Background: Supervise Learning in 25 Minutes

Pieter Abbeel
UC Berkeley

Jan Peters
TU Darmstadt

TECHNISCHE UNIVERSITÄT DARMSTADT

Big Picture

All of the steps involve function approximation

Reinforcement Learning Data

$$\mathscr{D} = \{(\mathbf{x}_i, \mathbf{u}_i, \mathbf{x}_{i+1}, r_i)\}\}$$

$$\mathbf{x}' \sim \mathscr{P}^{\mathbf{u}}_{\mathbf{x}\mathbf{x}'}$$
$$r \approx \mathscr{R}^{\mathbf{u}}_{\mathbf{x}\mathbf{x}'}$$

$$V^*(\mathbf{x})$$

$$\pi^*(\mathbf{u}|\mathbf{x})$$

Policy Search

$$V^*(\mathbf{x})$$

$$\pi^*(\mathbf{u}|\mathbf{x})$$

Value Function Methods

$$\pi^*(\mathbf{u}|\mathbf{x})$$

Optimal Control with Model Learning

Thursday, May 17, 2012

Purpose of this Lecture

- Supervised Learning approaches have been tremendously successful in a huge number of applications.
- A huge and highly dynamical area of research!
- We can only take a glimpse on one supervised learning problem: Regression!
- *Regression*: Approximate continuous functions from (noised) measured data.

!

Content of this Lecture

1. Introduction to Regression
2. Accuracy, Overfitting and Regularization
3. How to Avoid Handcrafting Features
4. Learning Inverse and Forward Dynamics Models

Motivational Example: Robot Arm

- You want to predict the torques of a robot arm:

$$y = I\ddot{q} + mlg\sin q - \mu\dot{q} = \left[\ \ddot{q}, \quad \sin q, \quad \dot{q}\ \right]\left[\ I, \quad mlg, \quad -\mu\ \right]^T$$

$$= \underbrace{\phi(\mathbf{x})^T}_{\text{Features}}\underbrace{\theta}_{\text{Parameters}}$$

- Can we do this with a data set $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i) | i = 1 \ldots n\}$?

- This is a linear regression problem!

Cost Function I: Least Squared Error

The classical cost function is the one of least-squares

$$J = \frac{1}{2}\sum_{i=1}^{N}(y_i - \mathbf{f}_\theta(\mathbf{x}_i))^2$$

Using

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

we can rewrite it as

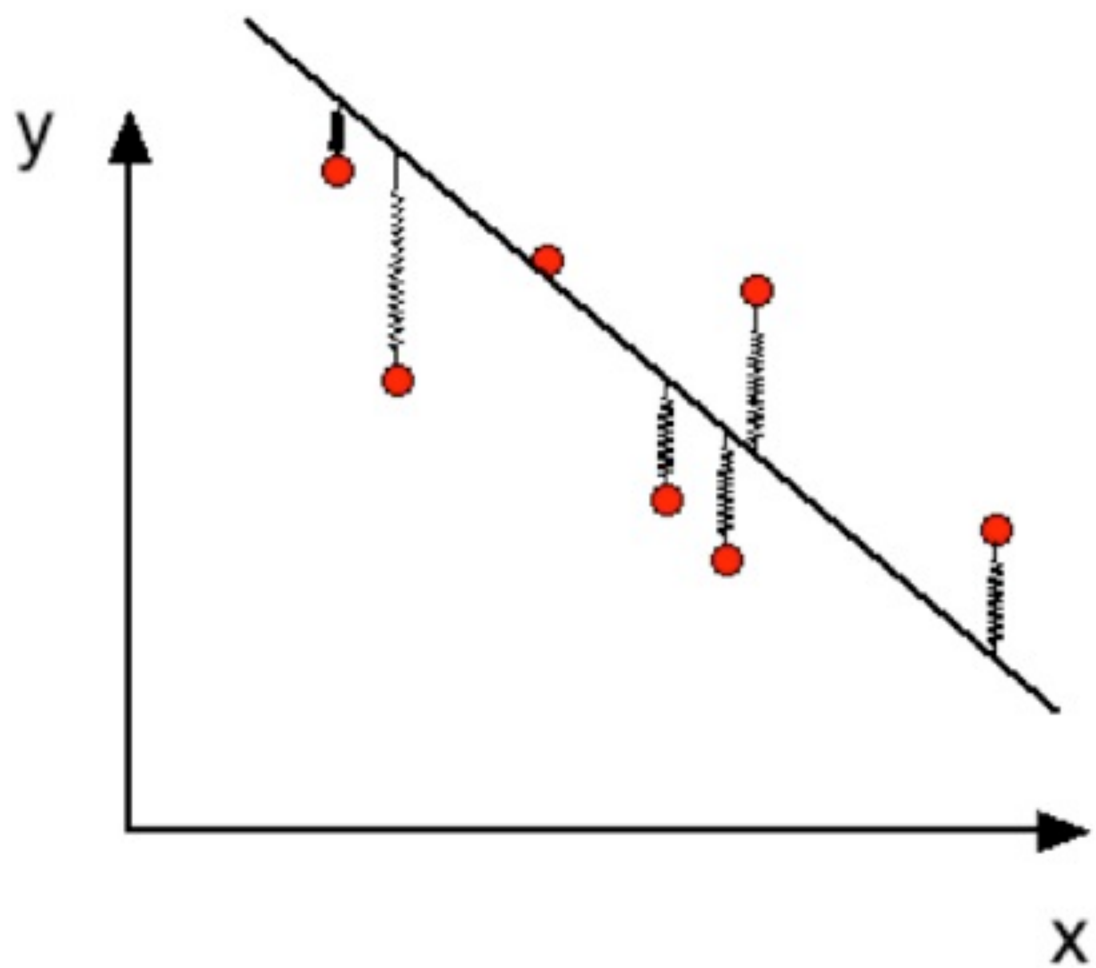
$$J = \frac{1}{2}(Y - \Phi\theta)^T(Y - \Phi\theta).$$

and solve it

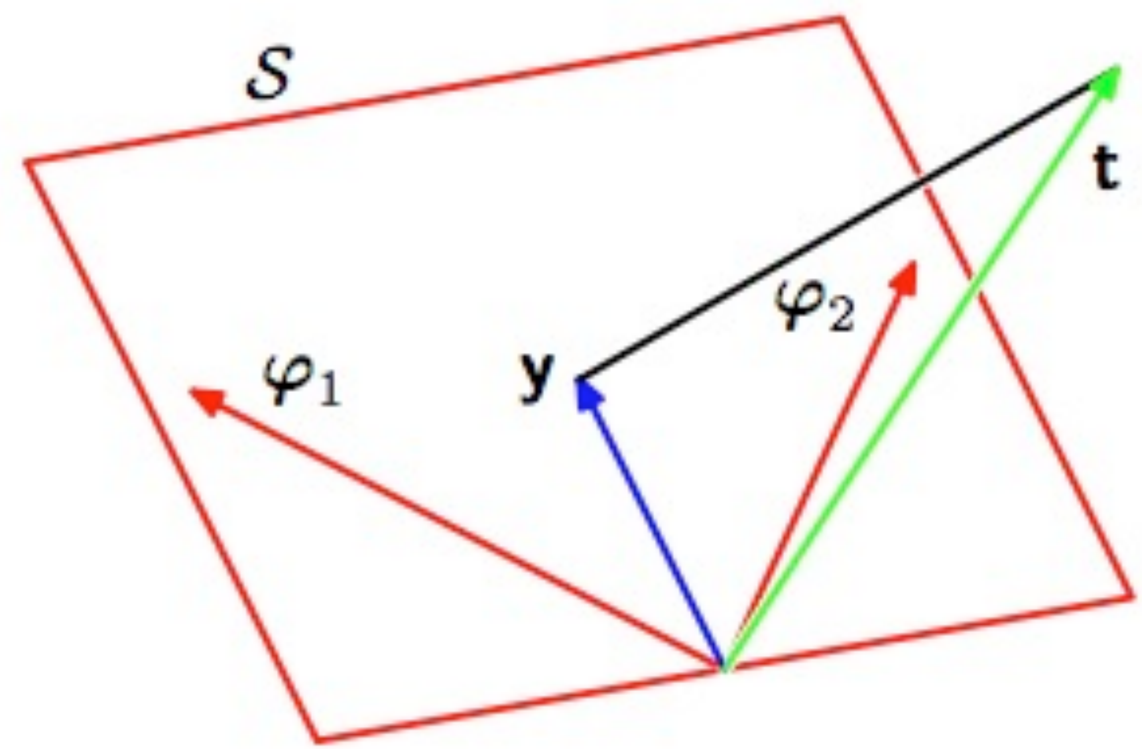
$$\theta = (\Phi^T\Phi)^{-1}\Phi^T Y$$

Classical Interpretations

Physical Interpretation



Geometric Interpretation



Maximum-Likelihood Interpretation

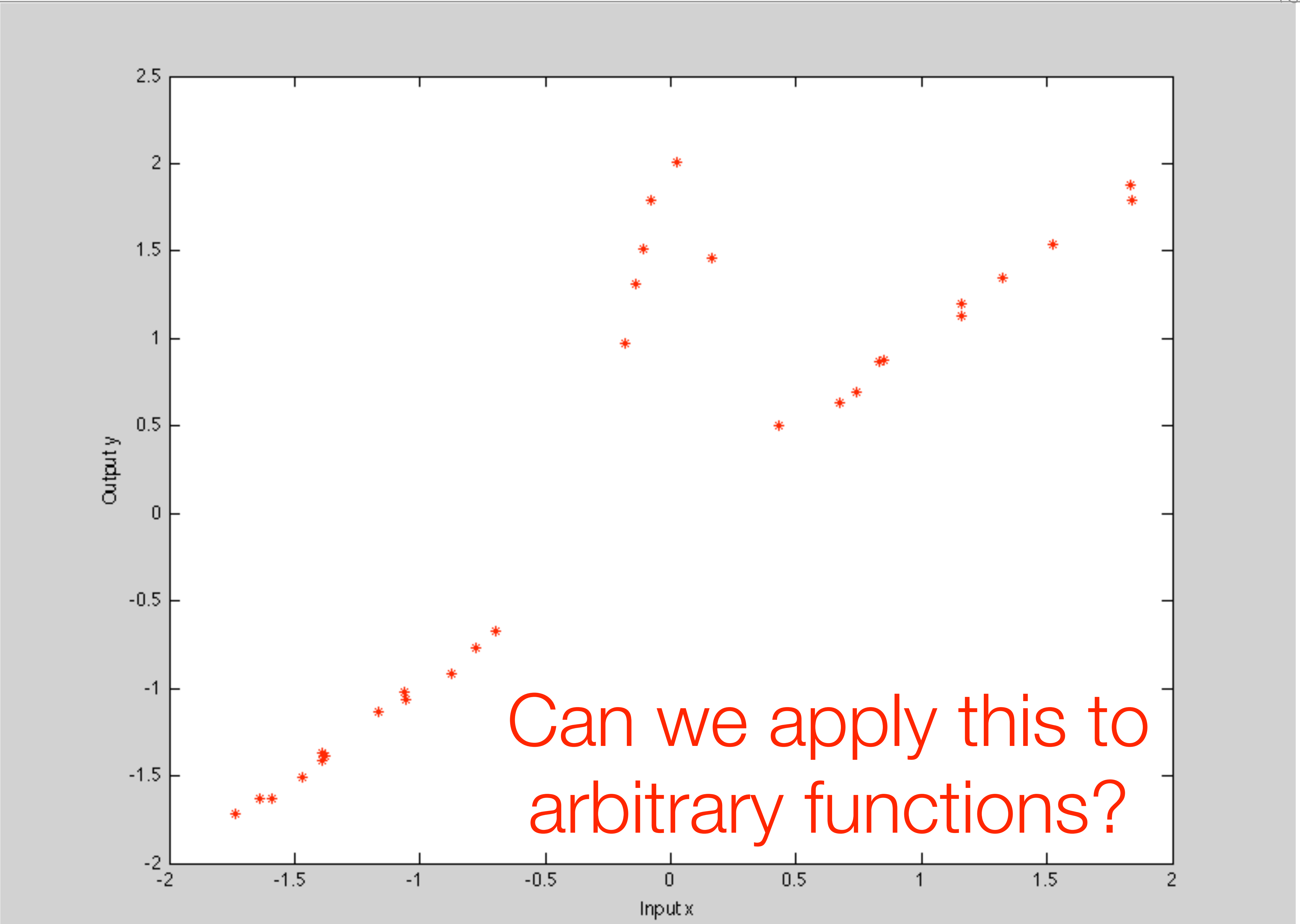
We could maximize the “likelihood” of the data:

$$\text{argmax}_\theta\, p(\mathscr{D}|\theta) = \text{argmax}_\theta \prod_{i=1}^{N} p(y_i|\mathbf{x}_i, \theta) p(\mathbf{x}_i).$$

This yields:

$$\begin{aligned} \text{argmax}_\theta \log p(\mathscr{D}|\theta) &= \text{argmax}_\theta \sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i, \theta) + \log p(\mathbf{x}_i), \\ &= \text{argmax}_\theta \frac{1}{2\sigma^2} \sum_{i=1}^{N} (y_i - \mathbf{f}_\theta(\mathbf{x}_i))^2. \end{aligned}$$

Example Problem: a Data Set



Content of this Lecture

1. Introduction to Regression
2. Accuracy, Overfitting and Regularization
3. How to Avoid Handcrafting Features
4. Learning Inverse and Forward Dynamics Models

Model Assumptions: Noise

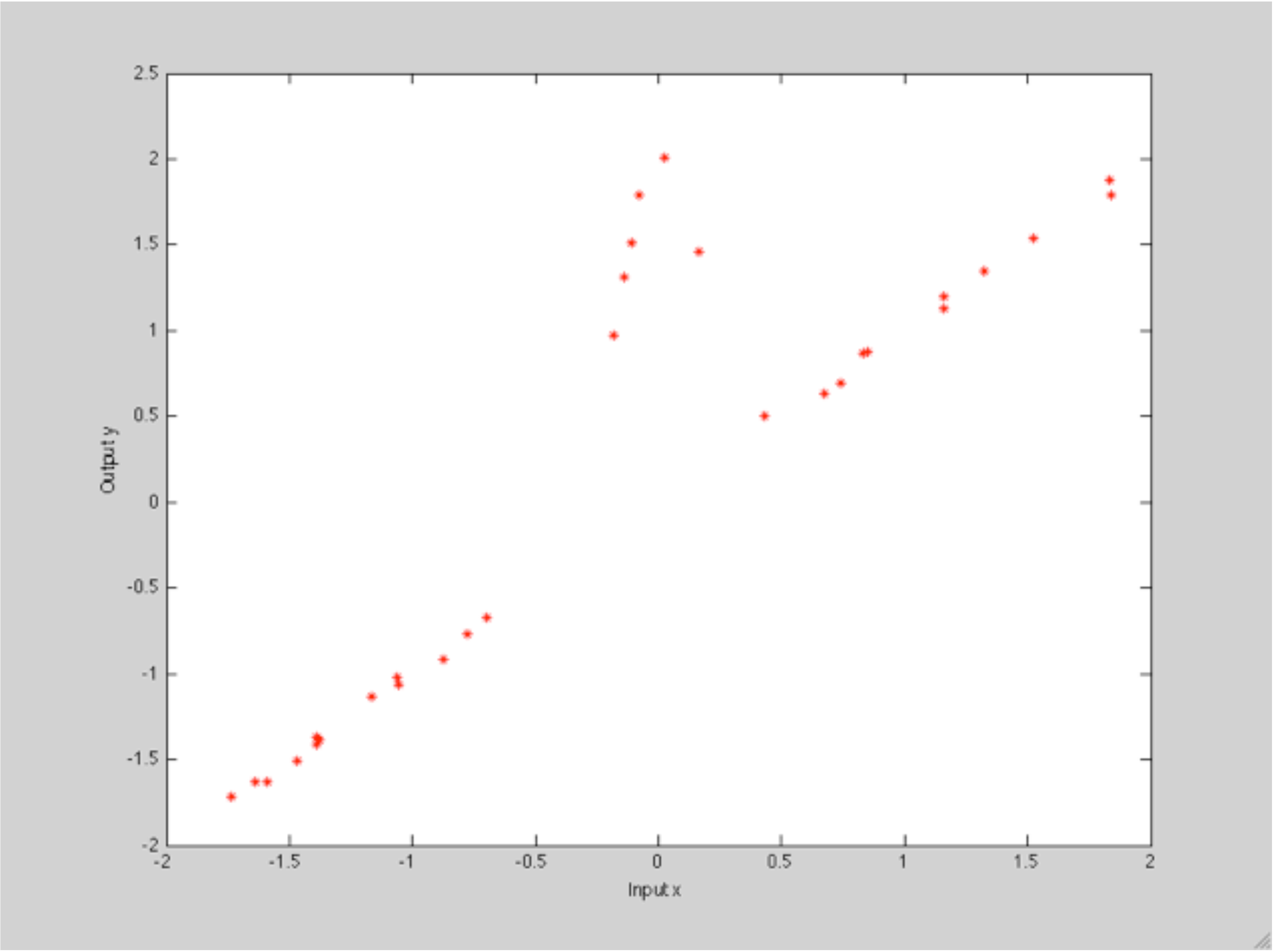
Additive Noise:

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

Linear in Features:

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

Let us fit our data ...

