Model learning for robot control: a survey

Duy Nguyen-Tuong, Jan Peters

2011

Presented by Evan Beachly

Motivation

Robots that can learn how their motors move their body

Complexity

Unanticipated Environments

Degradation

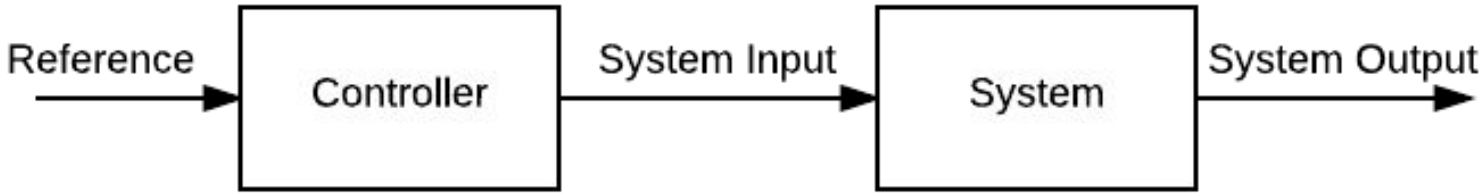
Images: Boston Dynamics Atlas, Boston Dynamics Big Dog, Ascending Technologies Hummingbird

Outline

- Background: Controllers
- Modelling
  - Model Types
  - Learning Architectures
- Challenges of applying machine learning to Robotics
- Machine Learning Methods
- 3 Examples of ways model learning has been applied to robotics
  - Simulation-based optimization
  - Approximation-based inverse dynamics control
  - Learning operational space control
- Future Directions

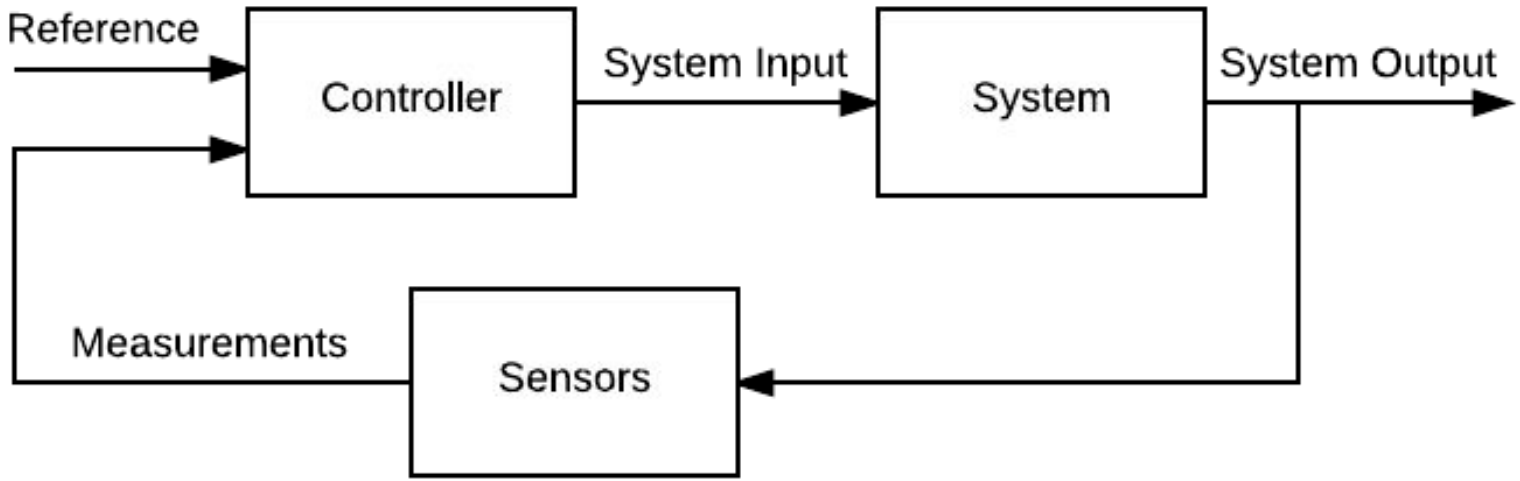
Background: Controllers

Open Loop Control



Example: Microwave Oven

Closed Loop Control

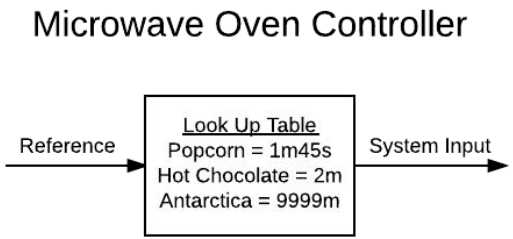


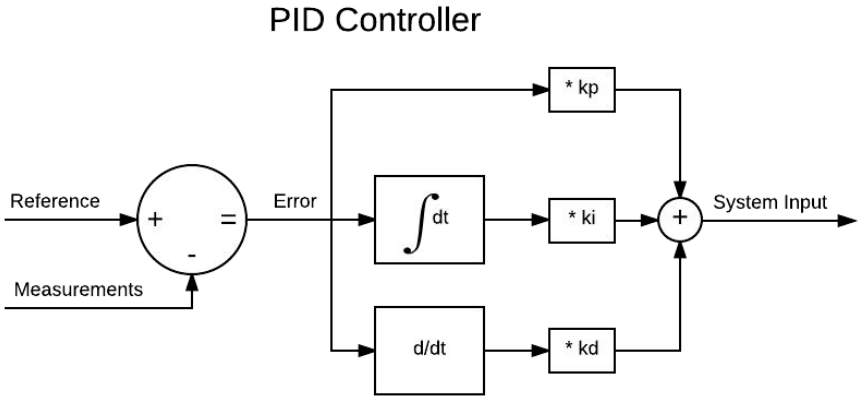
Example: Thermostat, Cruise Control

Implicit Models

A model describes how the system is expected to react

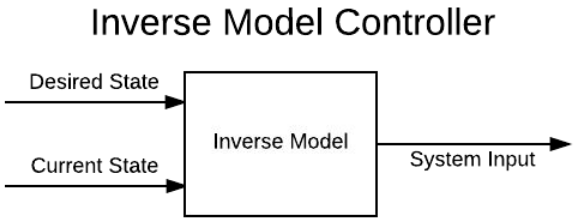
It might be implicitly coded into the rules of the controller

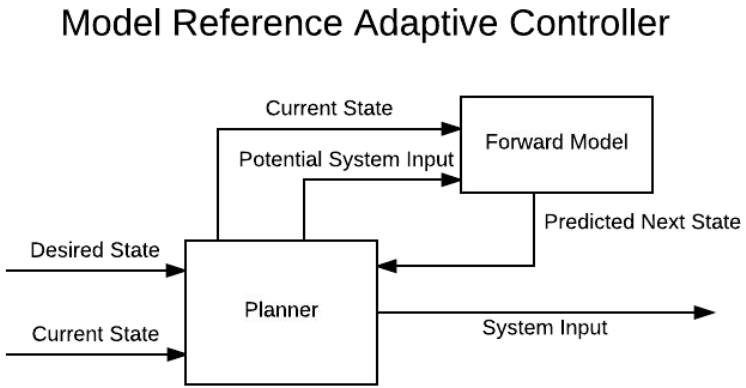




Explicit Models

We will discuss more advanced control algorithms that explicitly model the system's behavior





