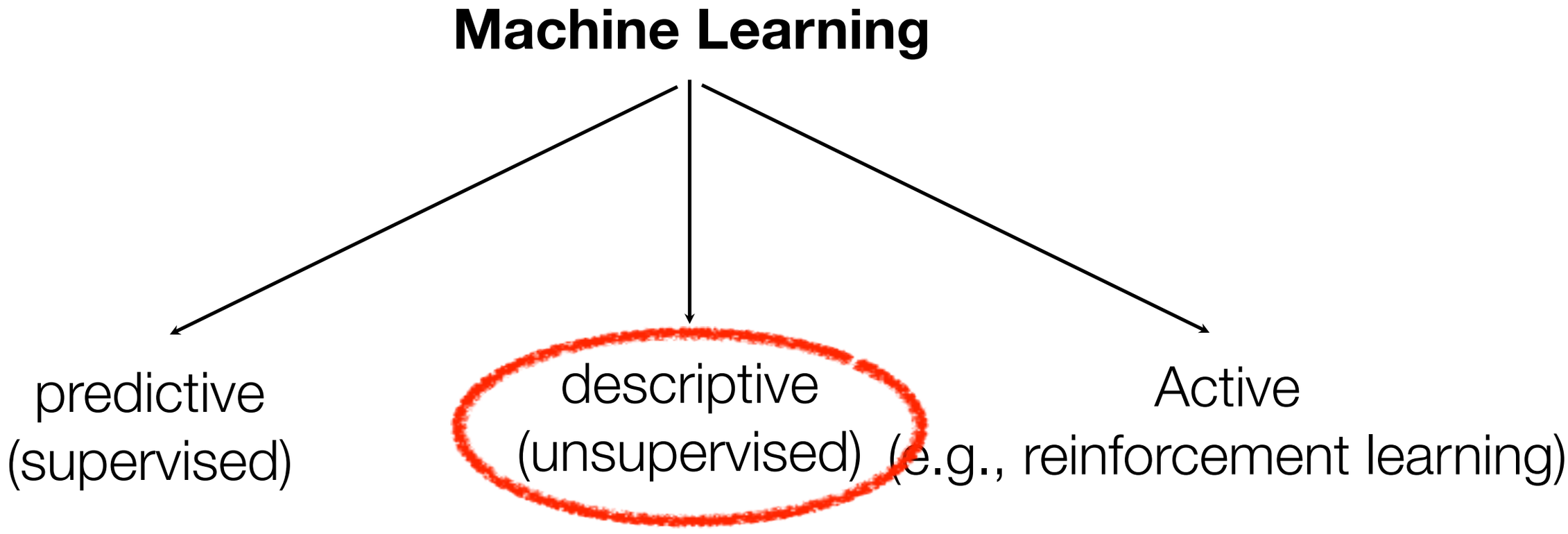
- Predict the amount of prostate specific antigen (PSA) in the body as a function of a number of different clinical measurements.

- Predict the temperature at any location inside a building using weather data, time, door sensors, etc.

- Predict the age of a viewer watching a given video on YouTube.

➡ Many problems in robotics can be addressed by regression!

Types of Machine Learning



Formalization of Descriptive Problems

In descriptive problems, we have

$$\mathscr{D} = \left\{ \mathbf{x}_i \middle| i = 1, 2, 3, \ldots, n \right\}$$

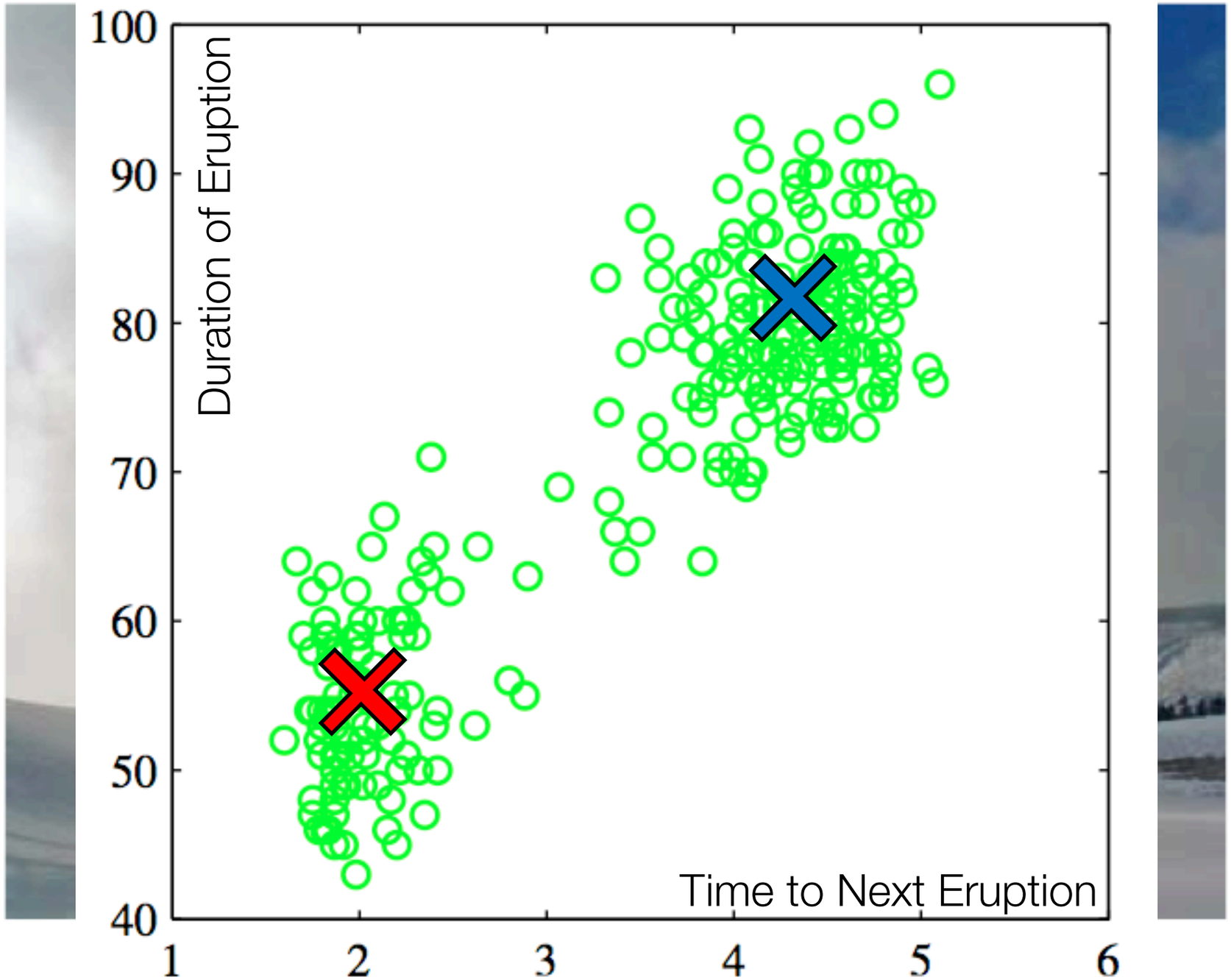
Three prominent examples are:

1. *Clustering*: Find groups of data which belong together.

2. *Dimensionality Reduction*: Find the latent dimension of your data.

3. *Density Estimation*: Find the probability of your data...

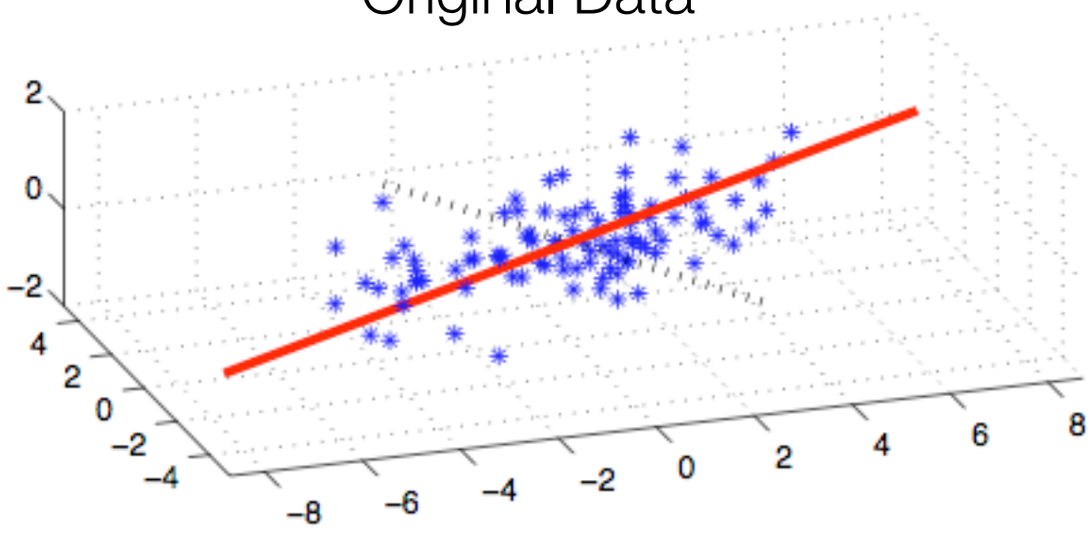
Old Faithful



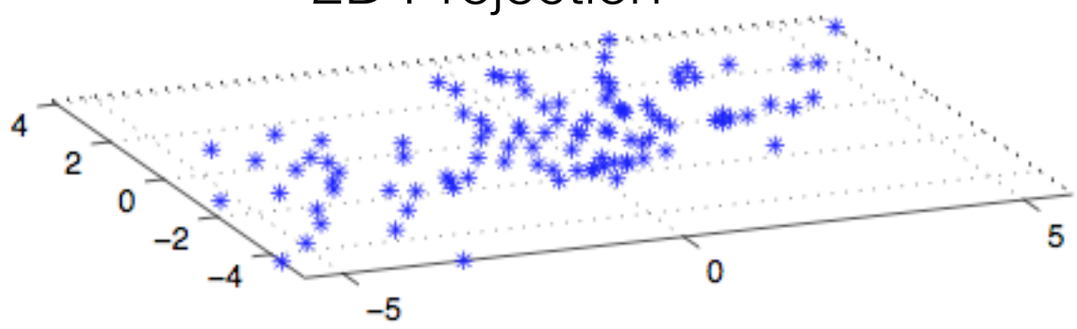
This is called Clustering!

Dimensionality Reduction

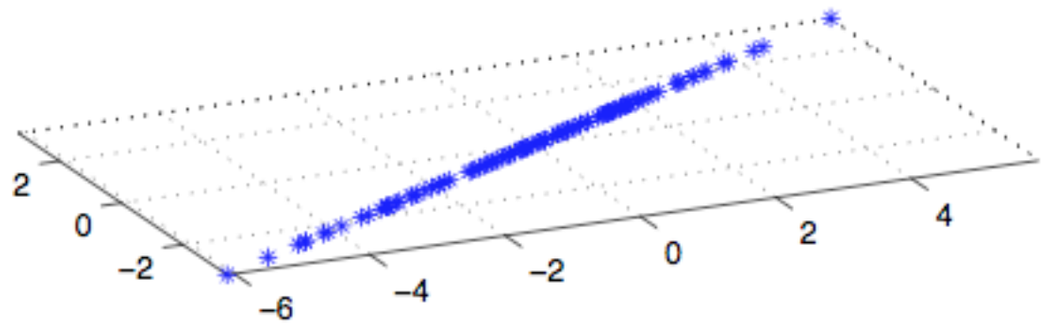
Original Data



2D Projection



1D Projection



This is called Dimensionality Reduction!

Dimensionality Reduction Example: Eigenfaces

How many faces do you need to characterize these?

Example: Eigenfaces

mean

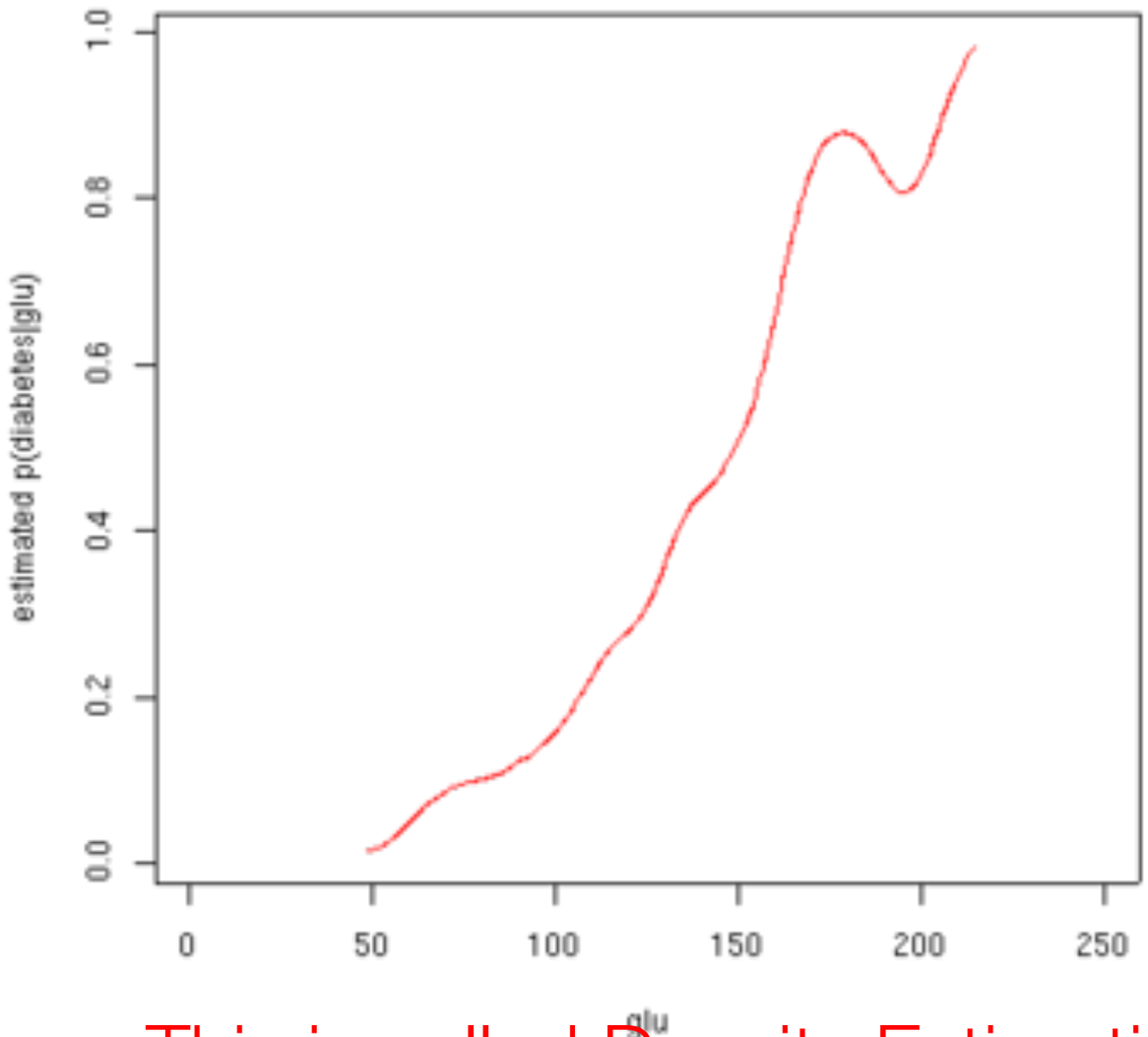
principal basis 1

principal basis 2

principal basis 3

Example: density of glu (plasma glucose concentration) for diabetes patients

Estimate relative occurance of a data point



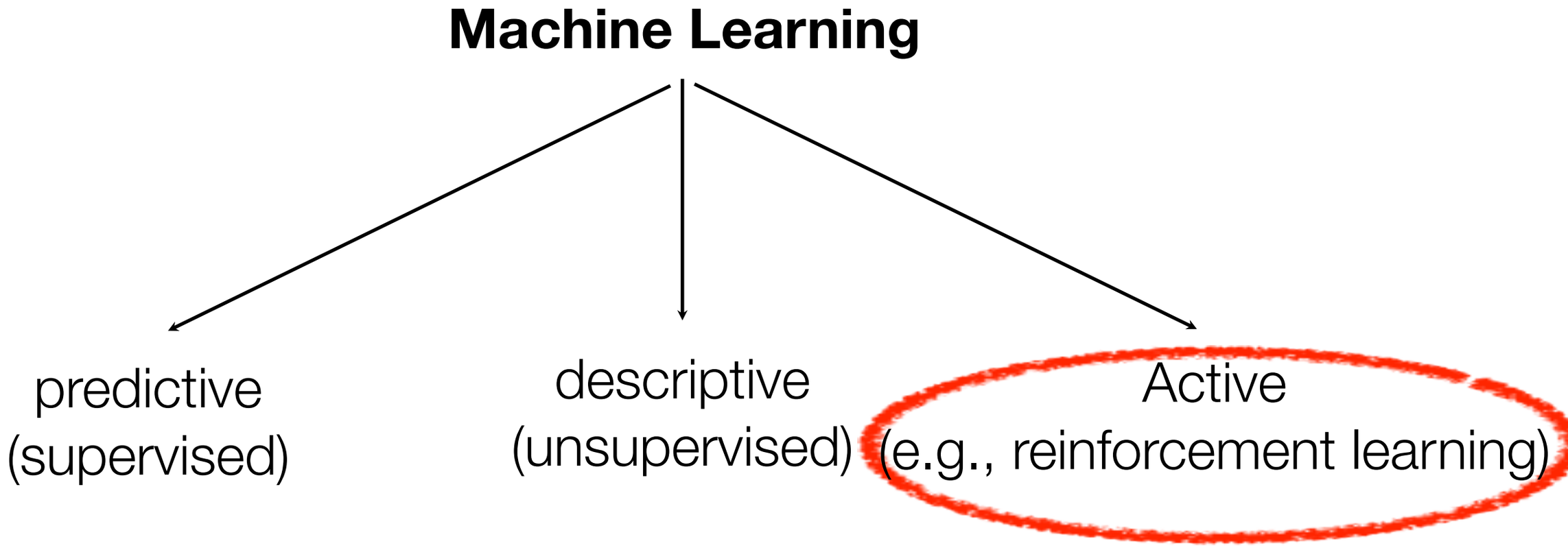
This is called Density Estimation!

The bigger picture...

When we're learning to see, nobody's telling us what the right answers are – we just look. Every so often, your mother says 'that's a dog,' but that's very little information. You'd be lucky if you got a few bits of information — even one bit per second — that way. The brain's visual system has 10^{14} neural connections. And you only live for 10^{9} seconds. So it's no use learning one bit per second. You need more like 10^{5} bits per second. And there's only one place you can get that much information: from the input itself.

— Geoffrey Hinton, 1996

Types of Machine Learning



That will be the main topic of the lecture!

How to attack a machine learning problem?

Machine learning problems essentially always are about two entities:

(i) *data model assumptions*:

- Understand your problem
- generate good features which make the problem easier
- determine the model class
- Pre-processing your data

(ii) *algorithms that can deal with* (i):

- Estimating the parameters of your model.

We are gonna do this for regression...

Content of this Lecture

- Math and Statistics Refresher
- What is Machine Learning?
- Model-Selection
  - Linear Regression
    - Gauss’ Approach
    - Frequentist Approach
    - Bayesian Approach

Important Questions

How does the data look like?

- Are you really learning a function?
- What data types do our outputs have?
- Outliers: Are there “data points in China”?

What is our model (relationship between inputs and outputs)?

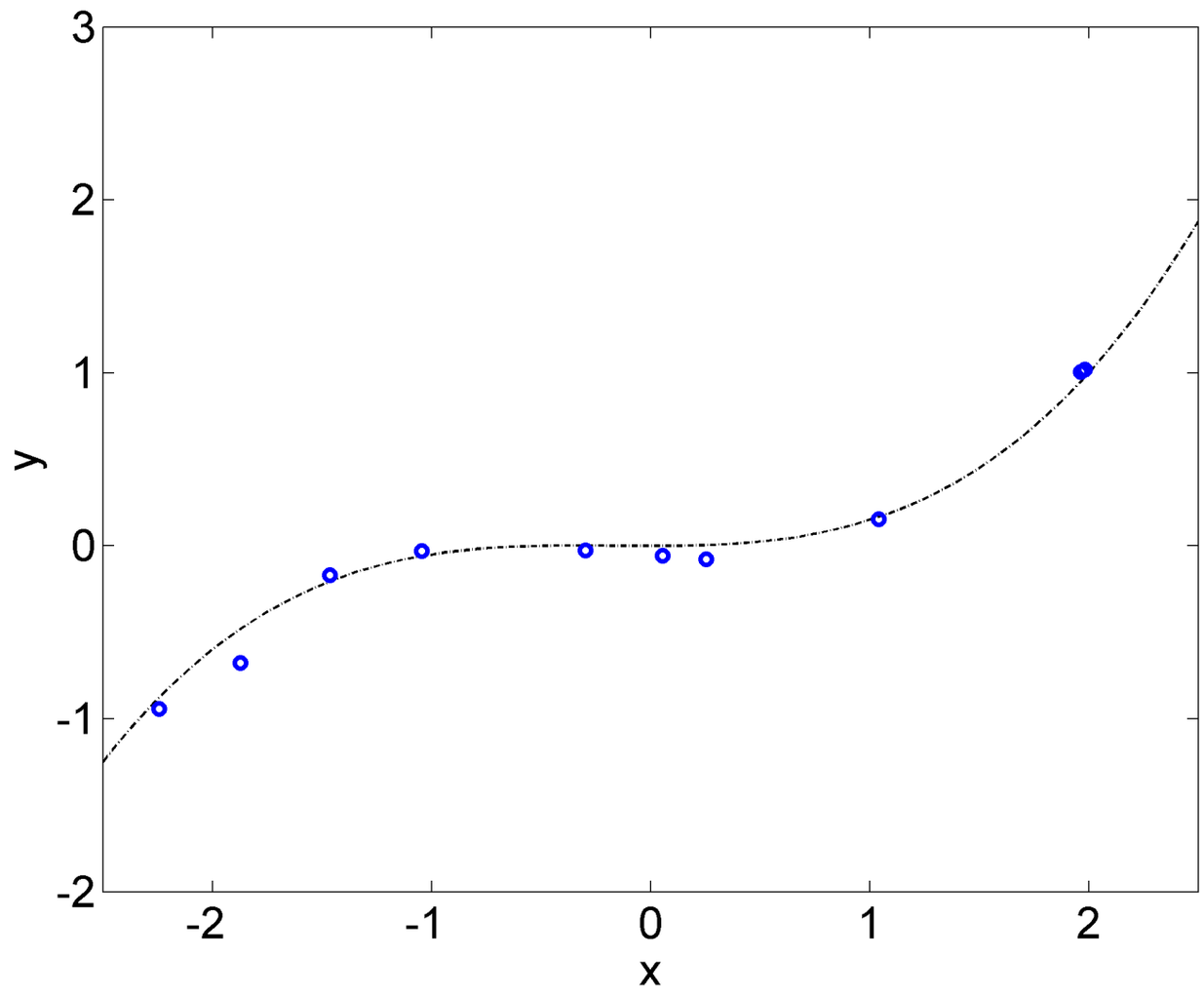
- Do you have features?
- What type of noise / What distribution models our outputs?
- Number of parameters?
- Is your model sufficiently rich?
- Is it robust to overfitting?

Important Questions

Requirements for the solution

- accurate
- efficient to obtain (computation/memory)
- interpretable

Example Problem: a data set



Task: Describe the outputs as a function of the inputs (**regression**)

Model Assumptions: Noise + Features

Additive **Gaussian** Noise:

$$y = \mathbf{f}_\theta(\mathbf{x}) + \epsilon \quad \text{with} \quad \epsilon \sim \mathcal{N}(0, \sigma^2)$$

Equivalent Probabilistic Model

$$p(y|\boldsymbol{x}) = \mathcal{N}(y|f_\theta(\boldsymbol{x}), \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\frac{(y - f_\theta(\boldsymbol{x}))^2}{\sigma^2}\right)$$

Lets keep in simple: **linear in Features**

$$\mathbf{f}_\theta(\mathbf{x}) = \phi(\mathbf{x})^T \theta$$

Important Questions

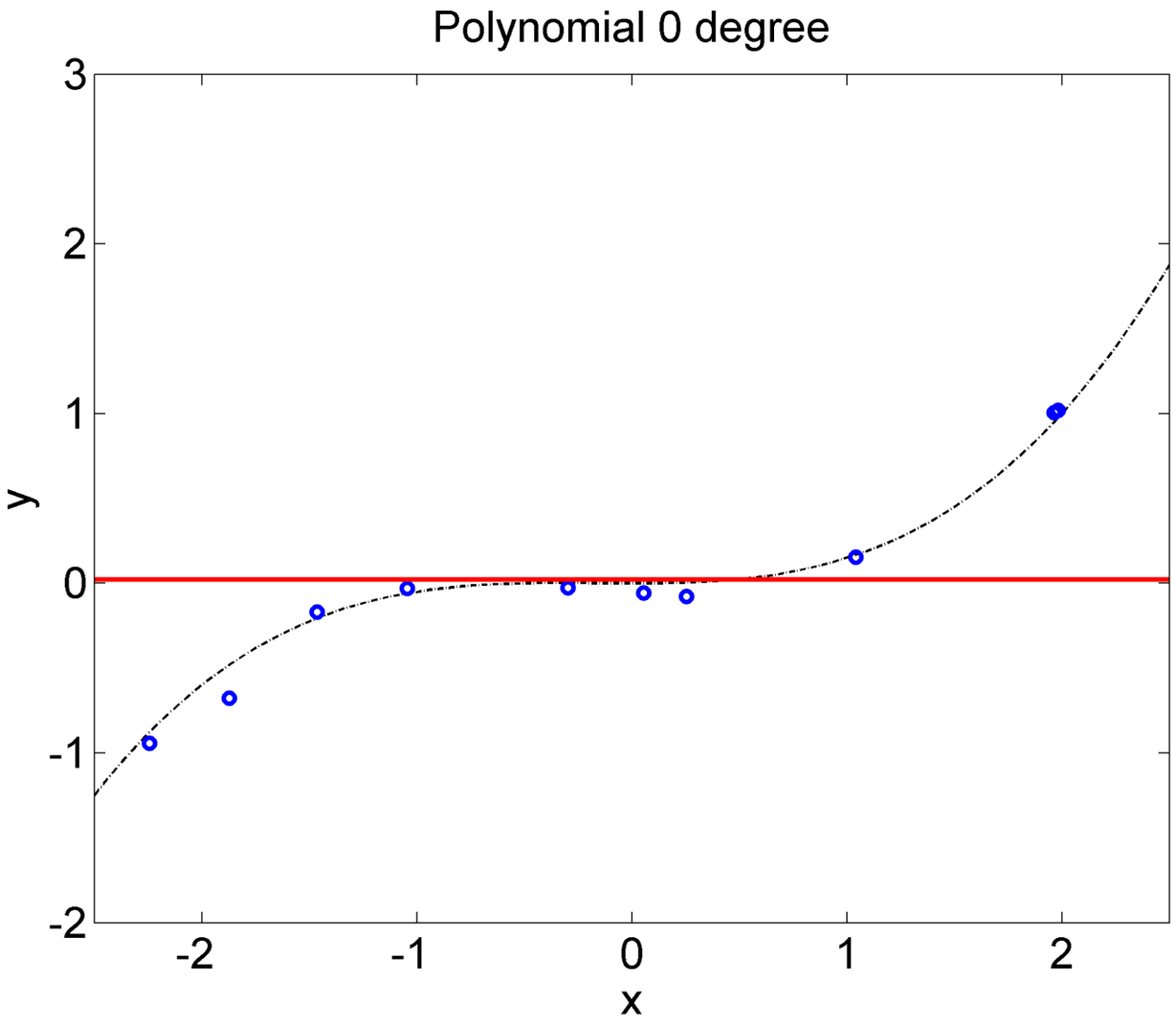
How does the data look like?

- What data types do our outputs have? $y_q \in \mathbb{R}$
- Outliers: Are there “data points in China”? NO
- Are you really learning a function? YES

What is our model? $y = \phi(\boldsymbol{x})^T \boldsymbol{\theta} + \epsilon$

- Do you have features? $\phi(\boldsymbol{x})$
- What type of noise / What distribution models our outputs? $\epsilon \sim \mathcal{N}(0, \sigma^2)$
- Number of parameters?
- Is your model sufficiently rich?
- Is it robust to overfitting?

Let us fit our model ...