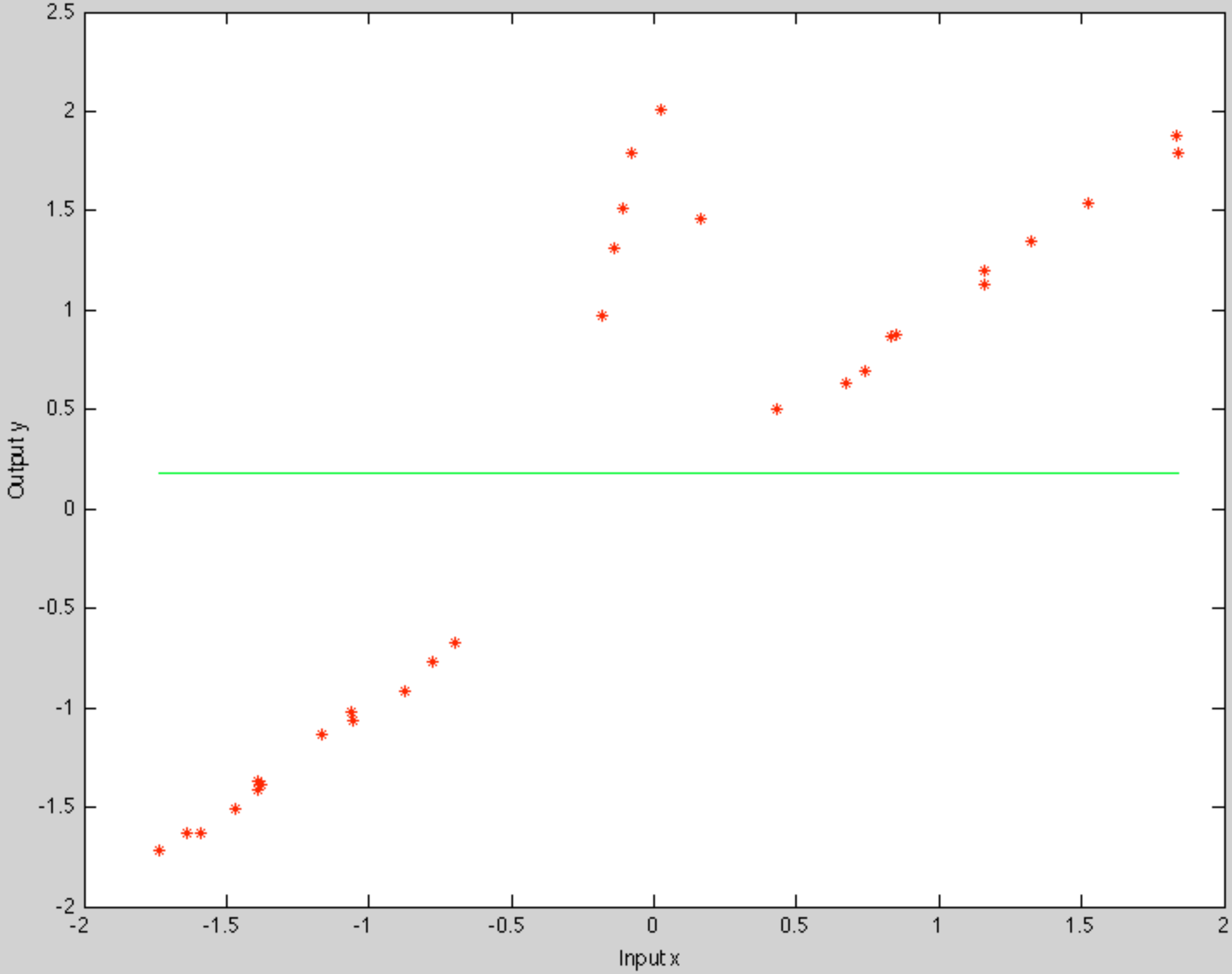


We need to answer:

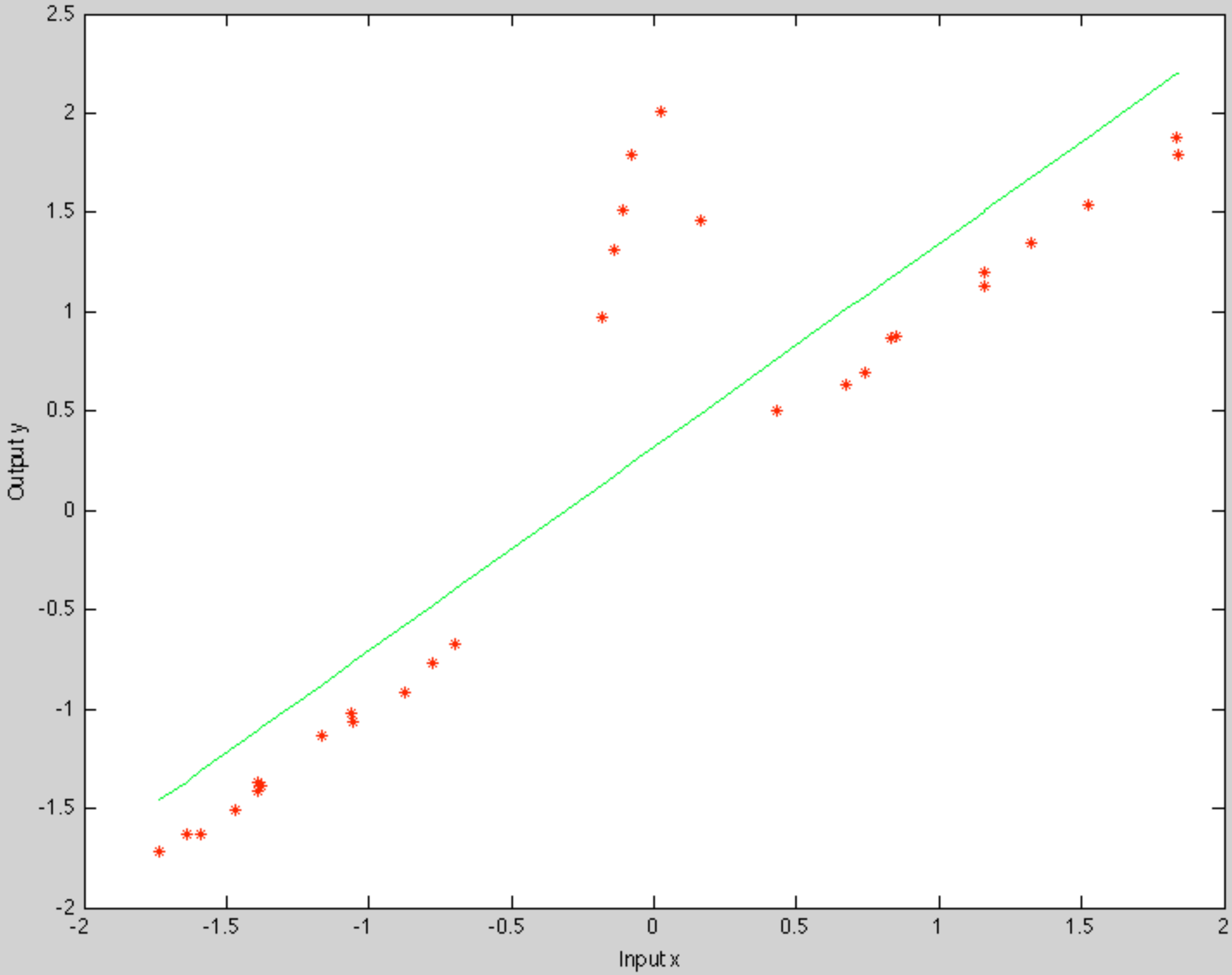
- Number of parameters?
- Is your model too rich?
- Does it allow overfitting?

We assume a model class: $y = \phi(x)^T\theta + \epsilon = [1, x, x^2, x^3, \ldots, x^n]^T\theta + \epsilon$