Modelling

Preview: Examples of Model Learning Controllers

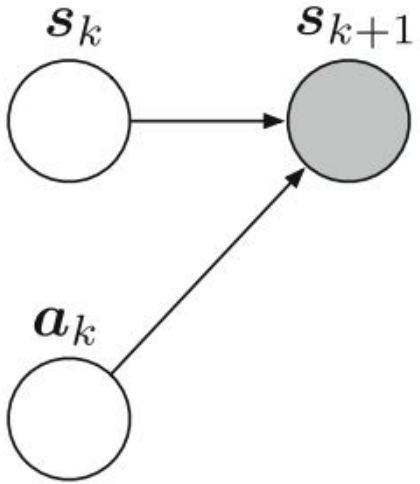
		Learning Architecture		
		Direct	Indirect	Distal Teacher
Model Type	Forward	Self-tuning regulator Model Reference Adaptive Control Model Predictive Control	N/A	N/A
	Inverse	Arm Joint Torque Controller	Feedback Error Learning	N/A
	Mixed	Multi Module Switching Controller	Gated network of experts	Distal Teacher
	Multi-Step	ARX, ARMAX variants	N/A	N/A

Model Types

- Forward
  - Given the current state and an action, predicts the next state
- Inverse
  - Given the current state and the desired state, predicts the action
- Mixed
  - Pairs a forward and inverse model together
- Multi-step prediction
  - Predicts multiple future states/actions

Forward Model

Given the current state and an action, predict the next state
Corresponds to the state transfer function of the environment



- Model Reference Adaptive Control (MRAC): Uses forward model to choose the action that gets it closest to the destination

$$\pi(s) = \arg\min_{a} \| f_{\text{forward}}(s_t, a) - s_{t+1}^{\text{des}} \|$$

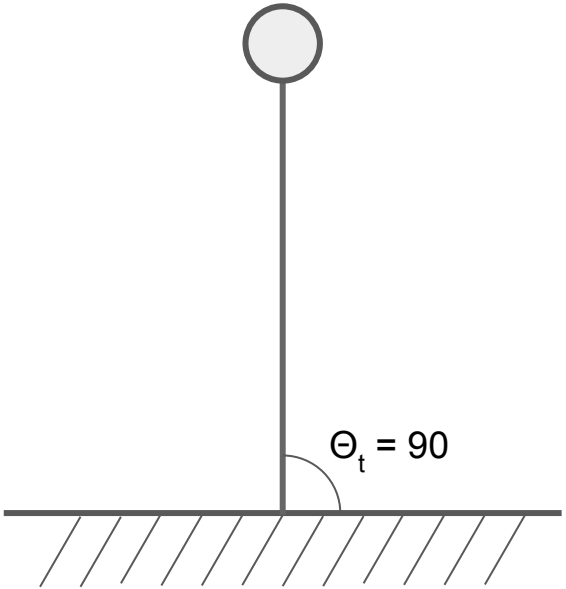
- Model Predictive Control (MPC): Uses forward model to choose the sequence of N actions that gets it closest to the destination

$$\pi(s) = \arg\min_{a_{t:t+N}} \sum_{k=t}^{t+N} F_{\text{cost}}\left(f_{\text{forward}}(s_k, a_k) - s_{k+1}^{\text{des}}\right)$$

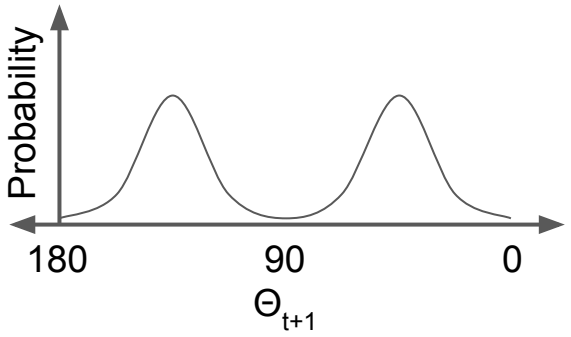
Problems with Forward Models

There may be many possible next states. Forward Models return the mean of the distribution, even if it is unlikely.

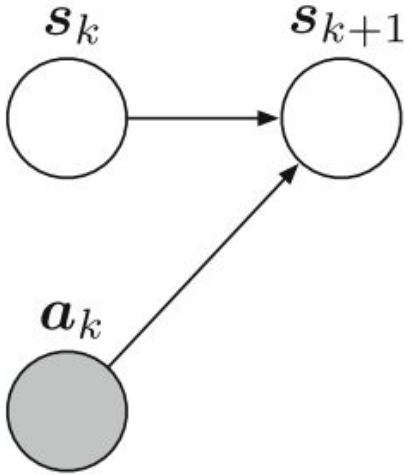
Inverted Pendulum Example



Likelihood of next states of the unstable inverted pendulum



Inverse Model



Given the current state and the next state, what is the action?

Easy and Fast policy: take the action given by the inverse model

$$\pi(s) = f_{\text{inverse}}(s, s_{\text{des}}) + k(s - s_{\text{des}})$$

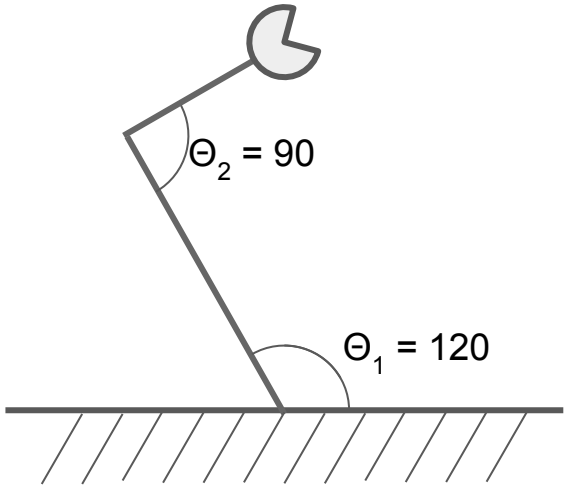
PD-controller that helps stabilize the robot.
Not necessary if the inverse model is good

Problems with Inverse Models

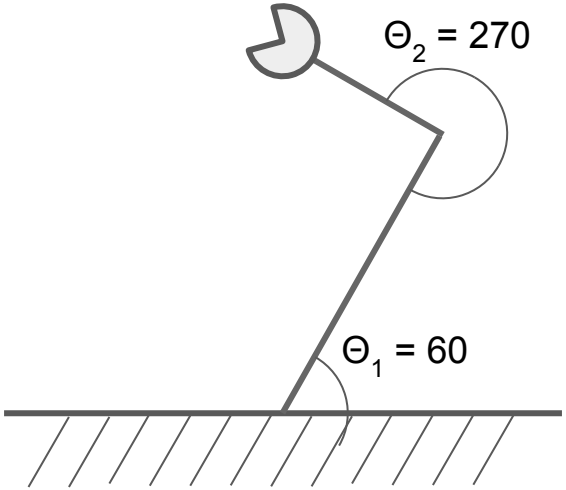
An action that will get you from the current state to the desired state may not exist.