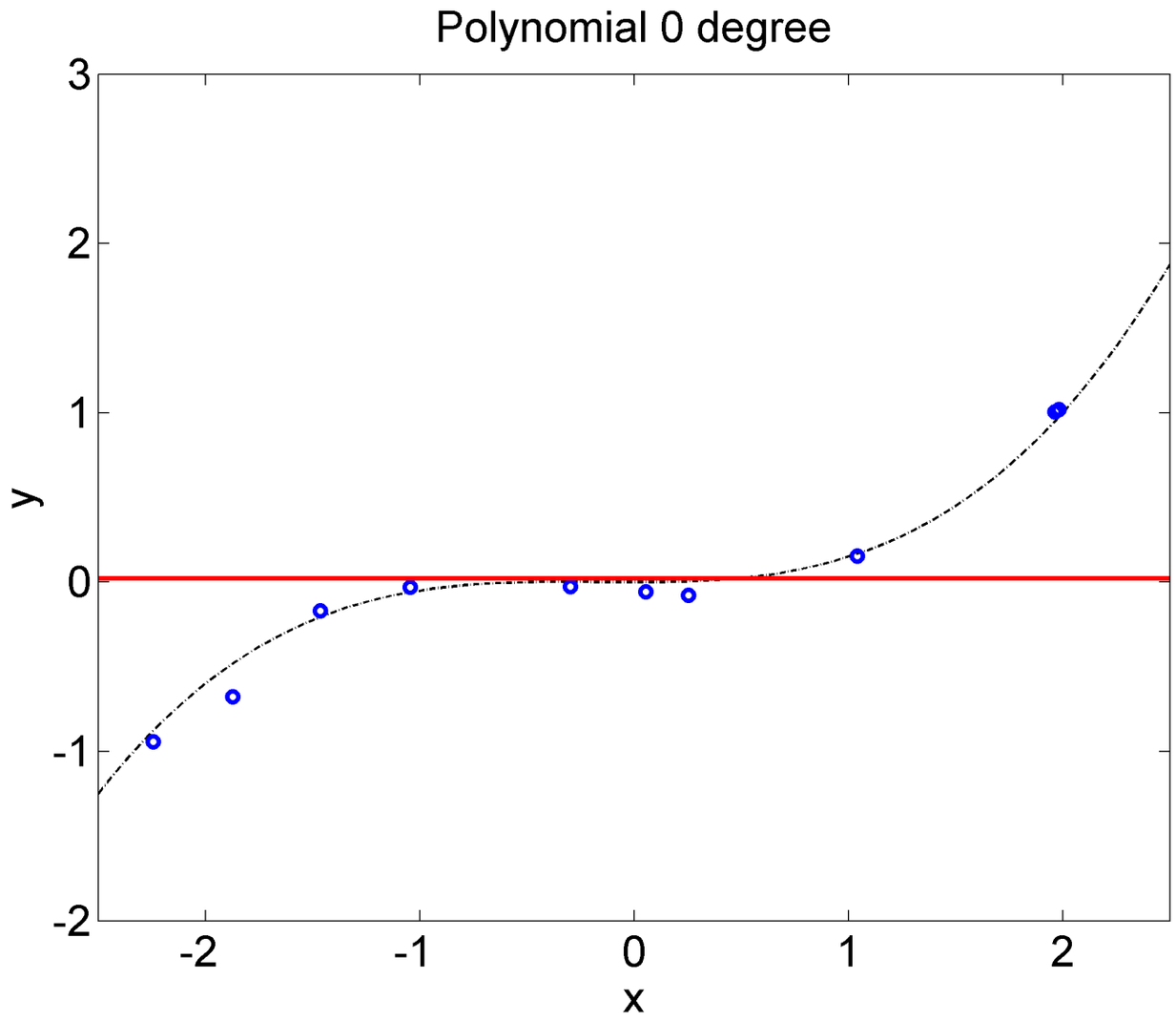
We need to answer:

- How many parameters?
- Is your model sufficiently rich?
- Is it robust to overfitting?

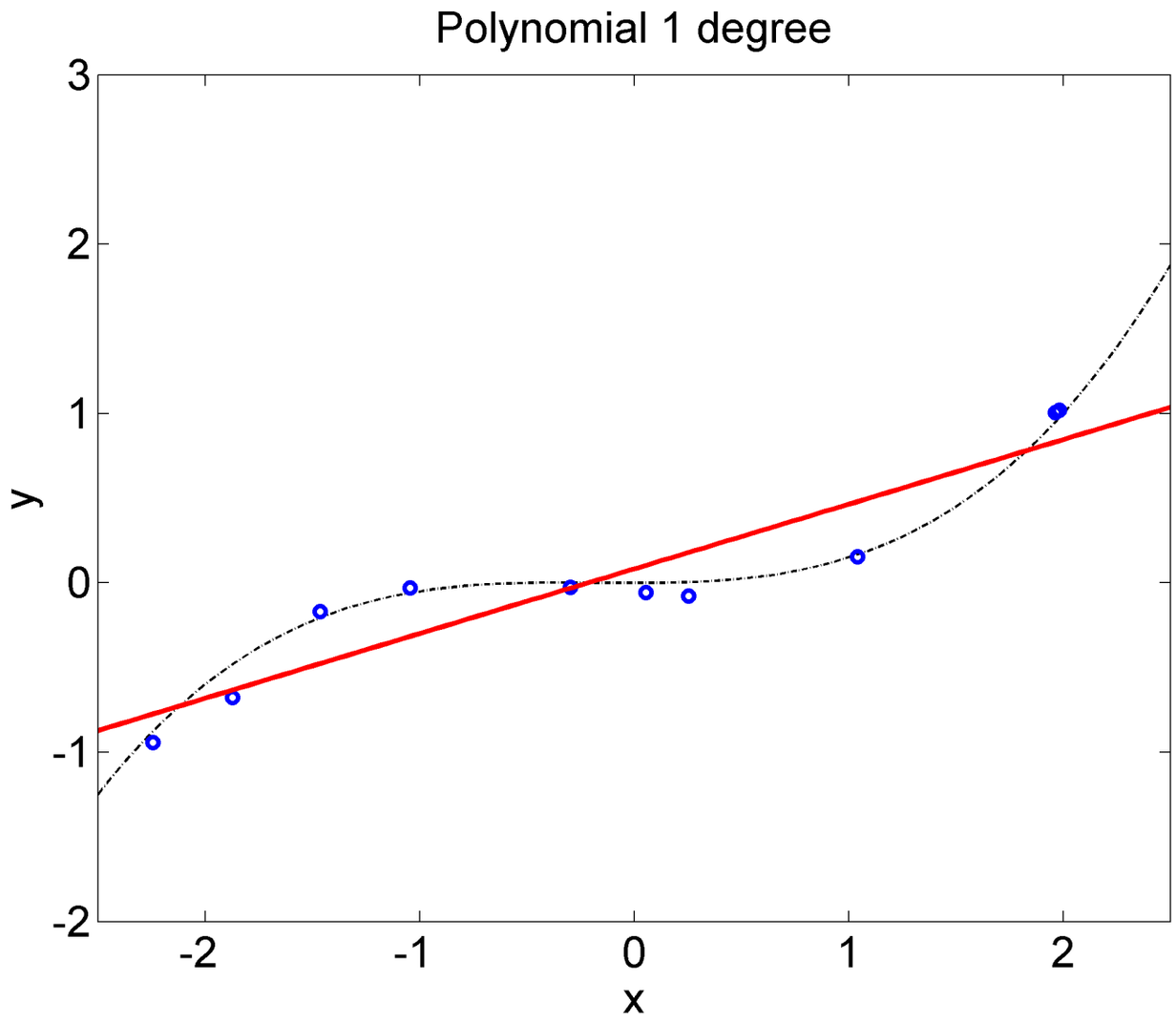
We assume a model class: polynomials of degree n

$$y = \phi(x)^T\theta + \epsilon = [1, x, x^2, x^3, \ldots, x^n]^T\theta + \epsilon$$

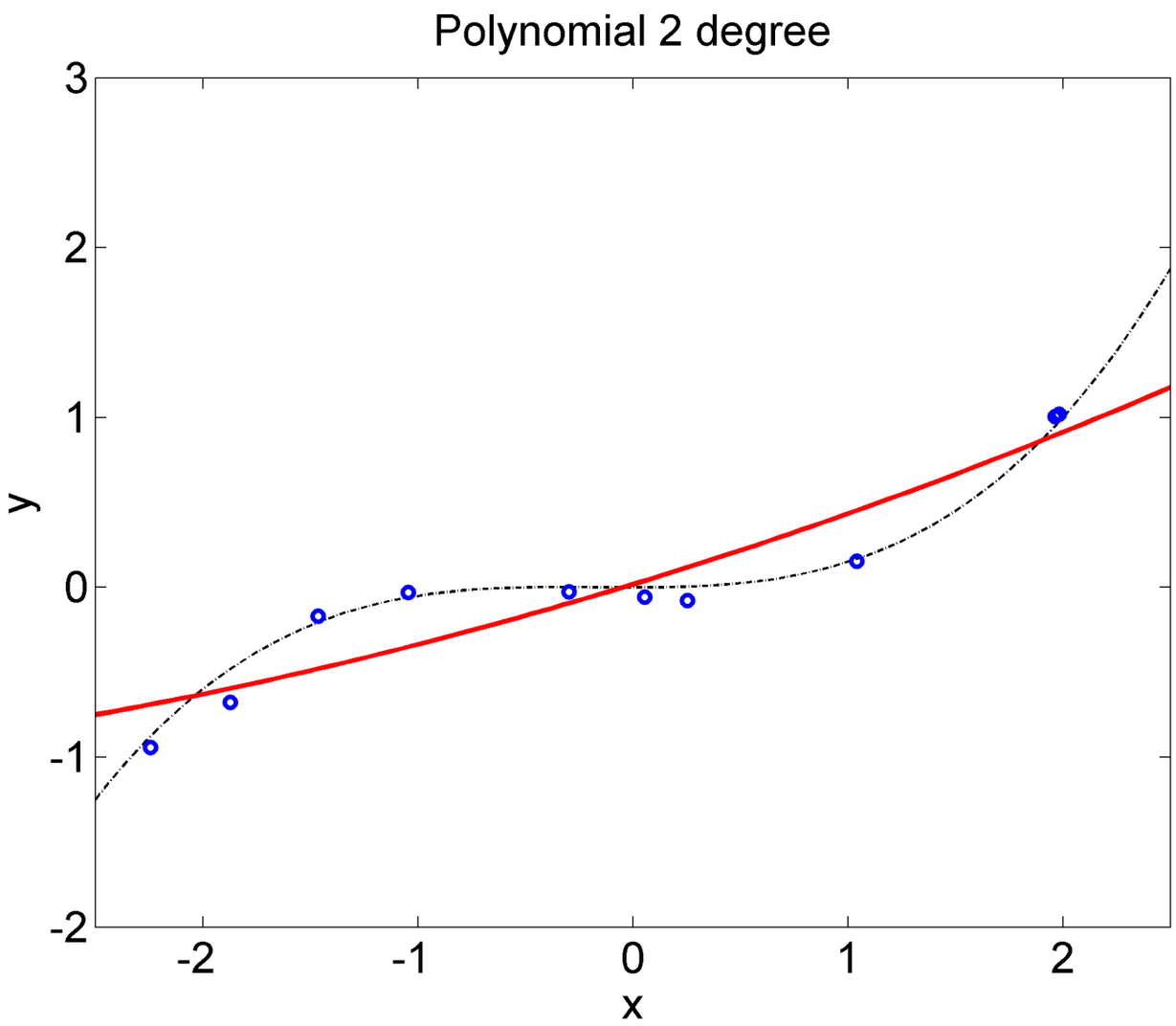
Fitting an Easy Model: n=0

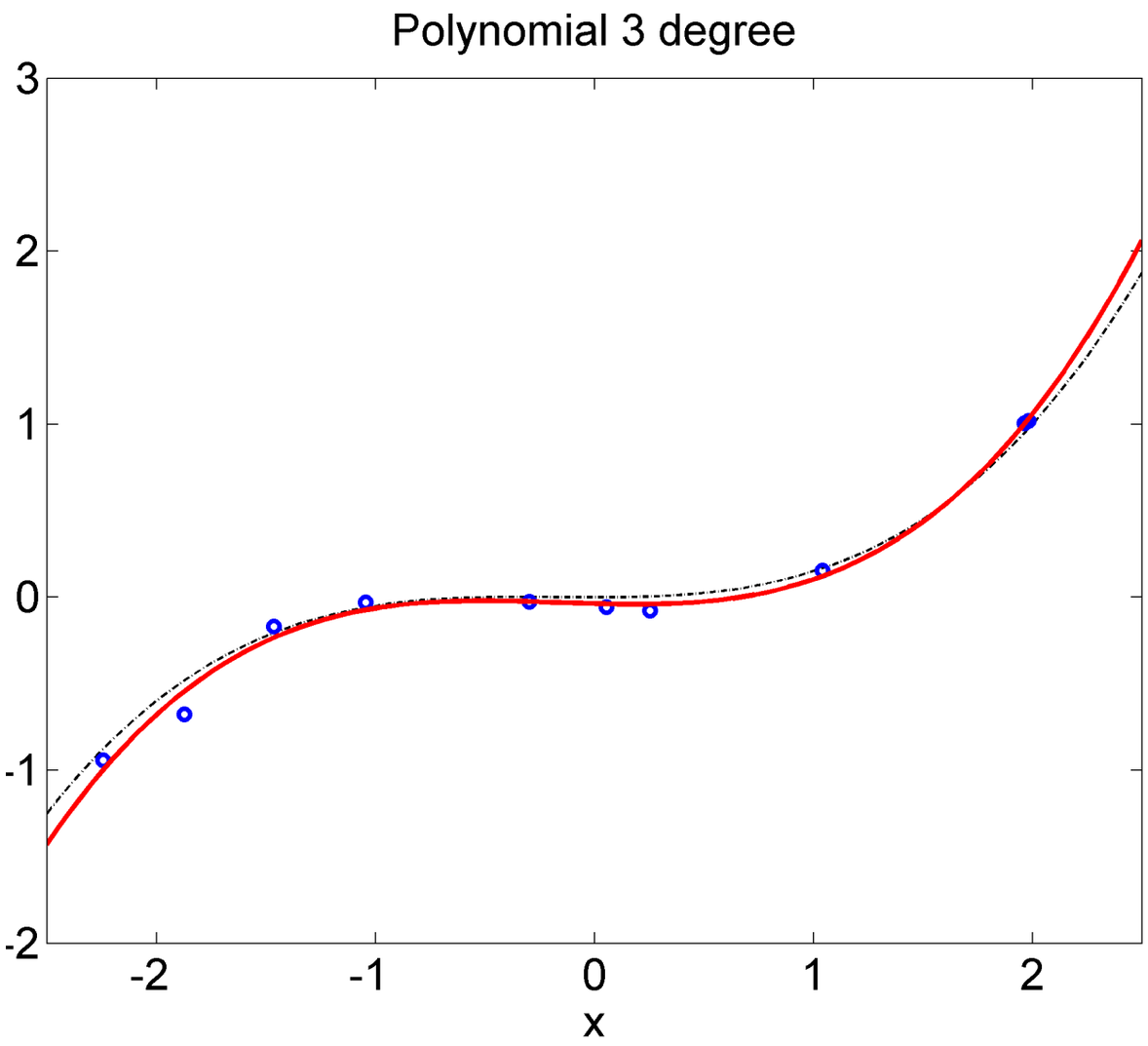


Add a Feature: n=1

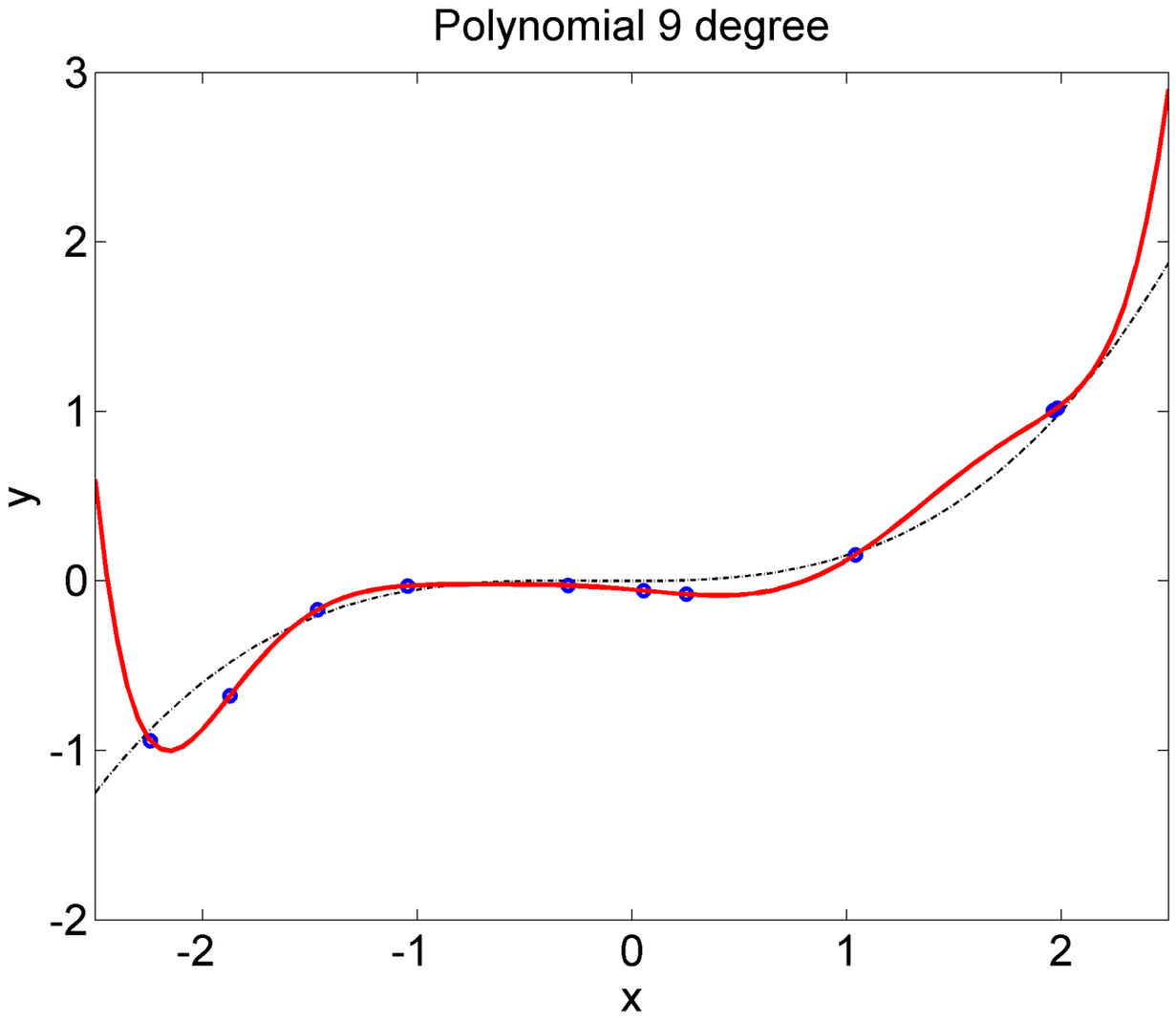


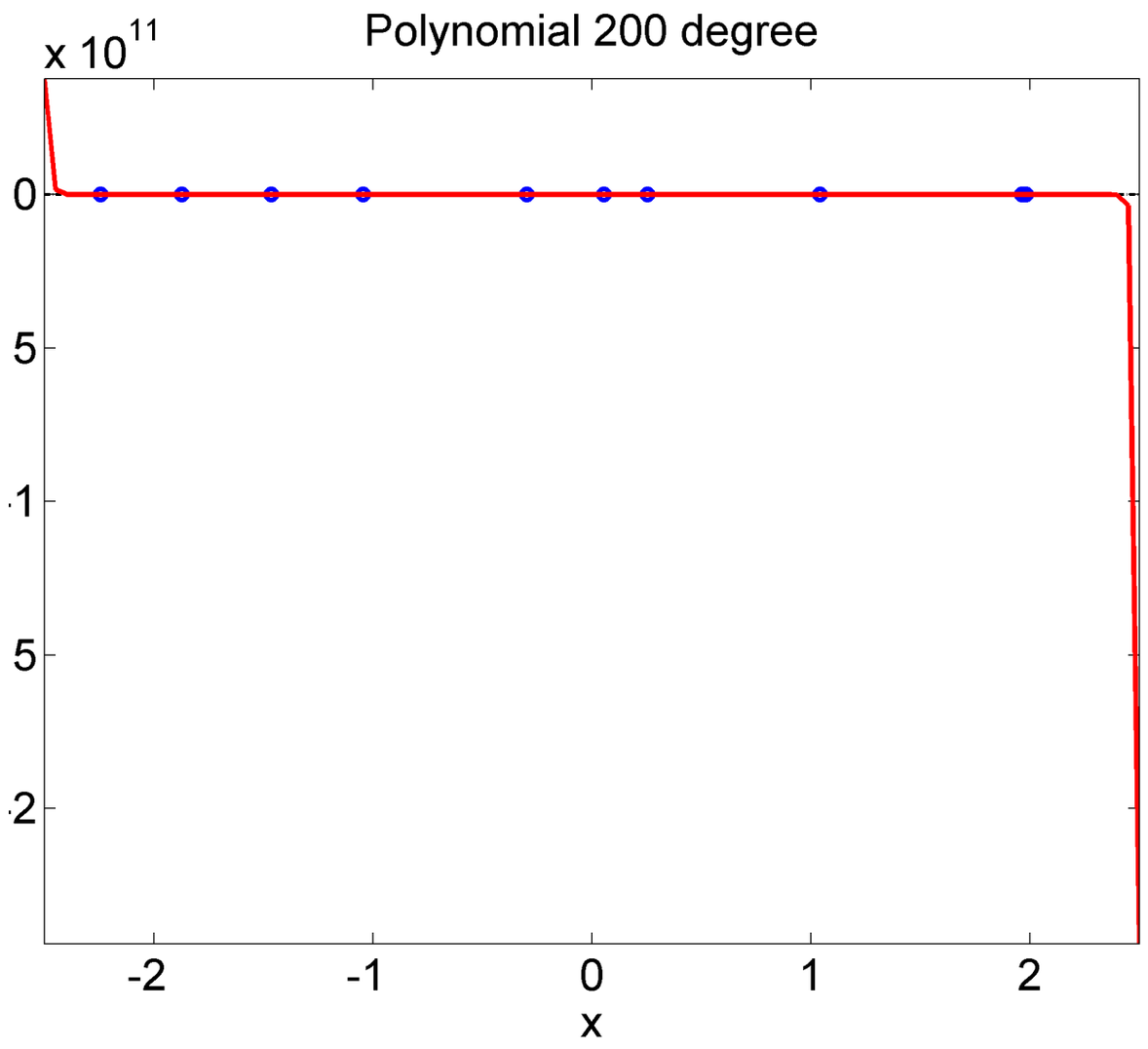
More features...



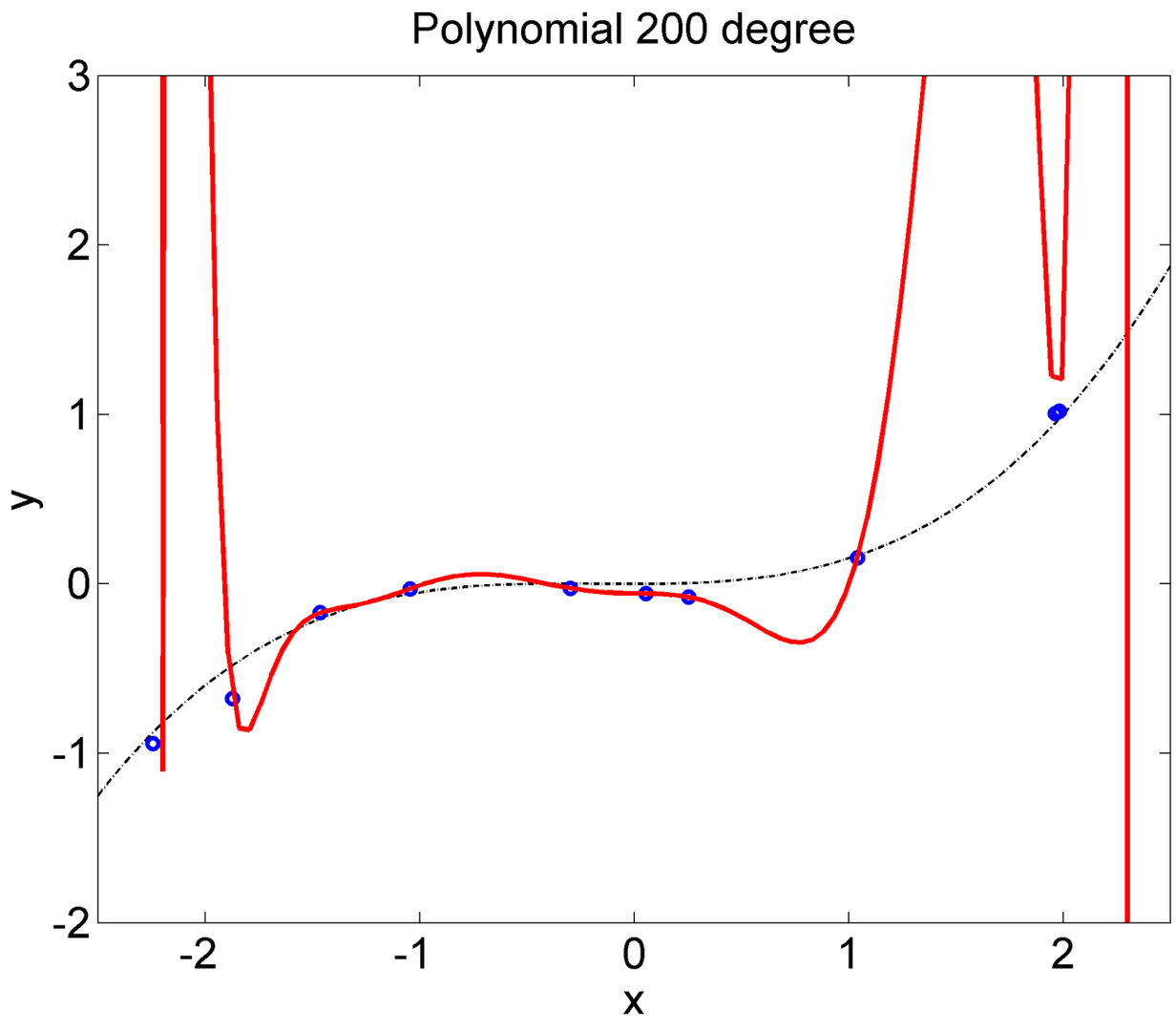


More features...



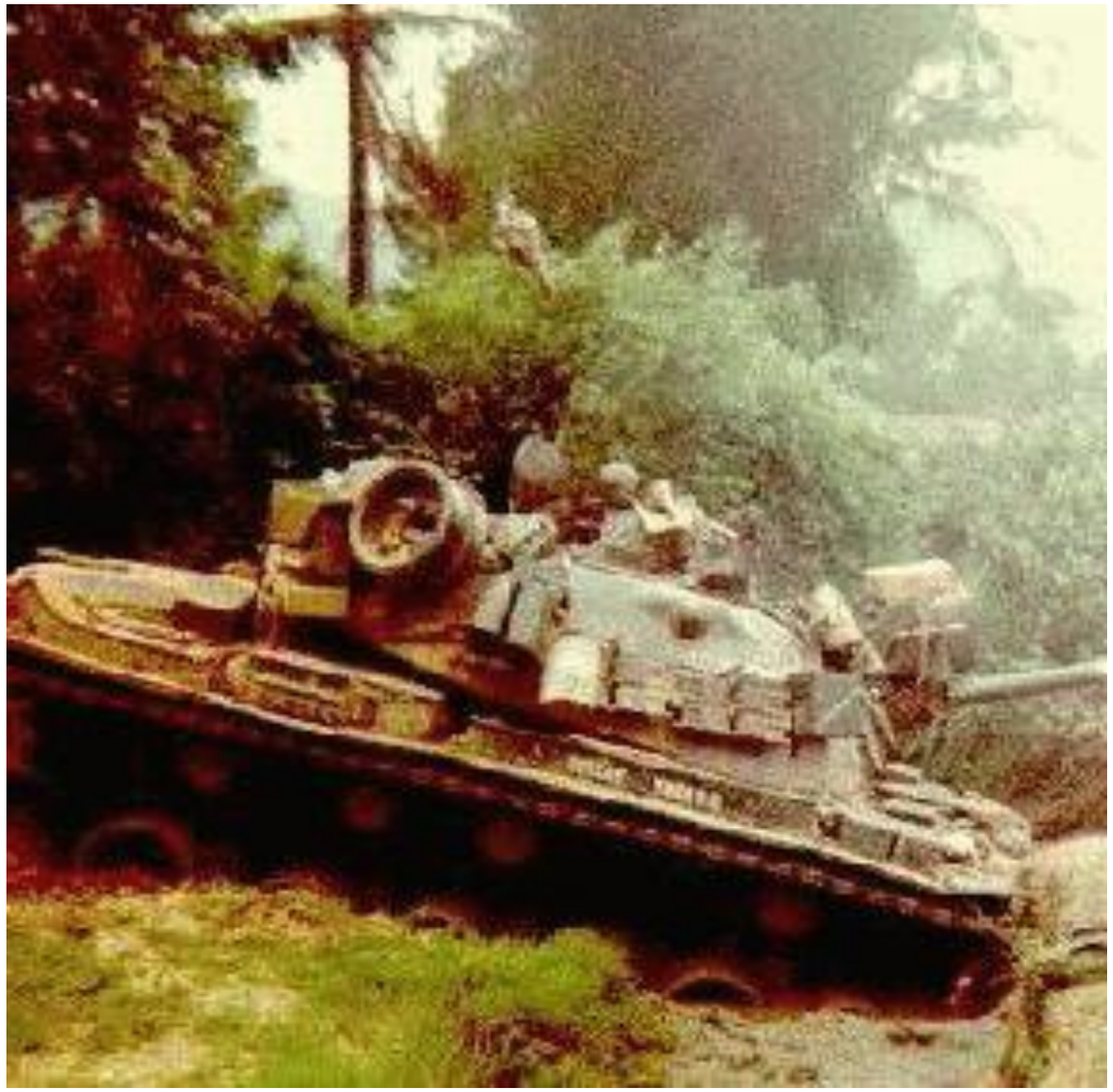


More features: n=200 (zoomed in)



overfitting and numerical problems

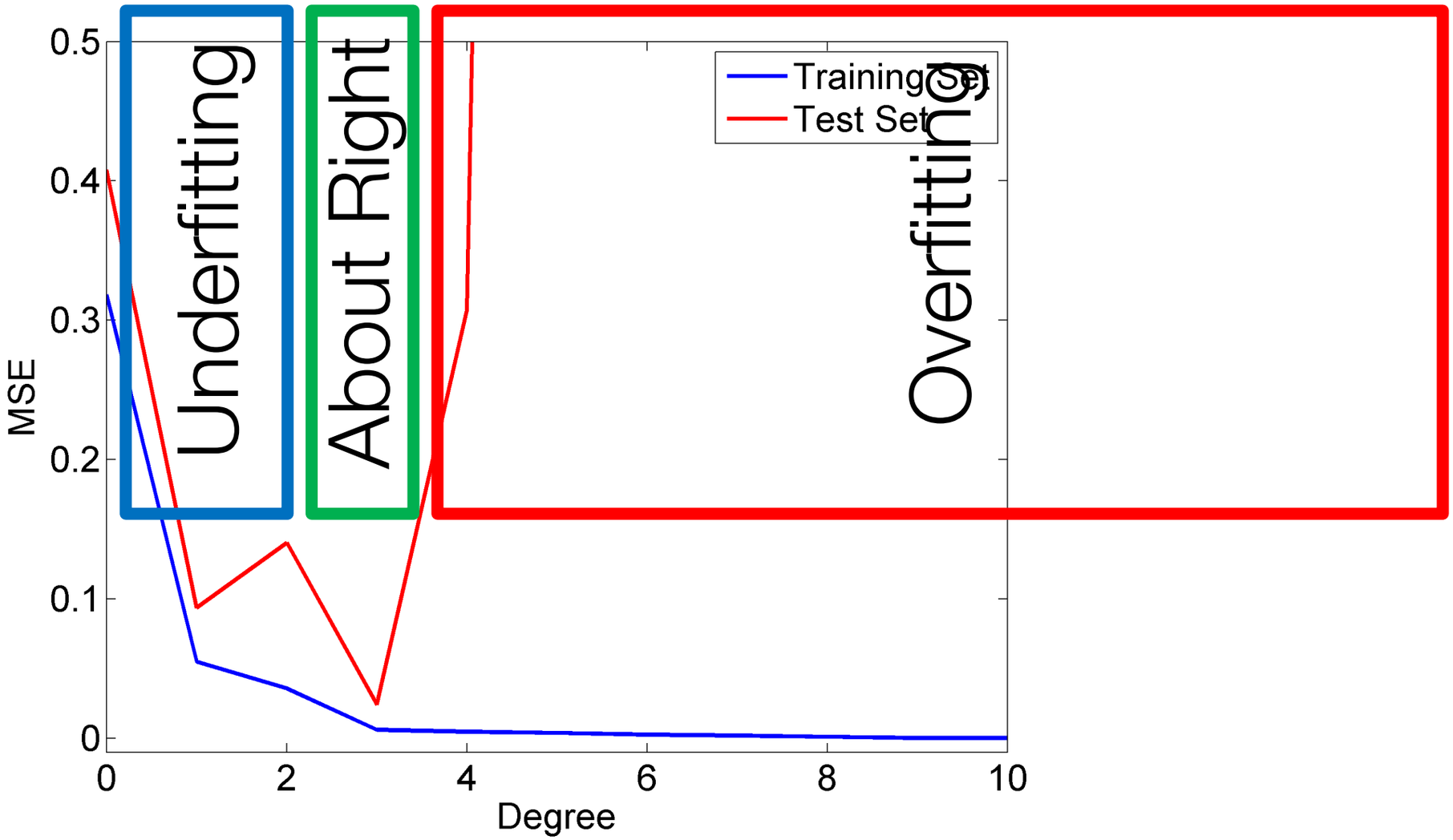
Prominent example of overfitting...