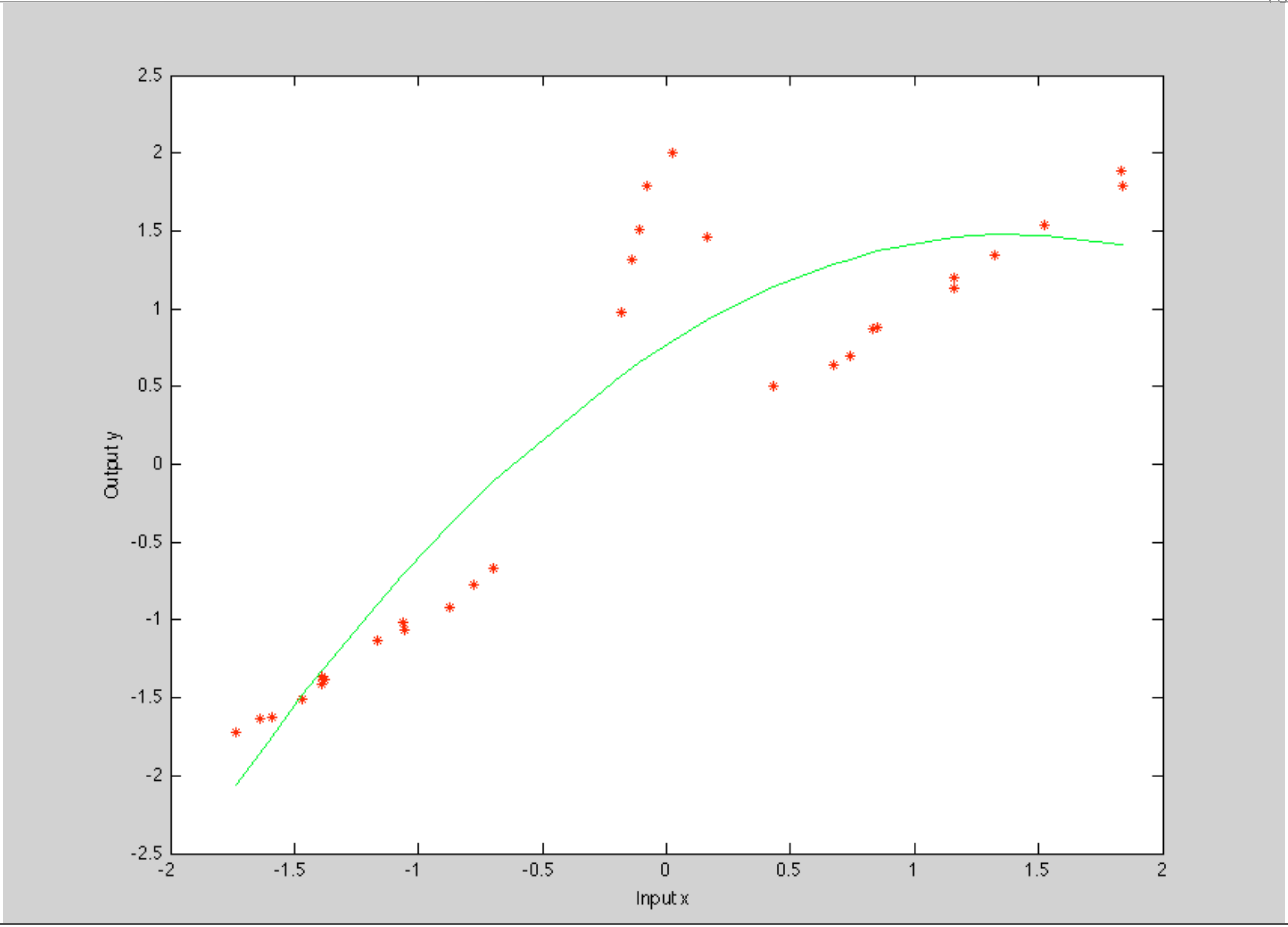
Fitting an Easy Model: n=0



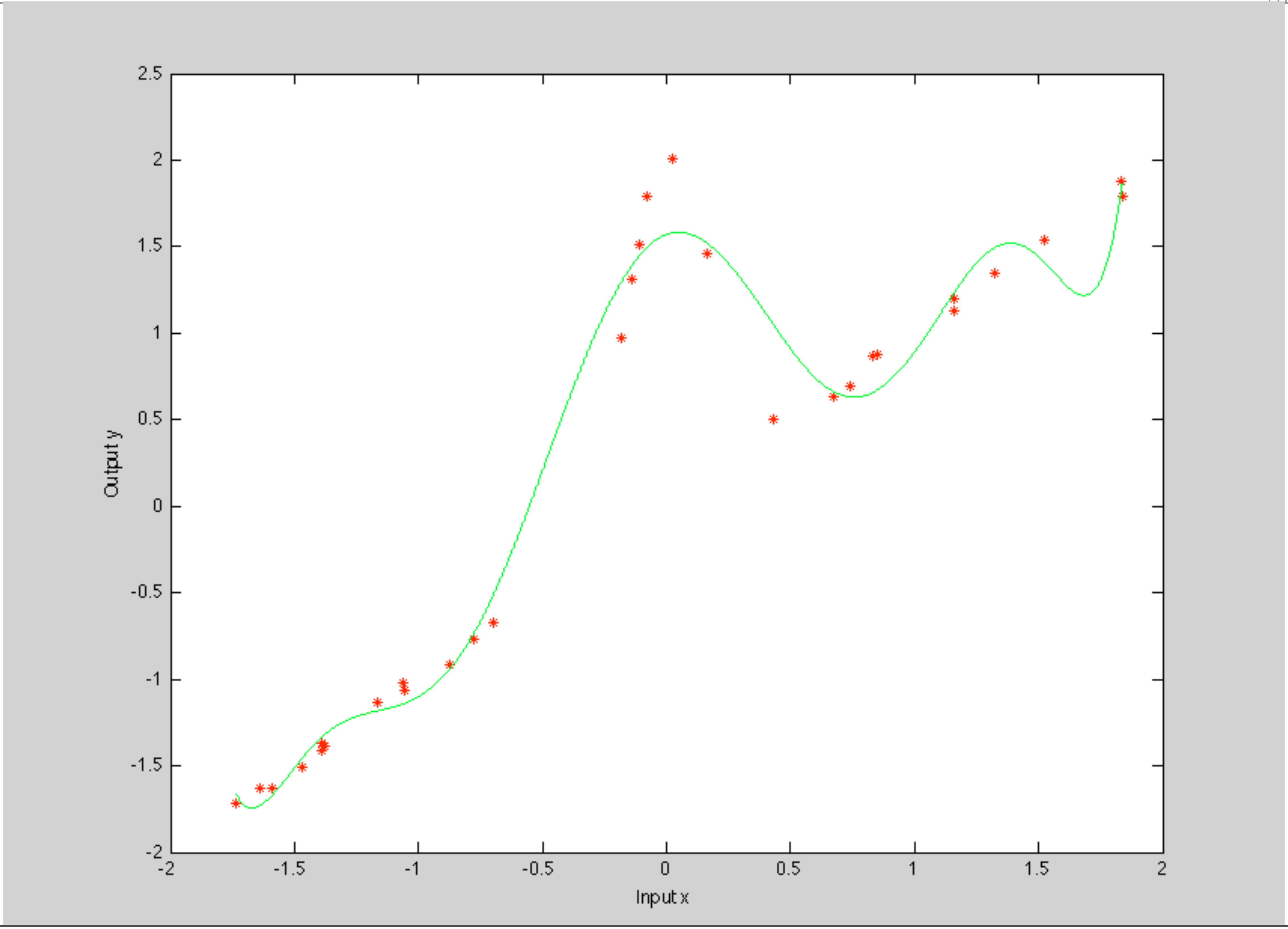
Add a Feature: n=1



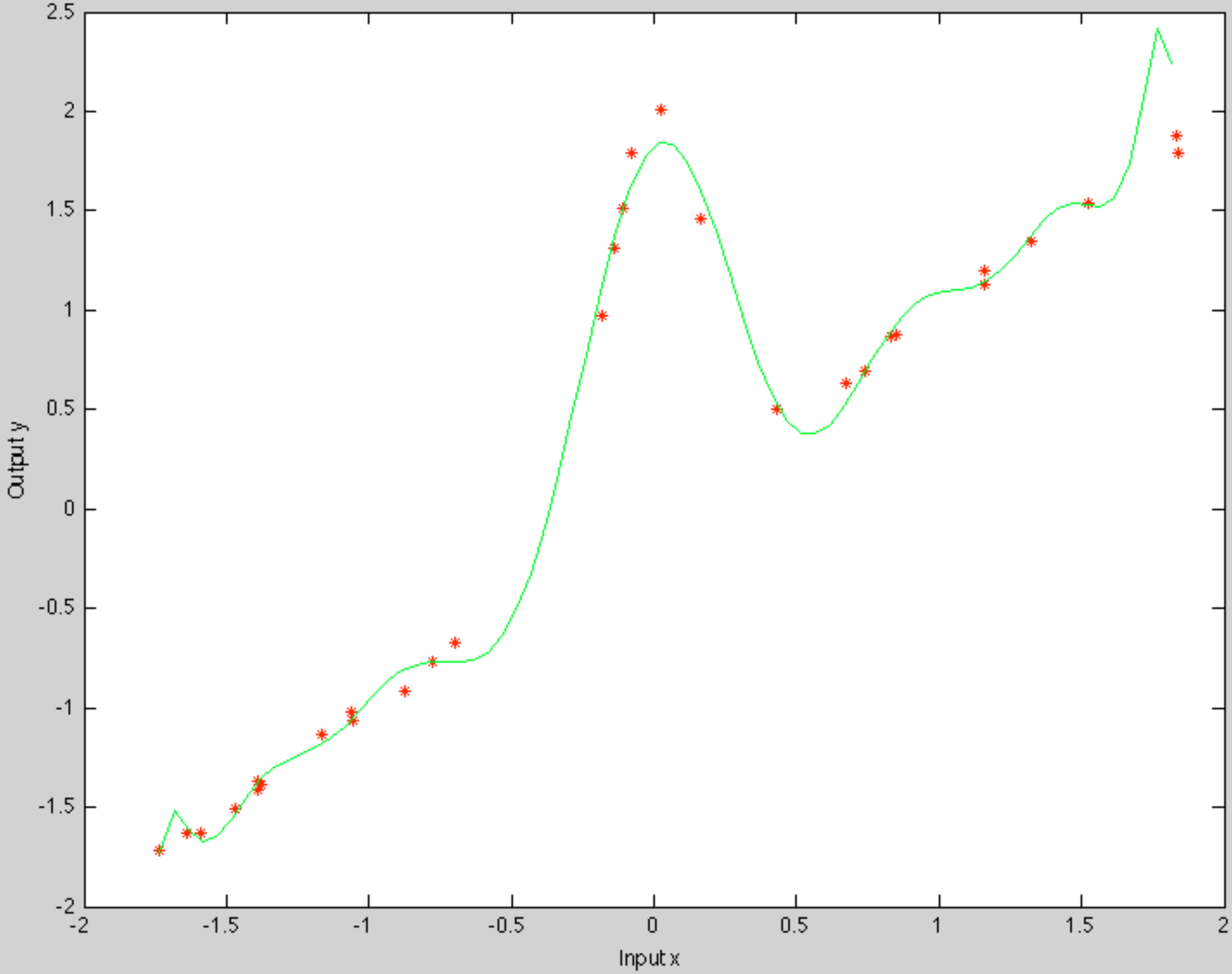
More features... n=2



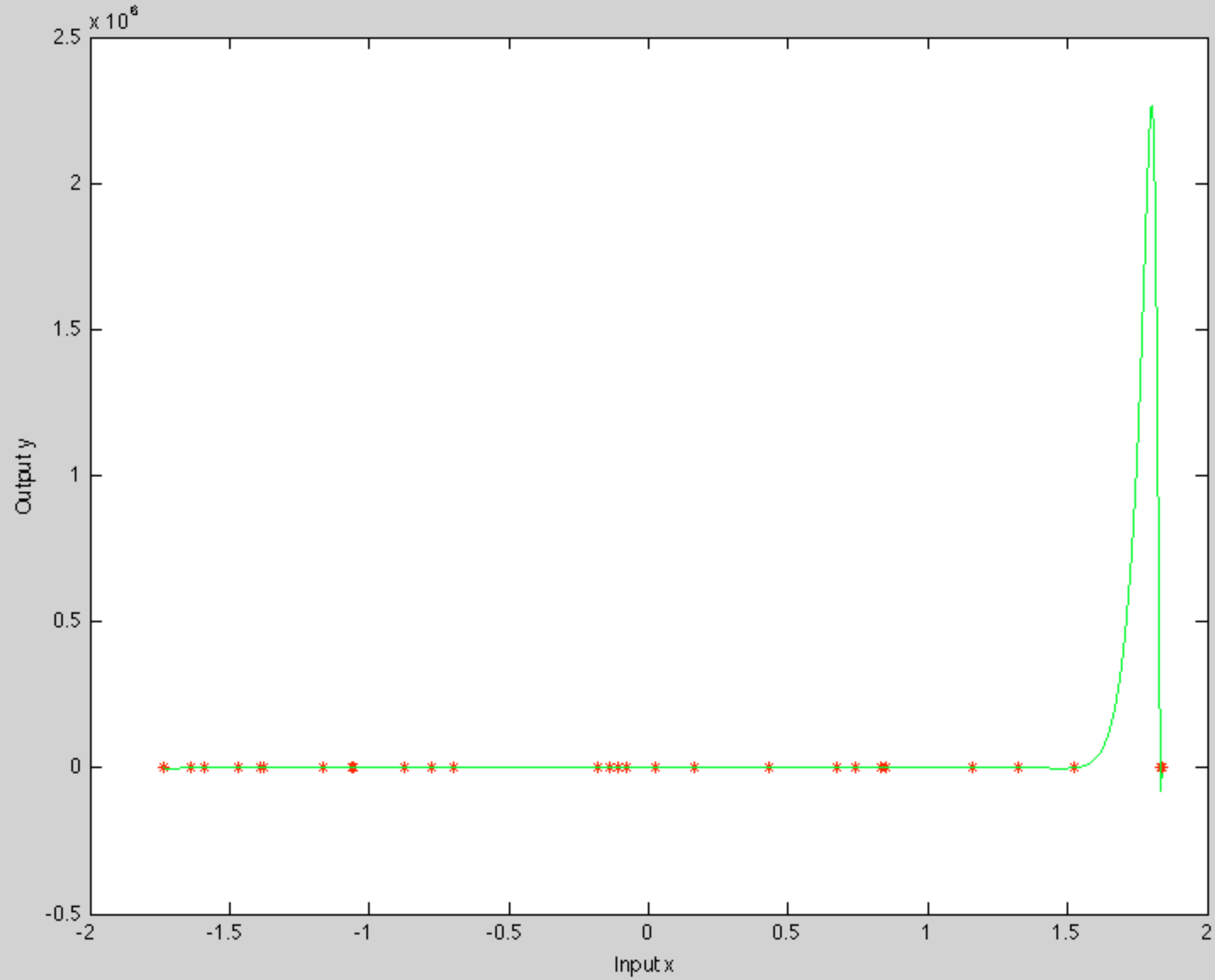
More features... n=8



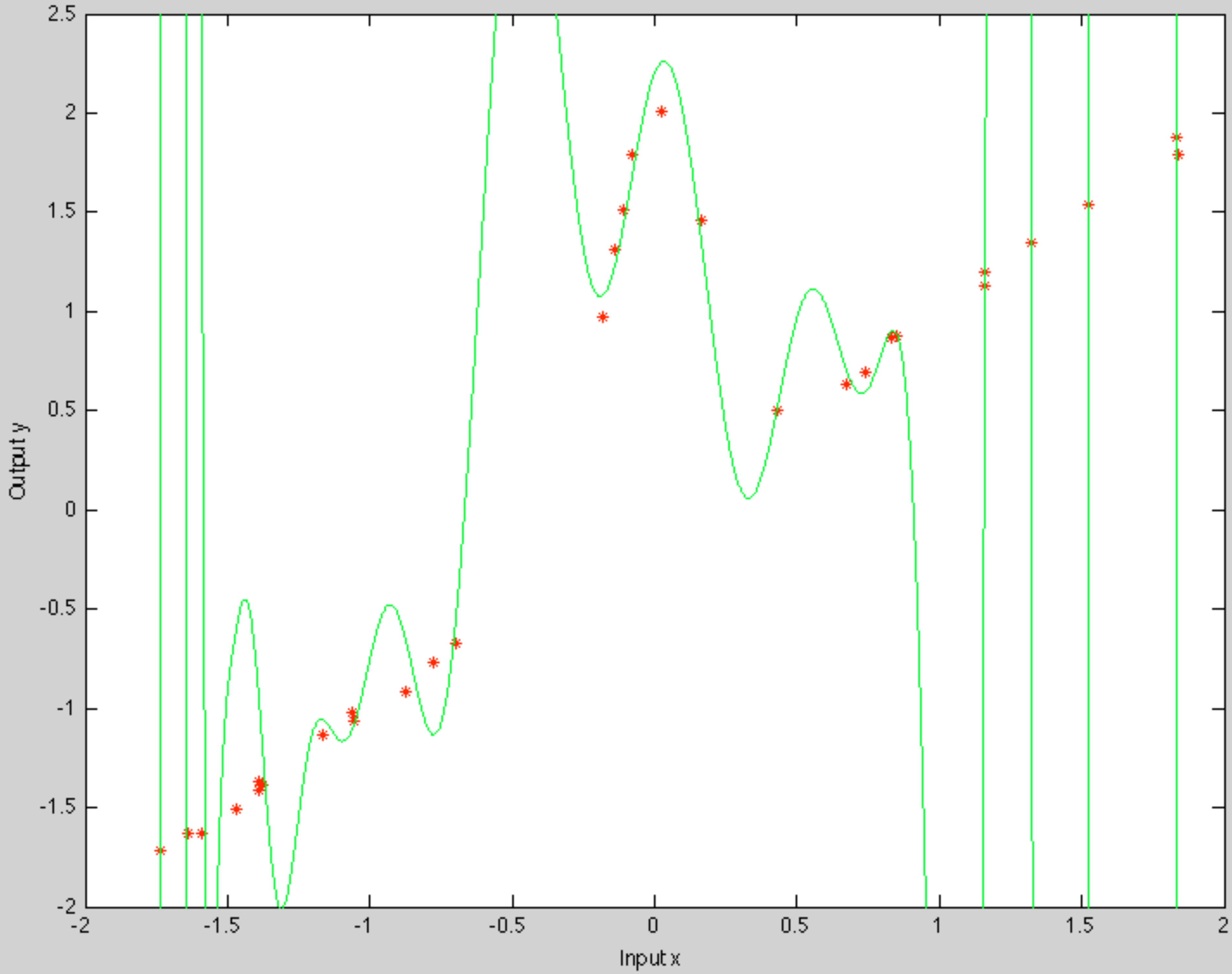
More features... n=15



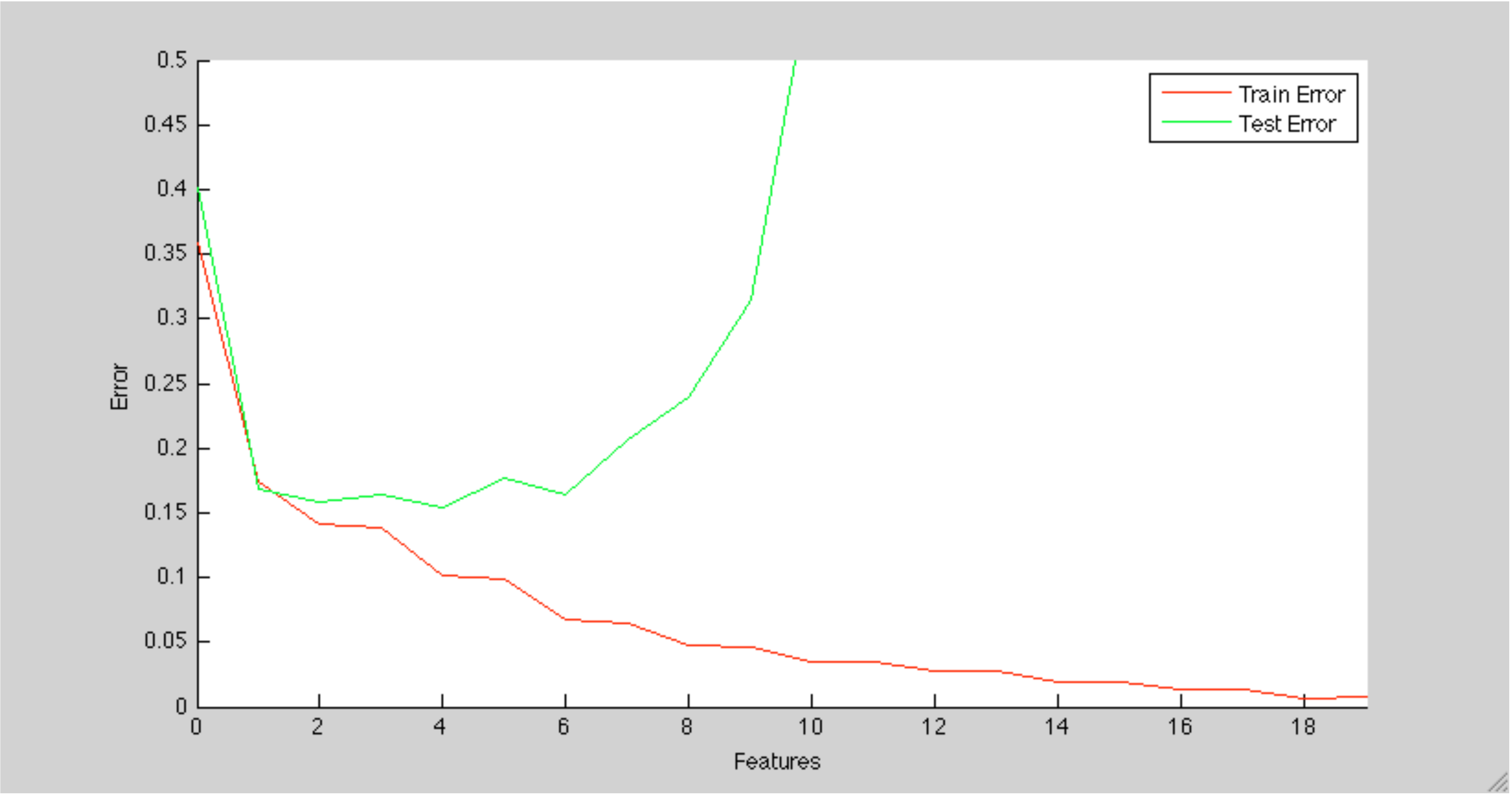
More features: n=200



More features: n=200

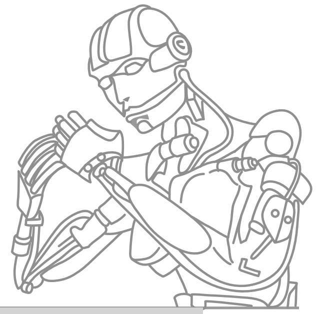


Test Error vs Training Error



“Magic” Tool: Leave-one-out-cross-validation (LOOCV)

What was the problem with n=200?