Ill Posedness: Multiple actions from the current state to the desired state.

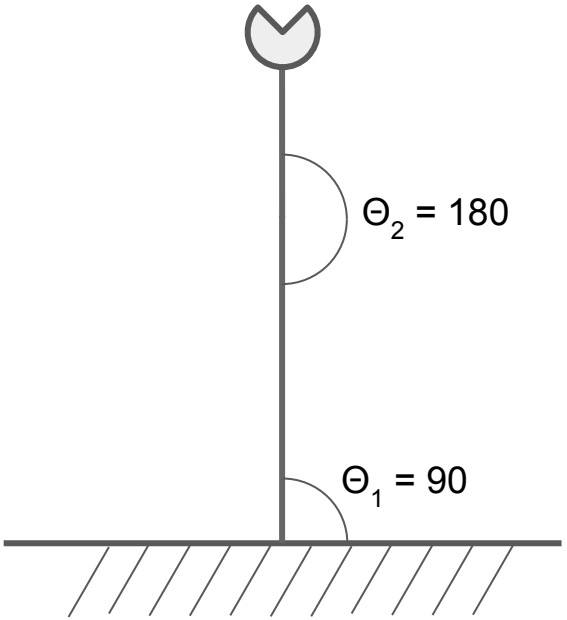
Robot Arm Example



One way to grip the desired position



Another way



The average

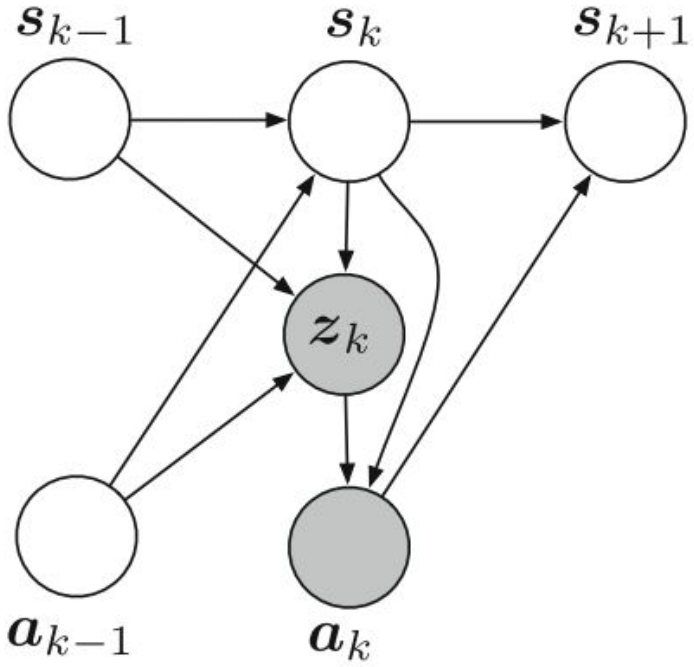
Link to Direct Modelling

Mixed Model

Use a forward model to resolve ill-posedness of the inverse model

In figure: Use forward model to estimate a "latent" state variable z_k from the previous state, previous action, and current state. Use z_k to disambiguate multiple possible a_k's in the inverse model for s_k and s_{k+1}.

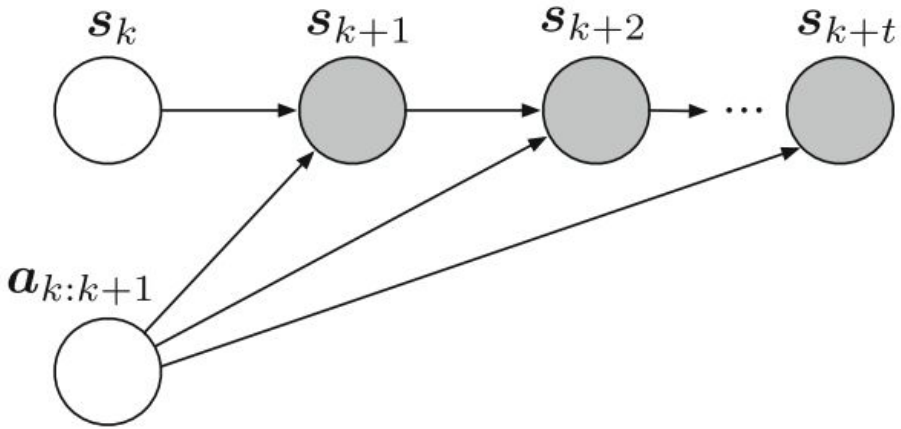
Distral Teacher: Use forward model to resolve ill-posedness when training the inverse model.

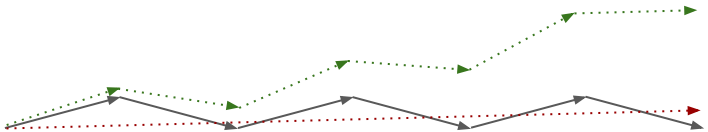


Multi-step Prediction Model

Uses a different model for each time step.

Avoids accumulating prediction error





True path for the planned actions

Path Predicted by Model Predictive Control with a forward model biased to the left

Path Predicted by the last model in a multi-step model

Learning Architectures

- Direct Modelling
- Indirect Modelling
- Distal Teacher Learning

Direct Modelling

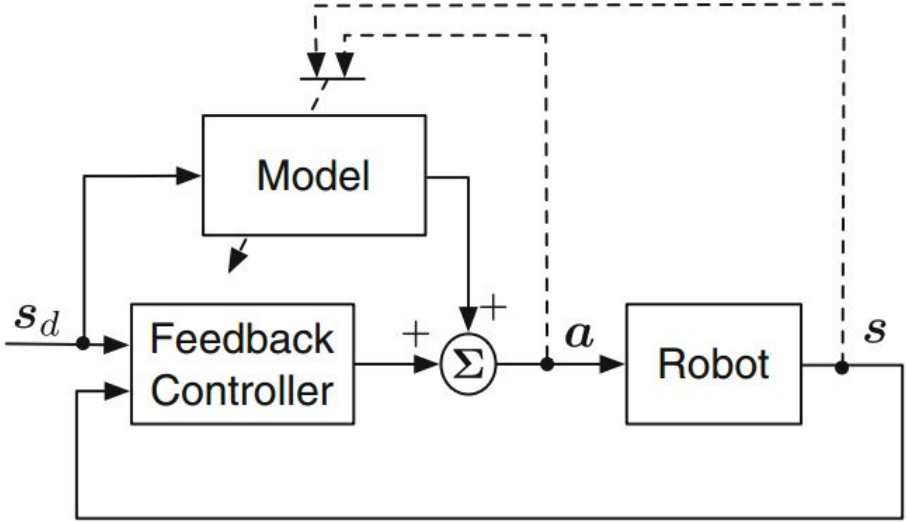
Use observed input/output pairs to train the model

Can be done off-line or on-line

- If on-line, use a feedback controller to guide the robot during learning

Difficult to train inverse models in ill-posed problems unless:

- the data is generated in an intelligent way
- local models are used.