Is there a tank in the picture?

DARPA Neural Network Study (1988-89), AFCEA International Press

Test Error vs Training Error



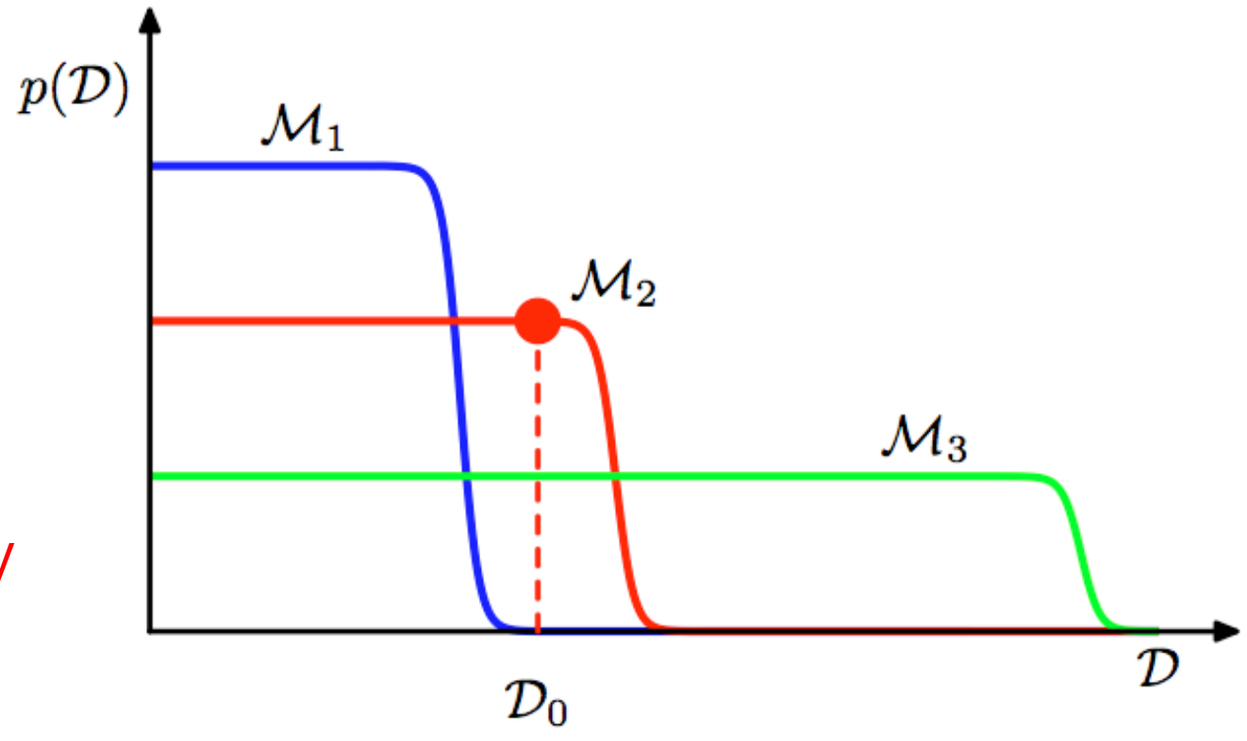
Does a small training error lead to a good model ??

NO ! We need to do model selection

Occam's Razor and Model Selection

Model Selection: How can we…

- choose number of features/parameters?
- choose type of features?
- prevent overfitting?



Some insights:

Always choose the model that fits the data and has the smallest model complexity

➡ called occam's razor

Bias-Variance Tradeoff

$$\text{Expected Total Error} = \text{Bias}^2 + \text{Variance}$$

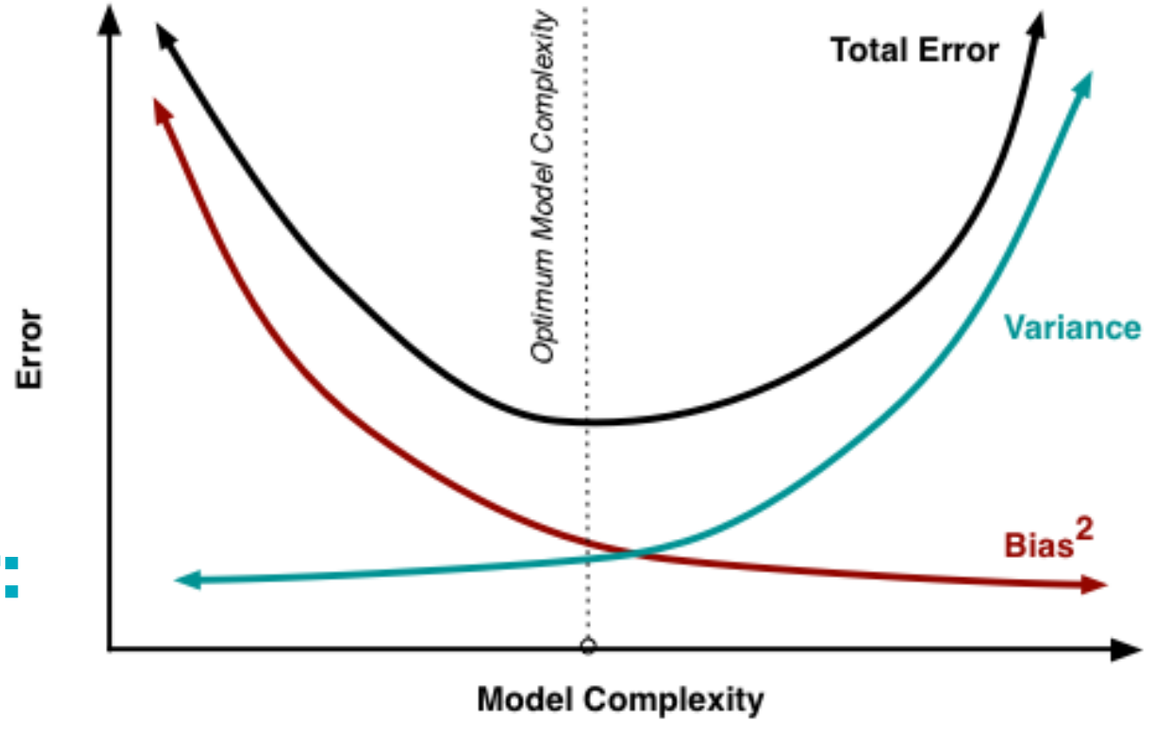
Typically, you can not minimize both!

Bias / Structure Error:

Error because our model can not do better

Variance / Approximation Error:

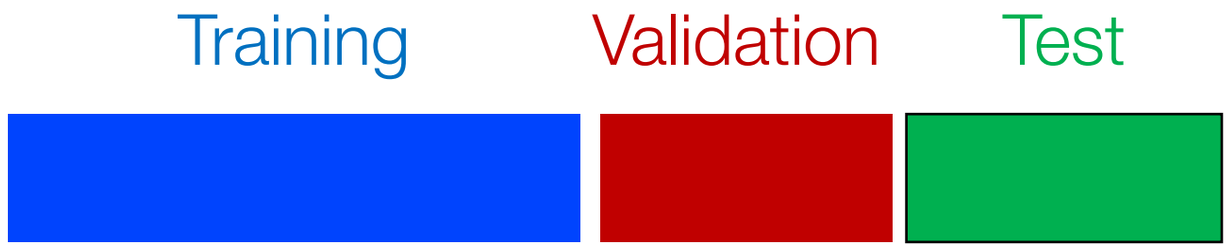
Error because we estimate parameters on a limited data set



How do choose the model?

Goal: Find a good model $\mathcal{M}$ (e.g., good set of features)

Split the dataset into:



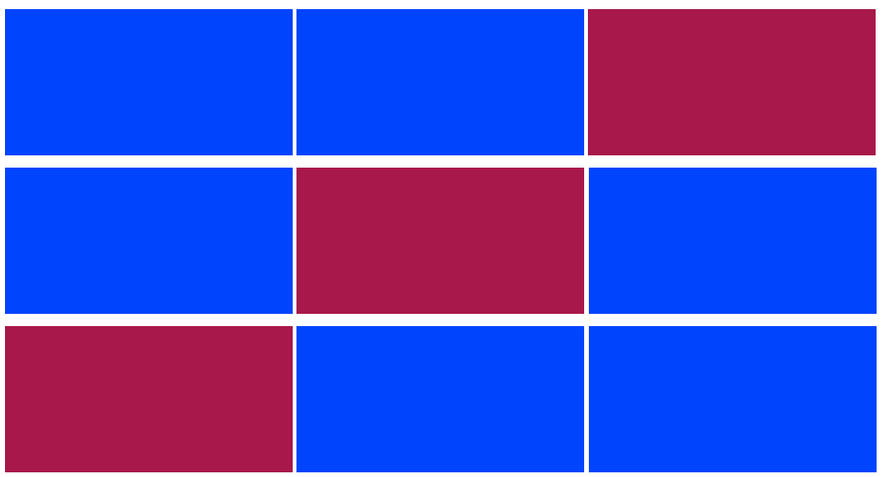
1. Training Set: Fit Parameters
2. Validation Set: Choose model class or single parameters
3. Test Set: Estimate prediction error of trained model

➡ **Error needs to be estimated on independent set!**

Model Selection:
K-fold Cross Validation

Partition data into K sets, use K-1 as **training set** and 1 set as **validation set**

For all possible ways of partitioning compute the **validation error** *J* ➡ computationally expensive!!



$$J(p_1, \mathcal{M}_i)$$

$$J(p_2, \mathcal{M}_i)$$

$$J(p_3, \mathcal{M}_i)$$

Choose model $\mathcal{M}_i$ with **smallest average validation error**

Content of this Lecture

- Math and Statistics Refresher
- What is Machine Learning?
- Model-Selection
- Linear Regression
  - Gauss’ Approach
  - Frequentist Approach
  - Bayesian Approach

How to find the parameters $\boldsymbol{\theta}$?

Gauss

Let’s find parameters through a cost function!

$$\boldsymbol{\theta}^* = \text{argmin}_{\boldsymbol{\theta}} \sum_{i=1}^{N} (f_{\boldsymbol{\theta}}(\boldsymbol{x}_i) - y_i)^2$$

Objective is defined by minimizing a certain cost function

Gauss' view: Least Squares

The classical cost function is the one of **least-squares**

$$J = \frac{1}{2}\sum_{i=1}^{N}(y_i - \mathbf{f}_\theta(\mathbf{x}_i))^2 \qquad y_i = \phi(\boldsymbol{x}_i)^T\boldsymbol{\theta} + \epsilon$$

Using

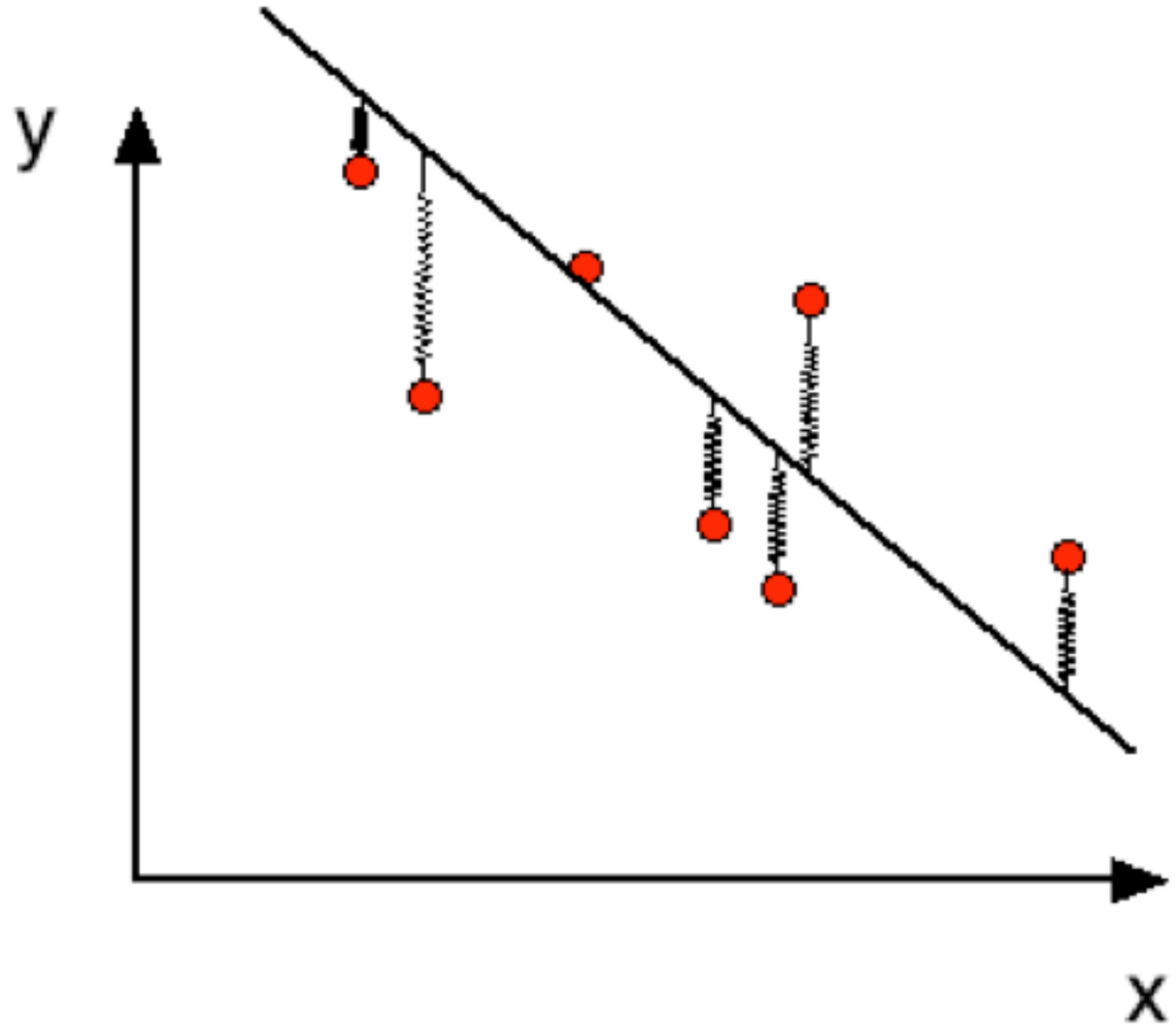
$$\begin{aligned}\Phi &= \left[\ \phi(\mathbf{x}_1),\ \ \phi(\mathbf{x}_2),\ \ \phi(\mathbf{x}_3),\ \ \ldots,\ \ \phi(\mathbf{x}_n)\ \right]^T,\\ \mathbf{Y} &= \left[\ y_1,\ \ y_2,\ \ y_3,\ \ \ldots,\ \ y_n\ \right]^T.\end{aligned}$$

we can rewrite it as $J = \frac{1}{2}(Y - \Phi\theta)^T(Y - \Phi\theta)$ → Scalar Product

and solve it $\theta = (\Phi^T\Phi)^{-1}\Phi^T Y$

Least Squares solution contains left pseudo-inverse

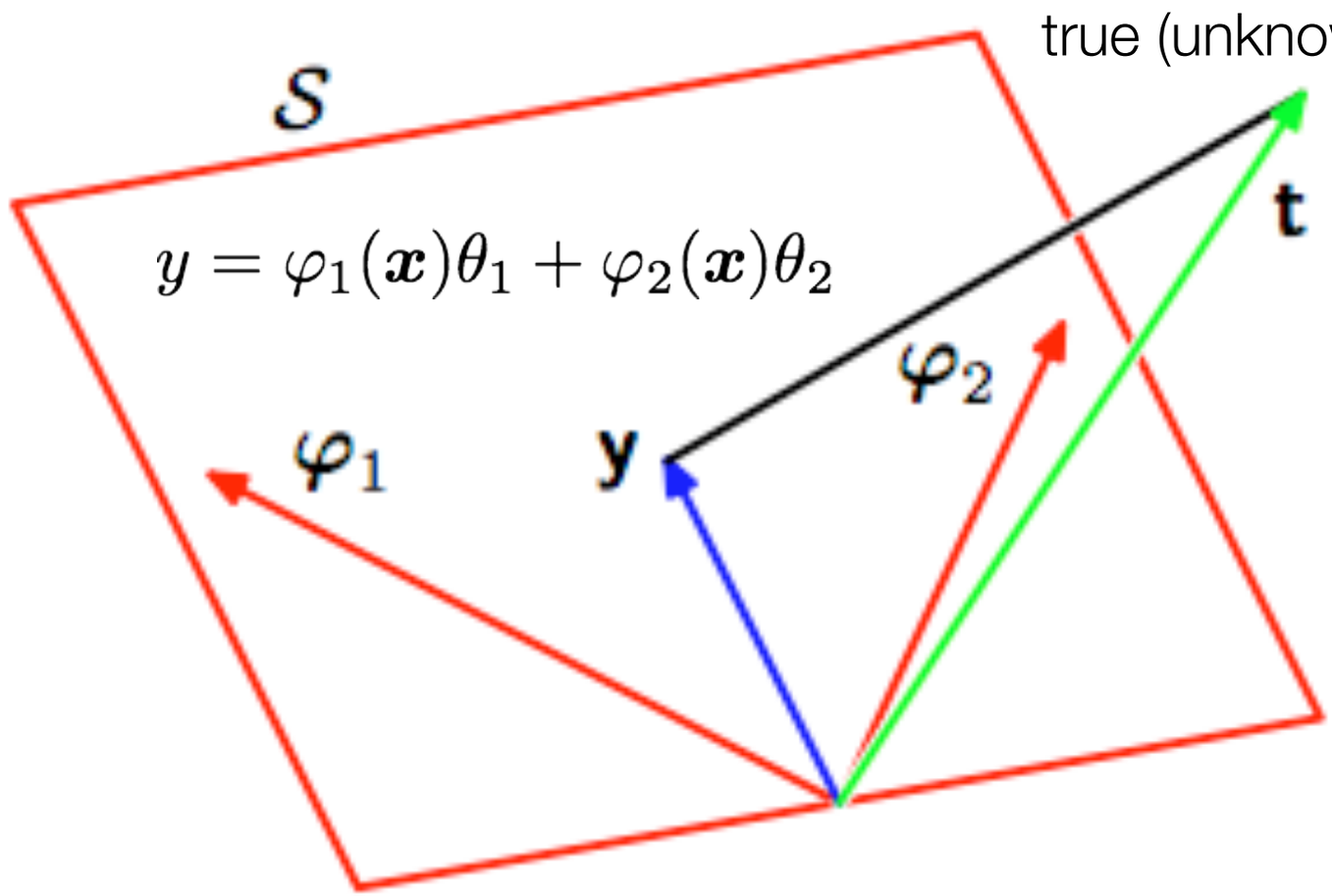
Physical Interpretation



Energy of springs ~ squared lengths

⇨ minimize energy of system

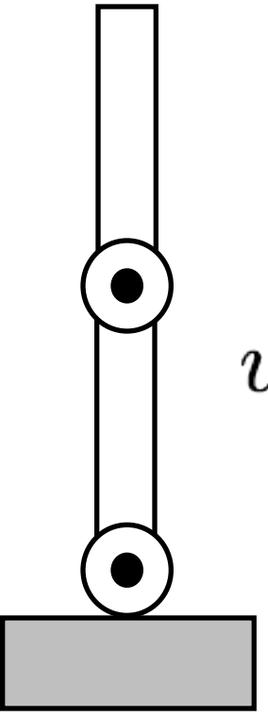
Geometric Interpretation



Minimize projection error ➡ orthogonal projection

Robotics Example: Rigid-Body Dynamics Known Features

$$
\begin{aligned}
u_1 &= [m_1 l_{g1}^2 + J_1 + m_2(l_1^2 + l_{g2}^2 + 2 l_1 l_{g2} \cos\theta_2) + J_2]\ddot{\theta}_1 \\
&+ [m_2(l_{g2}^2 + l_1 l_2 \cos\theta_2) + J_2]\ddot{\theta}_2 \qquad \text{Inertial Forces} \\
&- 2 m_2 l_1 l_{g2} \dot{\theta}_1 \dot{\theta}_2 \sin\theta_2 \qquad \text{Coriolis Forces} \\
&- 2 m_2 l_1 l_{g2} \dot{\theta}_1^2 \sin\theta_2 \qquad \text{Centripetal Forces} \\
&+ m_1 g l_{g1} \cos\theta_1 + m_2 g(l_1 \cos\theta_1 + l_{g2} \cos(\theta_1 + \theta_2)) \qquad \text{Gravity}
\end{aligned}
$$



$$
\begin{aligned}
u_2 &= [m_2(l_{g2}^2 + l_1 l_{g2} \cos\theta_2) + J_2]\ddot{\theta}_1 \\
&+ (m_2 l_{g2}^2 + J_2)\ddot{\theta}_2 \qquad \text{Inertial Forces} \\
&- m_2 l_1 l_{g2} \dot{\theta}_1^2 \sin\theta_2 \qquad \text{Centripetal Forces} \\
&+ m_2 g l_{g2} \cos(\theta_1 + \theta_2) \qquad \text{Gravity}
\end{aligned}
$$

Robotics Example: Rigid-Body Dynamics

We realize that rigid body dynamics is **linear in the parameters**

We can rewrite it as

$$\boldsymbol{u} = \boldsymbol{\phi}(\theta, \dot{\theta}, \ddot{\theta})^T \boldsymbol{\psi}$$

accelerations, velocities, sin and cos terms

masses, lengths, inertia, ...

For finding the parameters we can apply even the first machine learning method that comes to mind: Least-Squares Regression

Cost Function II: Ridge Regression

We punish the magnitude of the parameters

➡ Controls model complexity

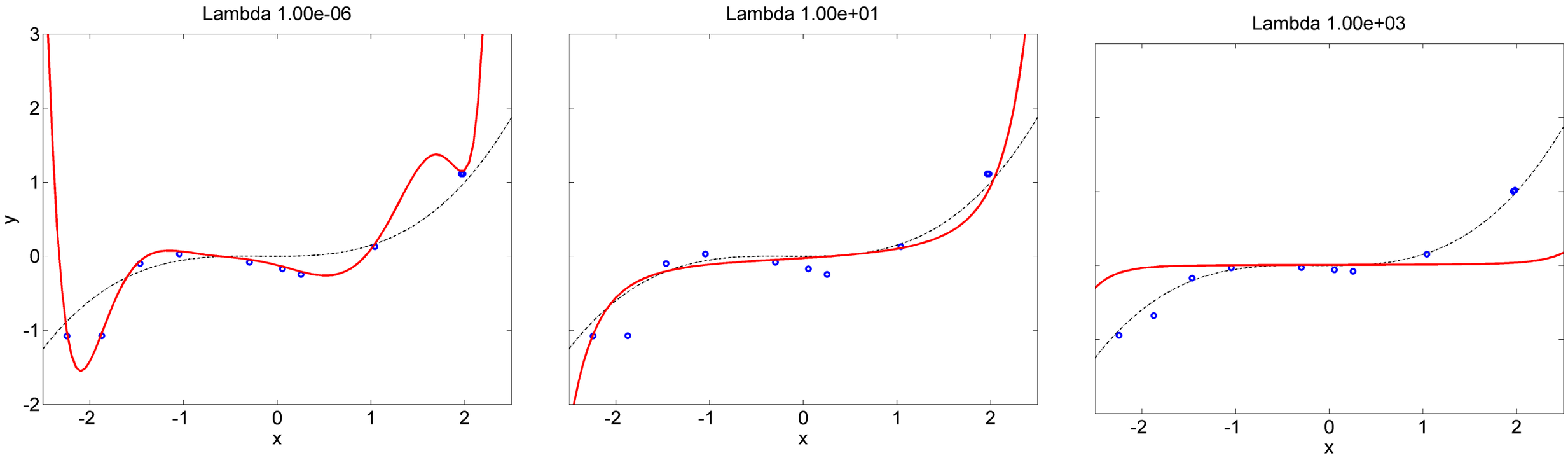
$$J_{\mathrm{RR}} = (\boldsymbol{y} - \boldsymbol{\Phi\theta})^T(\boldsymbol{y} - \boldsymbol{\Phi\theta}) + \boldsymbol{\theta}^T\boldsymbol{W}\boldsymbol{\theta}$$

This yields **ridge regression** $\theta = (\Phi^T\Phi + \mathbf{W})^{-1}\Phi^T Y$

with $\mathbf{W} = \lambda\mathbf{I}$, where λ is called ridge parameter. For features normalized by variance, typically $\lambda \in [10^{-9}, \ldots, 10^{-5}]$.