Prominent example of overfitting...

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

Cost Function II: Maximum-A-Priori = Ridge Regression

We could punish the size of the parameters (Complexity Control):

$$J = \frac{1}{2}(Y - \Phi\theta)^T(Y - \Phi\theta) + \theta^T \mathbf{W}\theta$$

This yields Ridge Regression

$$\theta = (\Phi^T\Phi + \mathbf{W})^{-1}\Phi^T Y$$

with

$$\mathbf{W} = \lambda \mathbf{I} \qquad \lambda < 10^{-6}$$

The probabilistic interpretation is called Maximum-A-Priori:

$$\text{argmax}_\theta\, p(\mathcal{D}|\theta)p(\theta) \qquad p(\theta) = \mathcal{N}(0, \mathbf{W})$$

“Full” Bayesian Regression

- Full Bayesian Regression wants to

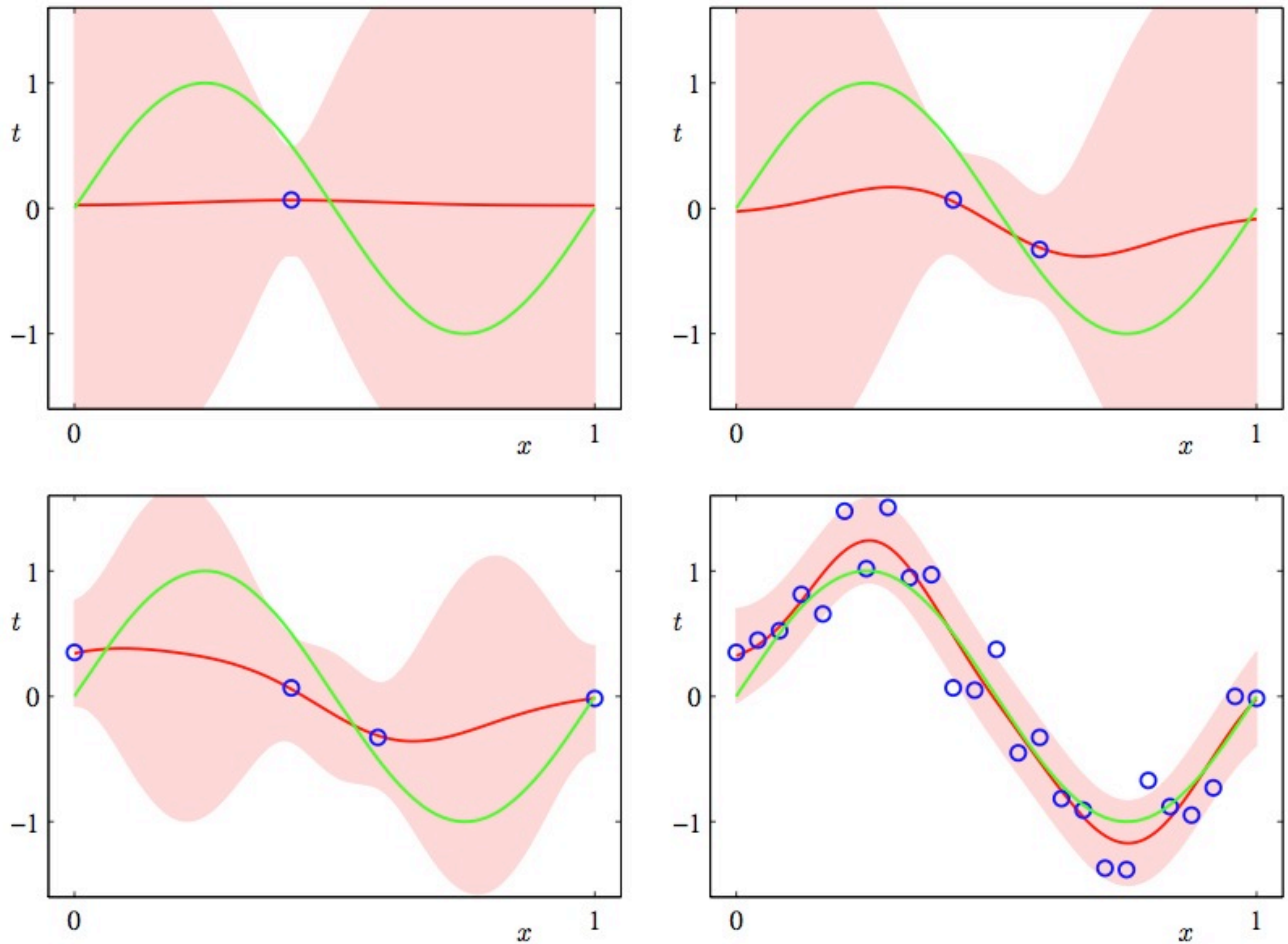
$$p(y|\mathscr{D},\mathbf{x}) = \int p(y|\mathbf{x},\theta)p(\theta|\mathscr{D})d\theta$$

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

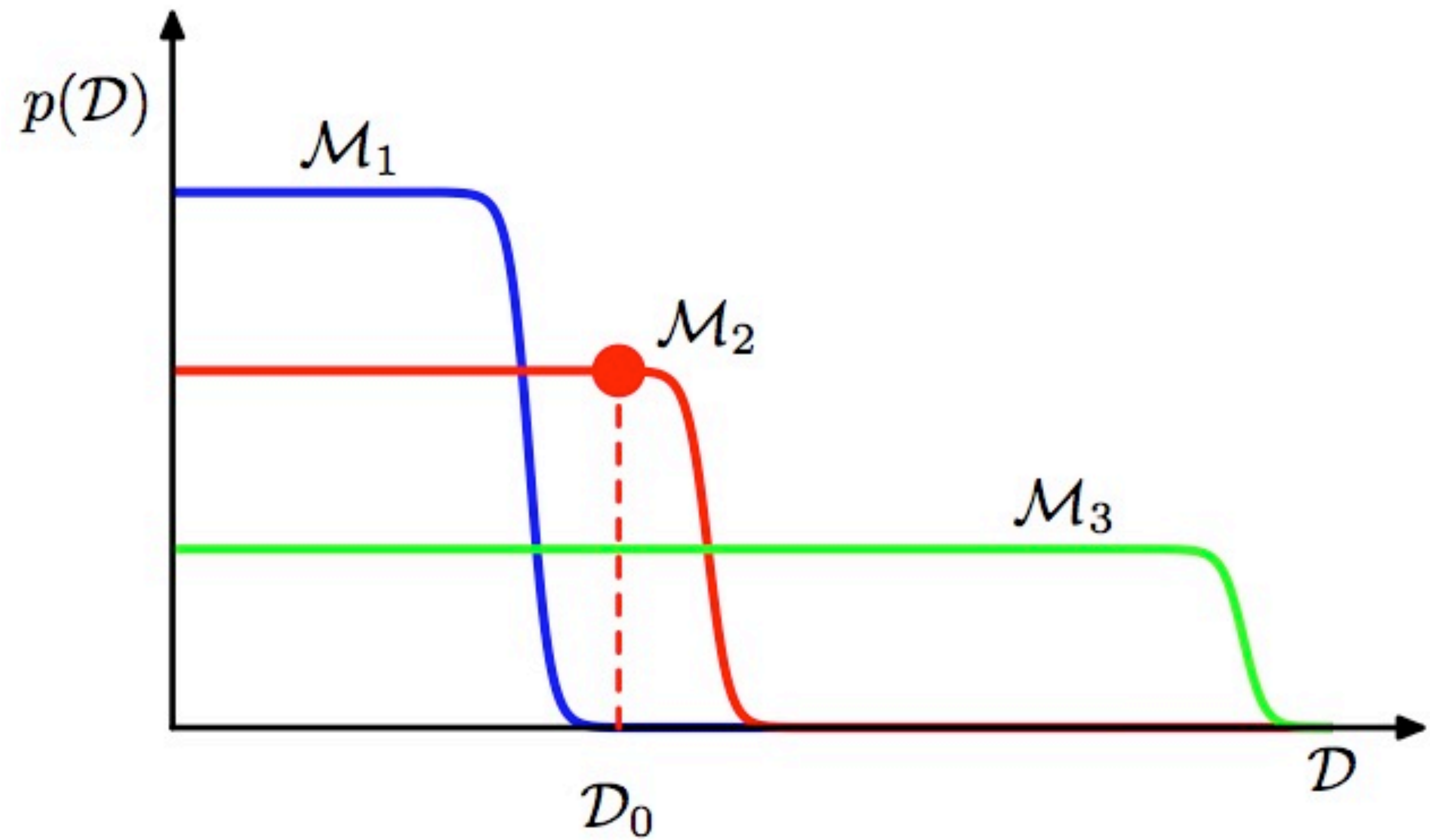
- Yields:

$$p(y|\mathcal{D},\mathbf{x}) = \mathcal{N}\left(\phi(\mathbf{x})^T\left(\frac{\lambda}{\beta}\mathbf{I}+\Phi^T\Phi\right)^{-1}\Phi^T\mathbf{Y}, \frac{1}{\beta}\left(1+\phi(\mathbf{x})^T\left(\frac{\lambda}{\beta}\mathbf{I}+\Phi^T\Phi\right)^{-1}\phi(\mathbf{x})\right)\right)$$

Example from Bishop (2006)



Basic Idea: Prior controls the Model Class and hence what Data Sets can be explained

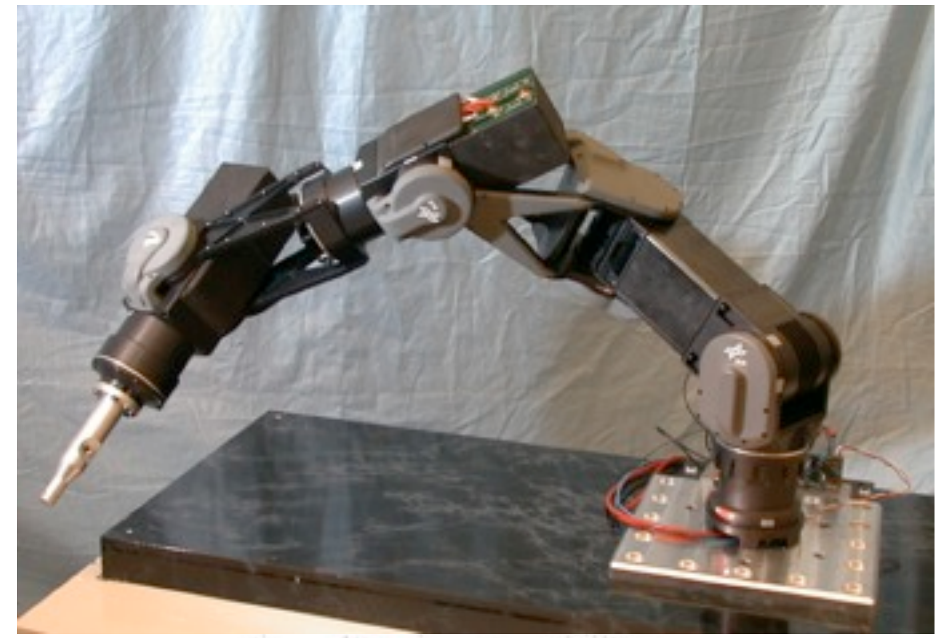


!

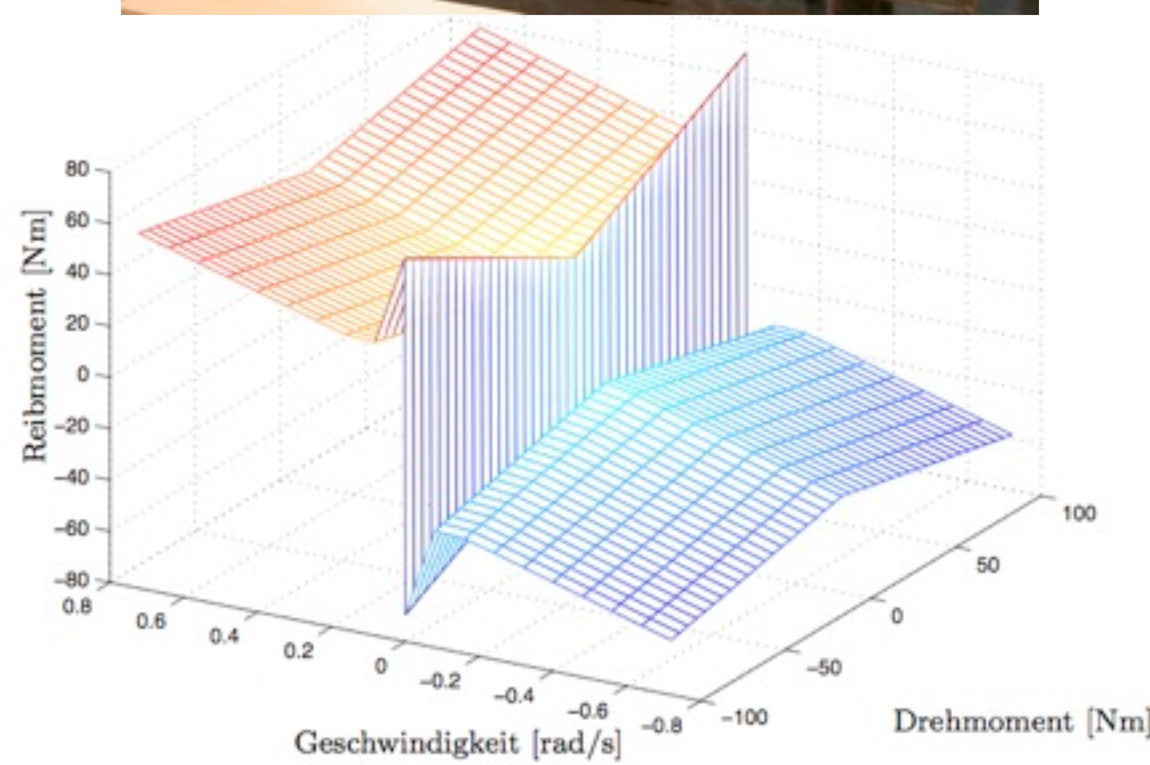
Content of this Lecture

1. Introduction to Regression
2. Accuracy, Overfitting and Regularization
3. How to Avoid Handcrafting Features
4. Learning Inverse and Forward Dynamics Models

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



- In most real applications, we know good features.
- However, we almost certainly don't know all features we need.
- ***Example***: Rigid body dynamics
  - Friction has no good features and may be self-referential.
  - 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 directly based on data.