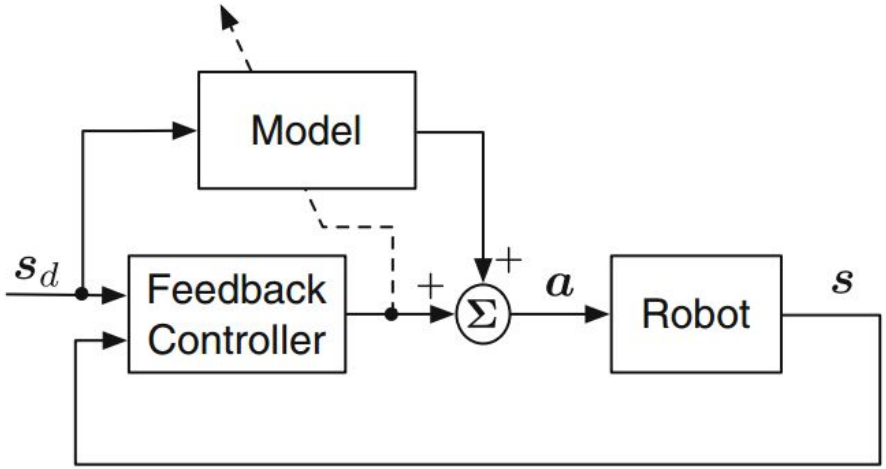
Link to Robot Arm Example

Indirect Modelling

Minimizes an error signal on-line, and therefore the inverse model converges to a single action in ill-posed problems.

In Figure: Feedback Error Model Learning

- Train an open-loop inverse model controller using the output of a feedback controller.
- As the model gets better, the output of the feedback controller will tend toward zero.

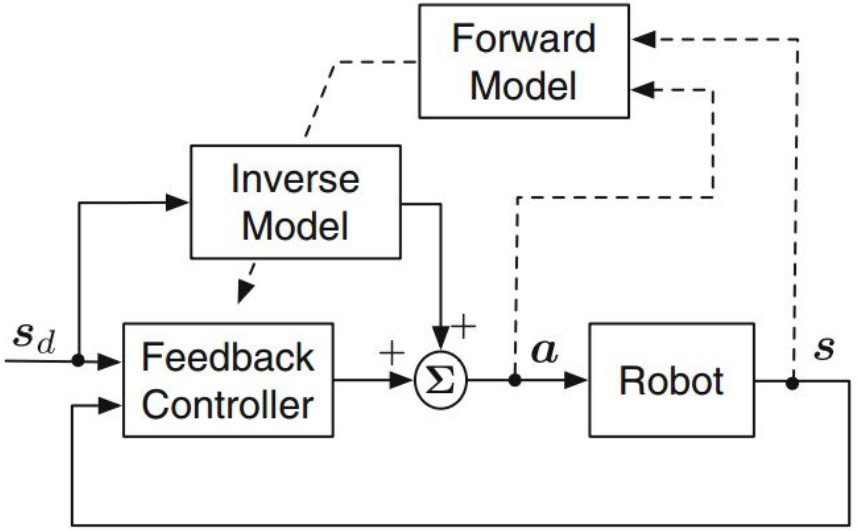


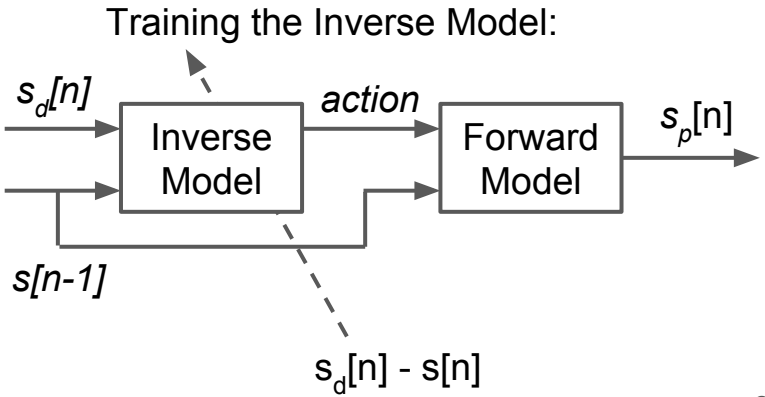
Distal Teacher Learning

Forward Model is trained directly to predict the next state.

Compose the Inverse and Forward models, and train the composed model while holding the Forward model fixed.

Advantage of adding the Forward model is that it results in a globally consistent inverse model instead of local on-policy optimization





Day 2
Model learning for robot control: a survey

Duy Nguyen-Tuong, Jan Peters
2011

Presented by Evan Beachly

Outline Day 1

- Background: Controllers
- Modelling
  - Model Types
    - Forward
    - Inverse
    - Mixed
    - Multistep
  - Learning Architectures
    - Direct
    - Indirect
    - Distal Teacher

Outline Day 2

- Challenges of applying machine learning to Robotics
- Machine Learning Methods
  - Global Methods
    - Artificial Neural Networks
    - Linear Regression
  - Local Methods
    - Locally Weighted Linear Regression
- 3 Examples of ways model learning has been applied to robotics
- Future Directions

Challenges of Applying Machine Learning to Robotics

Data Challenges

- Need to sample a large region of the state space
  - Make random movements
- Robotics may not have smooth dynamics (Static Friction to Kinetic Friction)
  - Use kernel methods that allow for unsmooth functions
  - Switch between local models (discontinuous at boundaries)
- High dimensionality
  - Dimensionality reduction
- Redundant data
  - Filter
- Noise and outliers
  - Estimate noise level in the data, and fit a model of appropriate complexity

Algorithmic Constraints

- Needs to run in real time
  - Don't use $\Omega(n^3)$ algorithms
  - Reduce data set size
  - Parallel computation
- Incorporate prior knowledge
  - Training data may be sparse or difficult to get
  - Specify apriori probability in probabilistic frameworks
- Online learning
  - May not know the important region of the state space beforehand
- Active learning
  - Labelling data may be time consuming
  - Figure out which regions of the state space need to be explored, and only ask the human to label those