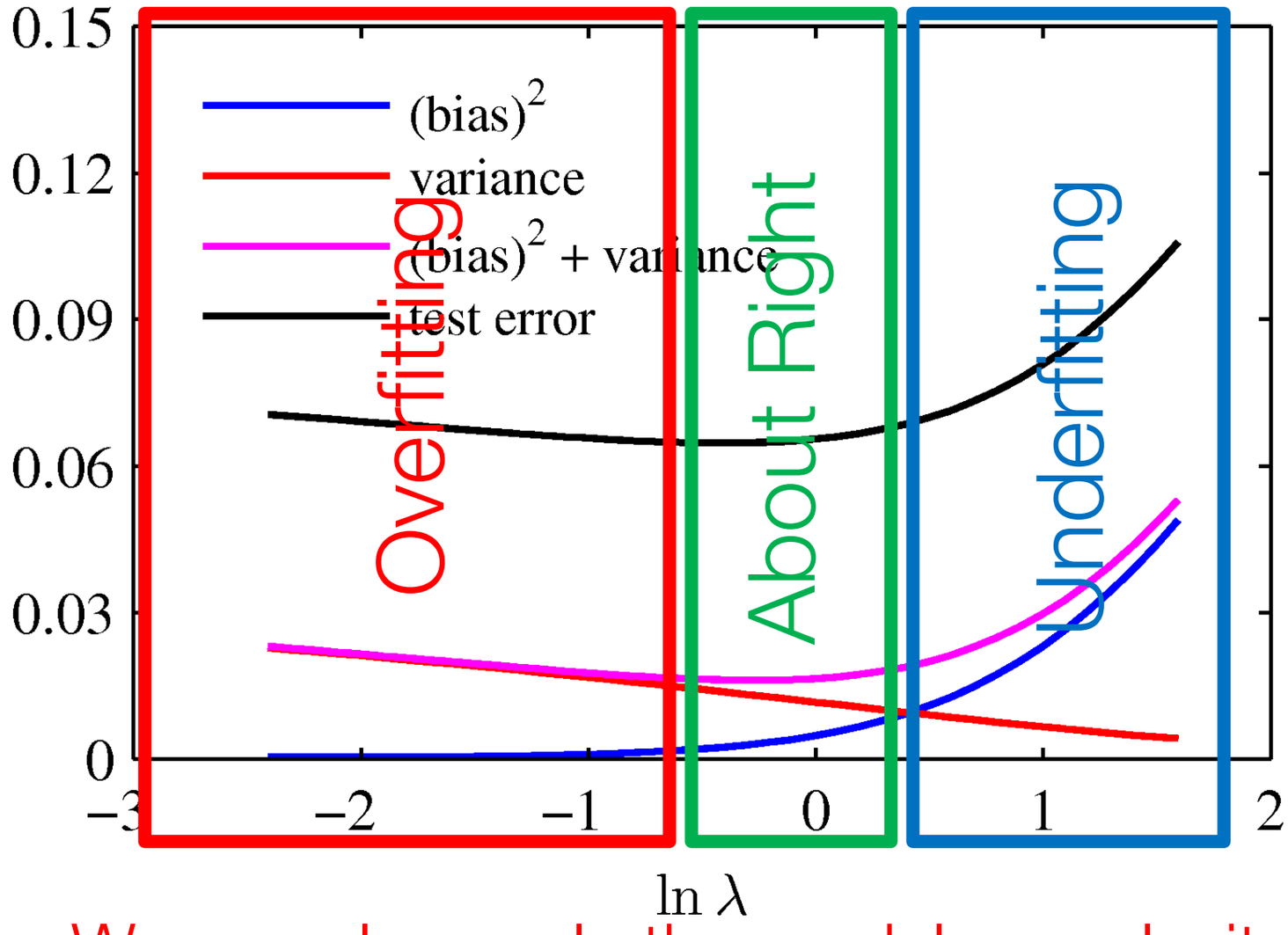
Numerically, this is much more stable! Even with redundant features.

Ridge regression: n=15



Influence of the regularization constant

MAP: Back to the Overfitting Problem



We can also scale the model complexity with the regularization parameter!

Smaller lambda ⇨ higher model complexity

Content of this Lecture

- Math and Statistics Refresher
- What is Machine Learning?
- Model-Selection
- Linear Regression
  - Gauss’ Approach
  - Frequentist approach
  - Bayesian Approach

How to find the parameters $\boldsymbol{\theta}$?

Fisher

Frequentist:

Probabilities are frequencies of a repeated experiment.

- There are some true parameters of the experiment which we cannot observe.
- They reveal themselves by the frequency (i.e., likelihood) at which we can repeat the outcome of the experiment $p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})$.
- We can obtain good parameters by maximizing likelihood of the outcome!

Maximum-Likelihood (ML) estimate

We can maximize the **likelihood of the outcome**:

$$\arg\max_{\theta} p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}) = \arg\max_{\theta} \prod_{i=1}^{N} p(y_i|\boldsymbol{x}_i,\boldsymbol{\theta})$$

That‘s hard!

Do the ‘log-trick’:

$$J_{\mathrm{ML}} = \arg\max_{\theta} \log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta}) = \arg\max_{\theta} \sum_{i=1}^{N} \log p(y_i|\boldsymbol{x}_i,\boldsymbol{\theta})$$

$$= \arg\min_{\theta} \frac{1}{2\sigma^2} \sum_{i=1}^{N} (y_i - f_\theta(\boldsymbol{x}_i))^2$$

That‘s easy!

$$\Rightarrow \boldsymbol{\theta}^*_{\mathrm{LS}} = \boldsymbol{\theta}^*_{\mathrm{ML}}$$

Least Squares Solution is equivalent to ML solution with Gaussian noise!!

Content of this Lecture

- Math and Statistics Refresher
- What is Machine Learning?
- Model-Selection
- Linear Regression
  - Gauss’ Approach
  - Frequentist Approach
  - Bayesian Approach

Does this make sense?

- Maximizing $p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})$ basically means we only care about the accuracy of the reproduction of the outcomes.

- What if there is no fully true $\boldsymbol{\theta}^*$?

- In this case, we rather need to study the probability of different $\boldsymbol{\theta}$.

- Thus, our quantity of interest is $p(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y})$.

- But where how can we obtain this quantity?

How to find the parameters $\boldsymbol{\theta}$?

Bayesian:

Bayes

Parameters are just random variables. We can encode our subjective belief in the prior.

likelihood

prior

$$p(\boldsymbol{\theta}|\boldsymbol{X},\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\boldsymbol{X}|\boldsymbol{y})}$$

posterior

evidence

***Intuition*:** If you assign each parameter estimator a "probability of being right", the average of these parameter estimators will be better than the single one

Maximum a posteriori (MAP) estimate

Put a prior on our parameters

- E.g., $\boldsymbol{\theta}$ should be small: $p(\boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, \boldsymbol{W}^{-1})$

Find parameters that maximize the posterior

$$p(\boldsymbol{\theta}|\boldsymbol{y}, \boldsymbol{X}) = \frac{p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\boldsymbol{y}|\boldsymbol{X})} \propto p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})p(\boldsymbol{\theta})$$

Do the ‘log trick’ again:

$$\begin{aligned}\arg\max_{\theta} p(\boldsymbol{\theta}|\boldsymbol{y}, \boldsymbol{X}) &= \arg\max_{\theta} p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta})p(\boldsymbol{\theta}) = \arg\max_{\theta} \log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) + \log p(\boldsymbol{\theta}) \\ &= \arg\min_{\theta} -\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) - \log p(\boldsymbol{\theta}) =: J_{\text{MAP}}\end{aligned}$$

Maximum a posteriori (MAP) estimate

The prior is just additive costs…

$$J_{\mathrm{MAP}} = -\log p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) - \log p(\boldsymbol{\theta})$$

Lets put in **our Model:**

$$p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = \prod_i p(y_i|\boldsymbol{x}_i, \boldsymbol{\theta}) = \prod_i \mathcal{N}(y_i|\boldsymbol{\theta}^T \phi(\boldsymbol{x}_i), \sigma^2) \qquad p(\boldsymbol{\theta}) = \mathcal{N}(\mathbf{0}, \boldsymbol{W}^{-1})$$

$$J_{\mathrm{MAP}} = \frac{1}{2}\sum_i \frac{(y_i - \phi^T(\boldsymbol{x}_i)\boldsymbol{\theta})^2}{\sigma^2} + \frac{1}{2}\boldsymbol{\theta}^T \boldsymbol{W} \boldsymbol{\theta}$$

$$= \frac{1}{2\sigma^2}(\boldsymbol{y} - \Phi(\boldsymbol{X})\boldsymbol{\theta})^T(\boldsymbol{y} - \Phi(\boldsymbol{X})\boldsymbol{\theta}) + \frac{1}{2}\boldsymbol{\theta}^T \boldsymbol{W} \boldsymbol{\theta} = \frac{1}{\sigma^2} J_{\mathrm{RR}}$$

$$\Rightarrow \quad \boldsymbol{\theta}^*_{\mathrm{RR}} = \boldsymbol{\theta}^*_{\mathrm{MAP}}$$

Ridge Regression is equivalent to MAP estimate with Gaussian prior

Predictions with the Model

We found an amazing parameter set $\boldsymbol{\theta}^*$ (e.g., ML, MAP)

Let's do **predictions**! parameter estimate

$$p(y_*|\boldsymbol{x}_*, \boldsymbol{\theta}^*) = \mathcal{N}(y_*|\mu(\boldsymbol{x}_*), \sigma^2(\boldsymbol{x}_*))$$

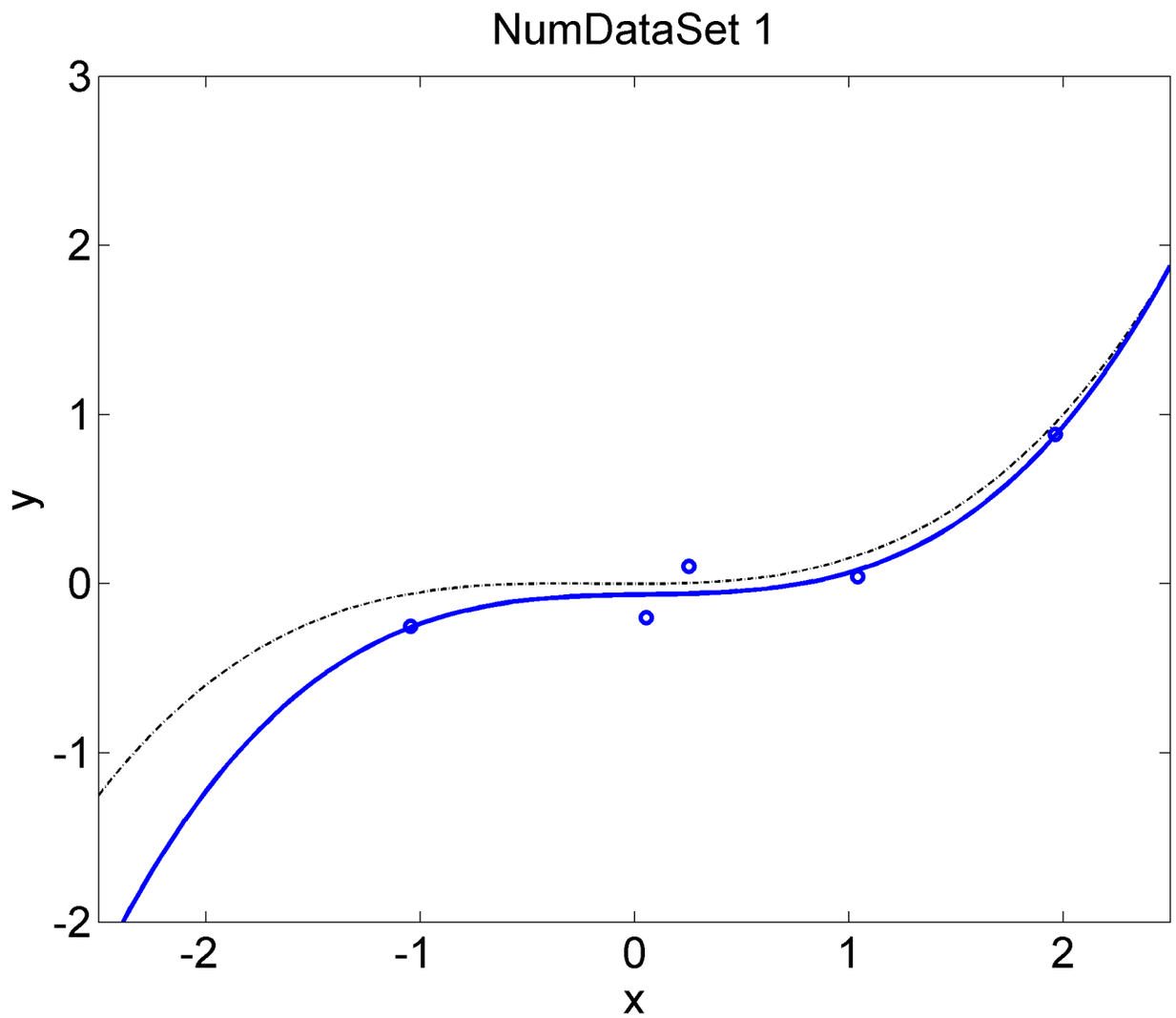
pred. function value — test input

Predictive mean: $\mu(\boldsymbol{x}_*) = \boldsymbol{\phi}^T(\boldsymbol{x}_*)\boldsymbol{\theta}^*$

Predictive variance: $\sigma^2(\boldsymbol{x}_*) = \sigma^2$

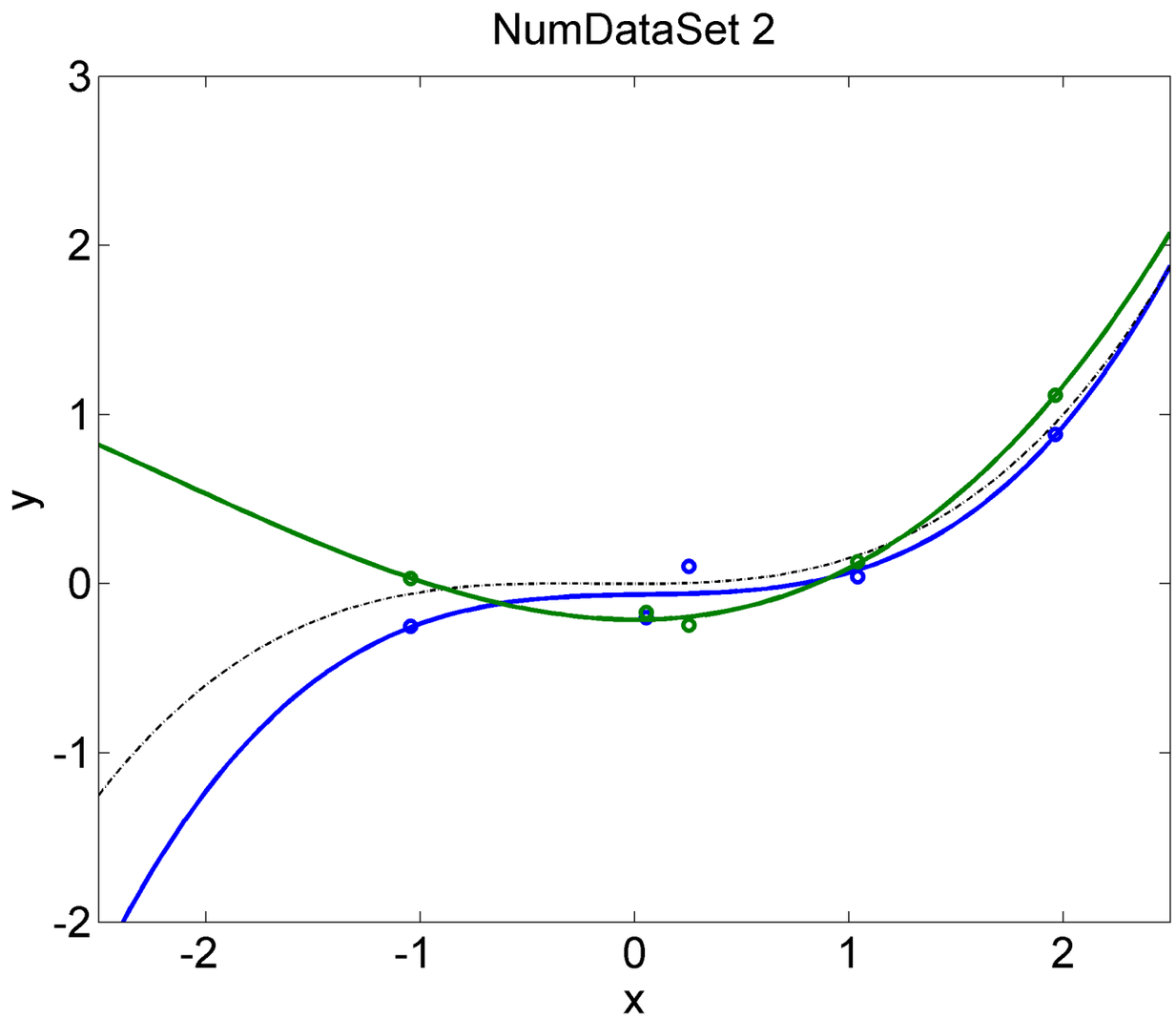
Comparing different data sets…

… with same input data, but different output values (due to noise):



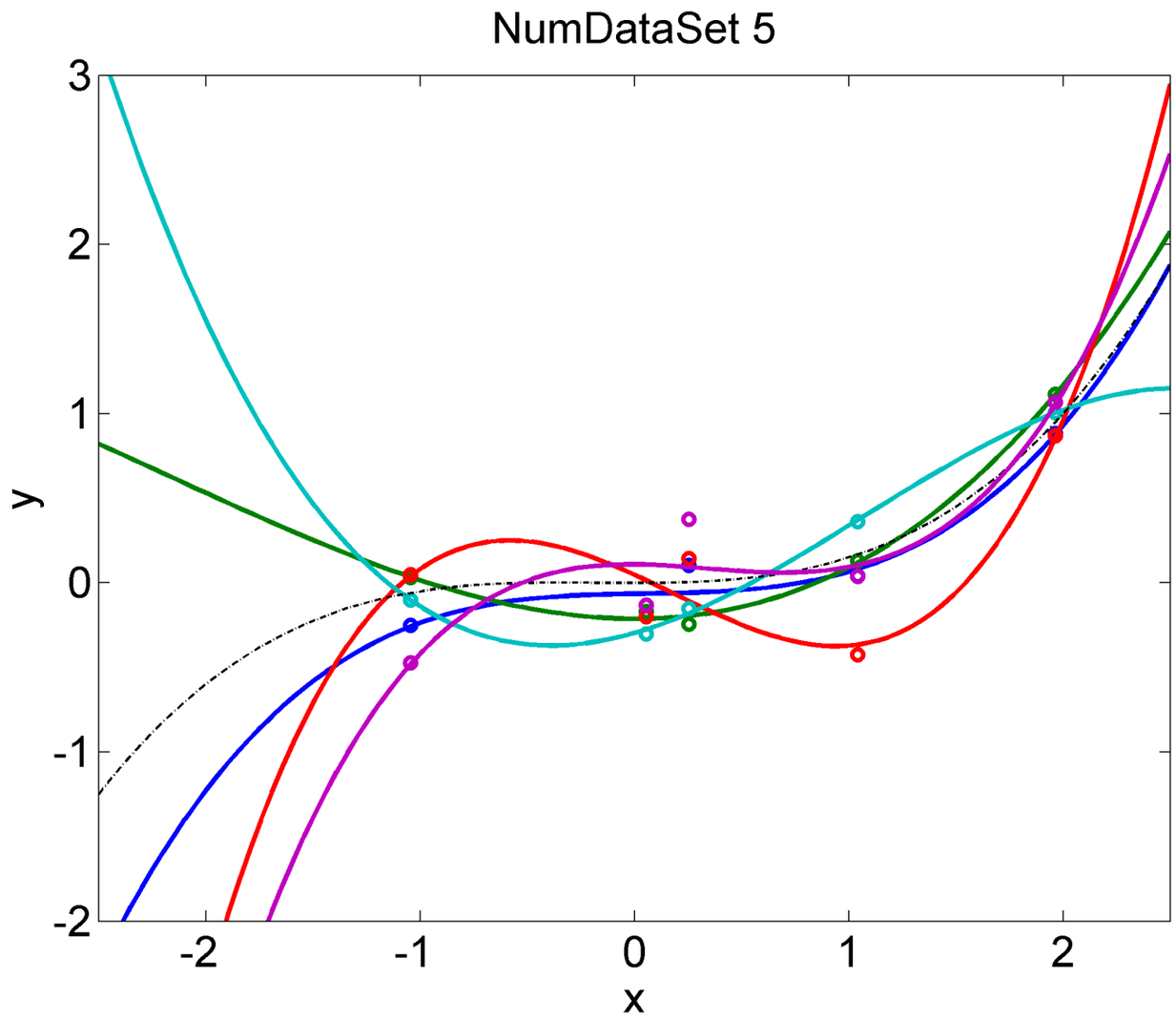
Comparing different data sets…

… with same input data, but different output values (due to noise):



Comparing different data sets…

… with same input data, but different output values (due to noise):



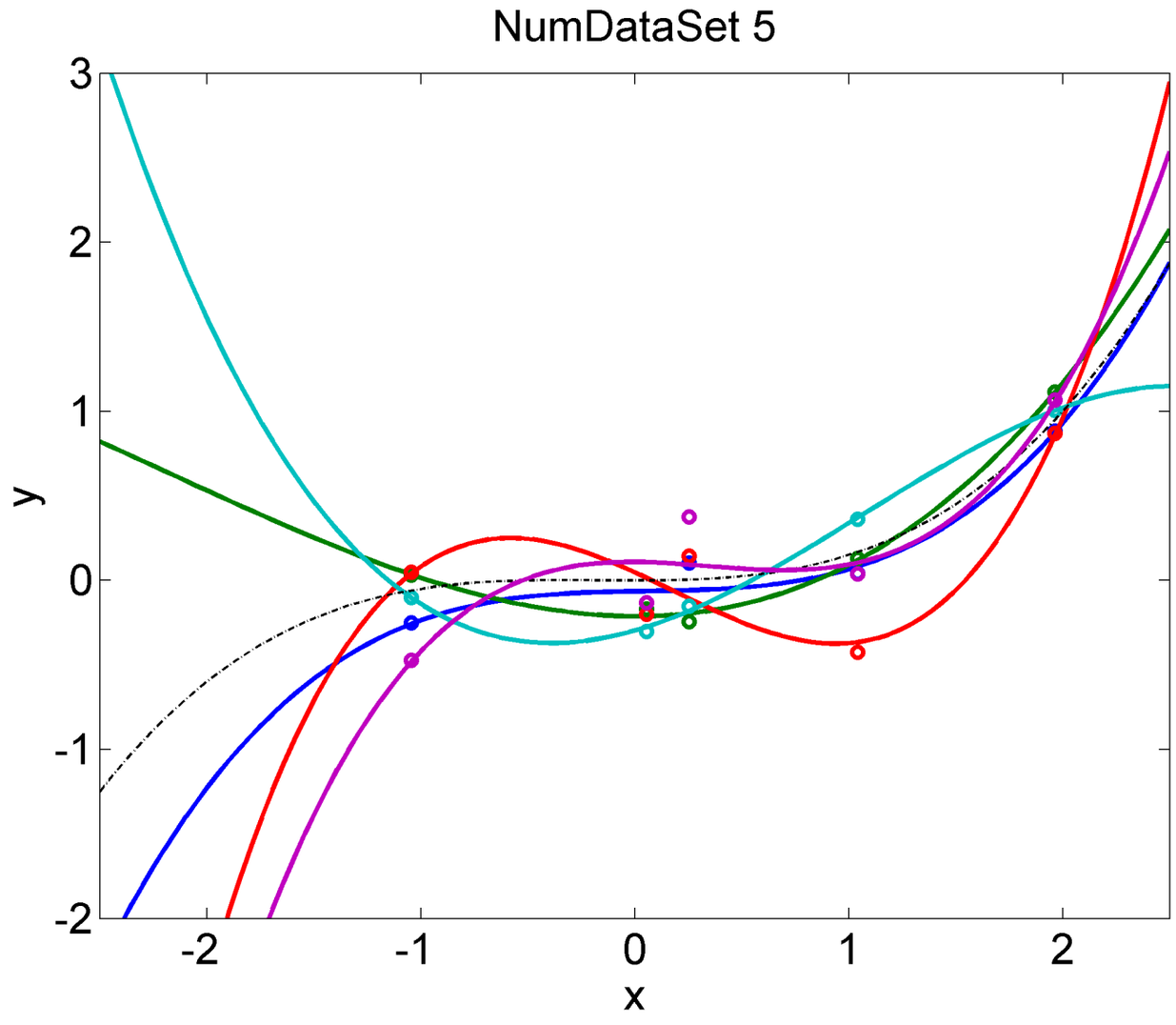
Our parameter estimate is also noisy!
It depends on the noise in the data

Comparing different data sets…

Can we also estimate our uncertainty in $\boldsymbol{\theta}$?

Compute probability of $\boldsymbol{\theta}$ given data

➡ $p(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y})$



How to get the posterior?

Bayes Theorem for Gaussians

$$p(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{0}, \boldsymbol{A})$$
$$p(\boldsymbol{y}|\boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{y}|\boldsymbol{F}\boldsymbol{x}, \sigma^2\boldsymbol{I}\right)$$

$$p(\boldsymbol{x}|\boldsymbol{y}) = \mathcal{N}\left(\boldsymbol{x}|\boldsymbol{\Sigma}\boldsymbol{F}^T\boldsymbol{y}, \sigma^2\boldsymbol{\Sigma}\right)$$
$$\boldsymbol{\Sigma} = (\boldsymbol{F}^T\boldsymbol{F} + \sigma^2\boldsymbol{A}^{-1})^{-1}$$

For our model:

Prior over parameters:

$$p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{0}, \lambda^{-1}\boldsymbol{I})$$

Data Likelihood

$$p(\boldsymbol{y}|\boldsymbol{X}, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{\Phi}\boldsymbol{\theta}, \sigma^2\boldsymbol{I})$$

Posterior over parameters:

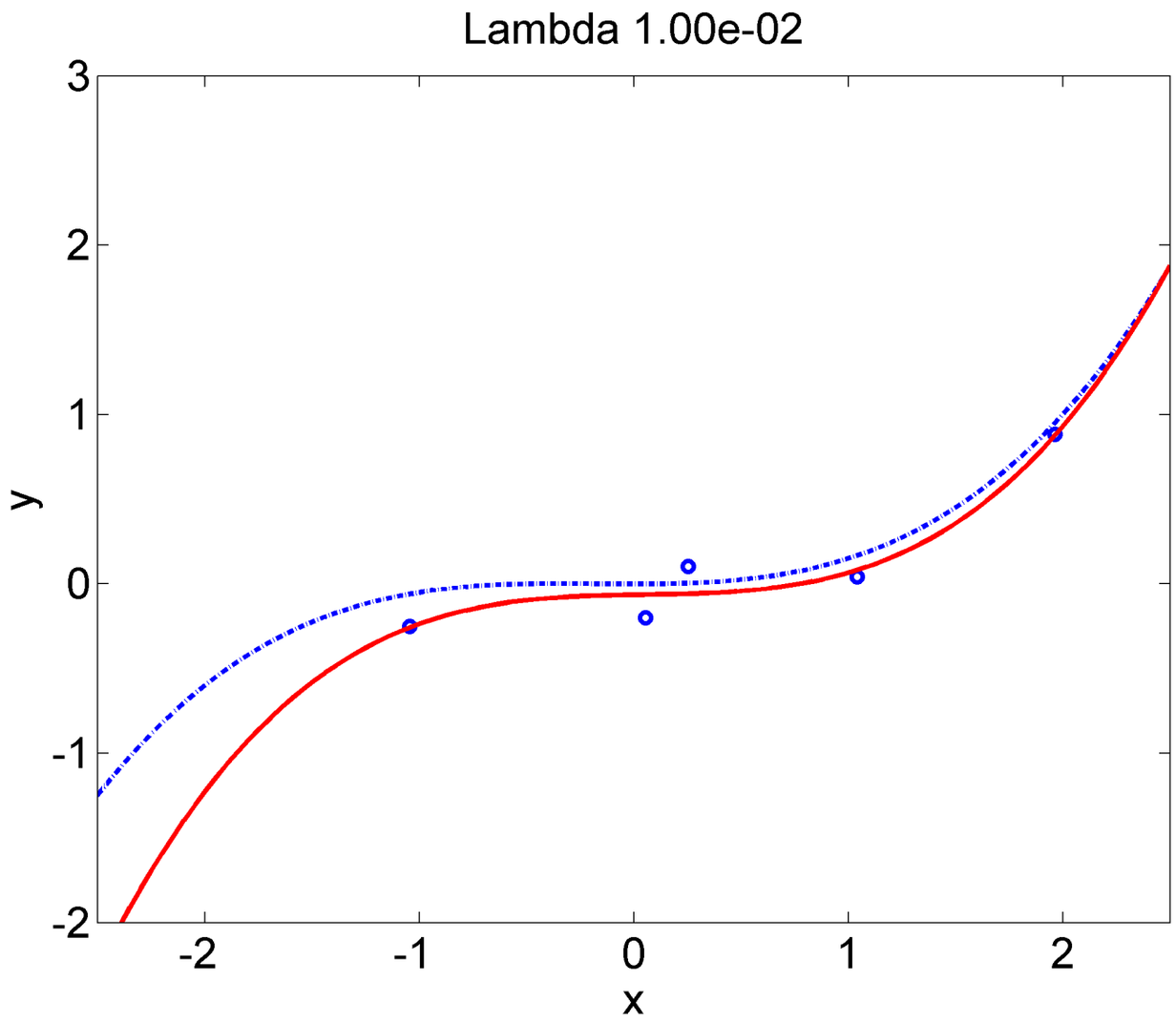
$$p(\boldsymbol{\theta}|\boldsymbol{y}, \boldsymbol{X}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_N, \boldsymbol{\Sigma}_N)$$

$$\boldsymbol{\Sigma}_N = (\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \boldsymbol{\sigma}^2\boldsymbol{\lambda}\boldsymbol{I})^{-1}$$
$$\boldsymbol{\mu}_N = \boldsymbol{\Sigma}_N\boldsymbol{\Phi}^T\boldsymbol{y}$$

What to do with the posterior?