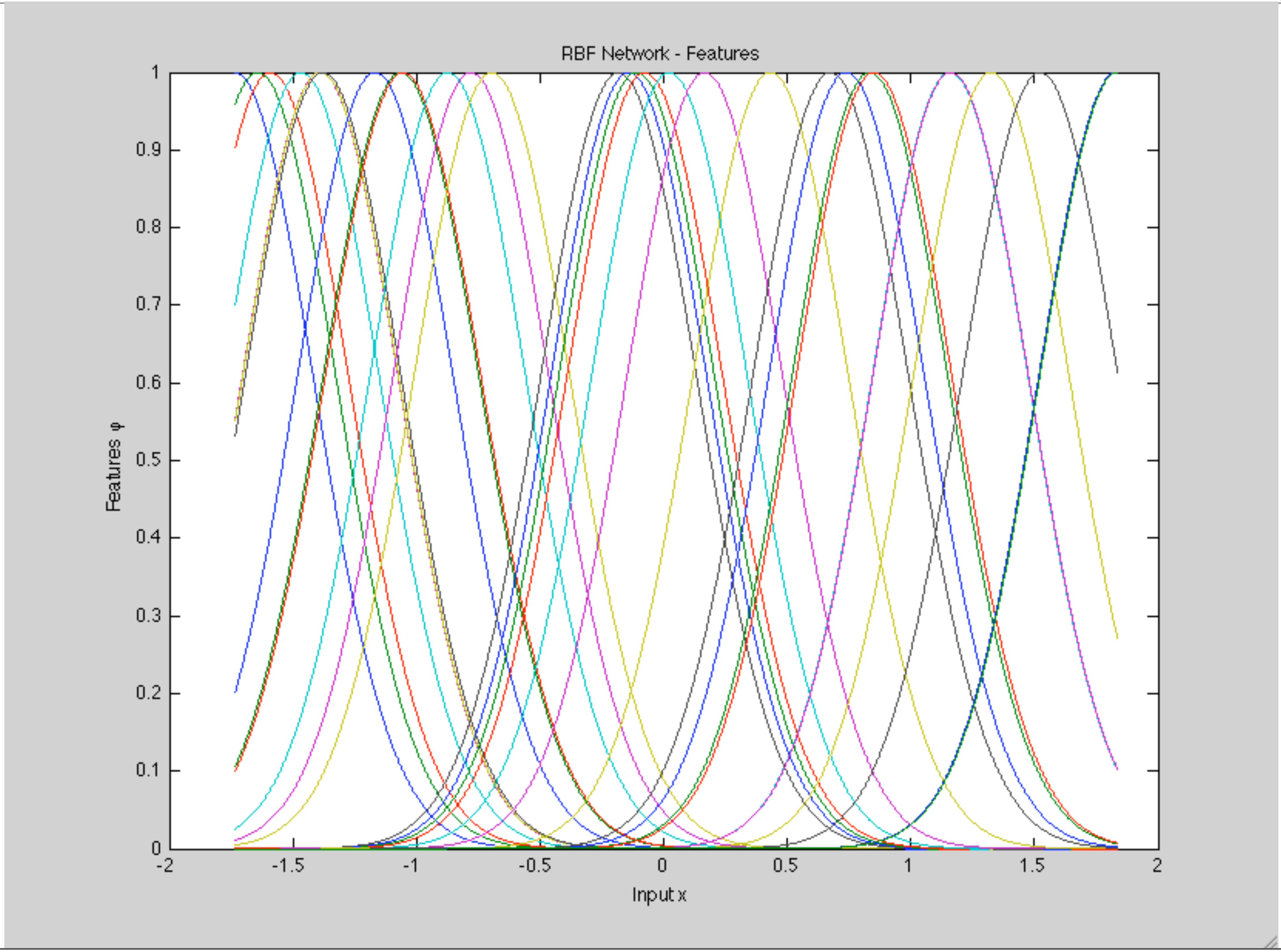
Example 1: *Radial basis functions* create an optimal smooth

Example 2: *Locally-Weighted Regression* localize in your data and try to interpolate with similar data.

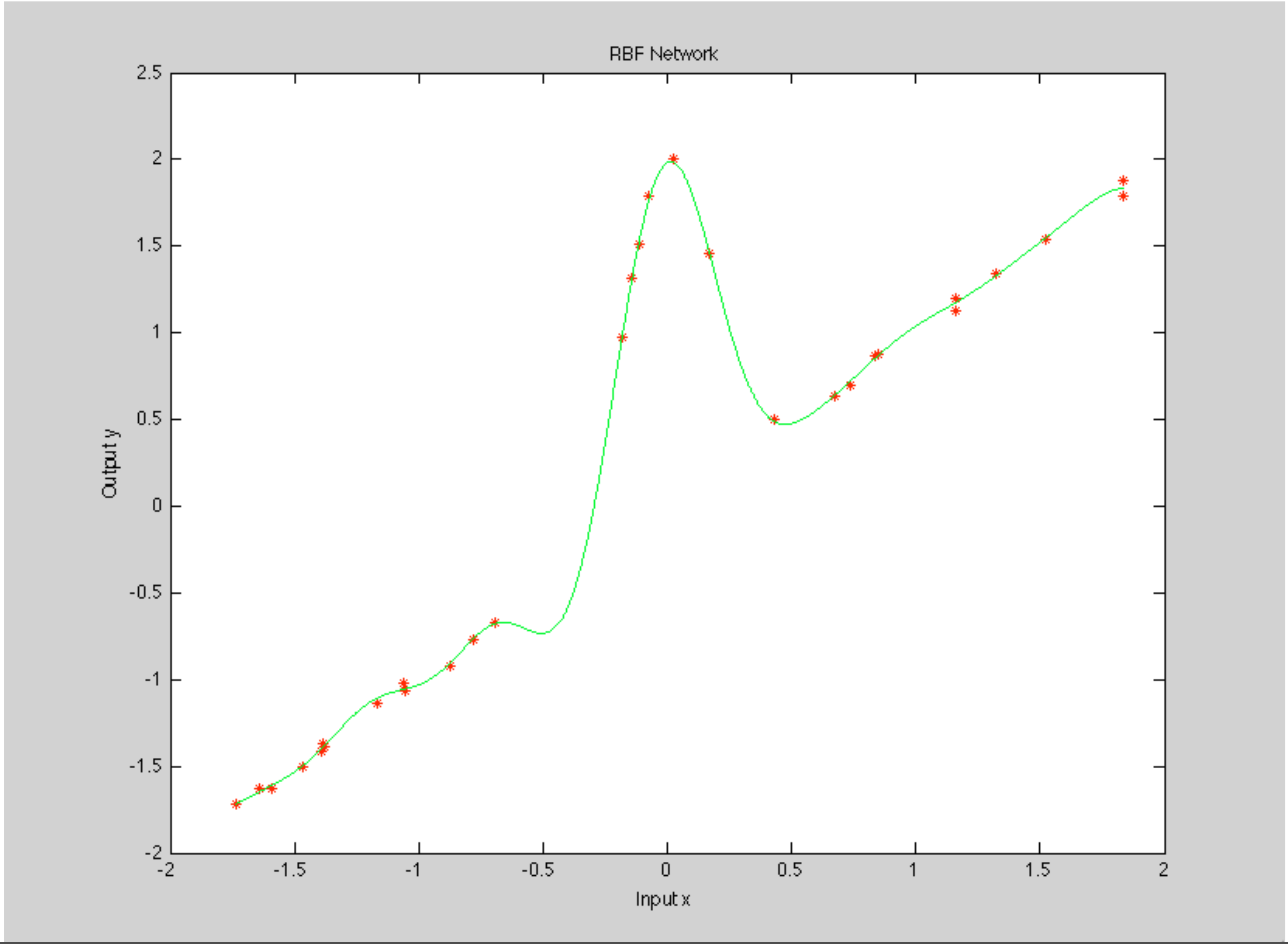
Example 3: *Kernel Regression* find the features by going into *function space* using a *kernel*?

!

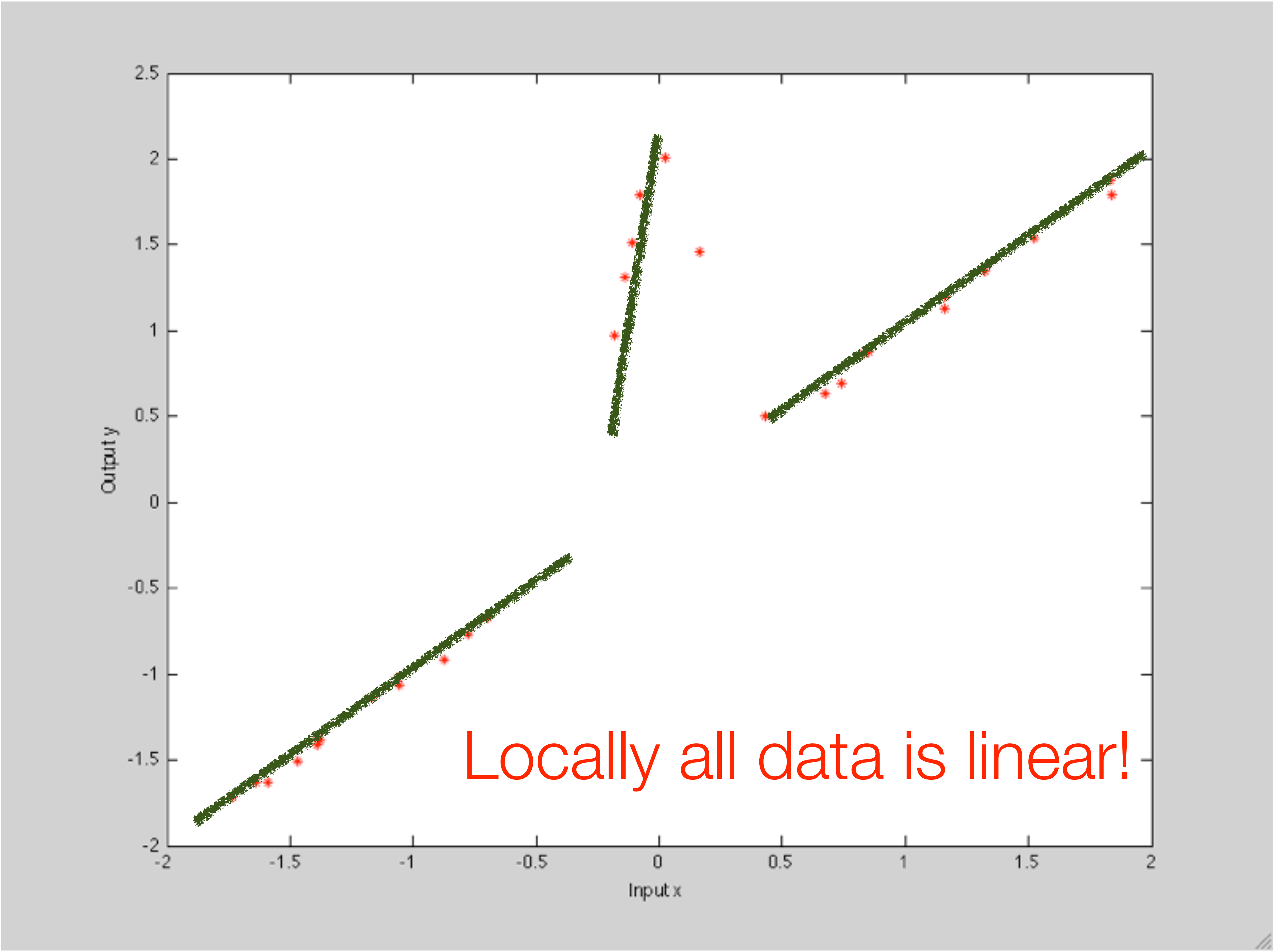
Example I: Radial Basis Function Features



Example I: Radial Basis Function Solution



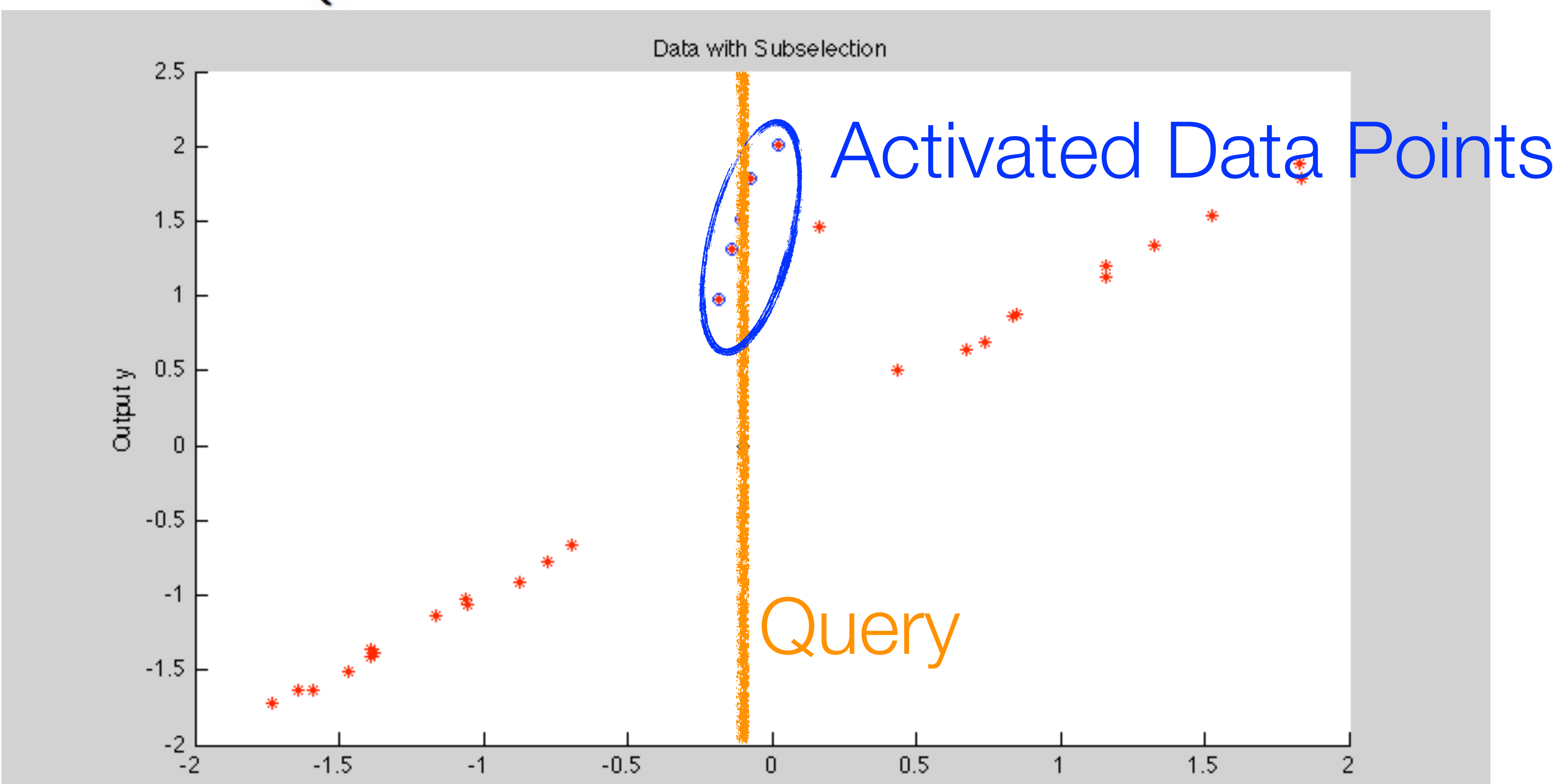
Example II: Locally Linear Solutions



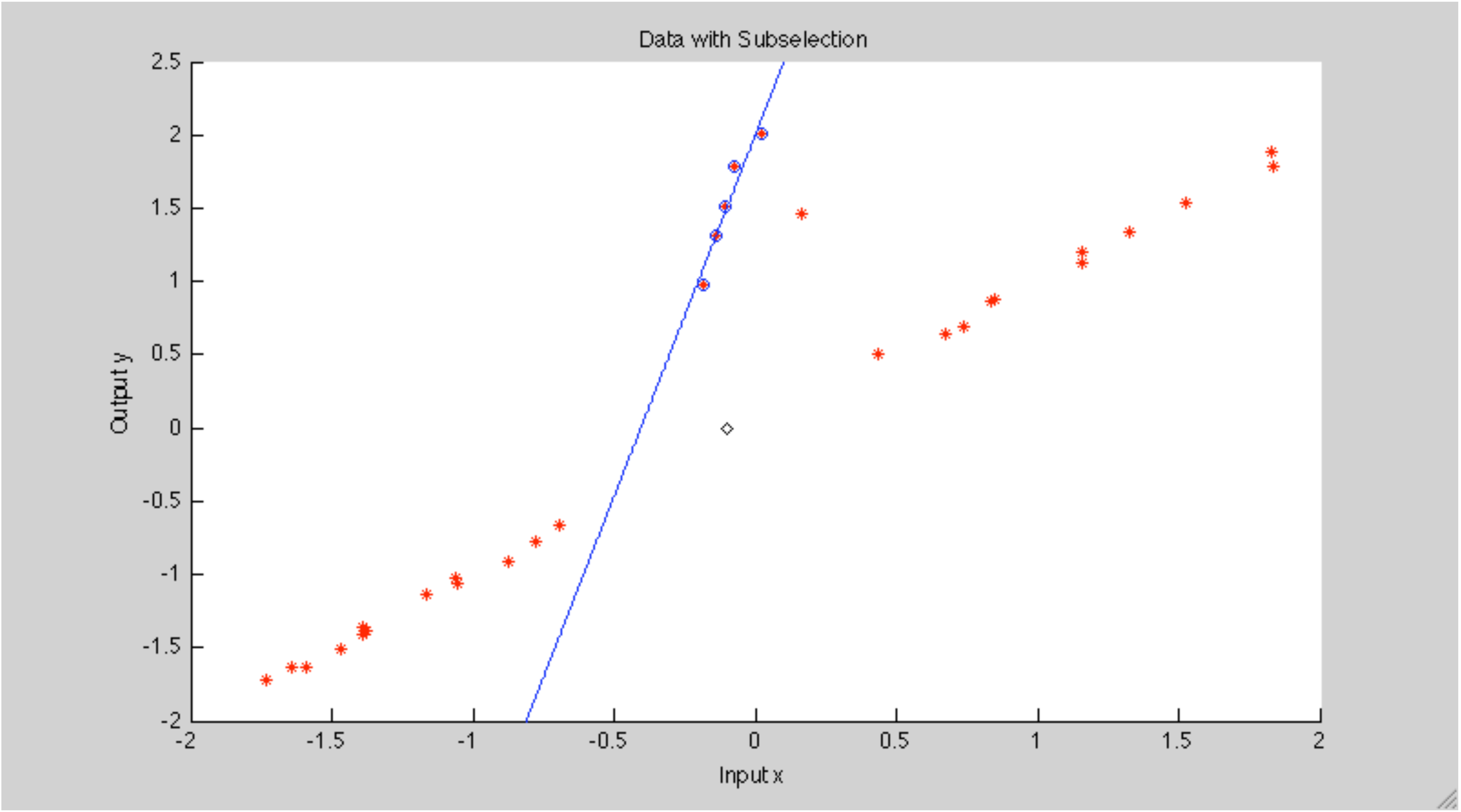
Example II: Locally Linear Solutions

- Locally all data is linear ... so why don't we take the next couple of data points to predict the solution?
- We select data points in a proximity and use only them in the prediction.