Real-world challenges

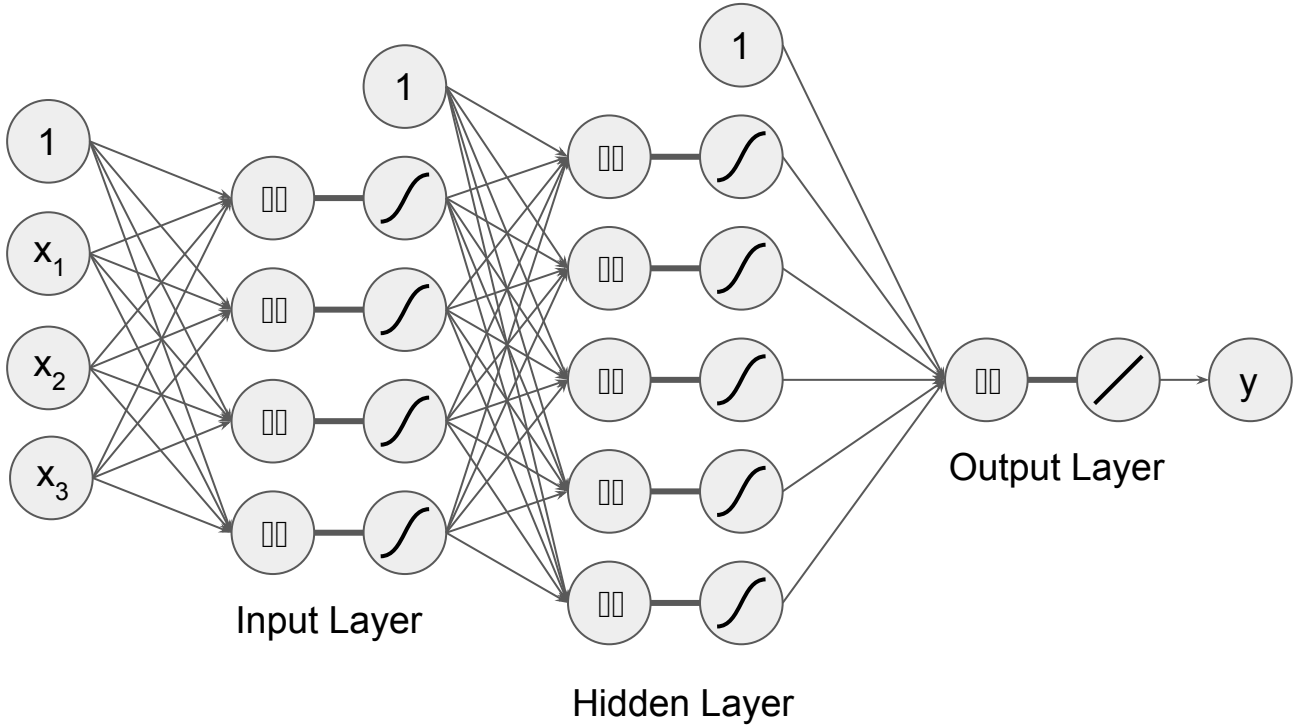
- Safety
  - Fail safe
- Robustness and Reliability
  - Feature selection to avoid overfitting
- Measurement Errors & Missing Data
  - Probabilistic learning methods
- Time-varying systems
  - Online learning

Machine Learning Methods

Global Regression Algorithms

- Estimate the form of the entire function
- Parametric Algorithms (model complexity is fixed):
  - Artificial Neural Network
  - Linear Regression
    - Fit a polynomial to the data
- Nonparametric Algorithms (model complexity depends on data):
  - Self-reconfiguring Artificial Neural Networks
  - Gaussian Process Regression
  - Sparse Gaussian Process Regression
  - Support Vector Regression
  - Incremental Support Vector Machine

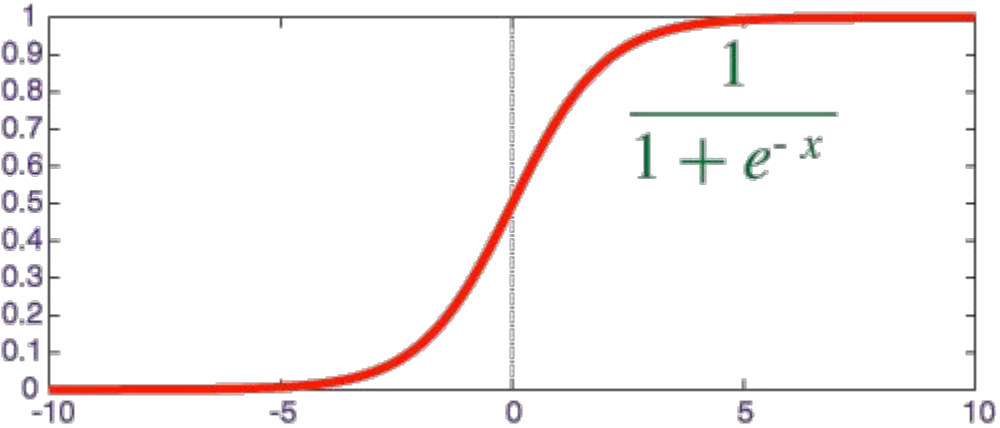
Artificial Neural Network

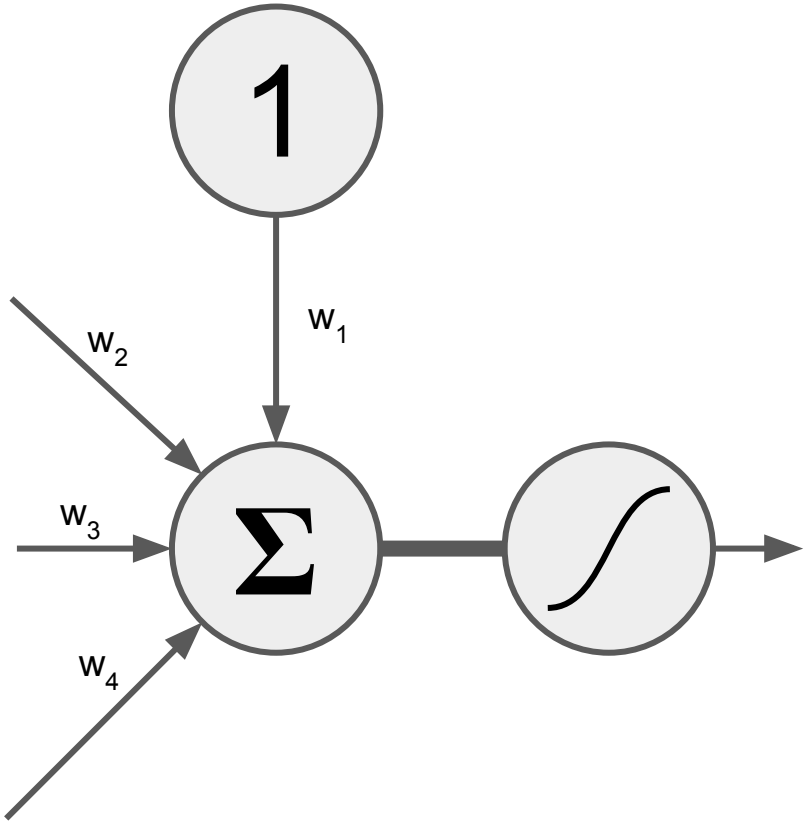


Perceptron

Computes a weighted sum of inputs

Sigmoid function:





The zero of the weighted sum is a hyperplane that divides the space.