We could sample from it to **estimate uncertainty**

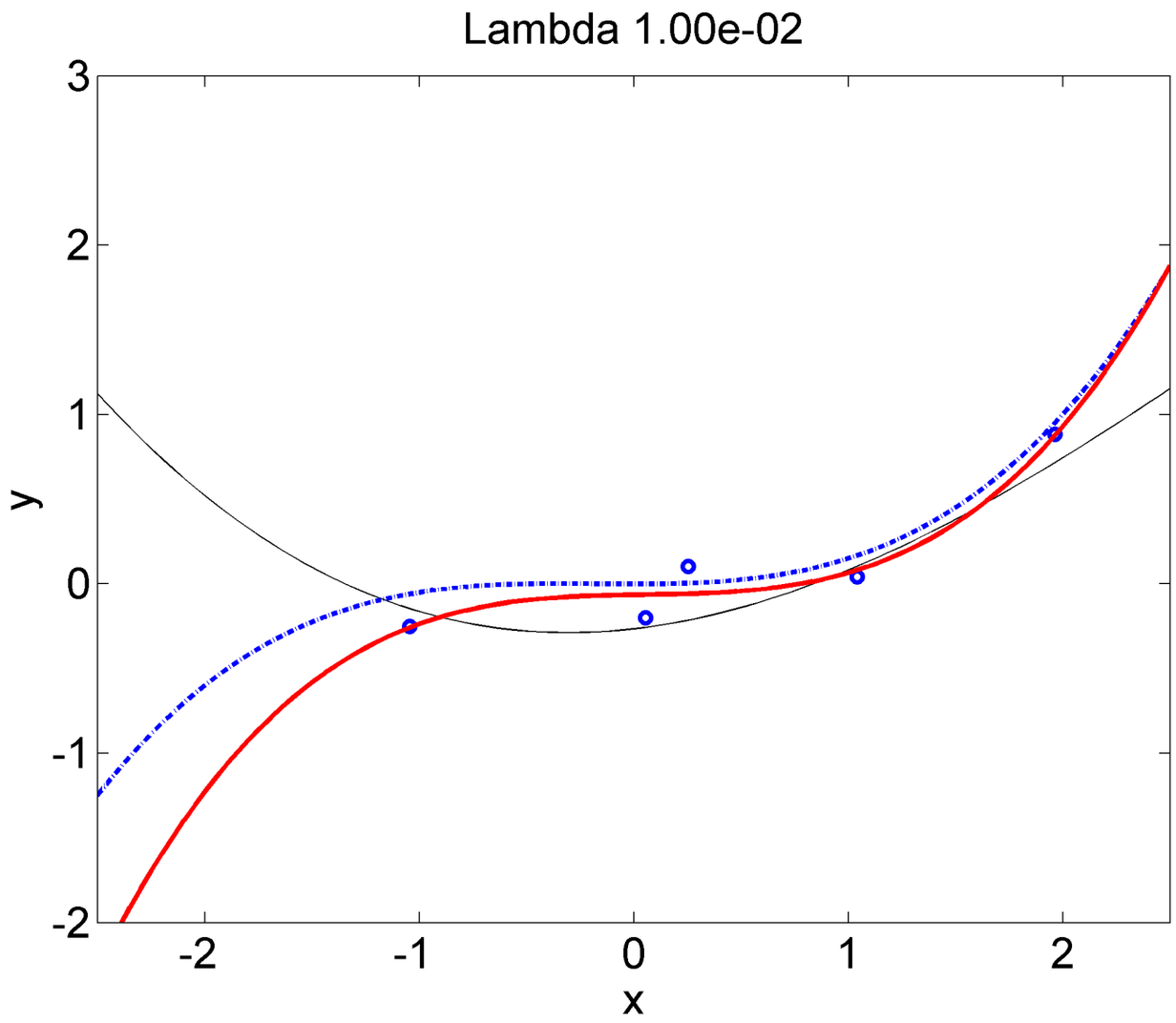
$$\boldsymbol{\theta}_i \sim p(\boldsymbol{\theta}|\boldsymbol{y}, \boldsymbol{X}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_N, \boldsymbol{\Sigma}_N)$$



What to do with the posterior?

We could sample from it to **estimate uncertainty**

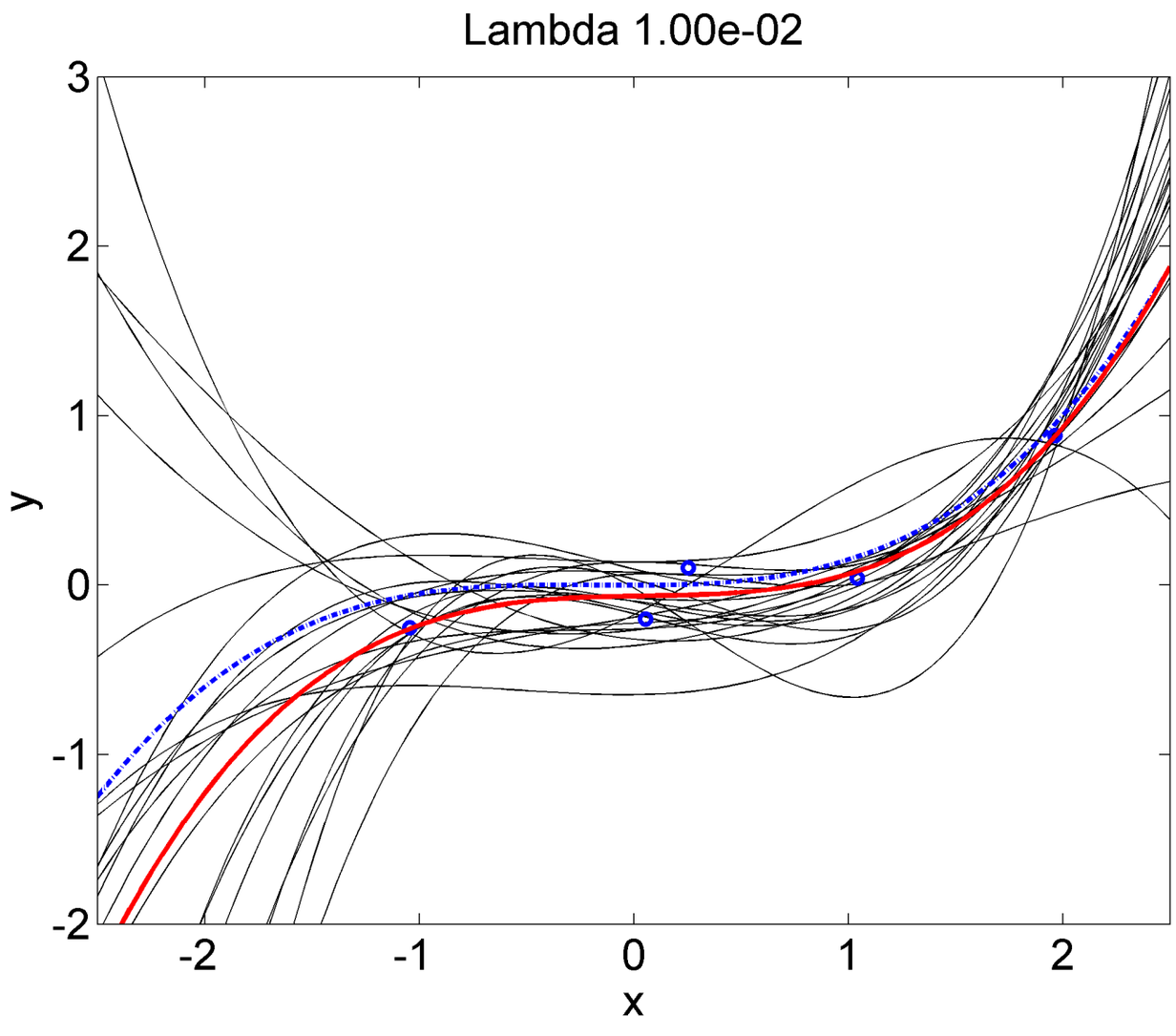
$$\boldsymbol{\theta}_i \sim p(\boldsymbol{\theta}|\boldsymbol{y}, \boldsymbol{X}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_N, \boldsymbol{\Sigma}_N)$$



What to do with the posterior?

We could sample from it to **estimate uncertainty**

$$\boldsymbol{\theta}_i \sim p(\boldsymbol{\theta}|\boldsymbol{y}, \boldsymbol{X}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_N, \boldsymbol{\Sigma}_N)$$



Can we avoid the parameters $\boldsymbol{\theta}$?

Bayesian Fundamentalism: We should not!

We don‘t care about parameters. We care about predictions!

$$p(y_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y})$$

Full Bayesian Regression

We can also do that in closed form: integrate out all possible parameters

$$p(y_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) = \int p(y_*|\boldsymbol{x}_*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y}) d\boldsymbol{\theta}$$

likelihood

parameter posterior

pred. function value

test input

training data

***Intuition*:** If you assign each parameter estimator a “probability of being right”, the average of these parameter estimators will be better than the single one

Full Bayesian Regression

We can also do that in closed form: integrate out all possible parameters

$$p(y_*|\boldsymbol{x}_*, \boldsymbol{X}, \boldsymbol{y}) = \int p(y_*|\boldsymbol{x}_*, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\boldsymbol{X}, \boldsymbol{y}) d\boldsymbol{\theta}$$

likelihood — parameter posterior

pred. function value — test input — training data

Predictive Distribution is again a Gaussian

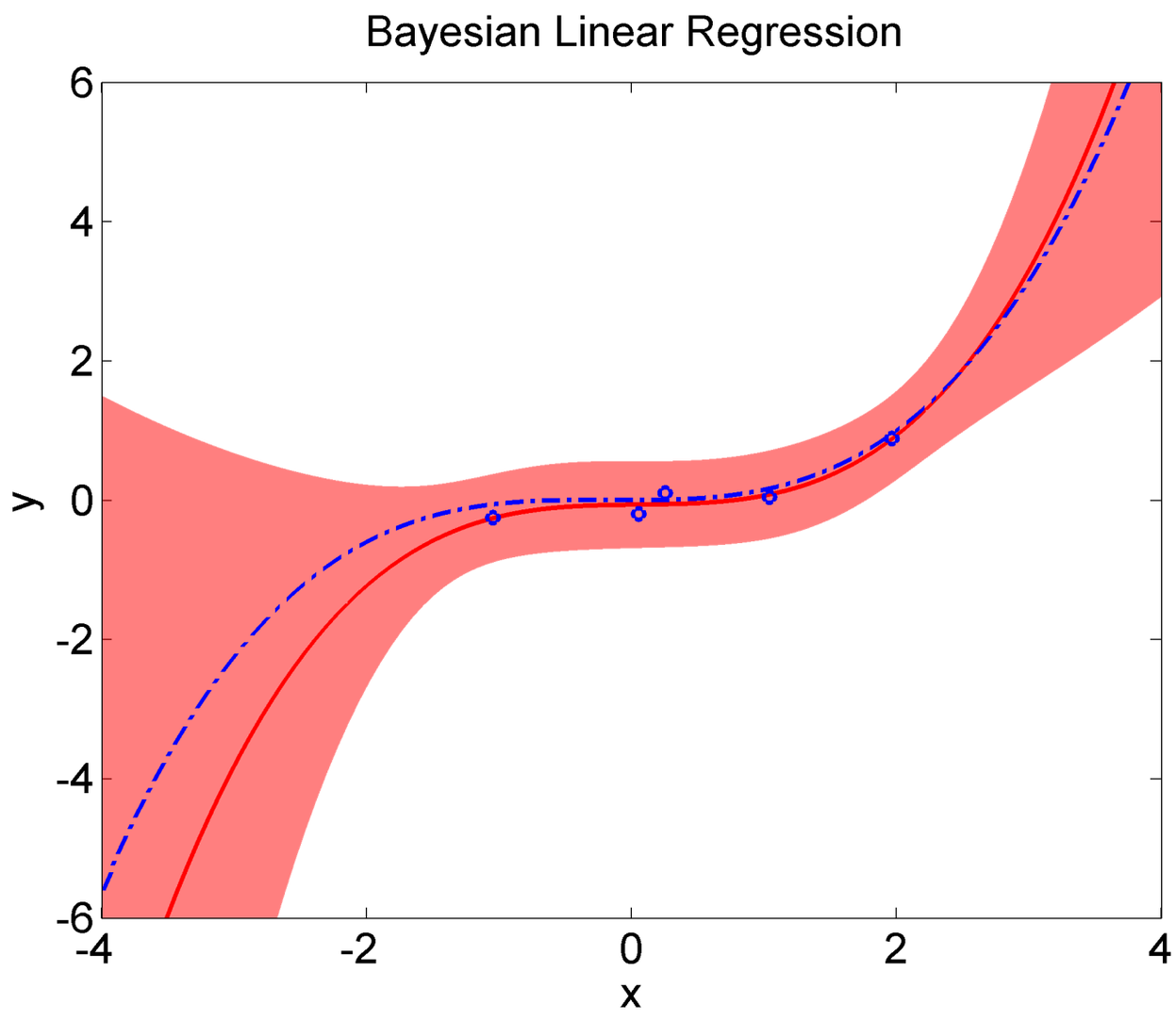
$$p(y_*|\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{x}_*) = \mathcal{N}(\boldsymbol{y}_*|\mu(\boldsymbol{x}_*), \sigma^2(\boldsymbol{x}_*))$$

$$\mu(\boldsymbol{x}_*) = \boldsymbol{\phi}^T(\boldsymbol{x}_*)(\boldsymbol{\Phi}^T\boldsymbol{\Phi} + \boldsymbol{\sigma}^2\lambda\boldsymbol{I})^{-1}\boldsymbol{\Phi}^T\boldsymbol{y}$$

$$\sigma^2(\boldsymbol{x}_*) = \sigma^2(1 + \boldsymbol{\phi}^T(\boldsymbol{x}_*)\boldsymbol{\Sigma}_N\boldsymbol{\phi}(\boldsymbol{x}_*))$$

State Dependent Variance!

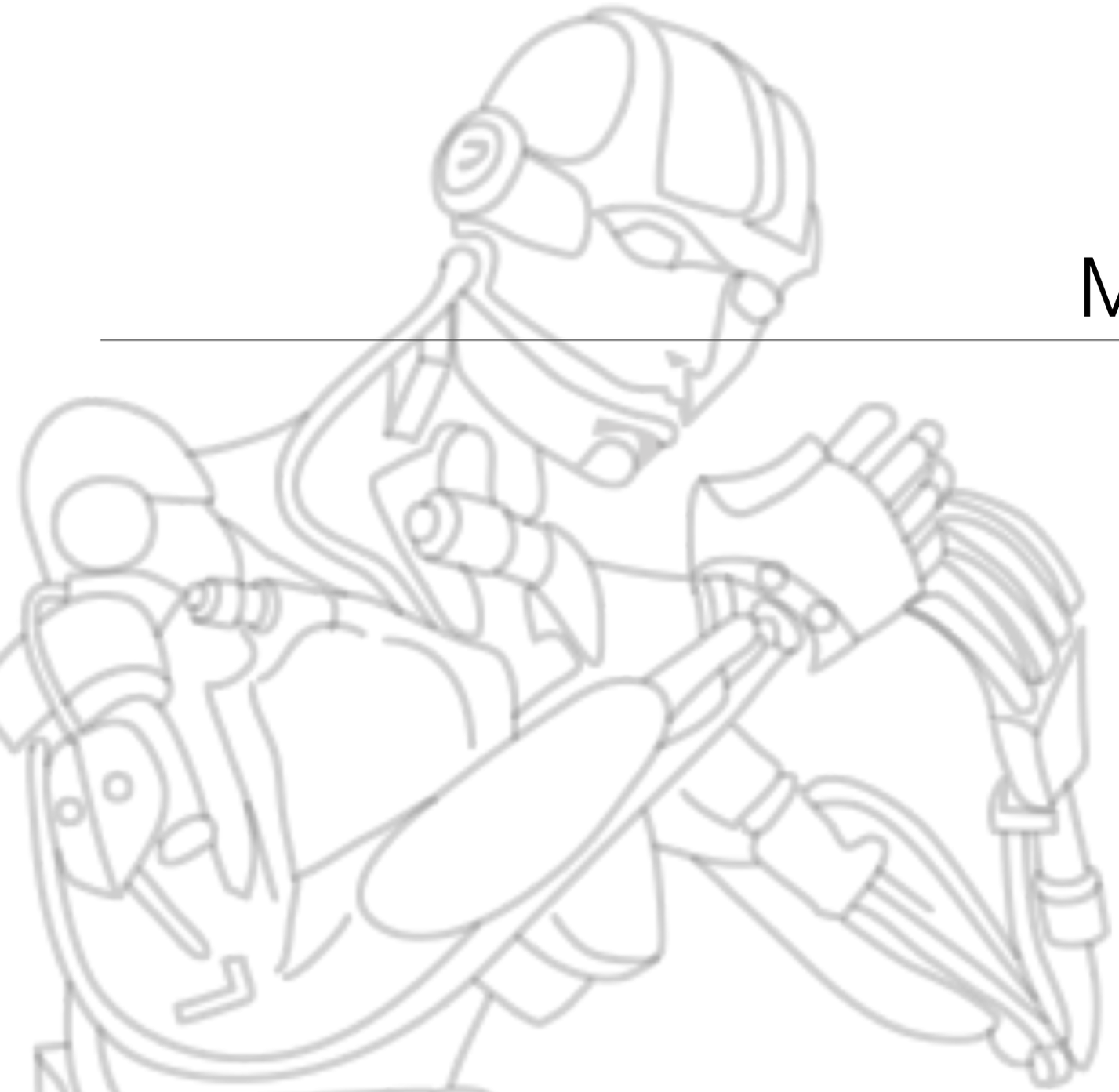
Integrating out the parameters



Variance depends on the information in the data!

Quick Summary

- Models that are linear in the parameters: $y = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{\theta}$
- **Overfitting** is bad
- **Model selection** (leave-one-out cross validation)
- **Parameter Estimation in Regression:** Frequentist vs. Bayesian
  - **Cost functions like Least Squares** go back to Gauss…
  - **Least Squares** ~ Maximum Likelihood estimation (ML; Frequentist)
  - **Ridge Regression** ~ Maximum a Posteriori estimation (MAP; Bayesian)
- **Full Bayesian Regression** integrates out the parameters when predicting
  - State dependent uncertainty



Machine Learning 101b

Jan Peters
Gerhard Neumann

Purpose of this Lecture

- How can we define such features for general machine learning problems?

- Can we avoid or automate the feature specification?

- Familiarize you with non-parametric models

Content of this Lecture

Constructing Basis Functions

- Radial Basis Function Networks

Non-Parametric Approaches

- Locally Weighted Regression
- Kernel Methods

What we did so far…

- Models that are linear in the parameters: $y = \boldsymbol{\phi}(\boldsymbol{x})^T \boldsymbol{\theta}$
- **Parameter Estimation in Regression**
  - **Least Squares** ~ Maximum Likelihood estimation (ML; Frequentist)
  - **Ridge Regression** ~ Maximum a Posteriori estimation (MAP; Bayesian)
  - **Full Bayesian Regression** integrates out the parameters when predicting
    - State dependent uncertainty

However, for most problems good features are not easy to find

What to do when you don't know the features?

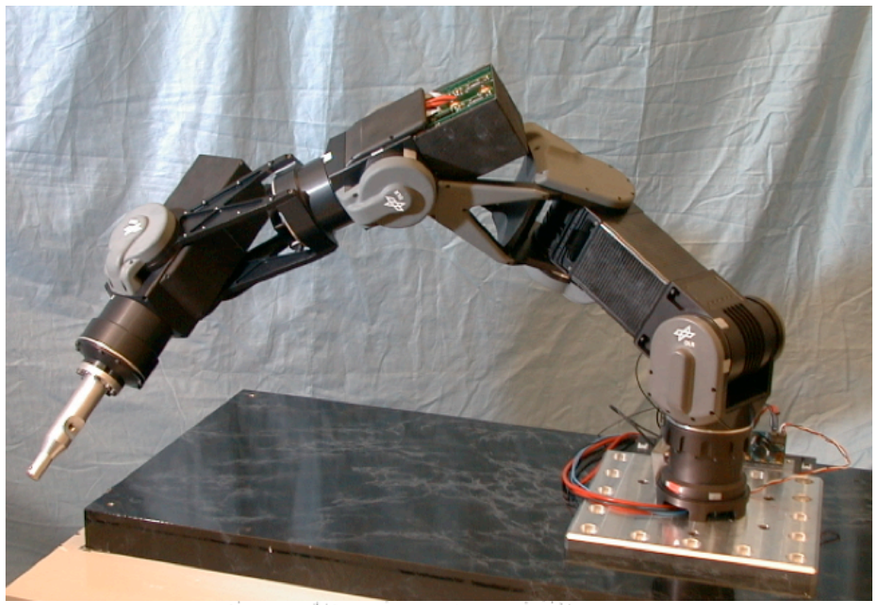
In most real applications, we know **some** good features.

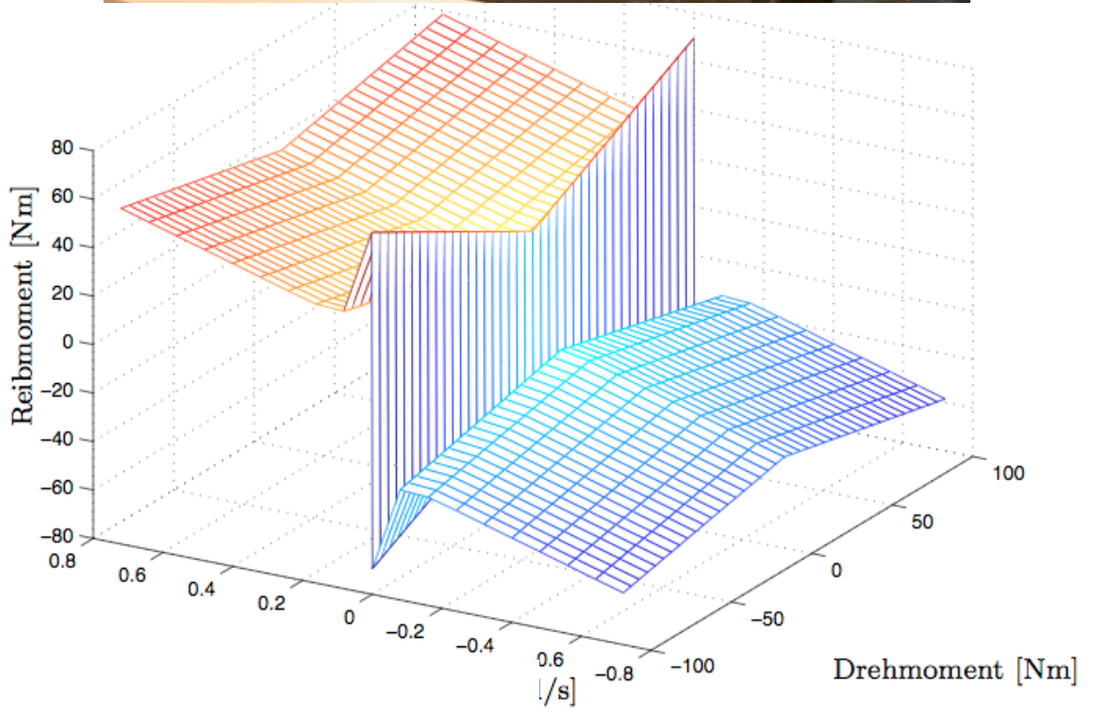
However, we almost certainly **don't know all** features we need.

Example: Rigid body dynamics

- Friction has no good features
- Unknown dynamics causes huge problems (requires more state variables).

There may also be way too many features!





Hand-crafted features are almost never enough...

Can we avoid having to find good features?

Yes, we can!

We need to find machine learning approaches that **generate the features automatically** from data.

- **Type 1**: *Automatic Basis Function Construction* constructs basis functions from data.
- **Type 2**: *Non-Parametric Regression* look at data locally and interpolate with similar data.
- **Type 3**: *Kernel Regression* finds the features implicitly by going into *function space* using a *kernel*

Type 1: Construct Basis Functions from Data

Classical idea behind "neural networks"

- Multi-Layer Perceptrons (see Machine Learning: Statistical Approaches)
- Radial Basis Function Networks

Radial Basis Function Network

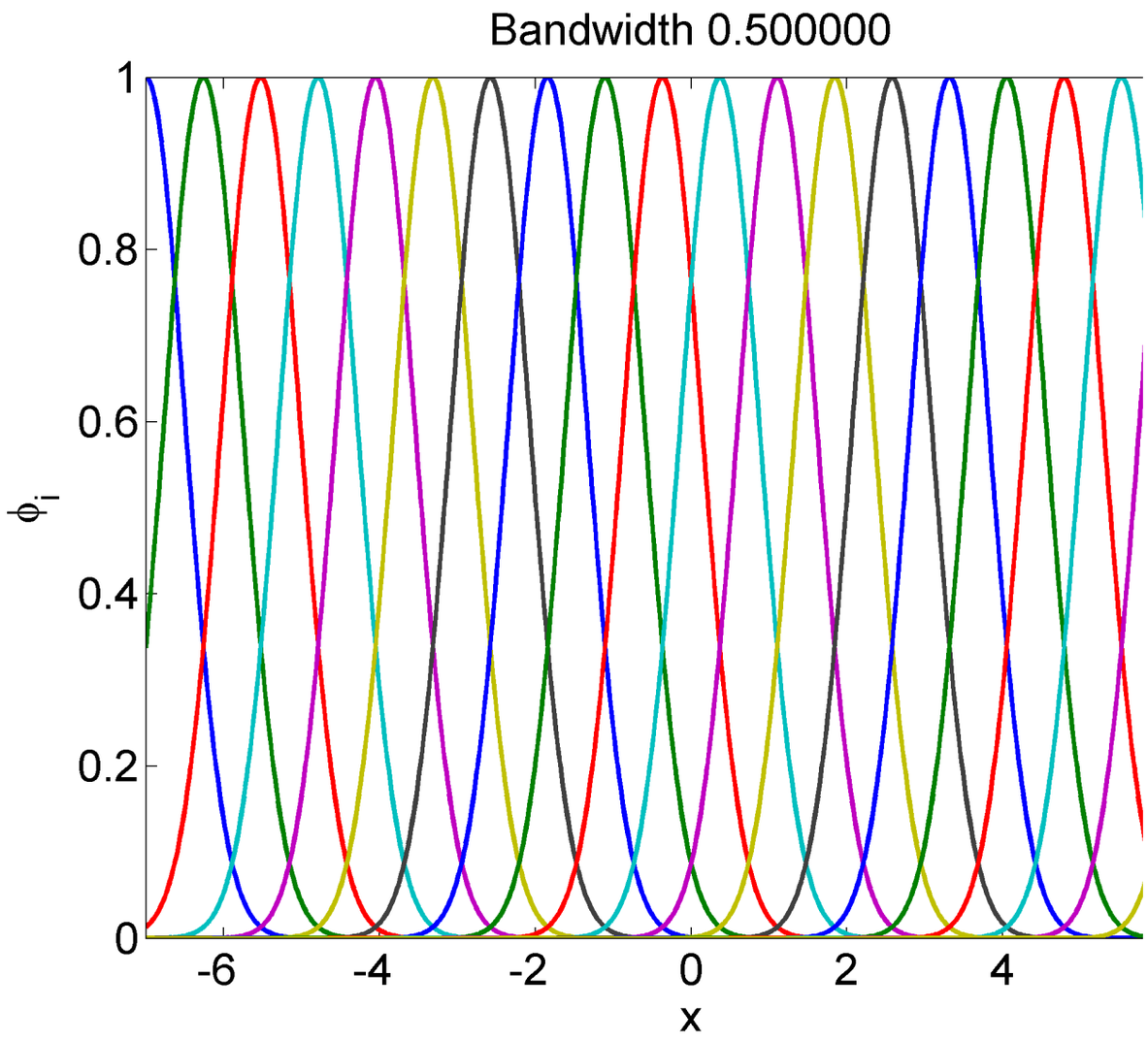
Assume a smoothness prior and obtain the cost function