$$w(\mathbf{x}) = \begin{cases} 1 & \text{if } \|\mathbf{x} - \mathbf{x}_q\| \leq \epsilon, \\ 0 & \text{otherwise.} \end{cases}$$



Example II: Locally Linear Solution for a query



Example II: Locally Linear Solutions

We can formalize this in a cost function. Let us use our on-off function in the cost function and we obtain:

$$J = \frac{1}{2} \sum_{i=1}^{N} w_i(\mathbf{x})(y_i - \mathbf{f}_\theta(\mathbf{x}_i))^2,$$

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

$$J = \frac{1}{2}(\mathbf{Y} - \Phi\theta)^T \mathbf{W} (\mathbf{Y} - \Phi\theta),$$

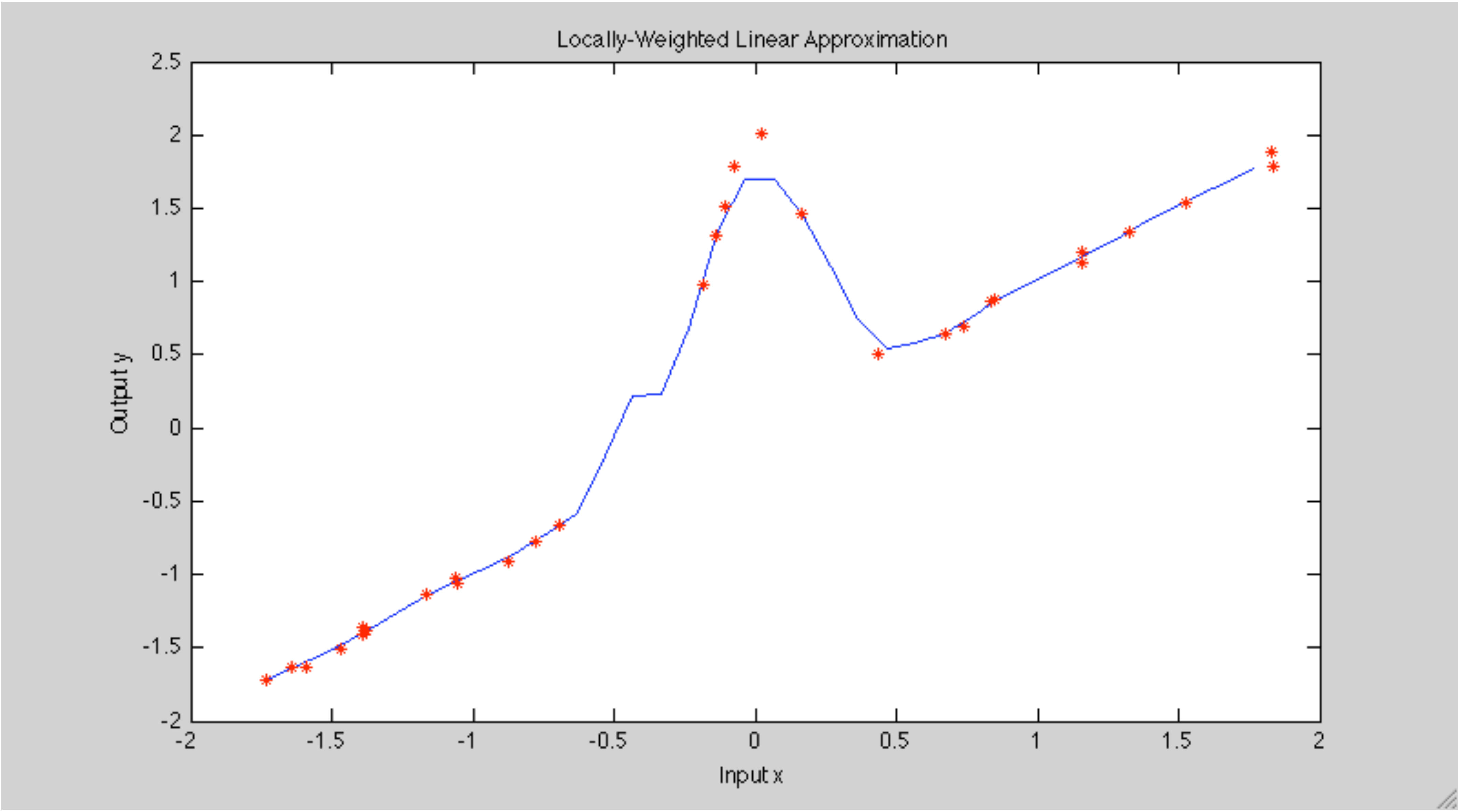
The solution to this problem

$$\theta = (\Phi^T \mathbf{W} \Phi)^{-1} \Phi^T \mathbf{W} \mathbf{Y}.$$

W can be large - don't implement it in MATLAB like this...

Solution with Locally-Weighted Regression

- We can use better weighting functions, e.g., $w_i(\mathbf{x}) = \exp\left(\frac{1}{\sigma^2}\left\|\mathbf{x}-\mathbf{x}_q\right\|^2\right)$
- Yes, just like in RBF networks.



Example III: Kernel Methods

- Look at the solution to linear regression again:

$$y(\mathbf{x}) = \phi(\mathbf{x})^T\theta = \phi(\mathbf{x})^T(\Phi^T\Phi)^{-1}\Phi^T\mathbf{Y}.$$

- We know from linear algebra that there is a left and a right pseudo-inverse

$$\Phi^{L\#} = (\Phi^T\Phi)^{-1}\Phi^T \qquad \Phi^{R\#} = \Phi(\Phi\Phi^T)^{-1}.$$

- and hence

$$\phi(\mathbf{x})^T(\Phi^T\Phi)^{-1}\Phi^T\mathbf{Y} = \phi(\mathbf{x})^T\Phi(\Phi\Phi^T)^{-1}\mathbf{Y}$$

- Even more general, the Woodbury matrix identity allows deriving:

$$(\Phi^T\mathbf{W}\Phi + \lambda\mathbf{I})^{-1}\Phi^T = \Phi(\Phi\Phi^T + \lambda\mathbf{W}^{-1})^{-1}$$

- This yields

$$\begin{aligned} y(\mathbf{x}) &= \phi(\mathbf{x})^T\theta = \phi(\mathbf{x})^T(\Phi^T\Phi + \lambda\mathbf{I})^{-1}\Phi^T\mathbf{Y}, \\ &= \phi(\mathbf{x})^T\Phi(\Phi\Phi^T + \lambda\mathbf{I})^{-1}\mathbf{Y}. \end{aligned}$$

This yields nearly the same solution as linear regression ... so why?

Example III: Kernel Methods

- Let us define the kernels:

$$k(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x})^T \phi(\mathbf{y}),$$
$$\begin{aligned} \mathbf{K}_{ij} &= k(\mathbf{x}_i, \mathbf{x}_j), \\ \mathbf{k}_i &= k(\mathbf{x}, \mathbf{x}_i), \end{aligned}$$

- Now we can rewrite the equation by

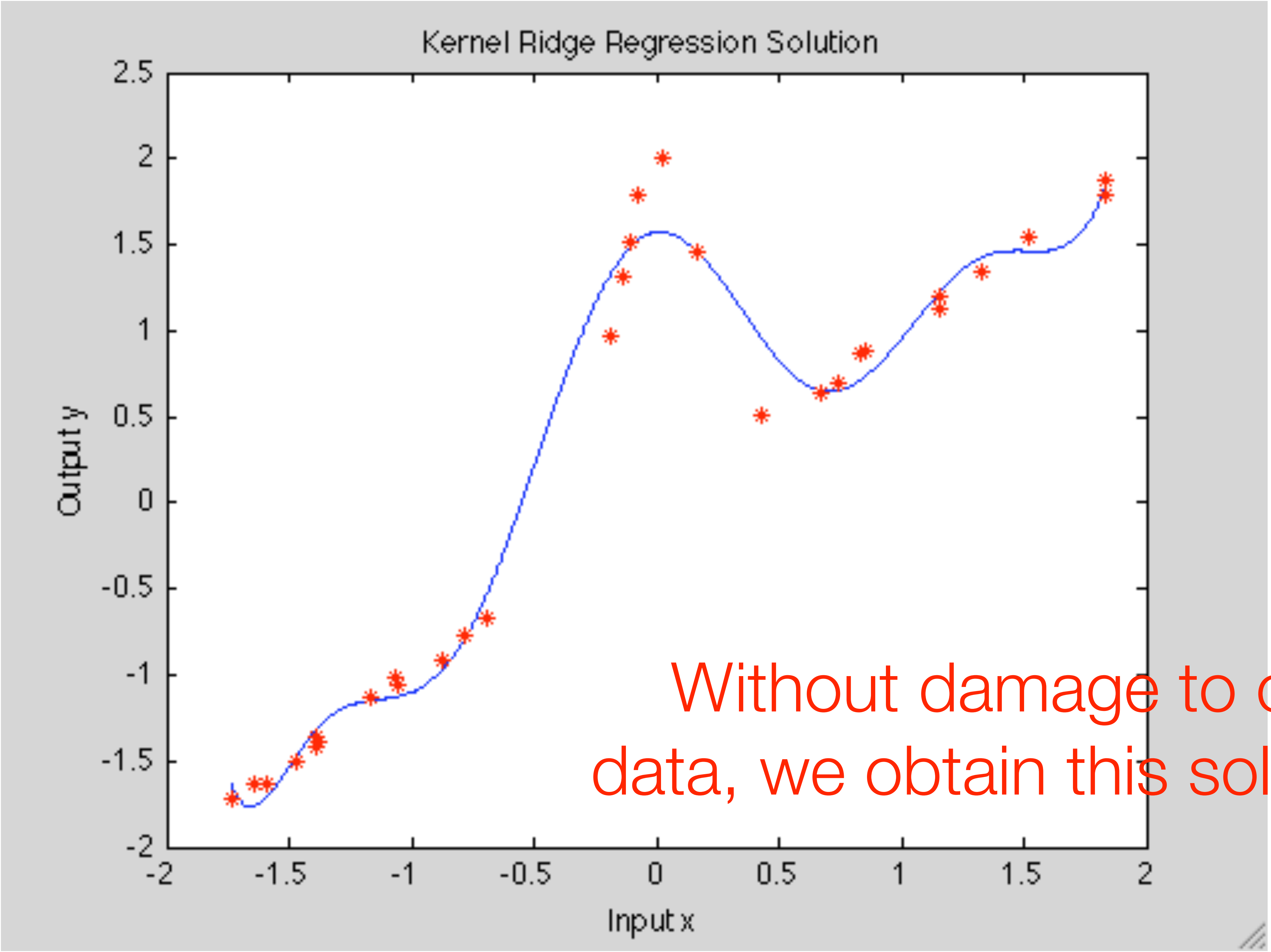
$$y(\mathbf{x}) = \phi(\mathbf{x})^T \Phi (\Phi\Phi^T + \lambda \mathbf{I})^{-1} \mathbf{Y} = \mathbf{k}(\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{Y}.$$

- This is called **kernel ridge regression**. Why would this be cool?

- Because we can use another kernel if we are unhappy with our features!

$$k(\mathbf{x}, \mathbf{y}) = \exp\left(\frac{1}{\sigma^2} \|\mathbf{x} - \mathbf{y}\|^2\right).$$

Example III: Using an exponential kernel...



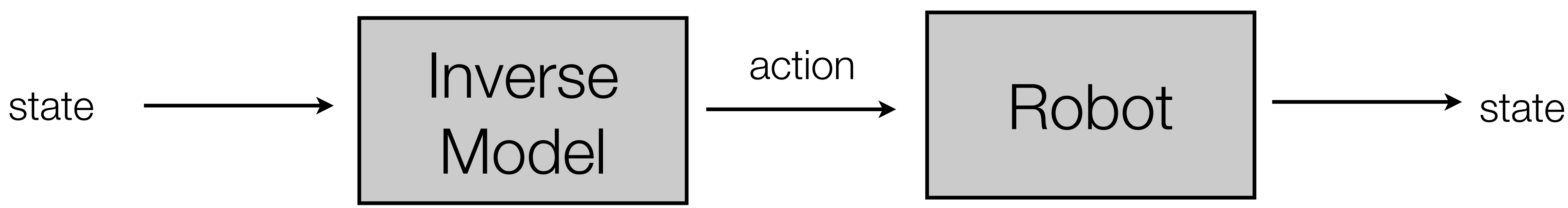
Without damage to our data, we obtain this solution!