On one half of the space, the sigmoid returns a value near 1, on the other, 0

Sigmoid function image: https://qph.ec.quoracdn.net/main-qimg-05edc1873d0103e36064862a45566dba

Backpropagation

An iterative method to set the weights in the network

1. Initialize all of the weights to random values
2. Select a training sample
3. Plug the training input into the network, and compute the error between the output and training sample's value
4. Compute the gradient of the error with respect to the weights of the network
5. Slightly change each weight along the gradient
6. If the network's performance is not good enough, goto 2

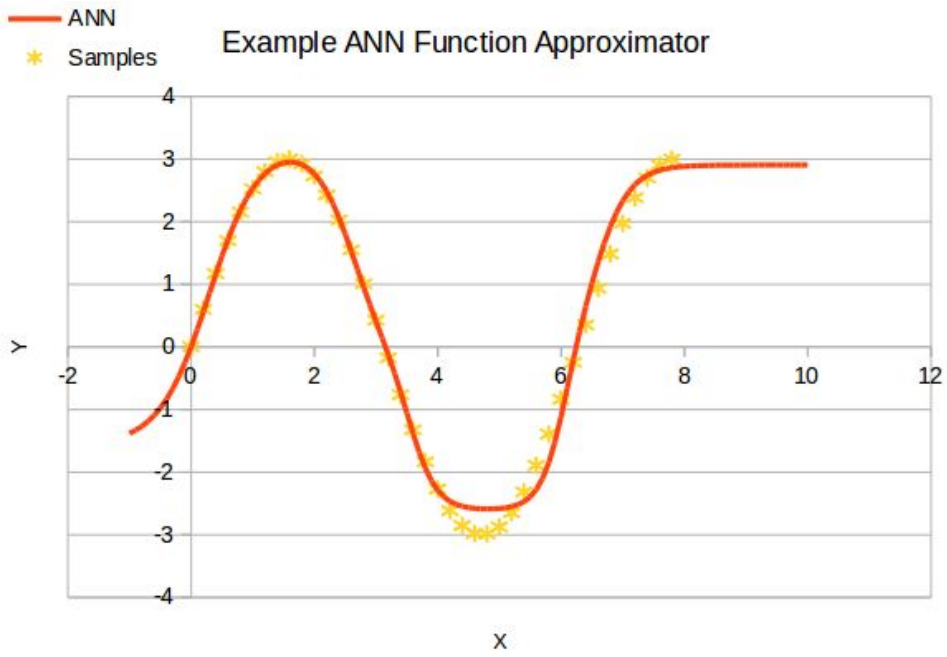
Gradient descent can run into local minima. Try many different initial weights

ANN as a function approximator

The sigmoid function constrains the output between 0 and 1. Use a linear transfer function at the output layer instead.

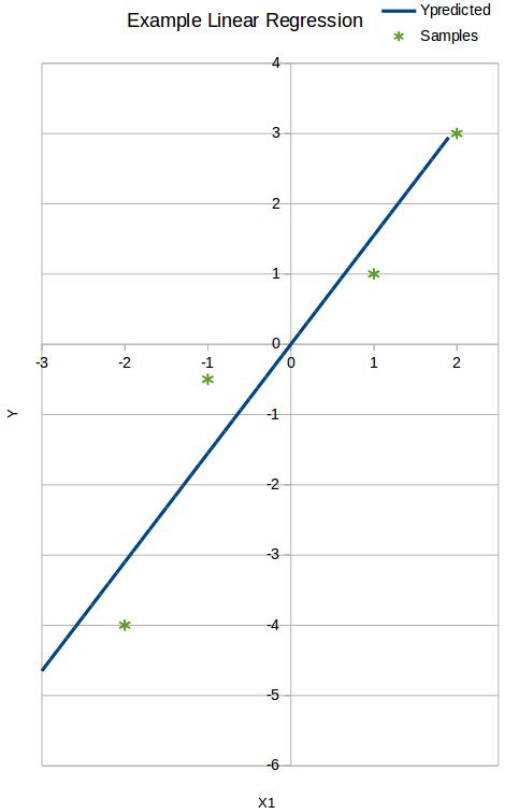
Network learns to use nonlinearities of the sigmoid function to fit the training data.

Image shows output from a network with 5 neurons in the input layer and no hidden layers.



Linear Regression Basics

y is the variable you are trying to predict
x_1 ... x_P are the variables you are using to predict y
You have N samples of y and x_1 ... x_P

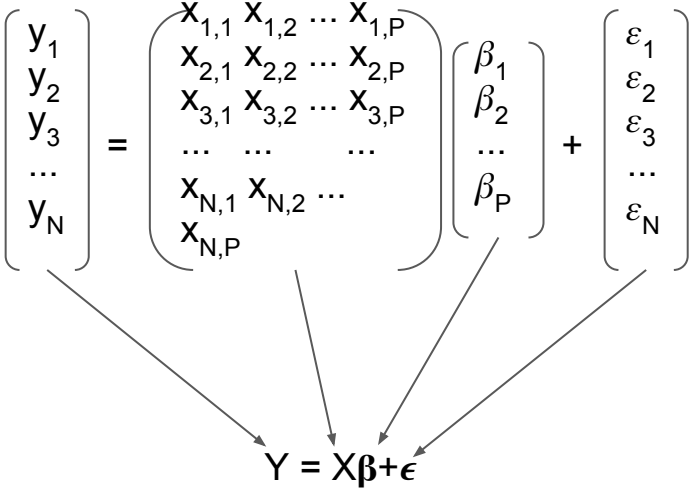


Linear Regression fits a linear model:

$y_{predicted} = \beta_1 x_1 + \beta_2 x_2 + ... + \beta_P x_P$

It finds coefficients $\beta_1 ... \beta_P$ that minimizes the sum of squared errors: $\sum_{i=0}^{N} (y_i - y_{predicted,i})^2$

Linear Regression Algorithm



Magic Formula:
$\beta = (X^T X)^{-1} X^T Y$

$O(P^2 N)$

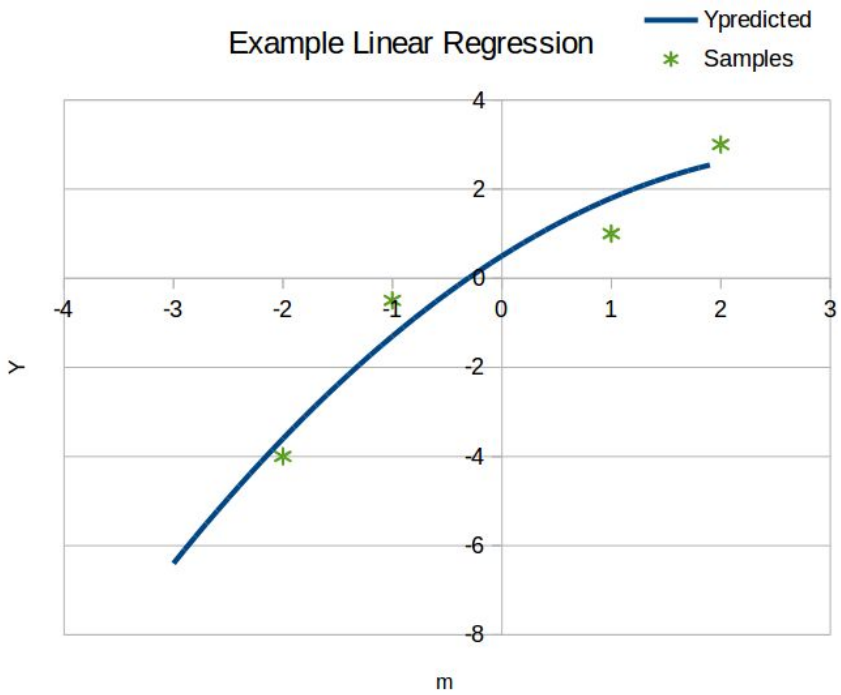
Linear Regression with more complex Models