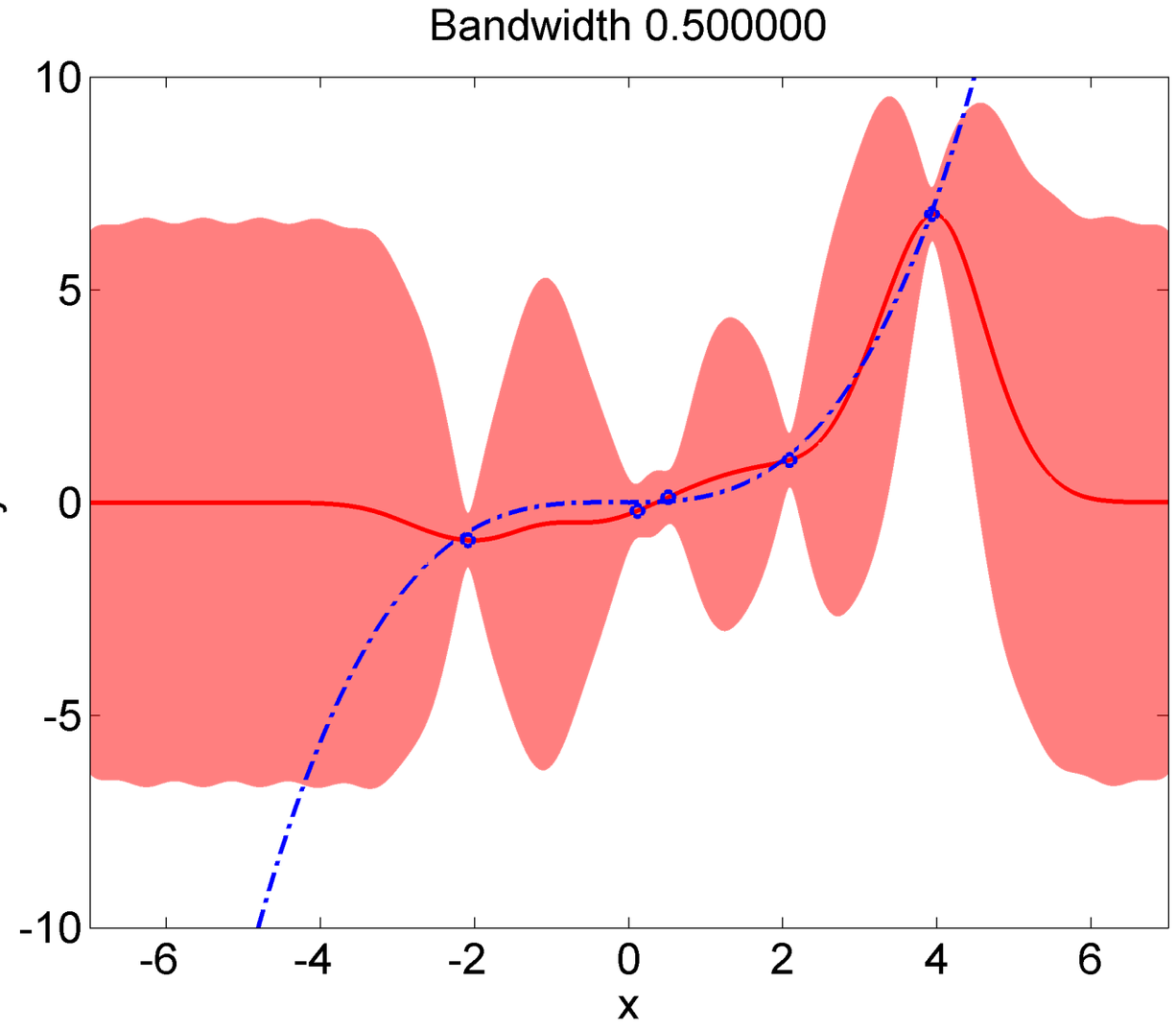
$$J = \frac{1}{2}\sum_{i=1}^{N}\left[(y_i - \mathbf{f}_\theta(\mathbf{x}_i))^2 + \left\|\frac{d^2}{dx^2}\mathbf{f}_\theta(\mathbf{x}_i)\right\|^2\right]$$

This prior yields radial basis functions as features:

$$\begin{aligned} f_{\boldsymbol{\theta}}(\boldsymbol{x}) &= \sum_i \theta_i \exp\left(\frac{-||\boldsymbol{x}-\boldsymbol{\mu}_i||^2}{2l^2}\right) \\ &= \boldsymbol{\phi}(\boldsymbol{x})^T\boldsymbol{\theta}, \quad \text{with } \phi_i(\boldsymbol{x}) = \exp\left(\frac{-||\boldsymbol{x}-\boldsymbol{\mu}_i||^2}{2l^2}\right) \end{aligned}$$

Example: Radial Basis Function Features





Radial Basis Functions Hyperparameters

Let's look again at

$$\phi_i(\boldsymbol{x}) = \exp\left(\frac{-||\boldsymbol{x} - \boldsymbol{\mu}_i||^2}{2l^2}\right)$$

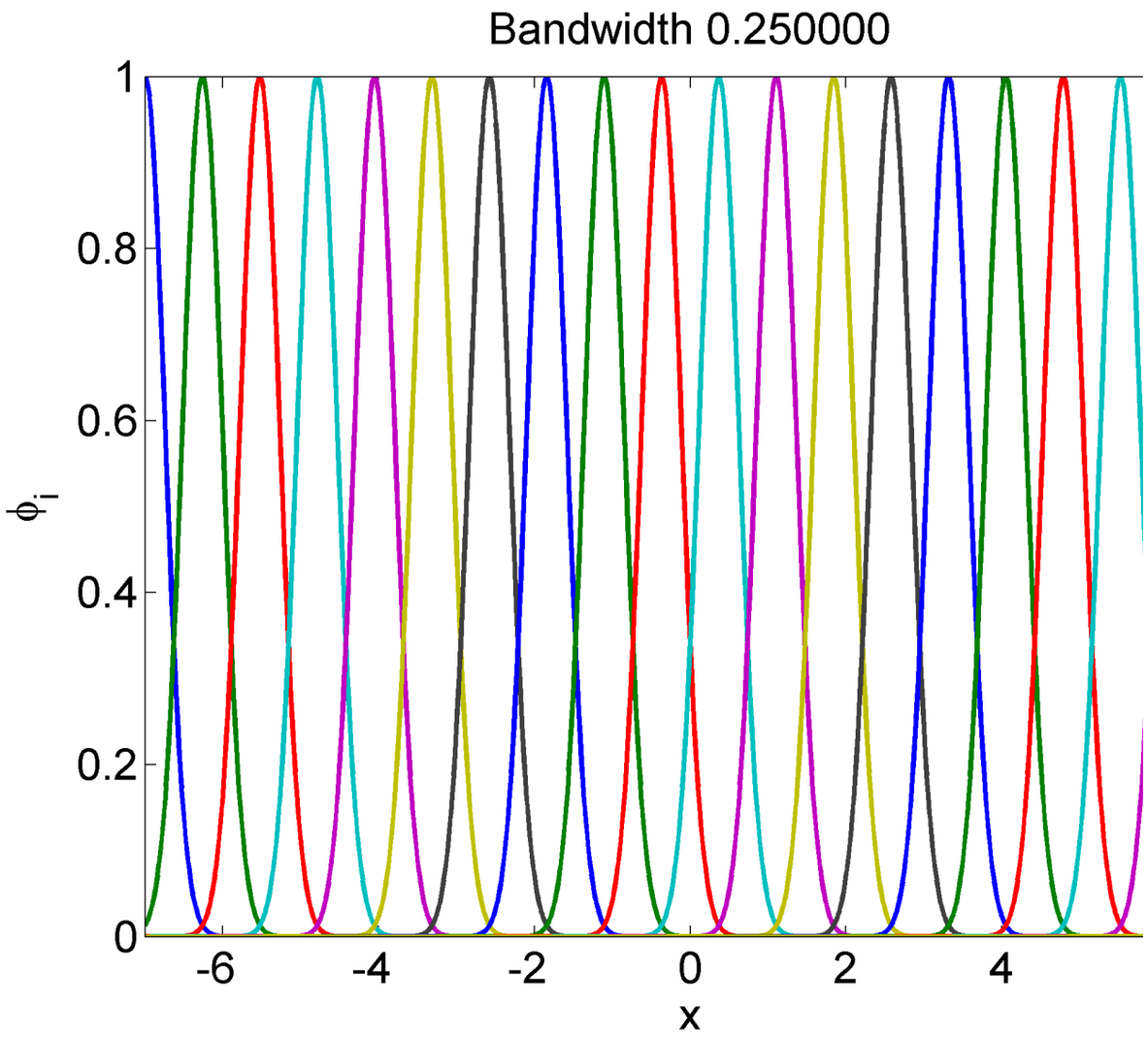
- How do I find the width *l* of the basis functions or the centers $\boldsymbol{\mu}_i$?

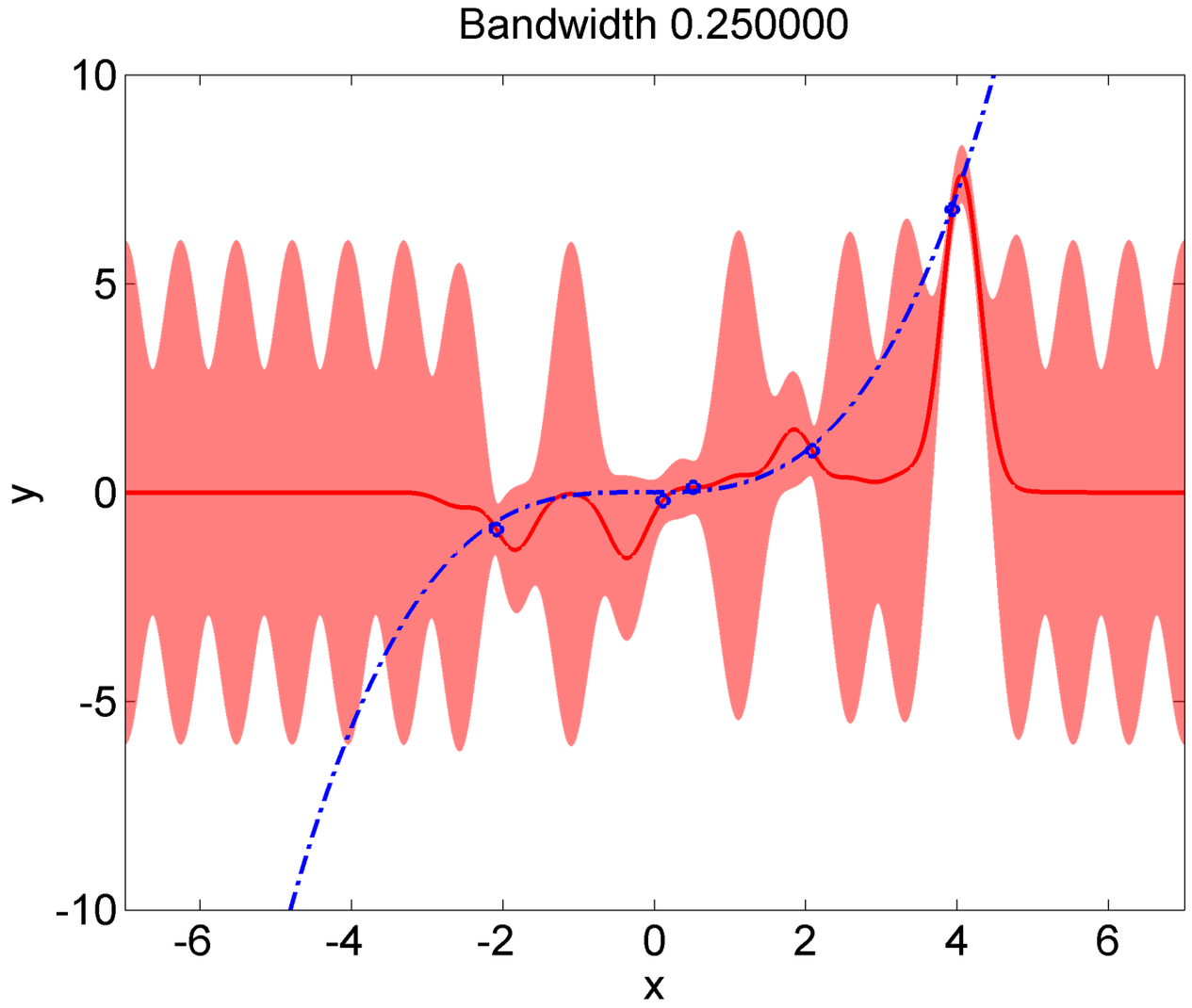
- Linear regression? Nope: not linear in *l or mu* !

- We need to optimize this width on the training set

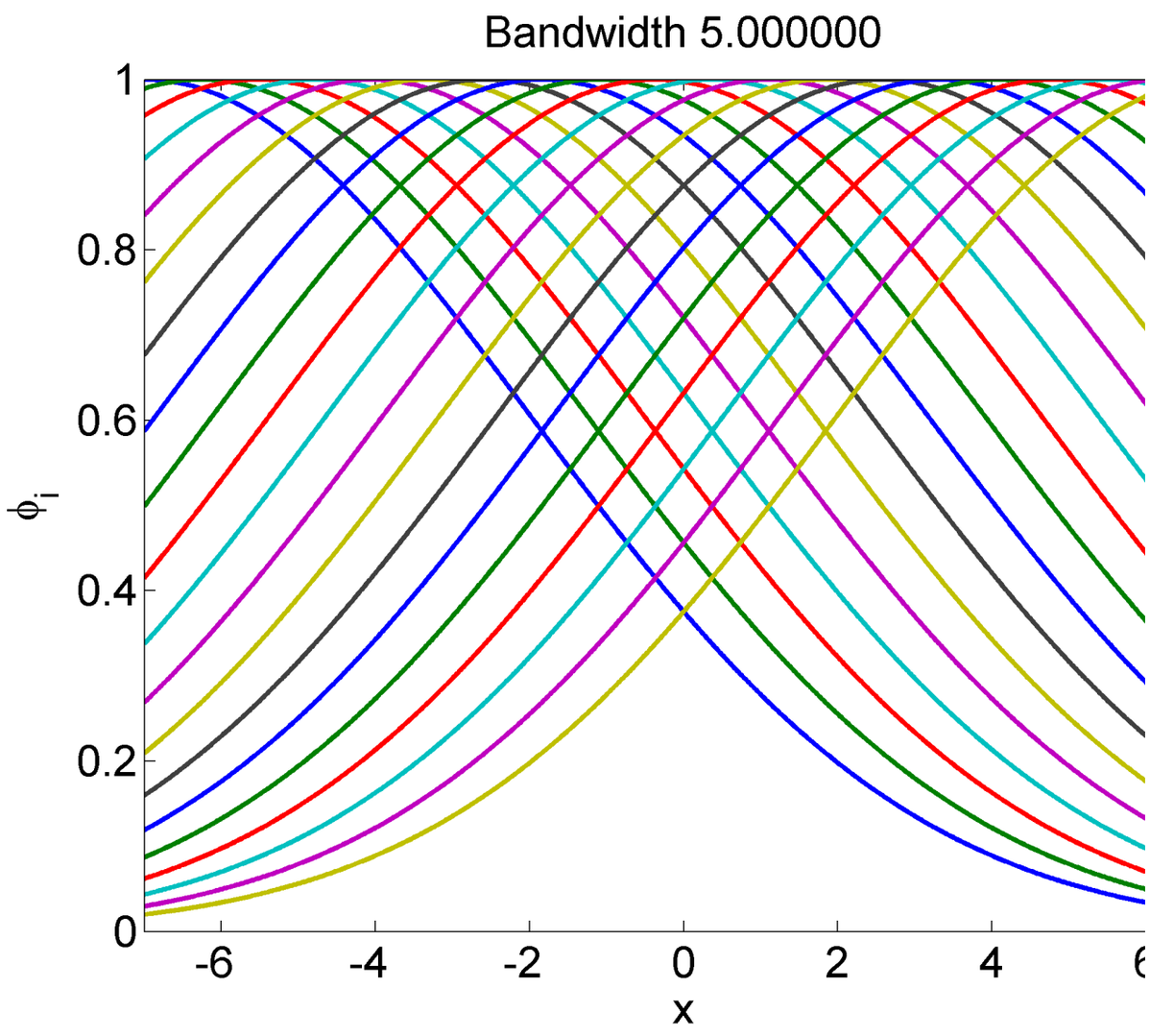
- We can do that by gradient descent: Write down a loss function, take the derivative w.r.t. *l*, and use an algorithm for non-convex optimization

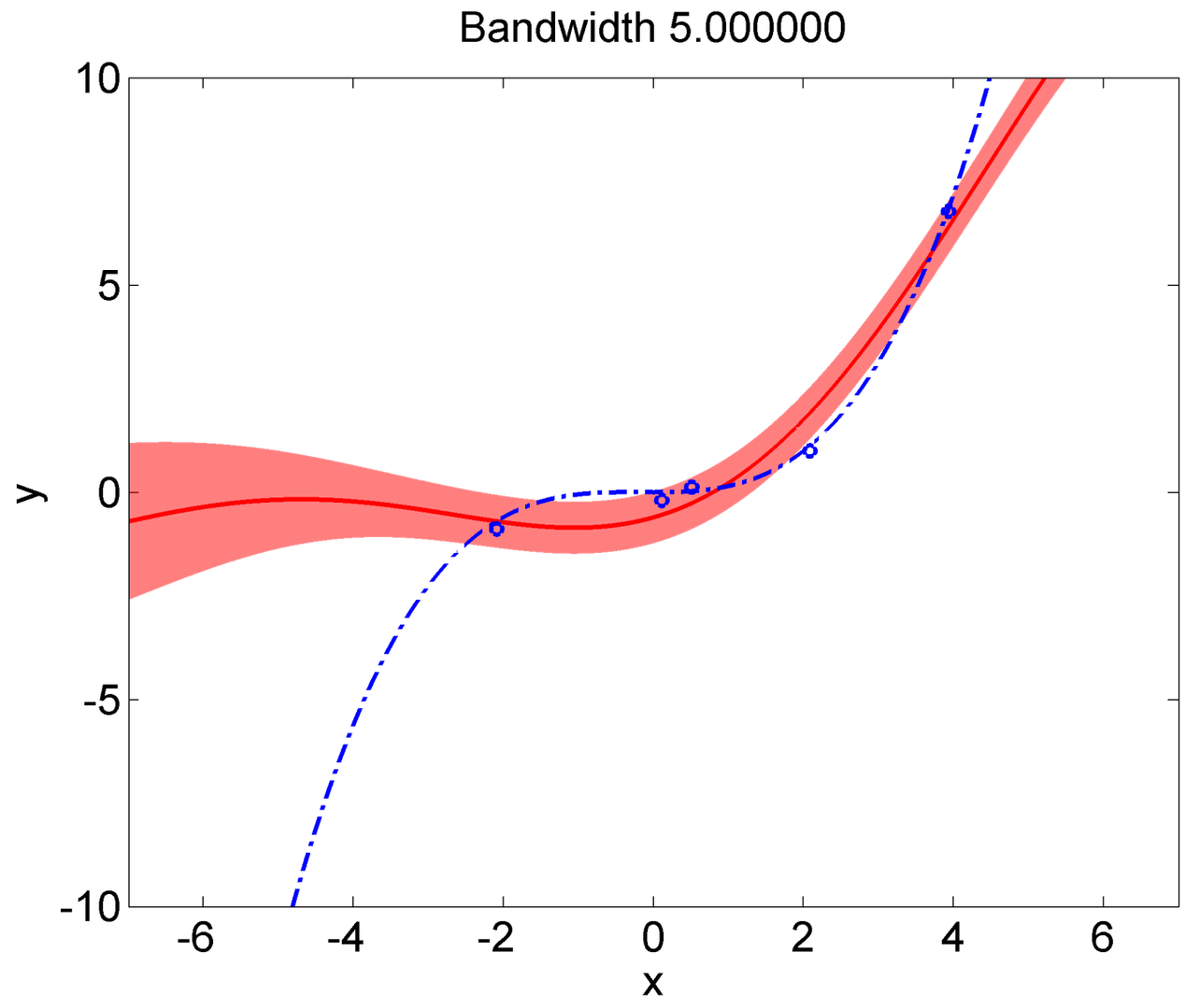
Example: Bandwidth too small





Example: Bandwidth too large





Content of this Lecture

Constructing Basis Functions

- Radial Basis Function Networks

Non-Parametric Approaches

- Locally Weighted Regression
- Kernel Methods

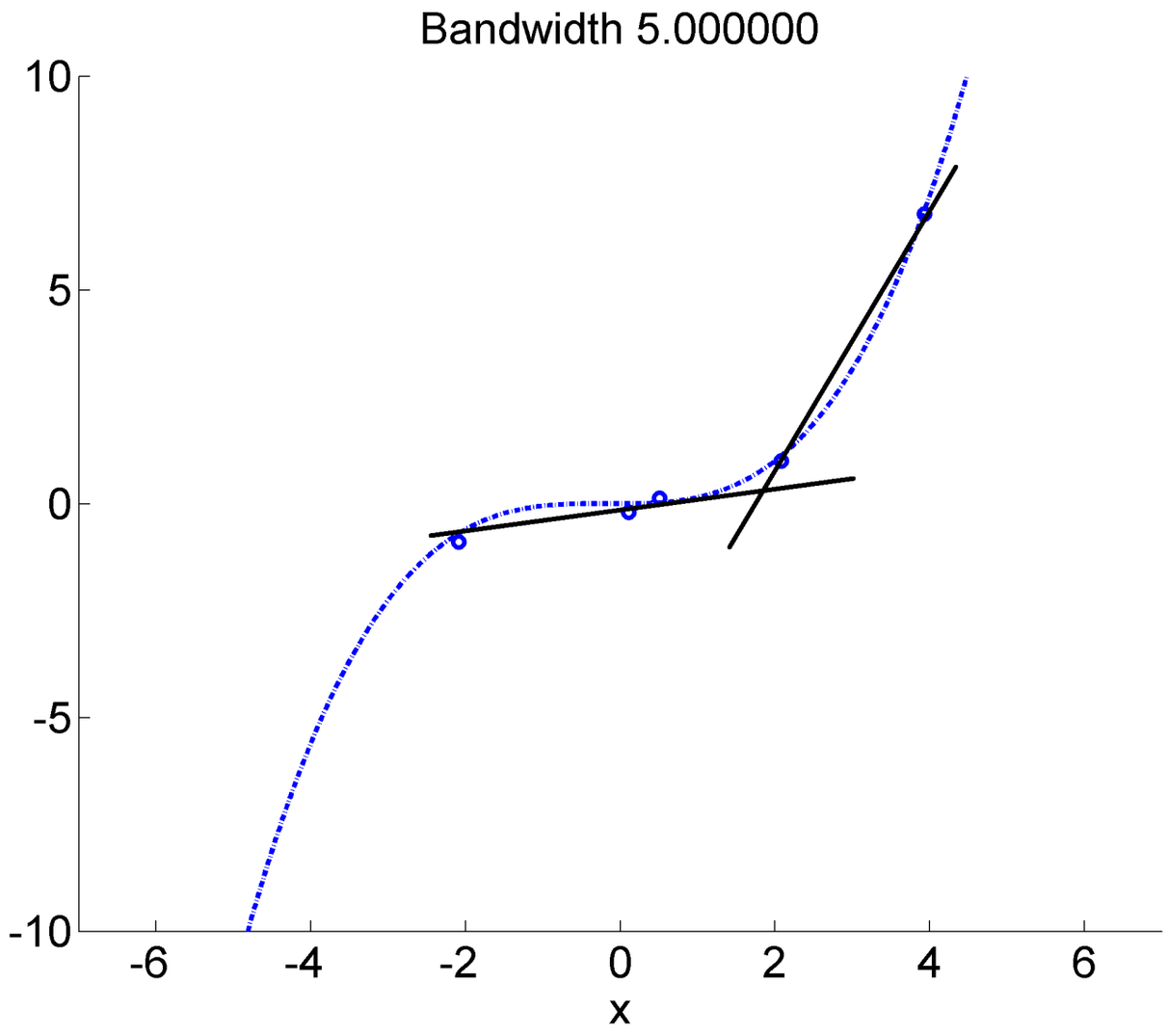
Type 2: Non-Parametric Regression

- If you choose to have one feature/basis function per sample, you have a "**non-parametric method**" ⇒ Don't need to select the number of bases

- Non-parametric means
  - infinitely many parameters not no parameters
  - expressiveness of the model depends on the number of data points
  - No predetermined "parametric" form necessary
    - (e.g., "5th-degree polynomial")

- One of them is **locally-weighted linear regression**...

Example: Locally Linear Solutions



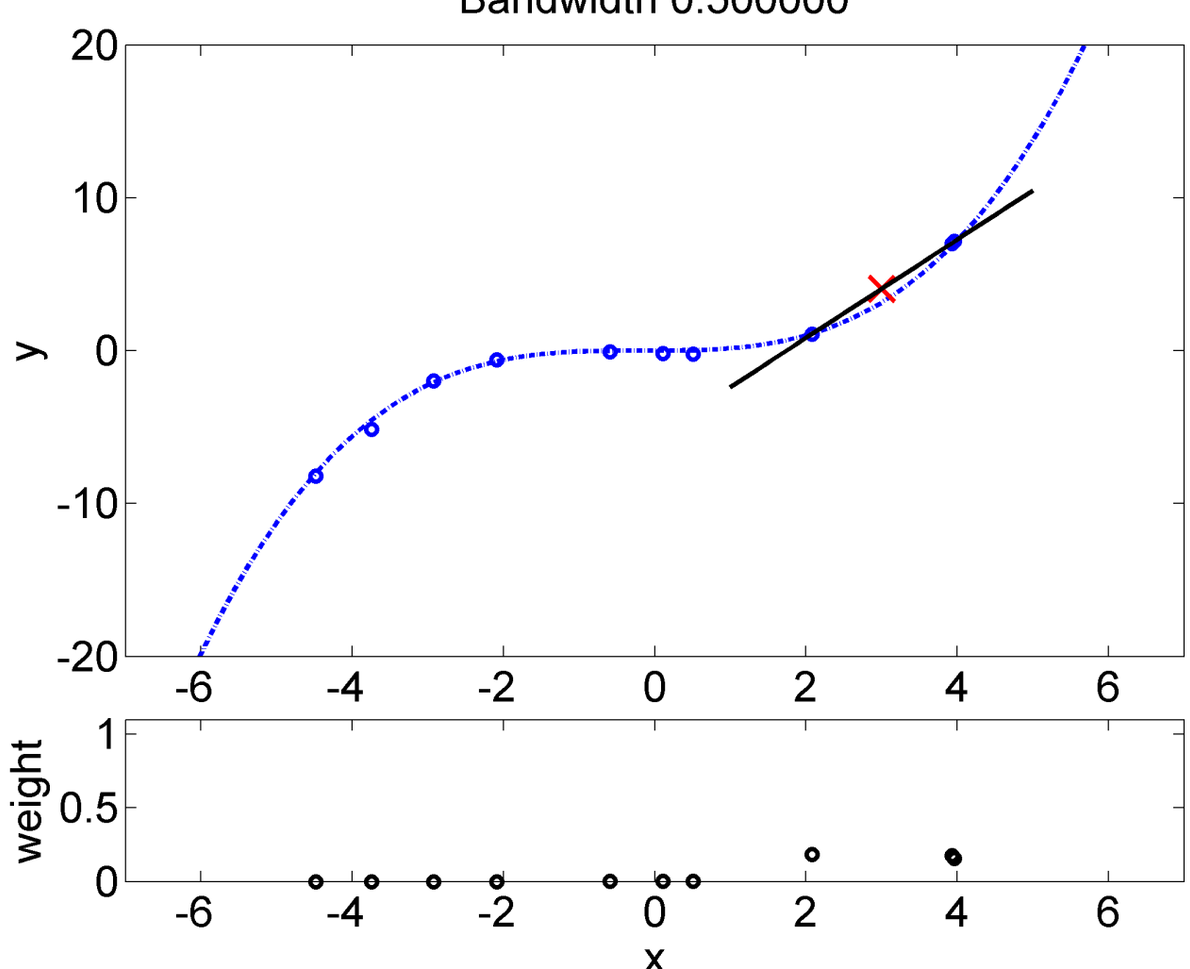
Locally all data is linear!

Locally all data is linear ...

so why don't we take the neighboring data points to predict the solution?

