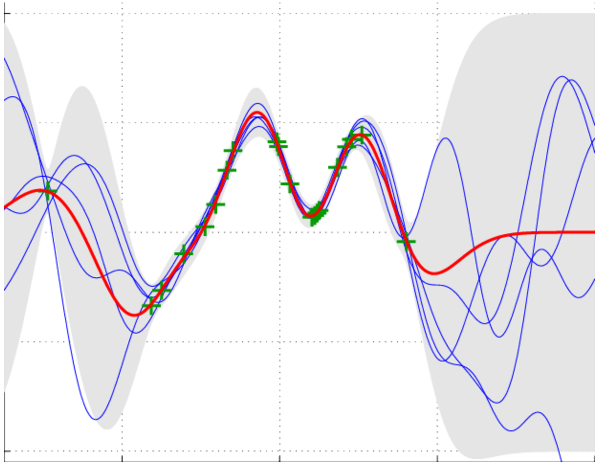
COMS W4733: Computational Aspects of Robotics

Lecture 26: Model Learning



Instructor: Tony Dear

Materials based on “Model Learning for Robot Control: A Survey” by D. Nguyen-Tuong and J. Peters

Robot Models

- We've assumed that we have knowledge of the **models** governing our robots, e.g. kinematics, transition, observation (even if noisy)
- Physics-based, empirical testing—all done offline and prior to robot operation

- Is this realistic / sufficient?

- Robot dynamics can be very *complex*, esp unconventional configurations
- Environments can be very *unstructured* and *stochastic*
- Robots, unlike many other AI systems, undergo *degradation* over time

Model Learning

- We assume transition and observation models of our robots as before:

$$\boldsymbol{s}_{k+1} = \boldsymbol{f}(\boldsymbol{s}_k, \boldsymbol{a}_k) + \boldsymbol{\epsilon}_f$$
$$\boldsymbol{y}_k = \boldsymbol{h}(\boldsymbol{s}_k, \boldsymbol{a}_k) + \boldsymbol{\epsilon}_h$$

- Problem: We may not necessarily know $\boldsymbol{f}$ and $\boldsymbol{h}$
- With both increasing computational power and complexity of robots, it is often desirable to **learn models from data** rather than derive them analytically

- *Parametric learning*: Try to fit open set of parameters to pre-defined models
- *Nonparametric learning*: No fixed model structure, try to adapt to data complexity

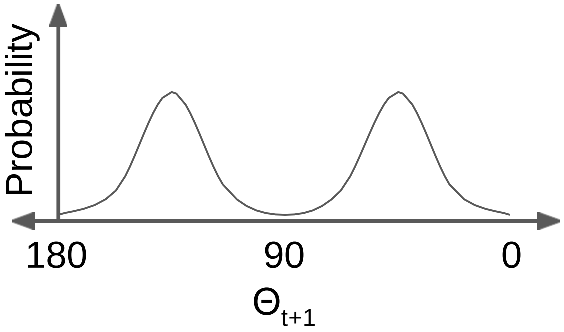
Model Learning

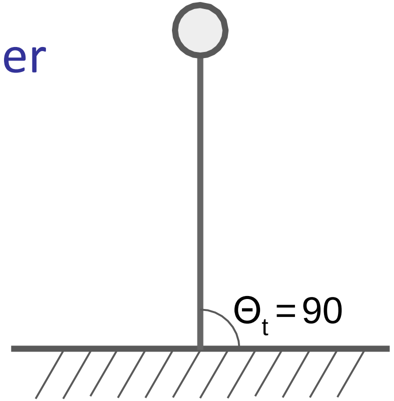
Model Type	Learning Architecture	Example Applications
Forward Model	Direct Modeling	Prediction, Filtering, Learning simulations, Optimization
Inverse Model	Direct Modeling, Indirect Modeling	Inverse dynamics control, Computed torque control, Feedback linearization control
Mixed Model	Direct Modeling (if invertible), Indirect Modeling, Distal-Teacher	Inverse kinematics, Operational space control, Multiple-model control
Multi-step Prediction Model	Direct Modeling	Planning, Optimization, Model predictive control, Delay compensation