x_1 ... x_P don't have to be the variables you measured, they can be a function of them.

Suppose m is the measured variable

To fit a parabola:

$x_1 = 1$ $x_2 = m$ $x_3 = m^2$



Local learning

- Estimate the form of the function near the point we're interested in (x_q)
- Model complexity is not fixed
- Local Algorithms:
  - Locally Weighted Linear Regression
  - Locally Weighted Projection Regression
  - Local Gaussian Process Regression
- Semi-local (combine both global and local methods):
  - Gaussian Mixture Model
  - Bayesian Committee Machine

Weighted Linear Regression

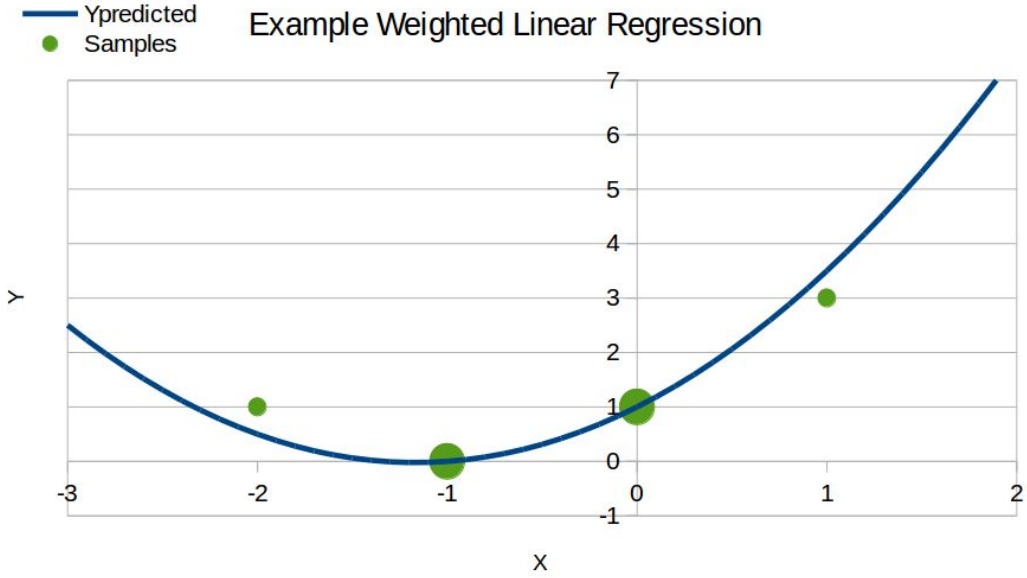
Minimize the *weighted* sum of squared errors: $\sum_{i=0}^{N} w_i(y_i - y_{predicted,i})^2$

$$W = \begin{pmatrix} w_1 & 0 & 0 & \dots & 0 \\ 0 & w_2 & 0 & \dots & 0 \\ 0 & 0 & w_3 & \dots & 0 \\ .. & .. & .. & .. & .. \\ 0 & 0 & 0 & \dots & w_N \end{pmatrix}$$

Magic Formula:

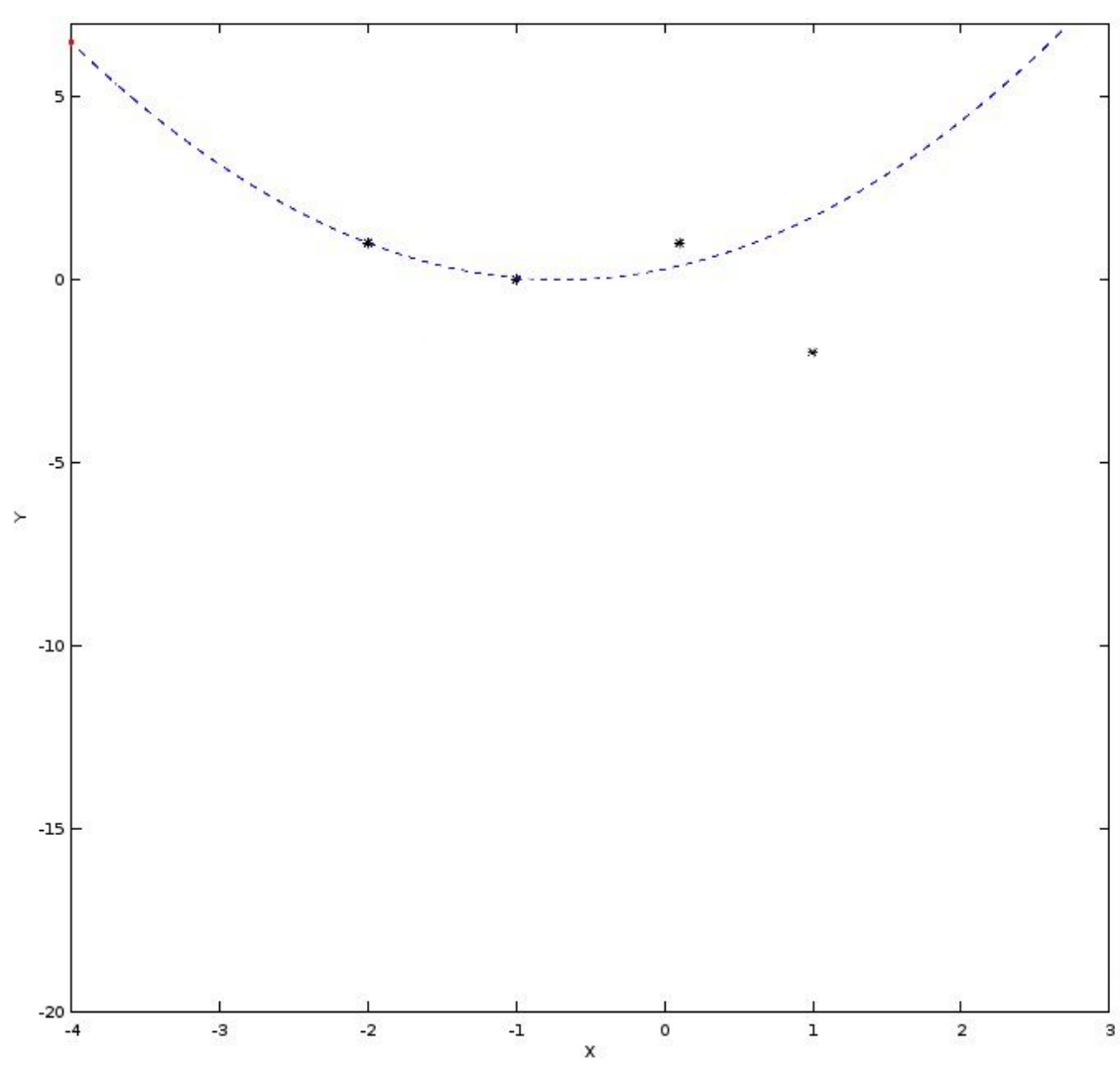
$\boldsymbol{\beta} = (X^T W X)^{-1} X^T W Y$