- Use higher importance or weighting of neighboring data points
- For each query point $\boldsymbol{x}$, weight training points $\boldsymbol{x}_i$ by

$$w_i(\boldsymbol{x}) = \exp\left(-\frac{||\boldsymbol{x}-\boldsymbol{x}_i||^2}{2l^2}\right) \ldots \text{squarred exponential weighting}$$



Weighted Linear Regression

Weighted cost function

$$J = \frac{1}{2}\sum_{i=1}^{N} w_i(\mathbf{x})(y_i - \mathbf{f}_\theta(\mathbf{x}_i))^2, \qquad w_i(\boldsymbol{x}) = \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{x}_i||^2}{2l^2}\right)$$

The function is linear in x

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{\theta}^T \begin{bmatrix} 1 \\ \boldsymbol{x} \end{bmatrix} = \boldsymbol{\theta}^T \tilde{\boldsymbol{x}}$$

In matrix form with $W = \text{diag}(w_1, w_2, w_3, \ldots, w_n)$:

$$J = \frac{1}{2}(\tilde{\boldsymbol{X}}\boldsymbol{\theta} - \boldsymbol{y})^T \boldsymbol{W}(\tilde{\boldsymbol{X}}\boldsymbol{\theta} - \boldsymbol{y})$$

Weighted Linear Regression

The solution to this problem: weighted pseudo inverse

$$\boldsymbol{\theta} = (\tilde{\boldsymbol{X}}^T \boldsymbol{W} \tilde{\boldsymbol{X}})^{-1} \tilde{\boldsymbol{X}}^T \boldsymbol{W} \boldsymbol{y}$$

- **W** can be large - don't implement it like this...
- Dismiss data points with small weights / use bsxfun

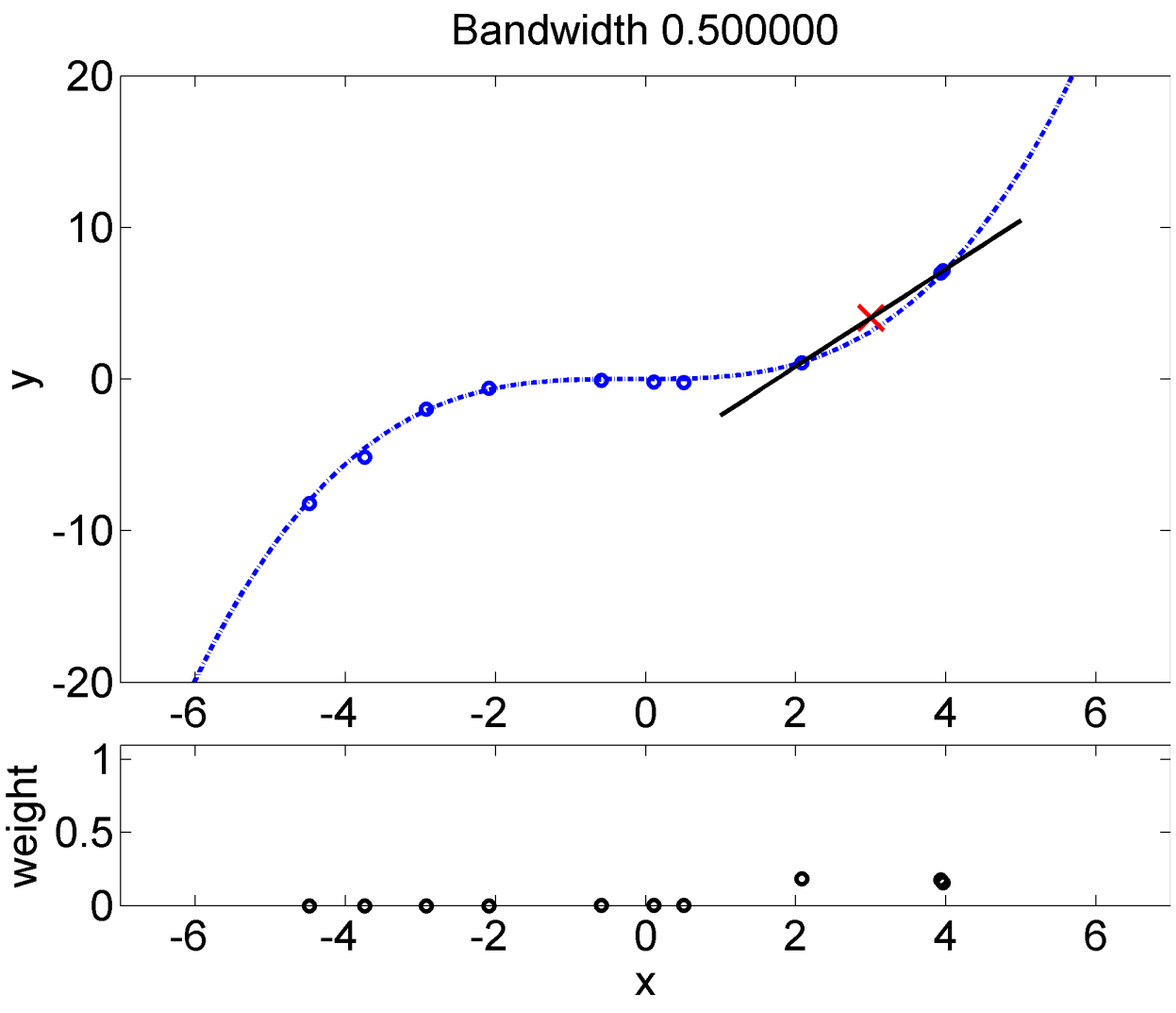
Local Ridge Regression:

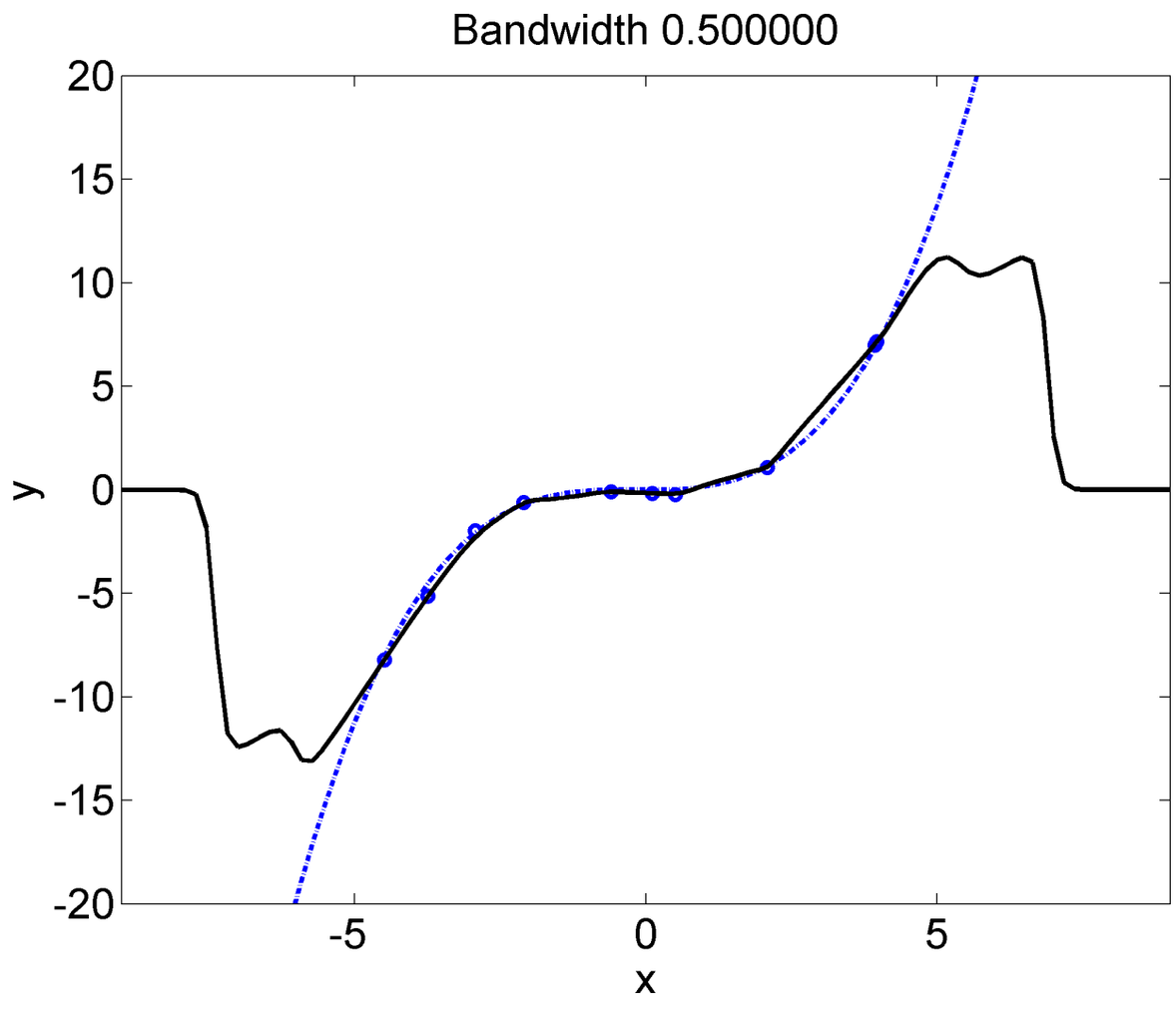
$$\boldsymbol{\theta} = (\tilde{\boldsymbol{X}}^T \boldsymbol{W} \tilde{\boldsymbol{X}} + \sigma^2 \boldsymbol{I})^{-1} \tilde{\boldsymbol{X}}^T \boldsymbol{W} \boldsymbol{y}$$

Advantages: Fast(real-time capable), scales(lots of data), interpolates linearly(useful in control)
Disadvantage: Tuning is not easy

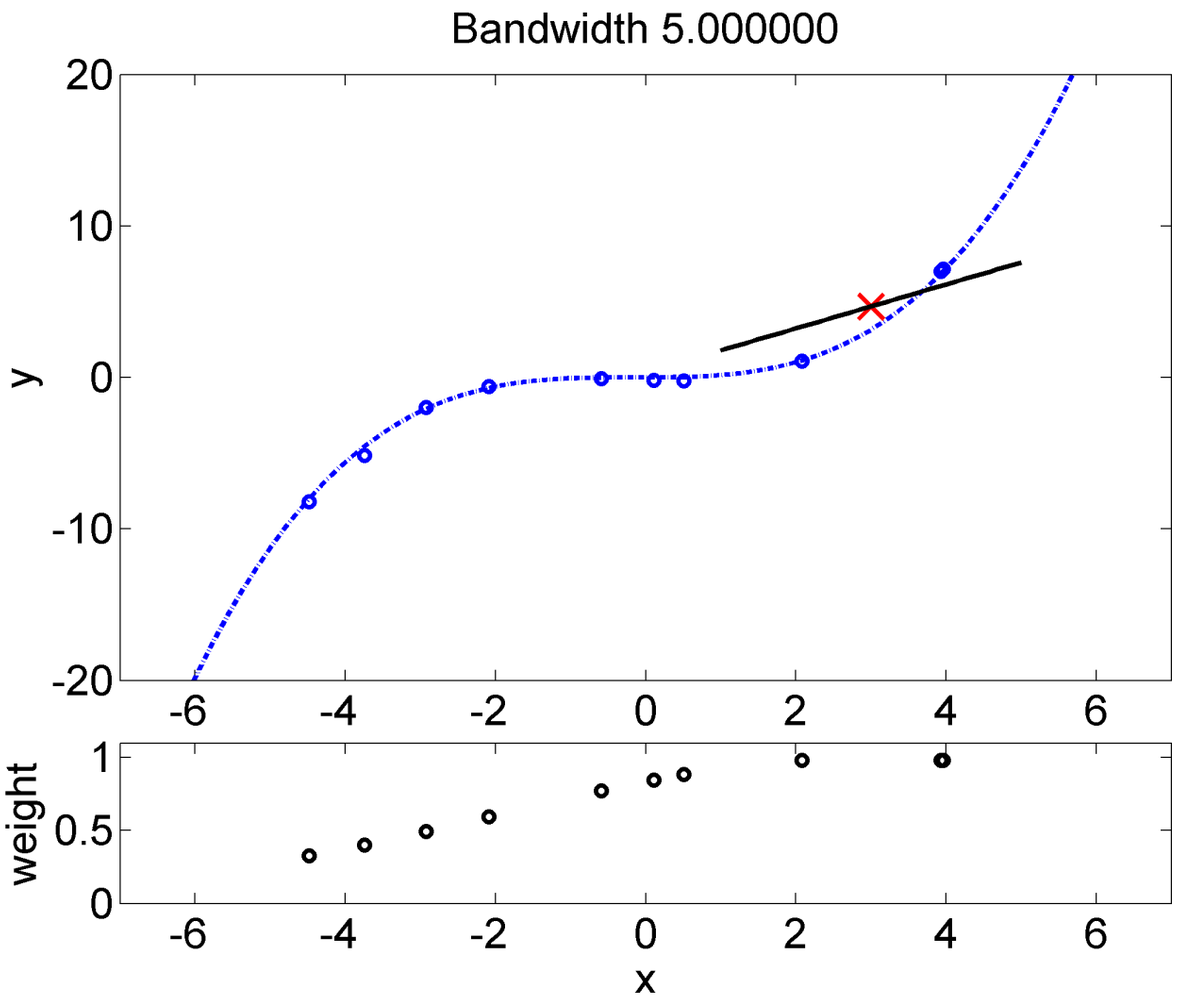
Frequently method of choice for control problems!

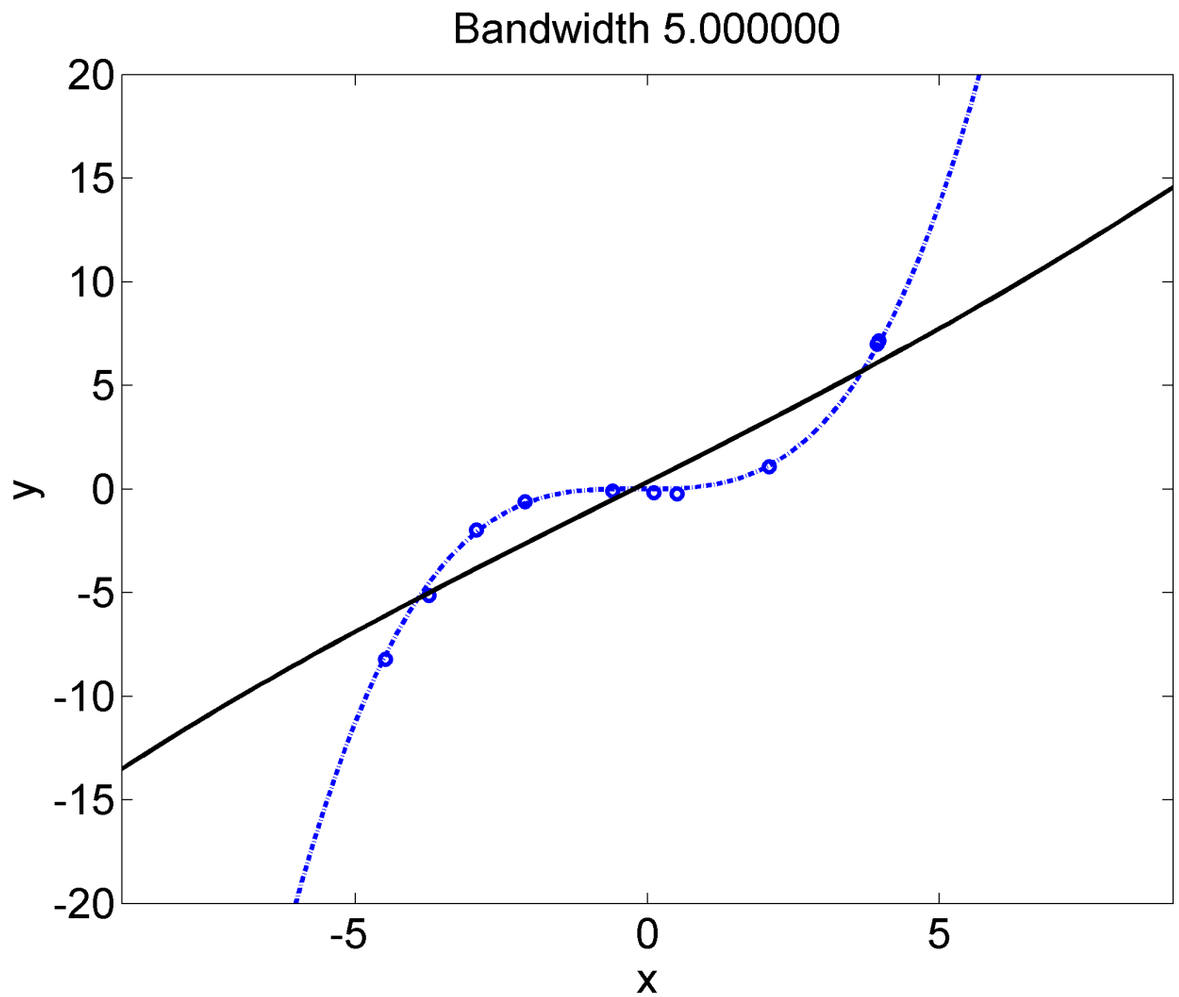
Solution with Locally-Weighted Regression



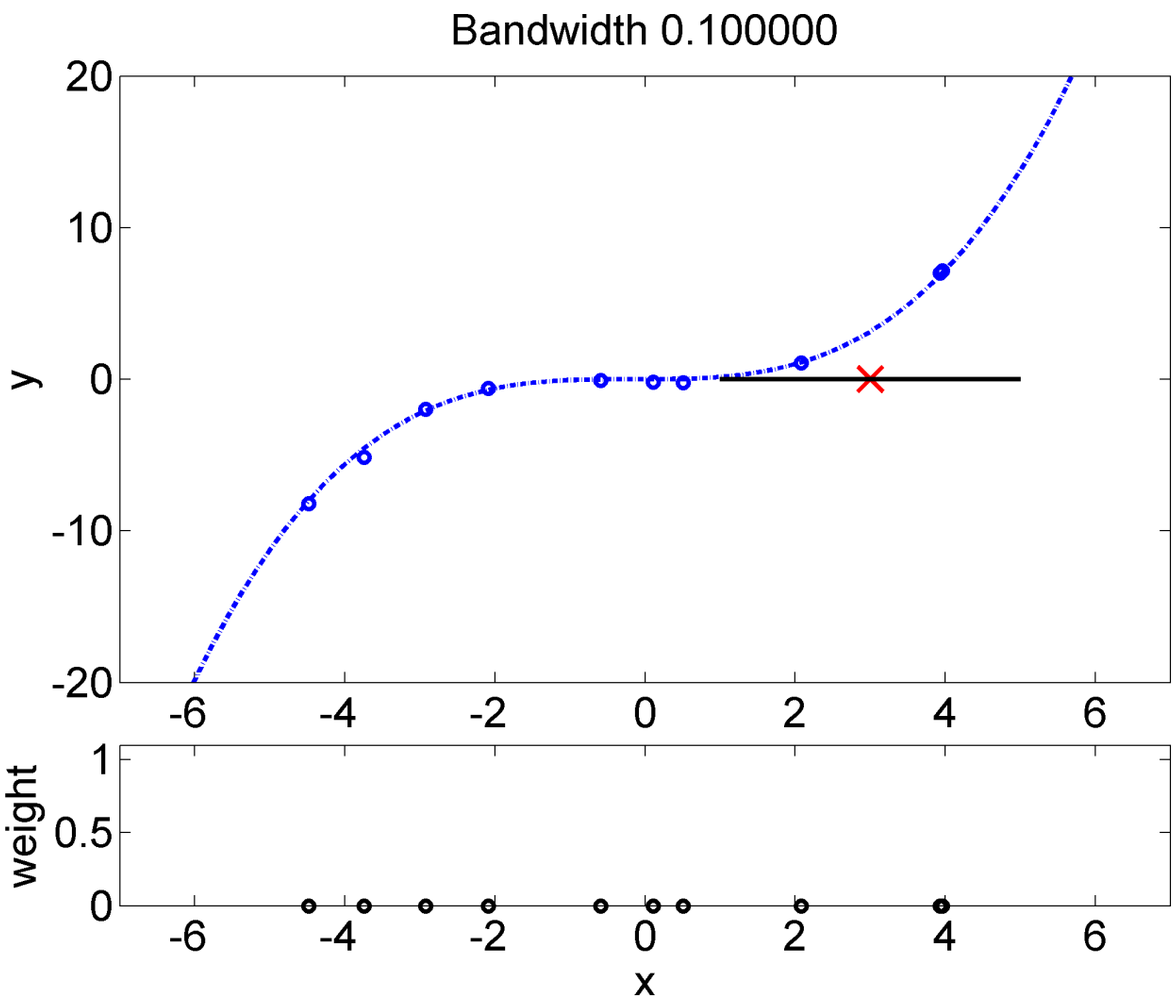


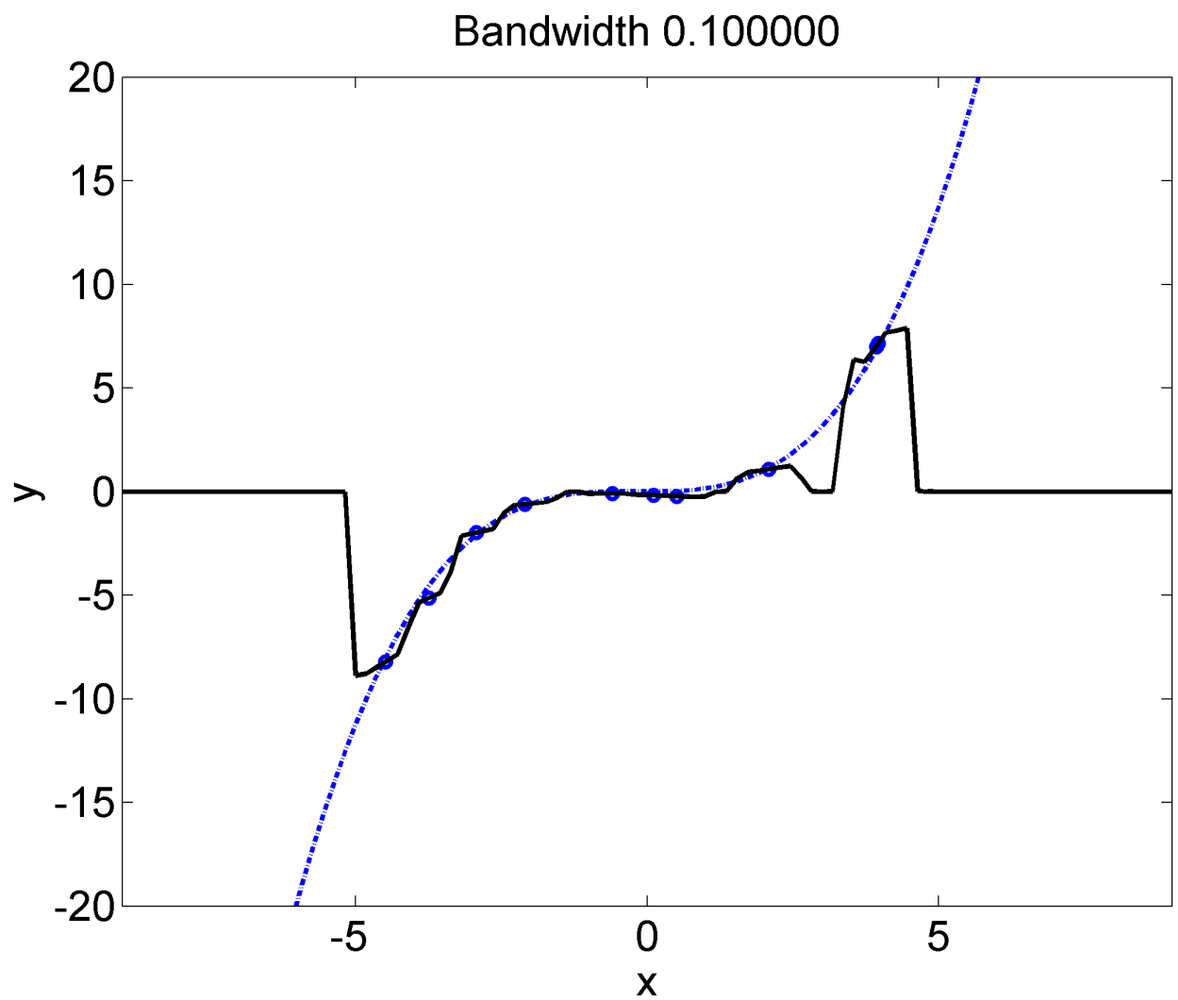
Solution with Locally-Weighted Regression





Solution with Locally-Weighted Regression



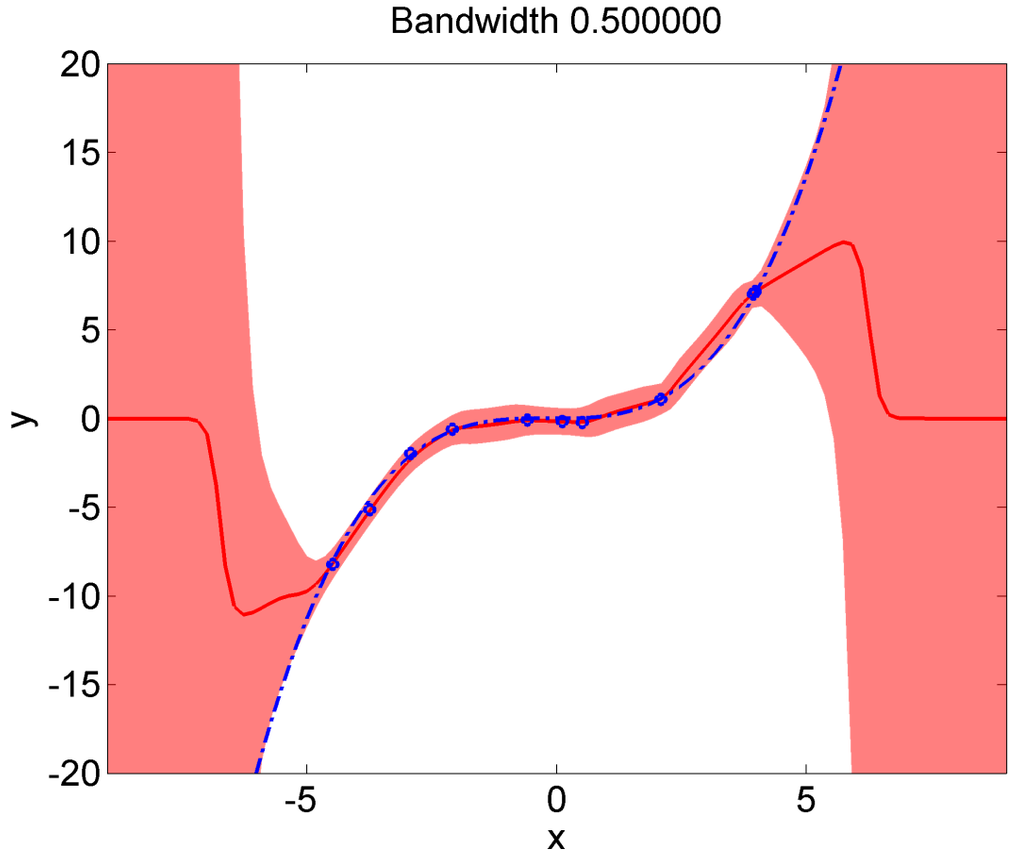


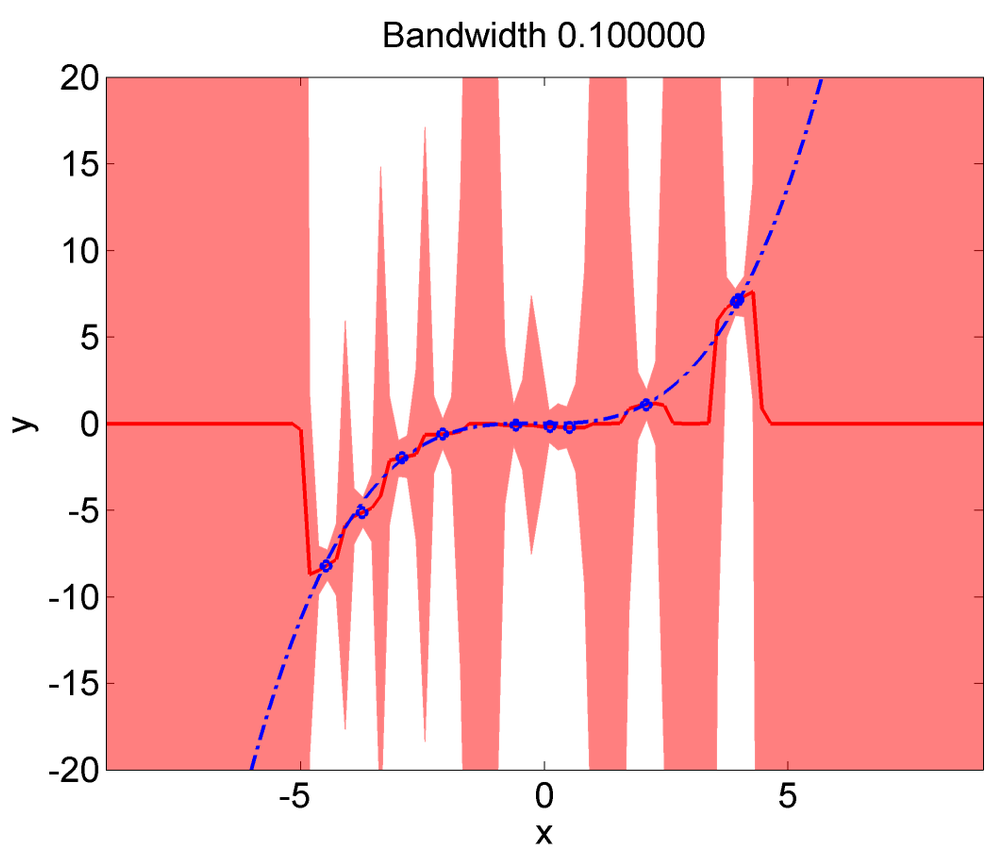
Weighted Linear Regression

Locally Weighted Bayesian Linear Regression

$$p(\boldsymbol{\theta}|\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{W}) = \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\mu}_N, \boldsymbol{\Sigma}_N)$$

$$\boldsymbol{\Sigma}_N = (\tilde{\boldsymbol{X}}^T \boldsymbol{W} \tilde{\boldsymbol{X}} + \sigma^2 \lambda \boldsymbol{I})^{-1} \qquad \boldsymbol{\mu}_N = \boldsymbol{\Sigma}_N \tilde{\boldsymbol{X}} \boldsymbol{W} \boldsymbol{y}$$





Content of this Lecture

Constructing Basis Functions

- Radial Basis Function Networks

Non-Parametric Approaches

- Locally Weighted Regression
- Kernel Methods

Type 3: Kernel Methods

Kernel methods rely on the ‘kernel trick’

➡ It is sufficient to evaluate the scalar product between two samples in feature space, called kernel

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{\phi}(\boldsymbol{x}_1)^T \boldsymbol{\phi}(\boldsymbol{x}_2)$$

Why is this useful?

- Kernels are easier to design than features
- The feature space can be possibly infinite dimensional.
- We just need to be able to compute the scalar product

Type 3: Kernel Methods

Example: One RBF feature at every position c

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \boldsymbol{\phi}(\boldsymbol{x}_1)^T \boldsymbol{\phi}(\boldsymbol{x}_2) = \int \exp\left(-\frac{||\boldsymbol{x}_1 - \boldsymbol{c}||^2}{2l^2}\right) \exp\left(-\frac{||\boldsymbol{x}_2 - \boldsymbol{c}||^2}{2l^2}\right) d\boldsymbol{c}$$

$$= \exp\left(-\frac{||\boldsymbol{x}_1 - \boldsymbol{x}_2||^2}{4l^2}\right)$$

➡ Reduces to an RBF feature at each sample

General conditions for kernels

- symmetric: $k(\boldsymbol{x}_1, \boldsymbol{x}_2) = k(\boldsymbol{x}_2, \boldsymbol{x}_1)$
- positive definite…

Kernel Ridge Regression (Kernels for Freqentists)

Look at the predictions with the MAP/RR estimator (linear regression) again:

$$y(\boldsymbol{x}_*) = \phi(\boldsymbol{x}_*)^T \boldsymbol{\theta} = \phi(\boldsymbol{x}_*)^T (\boldsymbol{\Phi}^T \boldsymbol{\Phi} + \lambda \boldsymbol{I})^{-1} \boldsymbol{\Phi}^T \boldsymbol{y}$$

Even more general, the **Woodbury identity for matrix inversion** yields

$$(\boldsymbol{\Phi}^T \boldsymbol{\Phi} + \lambda \boldsymbol{I}_D)^{-1} \boldsymbol{\Phi}^T = \boldsymbol{\Phi}^T (\boldsymbol{\Phi} \boldsymbol{\Phi}^T + \lambda \boldsymbol{I}_N)^{-1}$$

This yields

$$y(\boldsymbol{x}_*) = \phi(\boldsymbol{x}_*)^T \underbrace{(\boldsymbol{\Phi}^T \boldsymbol{\Phi} + \lambda \boldsymbol{I}_D)^{-1} \boldsymbol{\Phi}^T}_{D \times D} \boldsymbol{y}$$

$$= \phi(\boldsymbol{x}_*)^T \boldsymbol{\Phi}^T \underbrace{(\boldsymbol{\Phi} \boldsymbol{\Phi}^T + \lambda \boldsymbol{I}_N)^{-1}}_{N \times N} \boldsymbol{y}$$

Equivalent solution to ridge regression

Why is this potentially useful?