Outline of the Lecture

1. Introduction to Regression
2. Accuracy, Overfitting and Regularization
3. How to Avoid Handcrafting Features
4. Learning Inverse and Forward Dynamics Models

Application: Model Learning for Accurate Control in Joint-Space

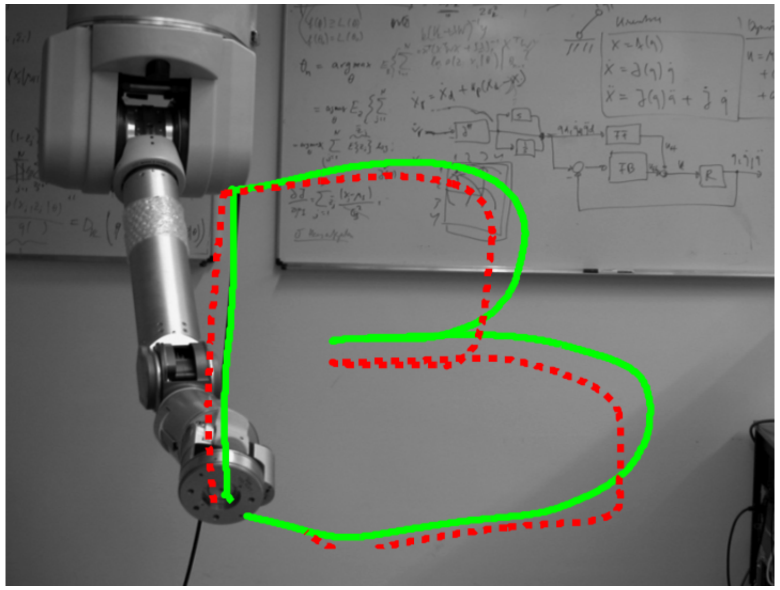
If you system that uniquely maps states to action, learning an Inverse Model directly yields a Policy



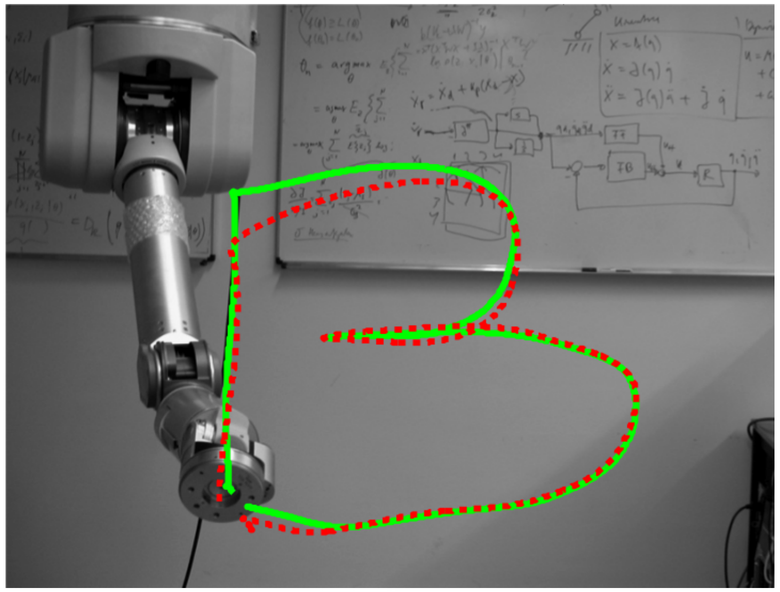
Learning to Control with Models

- Compliant, low-gain control of fast & accurate movements requires precise models.
- A changing world requires only adaption to altered dynamics.
- Control both directly in joint (here) and task space (next)

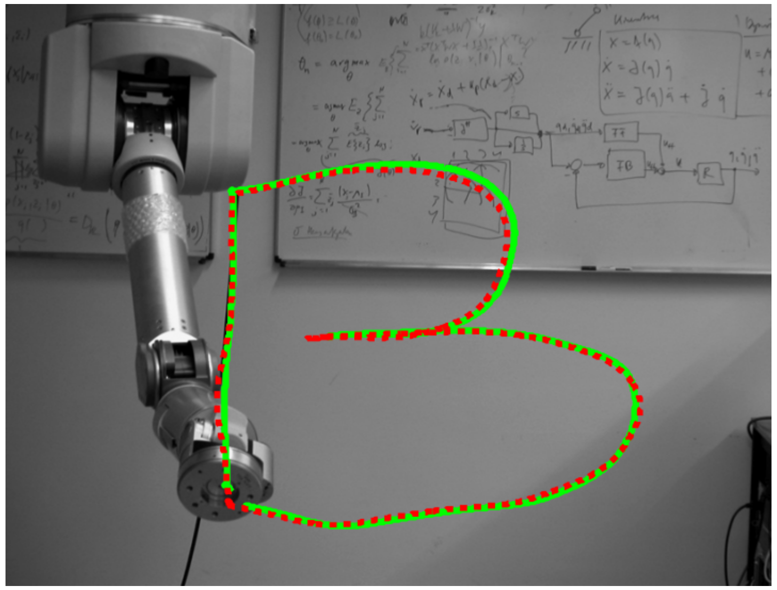
Analytical Rigid-Body Model with CAD data



Offline Trained



Online Trained



Nguyen-Tuong, Peters, IROS 2008 (Finalist for Best Paper Award)

Function Approximation Problem

Joint Accelerations, Velocities, Positions

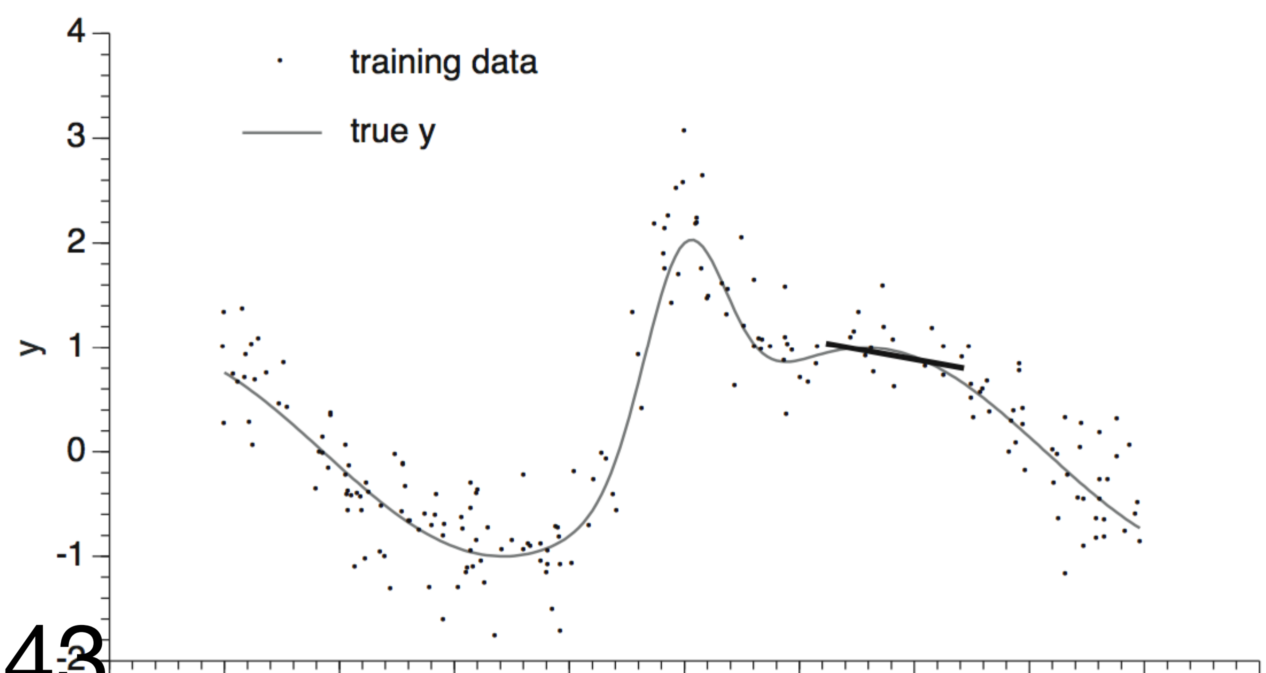
Torques

$$\mathbf{u} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{c}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q})$$

Mass Matrix

Gravity

Coriolis & Centripetal Forces



Inverse Dynamics is a giant *function approximation* problem

- **Robot arm**
  - 3 x 7 = 21 state dimensions,
  - 7 action dimensions
- **Humanoid**
  - 3 x 30 = 90 state dimensions
  - 30 action dimensions
- Learning in real-time!
- Unlimited continuous stream of data...

!

Function Approximation Problem

Joint Accelerations, Velocities, Positions

Torques

$$\mathbf{u} = \mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{c}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{g}(\mathbf{q})$$

Mass Matrix

Gravity

Coriolis & Centripetal Forces

What methods can deal with this problem?

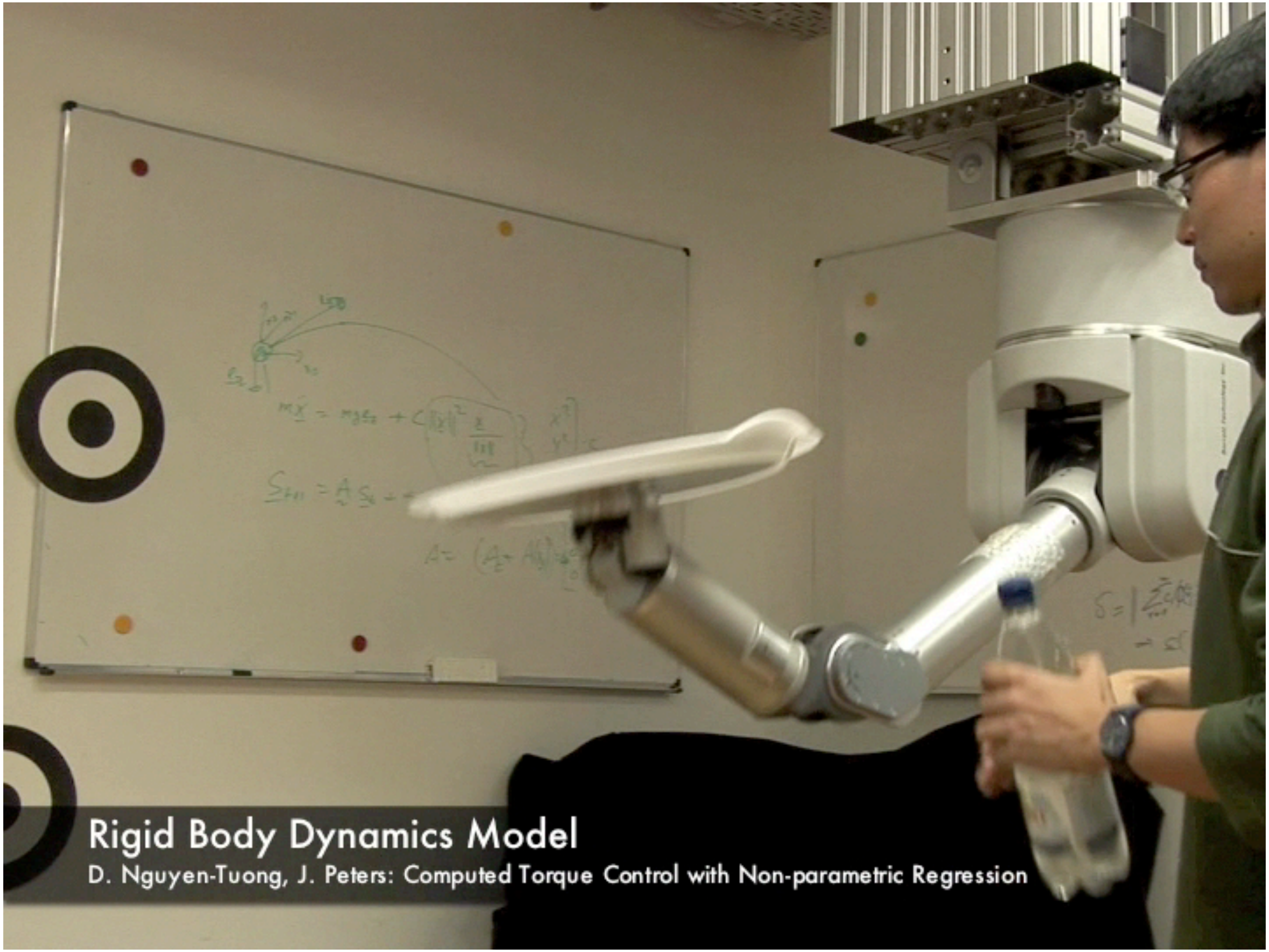
- **Neural networks?**
- **Mixture of Experts?**
- **Kernel Regression? SVR? GPR?**

✗These methods only in offline settings!!!

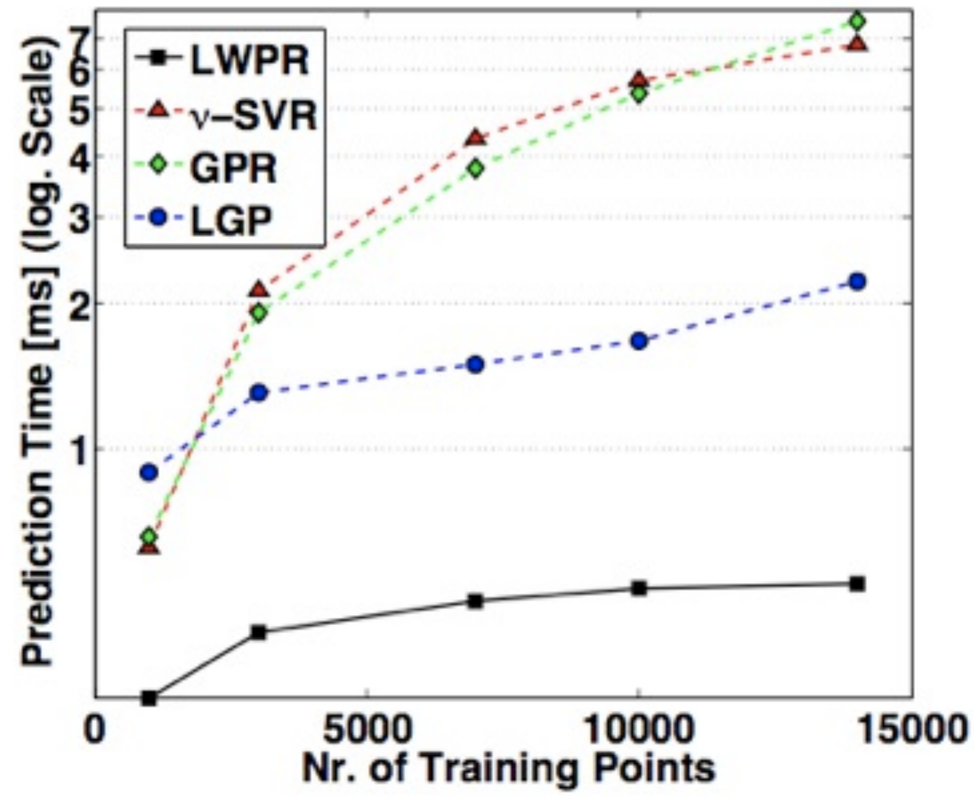
Local methods can perform online:

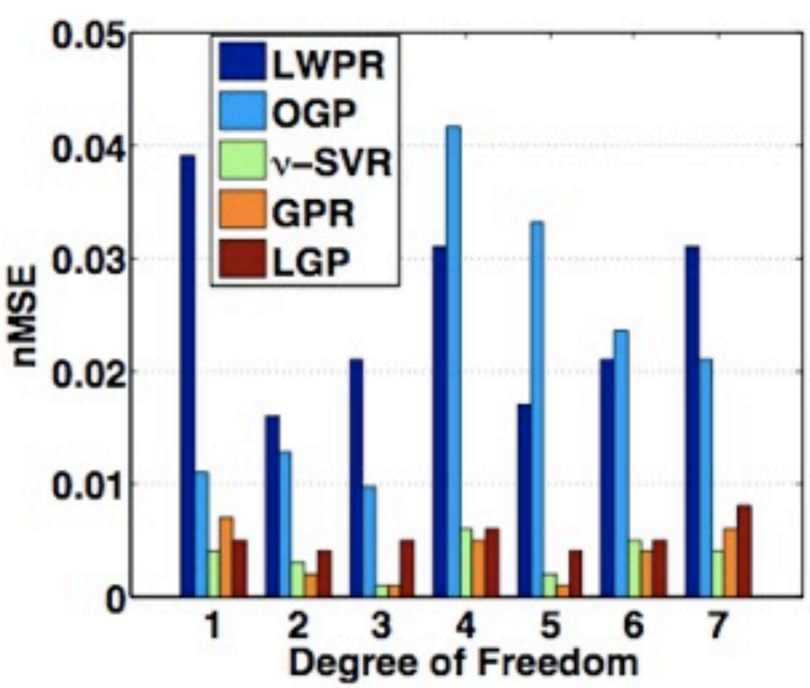
- **Locally Weighted PLS Regression (LWPR)** (Schaal, Atkeson & Vijayakumar, 2002)
- **Local Gaussian Processes (LGP)** (Nguyen-Tuong, Peters, 2008)