$\mathcal{O}(P^2 N)$



Useful for accomodating *Heteroskedasticity*: measurement variance

$w_i = 1/\sigma_i^2$



Locally Weighted Linear Regression

Weight samples according to how close they are to the point of interest.

Gaussian distribution of weights works well

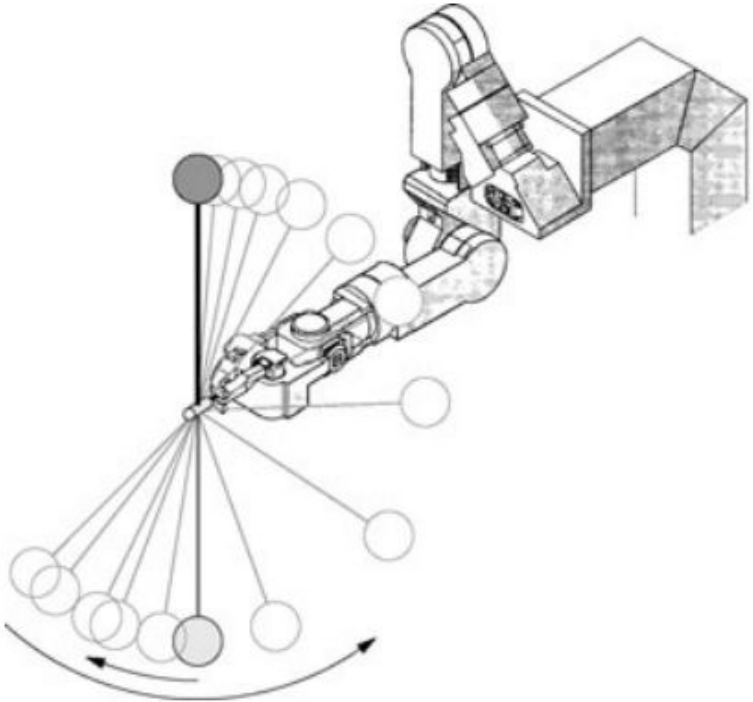
Problems with Local Learning Methods

- Need to determine a distance function $d(x_k,x_q) = (x_k - x_q) / h$
- Methods for choosing h:
  - Minimize cross validation error
  - Adaptive bandwidth selection
- Speed Optimization:
  - Partition space, and only consider points in the query point's partition
- Problems:
  - Notions of locality break down in high-dimensional space
    - Project data to lower dimensional space
    - Combine with a probabilistic regression to exploit its strengths in high dimensional space
  - Sensitive to noise

3 Examples of Ways Model Learning has been Applied to Robotics

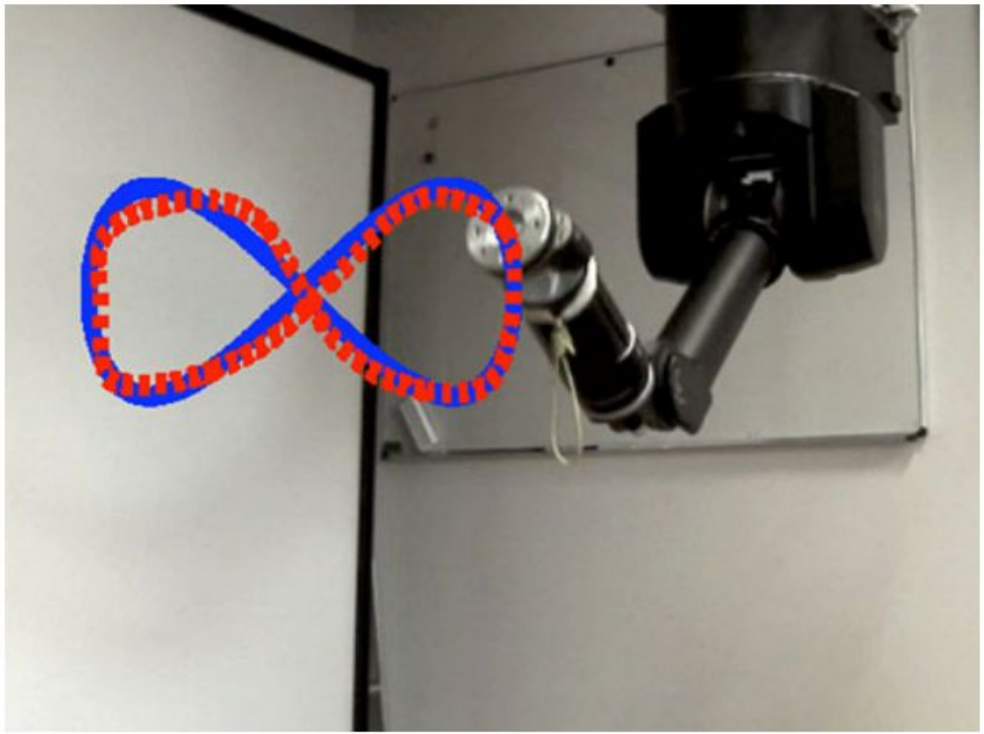
Simulation Based Optimization

Learn a forward model and use it for planning



Approximation Based Inverse Dynamics Control

Model learning accounts for nonlinearities that physics models don't account for.



Learning Operational Space Control

- Learn how to follow trajectories in operational space (not joint space)
- Normally this is ill-posed, but there are several ways this has been tackled
  - Local linearizations aren't ill-posed. Use local models
  - Learn the forward kinematics, then invert it, and combine it with an inverse dynamics model
  - Use a neural network to jointly learn the forward and inverse kinematics

Future Directions

Dealing with uncertainty in the environment
Transfer learning between tasks
Semi-supervised learning
Approximation based control needs more stability analysis
Learning nonunique mappings

Summary

- There are many benefits of robots that can learn how to control their body
- Controllers use a model of the system to determine what action to take
- Forward models are well-defined
- Inverse models have a simple policy
- Models can be trained directly from observations,
  or indirectly from error signals
- Learning a model is a regression problem
- Model learning in robotics has several challenges that general machine learning doesn't