Kernel Ridge Regression (Kernels for Freqentists)

Let's say, we have an inner product for our features:

$$k(x_i, x_j) = \phi(x_i)^T \phi(x_i)$$

We can use this to rewrite

$$y = \phi(x)^T \Phi (\Phi\Phi^T + \lambda^{-1} I)^{-1} Y$$

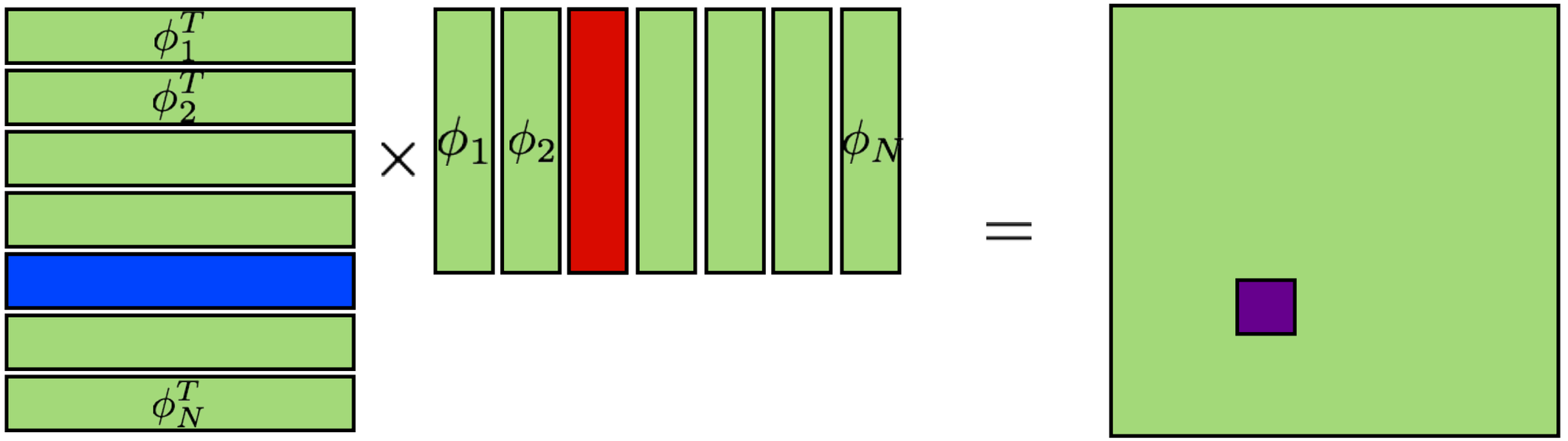
Into

$$y = k(x)(K + \lambda^{-1} I)^{-1} Y$$

This a "Kernelization" of regression! But why is this a good idea?

Kernels are scalar products in feature space!

$K_{ij} = \lambda^{-1}\boldsymbol{\phi}(\boldsymbol{x}_i)^T\boldsymbol{\phi}(\boldsymbol{x}_j) = k(\boldsymbol{x}_1, \boldsymbol{x}_2)$ scalar products in feature space

$\boldsymbol{K} =$ 

Kernels can measure the similarity between data points in feature space without evaluating or explicitly knowing all features!

Particularly useful when there are more features than data points as, e.g., in images…

Bayesian Linear Regression revisited

We have:

Data-Likelihood: $p(\boldsymbol{y}|\boldsymbol{\theta}, \boldsymbol{X}) = \mathcal{N}(\boldsymbol{y}|\boldsymbol{\Phi\theta}, \sigma^2\boldsymbol{I})$

Prior: $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta}|\mathbf{0}, \lambda^{-1}\boldsymbol{I})$

If we integrate out the weights, we get

$$p(\boldsymbol{y}|\boldsymbol{X}) = \int \mathcal{N}(\boldsymbol{y}|\boldsymbol{\Phi\theta}, \sigma^2\boldsymbol{I})\mathcal{N}(\boldsymbol{\theta}|\mathbf{0}, \lambda^{-1}\boldsymbol{I})d\boldsymbol{\theta}$$
$$= \mathcal{N}(\boldsymbol{y}|\mathbf{0}, \sigma^2\boldsymbol{I} + \lambda^{-1}\boldsymbol{\Phi\Phi}^T)$$

Defines a multivariate Gaussian distribution over the samples

⇒ Samples are correlated as the marginalized weight vector is the same for each sample

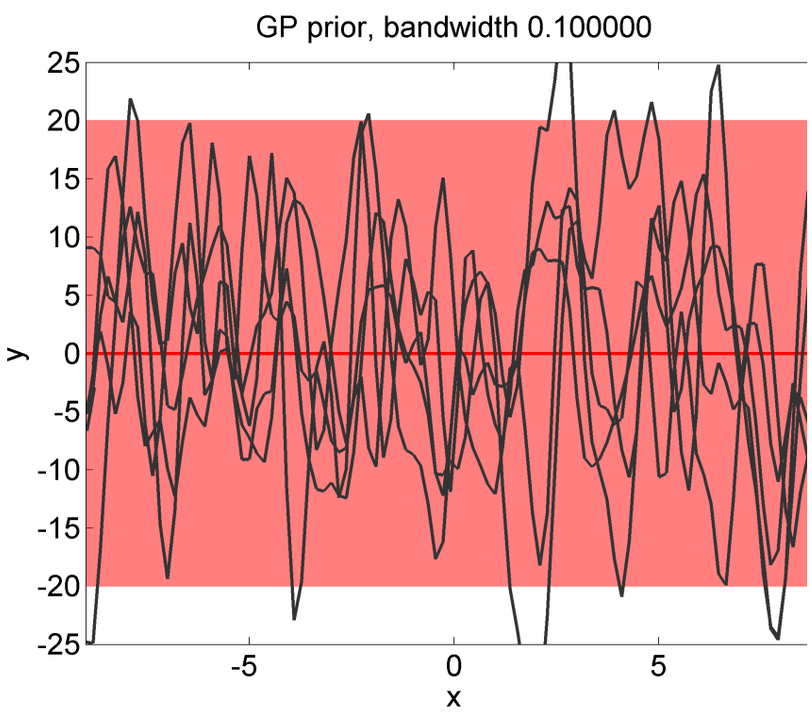
Bayesian Kernel Regression: Gaussian Processes (GPs)

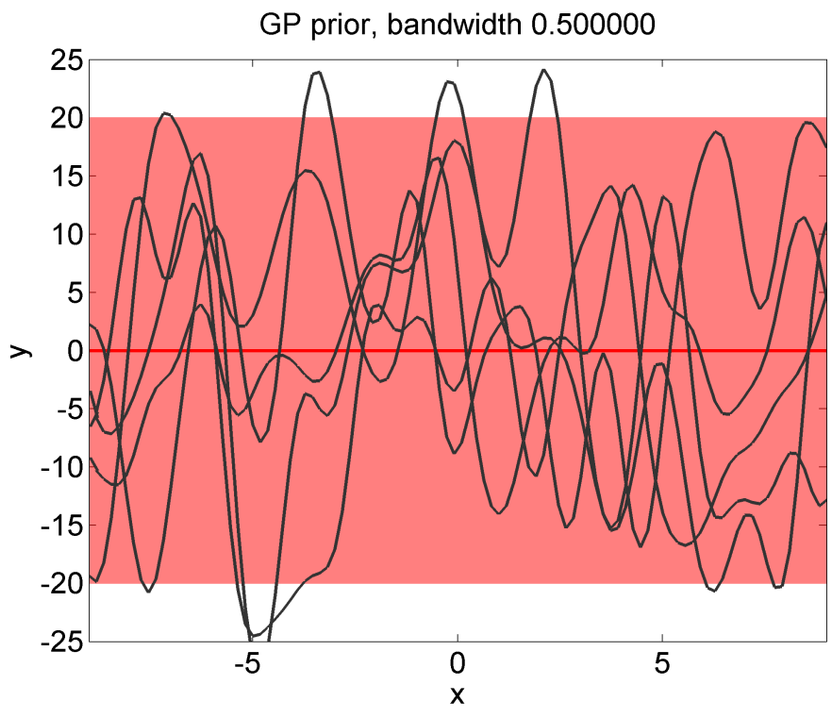
Replace the features in Bayesian Linear Regression by a Kernel and you obtain:

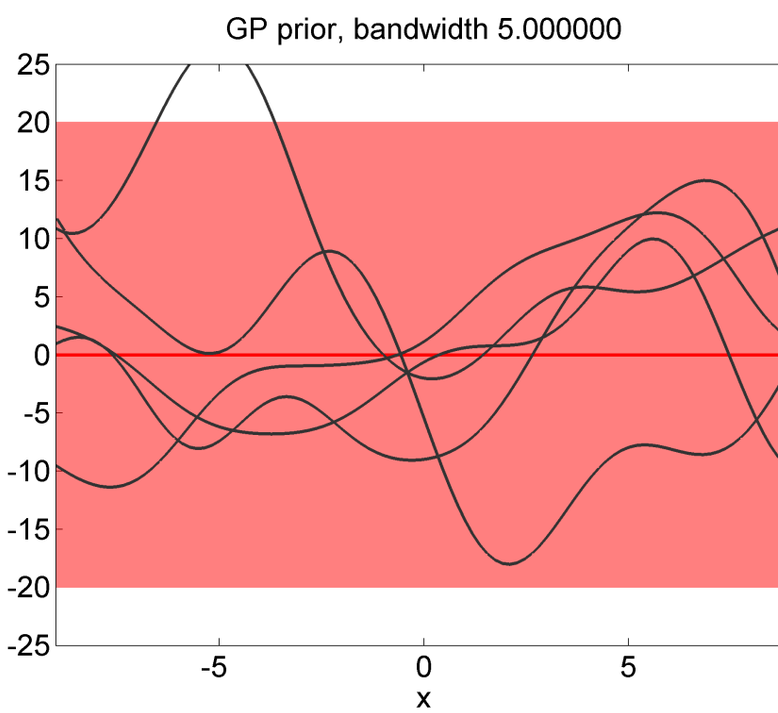
$$p(\boldsymbol{y}|\boldsymbol{X}) = \mathcal{N}(\boldsymbol{y}|\mathbf{0}, \sigma^2\boldsymbol{I} + \boldsymbol{K}) \text{ with } \boldsymbol{K} = \lambda^{-1}\boldsymbol{\Phi}\boldsymbol{\Phi}^T$$

This method is called a Gaussian Process $\mathcal{GP}(\mathbf{0}, \boldsymbol{K})$ with covariance function *k*

Sampling from the GP-Prior







The kernel bandwidth of the exponential kernel is a prior on the smoothness on the function

$$k(\boldsymbol{x}_1, \boldsymbol{x}_2) = \exp\left(-\frac{||\boldsymbol{x}_1 - \boldsymbol{x}_2||^2}{4l^2}\right)$$

GP-Posterior

Now we observe a data set given by $\boldsymbol{y}$ and $\boldsymbol{X}$ and we want to predict y^* for $\boldsymbol{x}^*$

- We can write down the GP prior for the concatenated data

$$p\left(\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix} \middle| \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{x}_* \end{bmatrix}\right) = \mathcal{N}\left(\begin{bmatrix} \boldsymbol{y} \\ y_* \end{bmatrix} \middle| \boldsymbol{0}, \begin{bmatrix} \boldsymbol{K} & \boldsymbol{k}(\boldsymbol{X}, \boldsymbol{x}_*) \\ \boldsymbol{k}(\boldsymbol{x}_*, \boldsymbol{X}) & k(\boldsymbol{x}_*, \boldsymbol{x}_*) \end{bmatrix} + \sigma^2 \boldsymbol{I}\right)$$

- We get the GP-posterior by Gaussian conditioning (see refresher)

$$p(y_*|\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{x}_*) = \mathcal{N}(\boldsymbol{y}_*|\mu(\boldsymbol{x}_*), \sigma^2(\boldsymbol{x}_*))$$

GP-Posterior

$$p(y_*|\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{x}_*) = \mathcal{N}(\boldsymbol{y}_*|\mu(\boldsymbol{x}_*), \sigma^2(\boldsymbol{x}_*))$$

Same solution as in Kernel Ridge Regression

Predictive mean

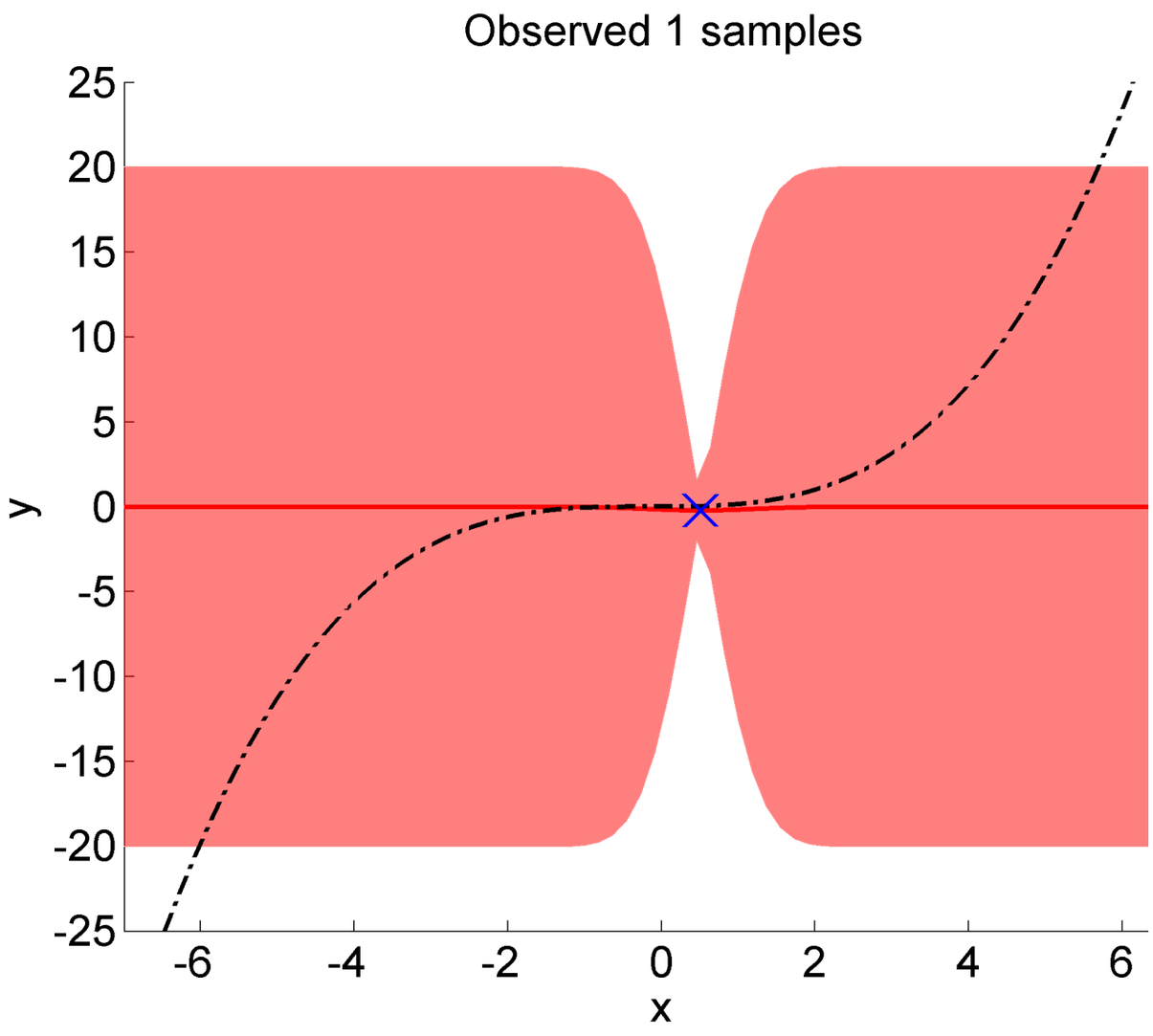
$$\mu(\boldsymbol{x}_*) = k(\boldsymbol{x}_*, \boldsymbol{X})(\boldsymbol{K} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{y}$$

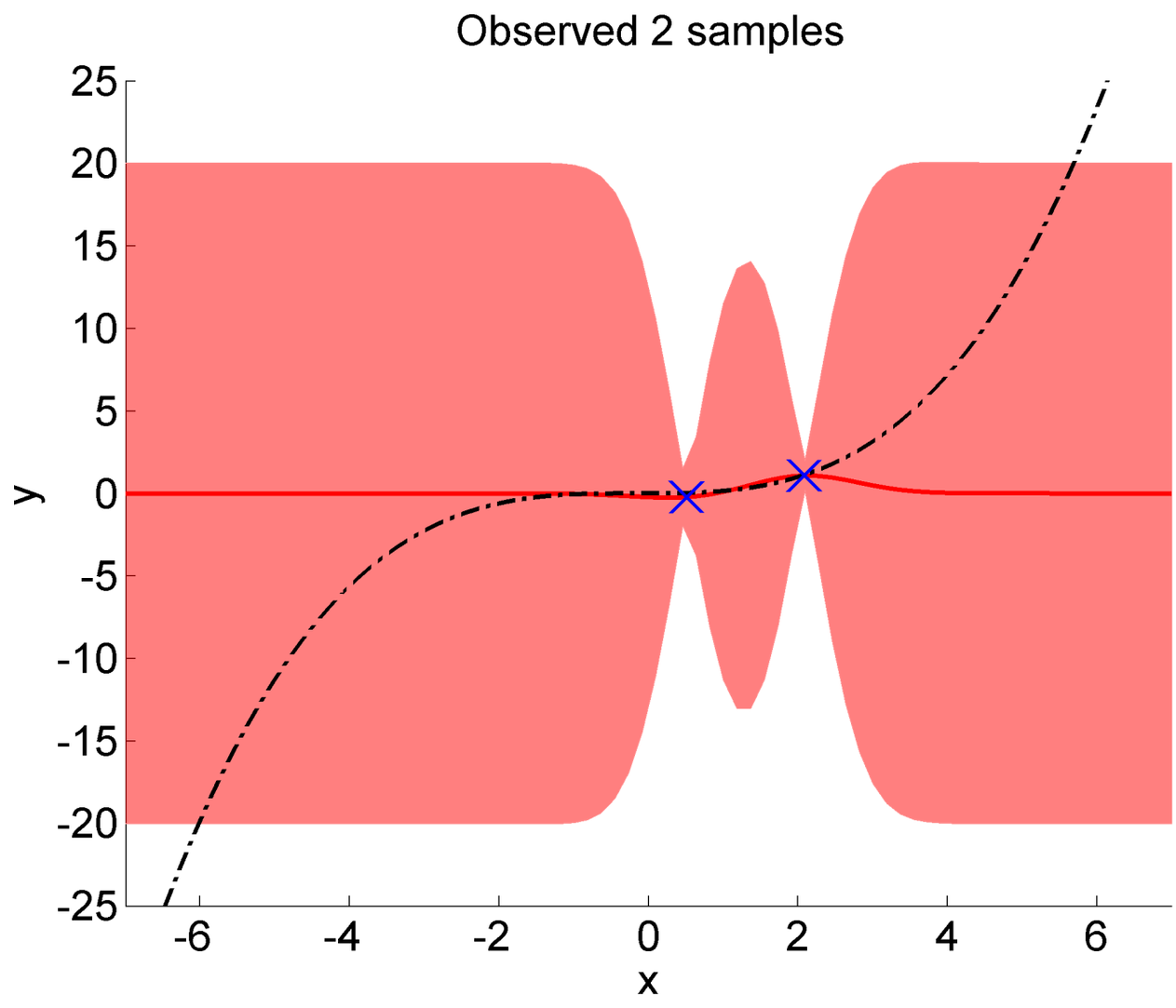
Predictive variance

$$\sigma^2(\boldsymbol{x}_*) = k(\boldsymbol{x}_*, \boldsymbol{x}_*) + \sigma^2 - \boldsymbol{k}(\boldsymbol{x}_*, \boldsymbol{X})(\boldsymbol{K} + \sigma^2\boldsymbol{I})^{-1}\boldsymbol{k}(\boldsymbol{X}, \boldsymbol{x}_*)$$

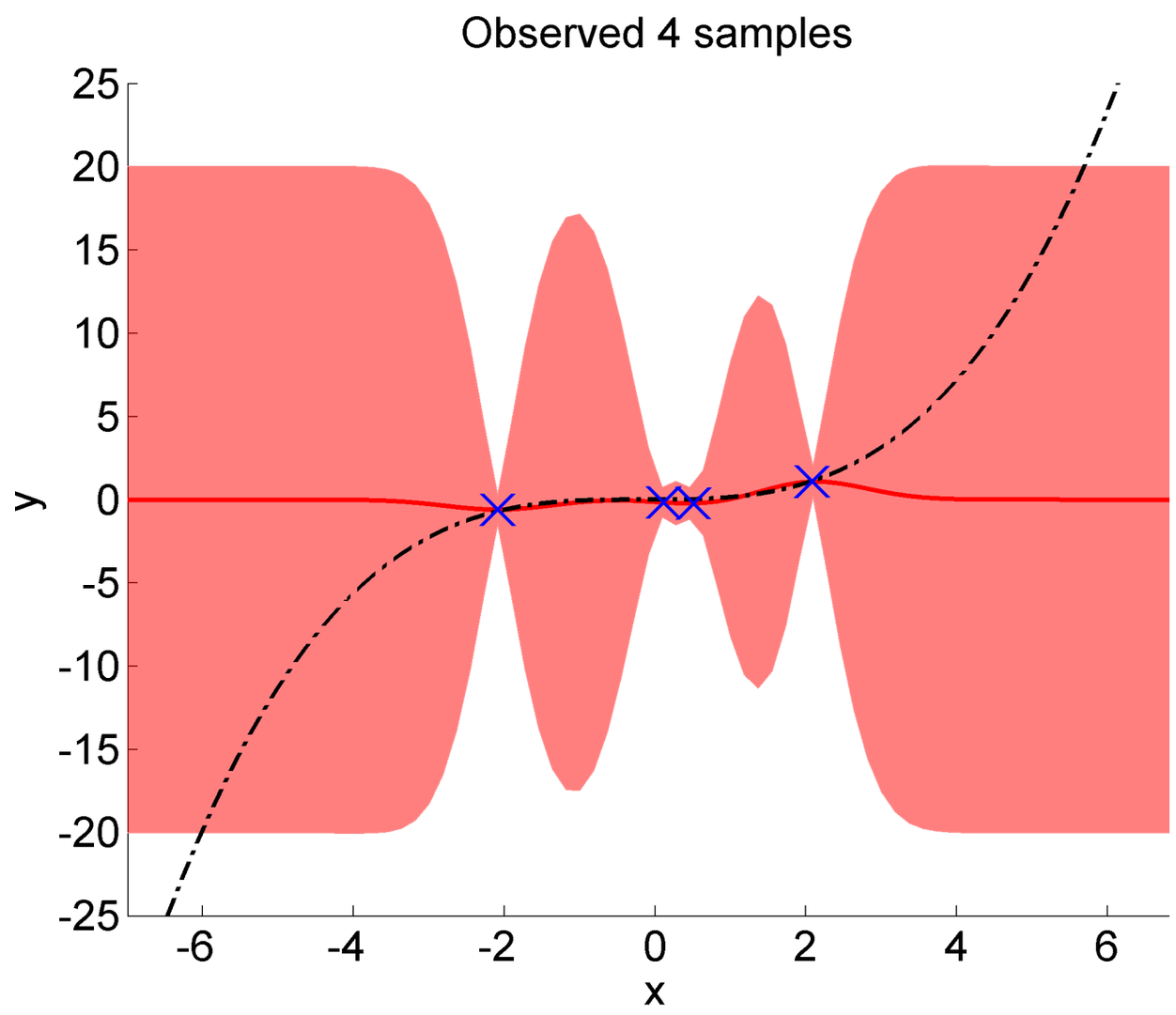
GPs = Kernel Ridge Regression + Knowledge on your Uncertainty!

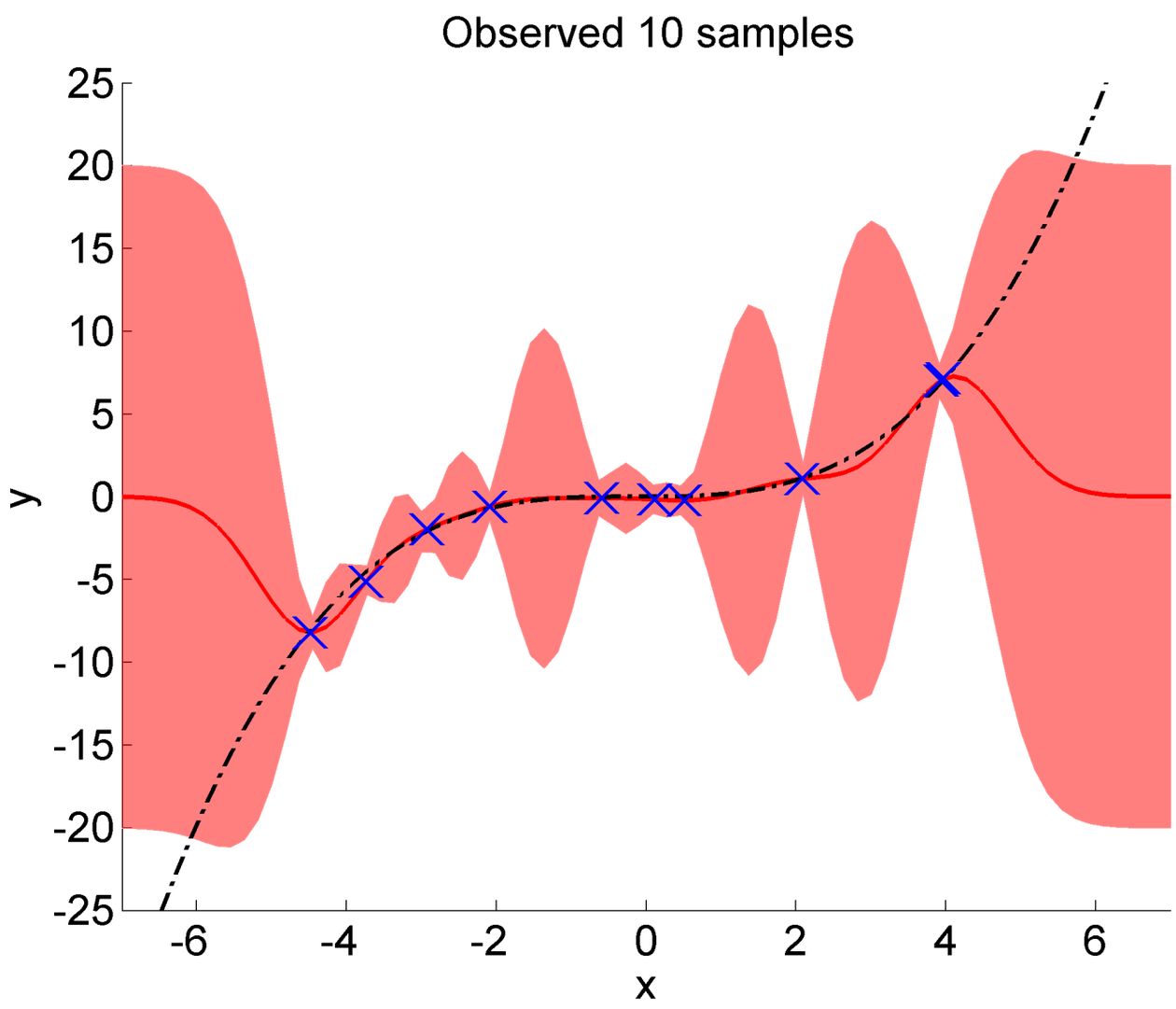
GP-Posterior





GP-Posterior





Gaussian Processes

A key advantage over
Kernel Ridge Regression

Optimization of Hyper-Parameters (All learning becomes optimization)

- The parameters of the kernel are called hyper-parameters
- Cross validation or maximization of marginal log-likelihood

GPs vs. Bayesian Linear Regression:

- GP is Kernel Ridge Regression with uncertainty! Same mean!
- GPs is kernelized Bayesian Linear Regression
- Kernels are often easier to use than features!

GPs are currently the gold standard for regression!

… if you do not have too many data points!

Summary

- You should have a really good overview of machine learning by now.
- You should remember the following **regression methods**
  - Least-Squares Regression / Ridge Regression
  - Bayesian Regression
  - Radial-Basis Function Regression
  - Locally-Weighted Linear Regression
  - Kernel Ridge Regression and GPs
- You should know how to **choose the right method** for a regression problem