Learning to Control: Inverse Dynamics

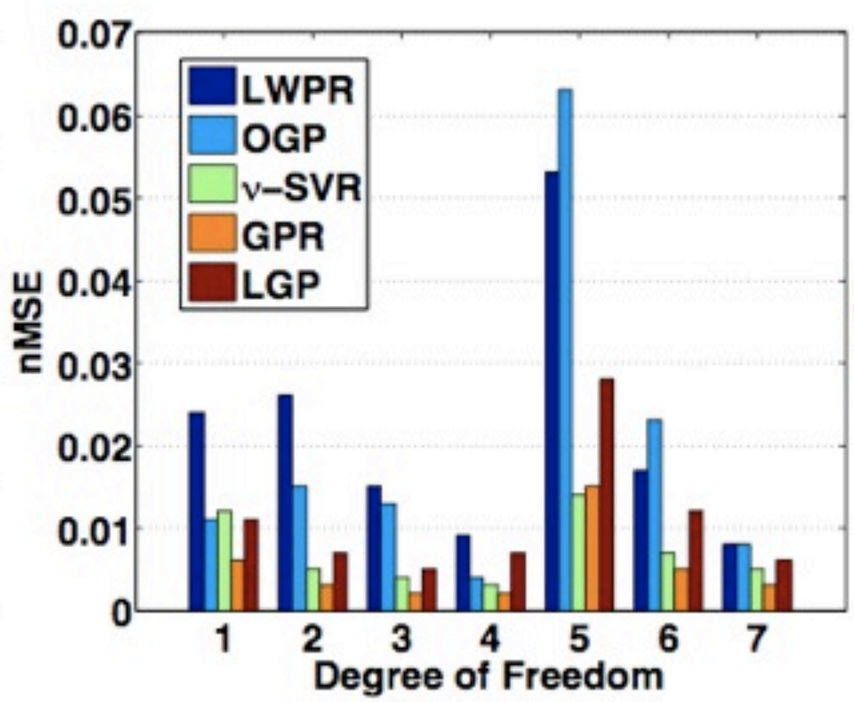


Comparison of Methods

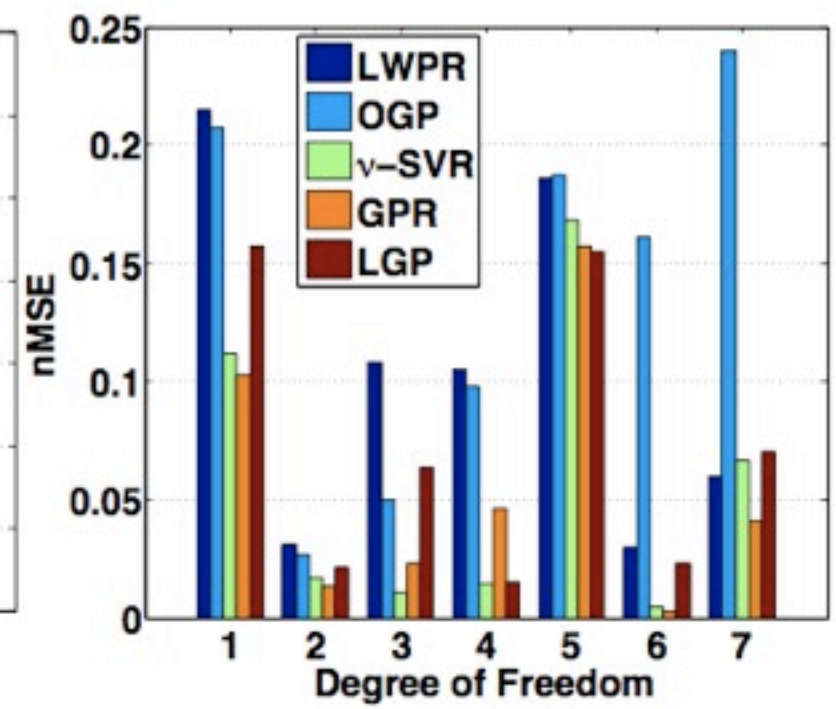




(a) Approximation Error on SL data (SARCOS model)

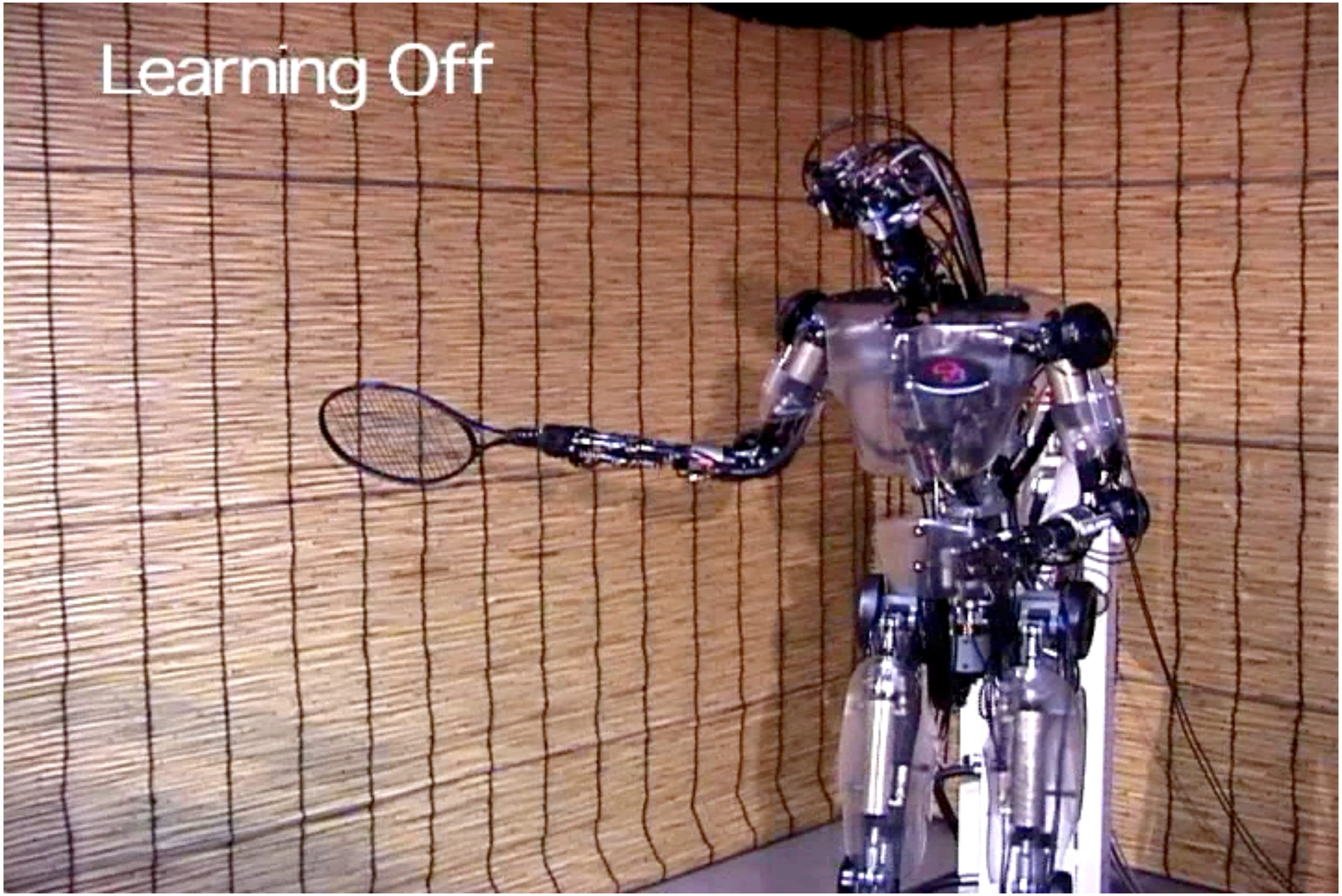


(b) Approximation Error on SARCOS data



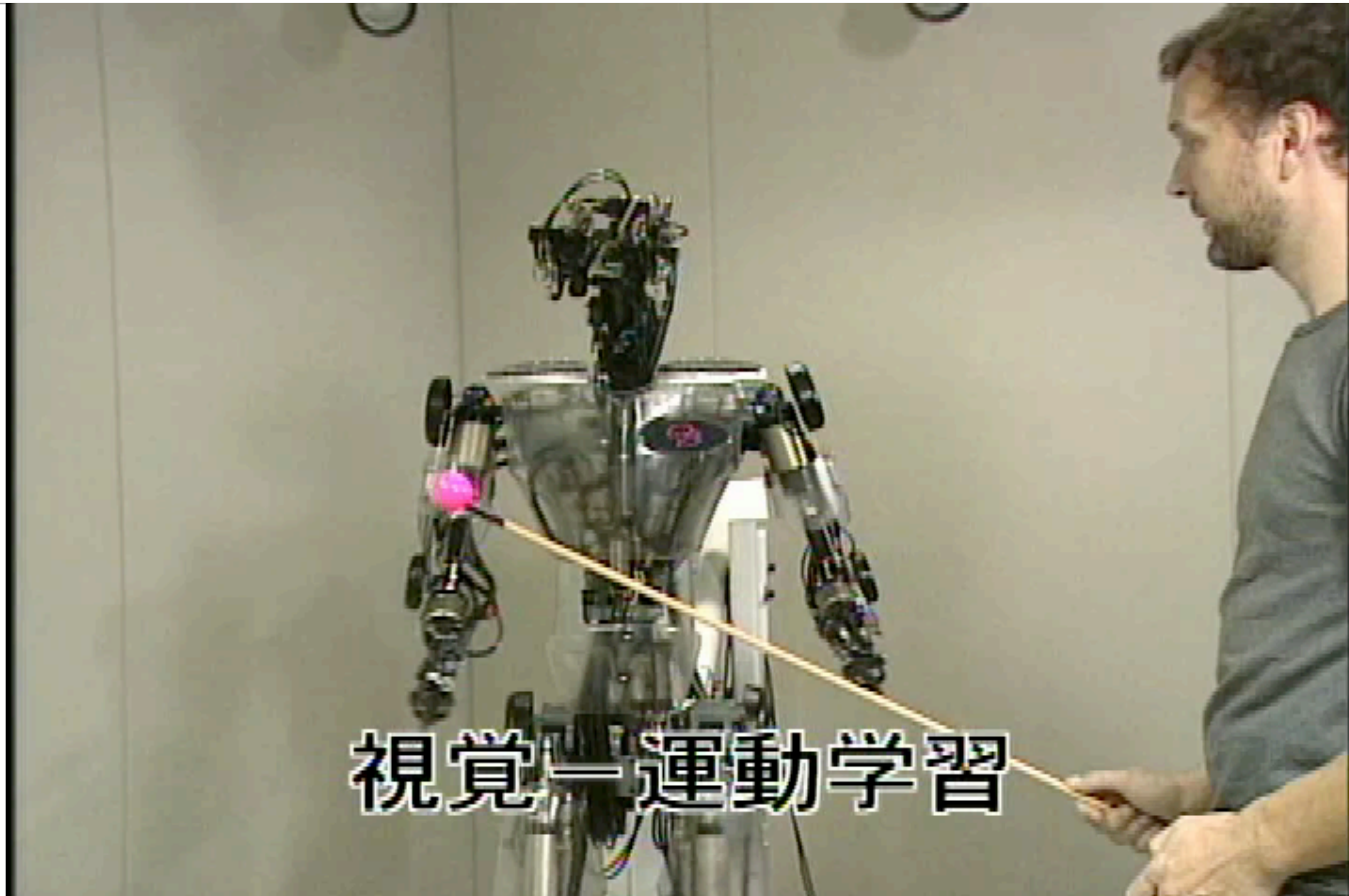
(c) Approximation Error on Barrett WAM data

Learning Inverse Dynamics for Humanoid Robots



Learns a model of the forces in the arm!
90 dimensional regression!

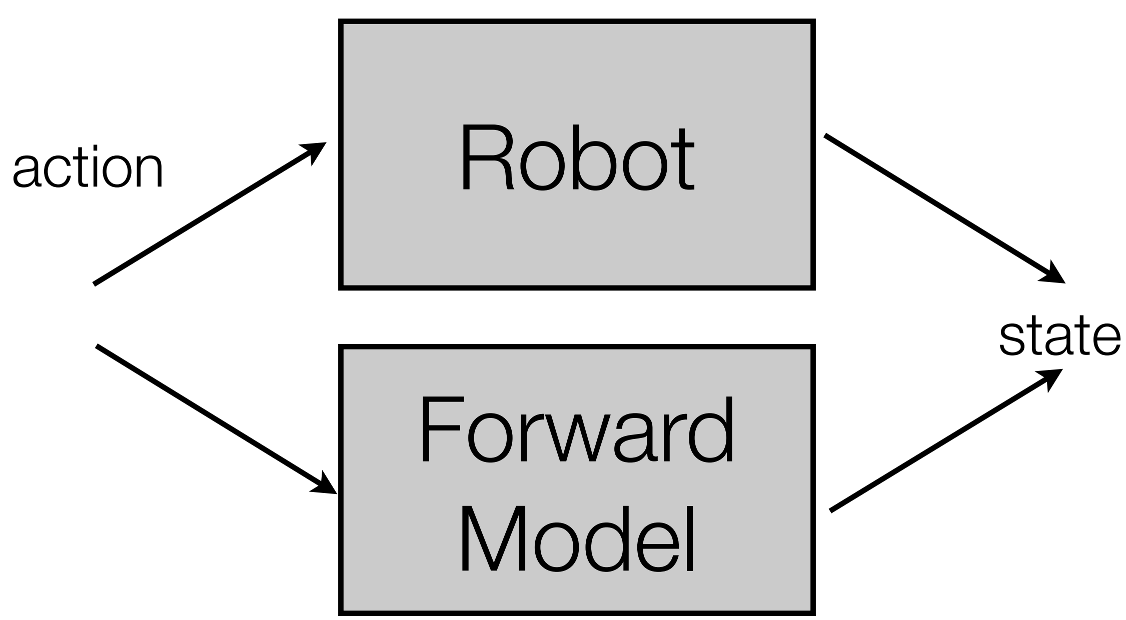
Learning Forward Kinematics for Humanoid Robots



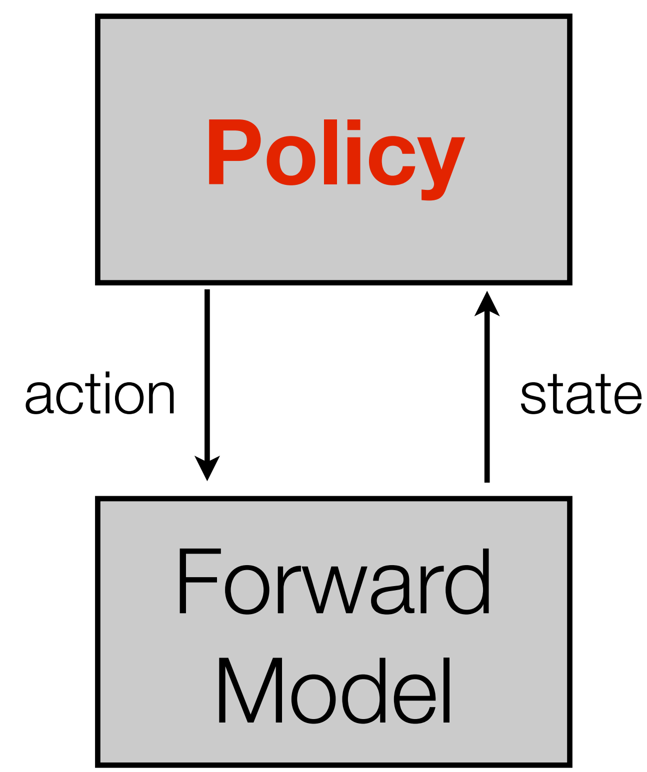
Learns a model of the position of the hand!
60 dimensional regression!

Goal of the Next Lecture

1. Step: Learn an Forward Model



2. Step: Use your favorite *Optimal Control Method* to get an optimal policy



The state-of-the-art

- Kernel-based Regression is currently yielding the highest accuracy in function approximation. The Bayesian version of Kernel Ridge Regression is called **Gaussian Process Regression (GPR)** and the current gold standard!
- The fastest accurate methods usually are locally linear weighted regression methods. The fastest off-the-shelf method that scales is **Locally Weighted Projection Regression**.
- It is very expensive - cubic in the data points while the others are cubic in the dimension and linear in the number of data points!
- If you have few data points (up to ~15.000), use GPR.
- If you want to be fast, rather use LWPR.
- If you need to make a trade-off, use LGP as the mix of LWPR and GPR.

Further Reading

- Schaal, S.;Atkeson, C. G.;Vijayakumar, S. (2002). *Scalable techniques from nonparameteric statistics for real-time robot learning*, Applied Intelligence, 17, 1, pp.49-60

- Nguyen-Tuong, D.; Seeger, M.; Peters, J. (2009). *Model Learning with Local Gaussian Process Regression*, Advanced Robotics, 23, 15, pp. 2015-2034.

- Nguyen-Tuong, D.; Peters, J. (2011). *Model Learning in Robotics: a Survey*, Cognitive Processing, 12, 4.