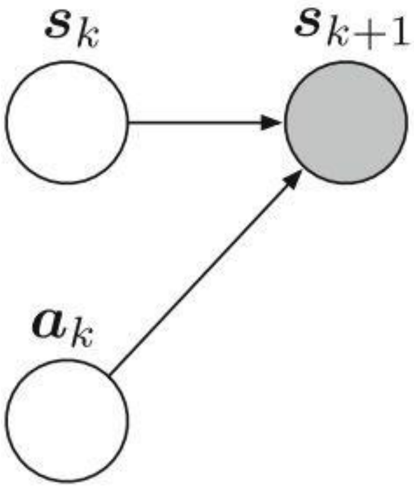
Forward Models

- **Forward model**: Given current state and action, predict next state
- Usually unique, correspond to causal relationships

- However, not always fully informative!
- E.g., unstable inverted pendulum
- Probabilistic model may be better







Forward Models

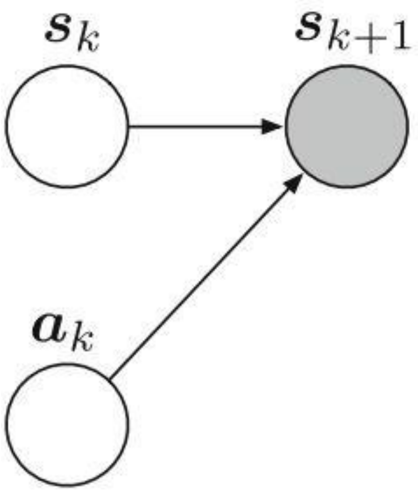
- Forward models by themselves can be useful for predictive control schemes

- **Model reference adaptive control** (MRAC)
- Policy based on model's prediction of future

$$\boldsymbol{\pi}(\boldsymbol{s}) = \arg\min_{\boldsymbol{a}} \left\| \boldsymbol{f}_{\text{forward}}(\boldsymbol{s}_t, \boldsymbol{a}) - \boldsymbol{s}_{t+1}^d \right\|$$

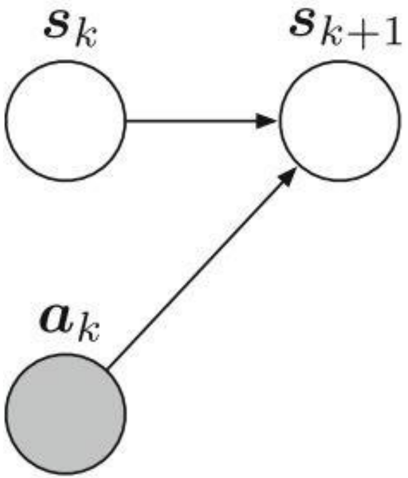
- **Model predictive control** (MPC)
- Optimal actions over a prediction horizon

$$\boldsymbol{\pi}(\boldsymbol{s}) = \arg\min_{\boldsymbol{a}_{t:t+N}} \sum_{k=t}^{t+N} F_{\text{cost}}\left(\boldsymbol{f}_{\text{forward}}(\boldsymbol{s}_k, \boldsymbol{a}_k) - \boldsymbol{s}_{k+1}^d\right)$$



Inverse Models

- Given current and future state, what is the action required to get there?
- Interpretation is often anticausal, may not have unique solutions
  - E.g., inverse kinematics
- Ill-posed models may require more constraints



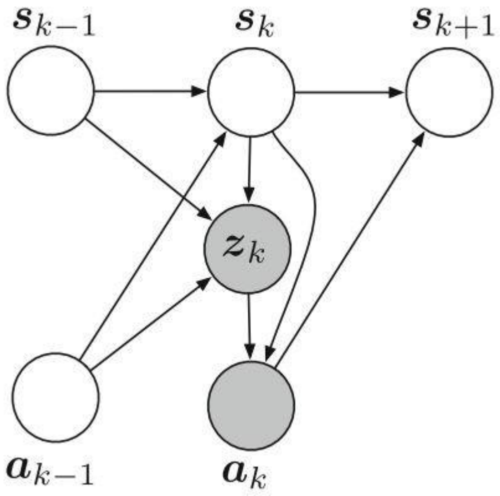
- Example of well-posed inverse model: inverse dynamics
- Predict torques required to move robot along trajectory

$$\boldsymbol{\pi}(\boldsymbol{s}) = \boldsymbol{f}_{\text{inverse}}(\boldsymbol{s}, \boldsymbol{s}^d)$$

Mixed Models

- If inverse model is ill-posed, combining with forward model to determine" solution validity is one solution
- Ex: Use forward kinematics to check solution from inverse kinematics
- Errors can be used to adjust inverse model

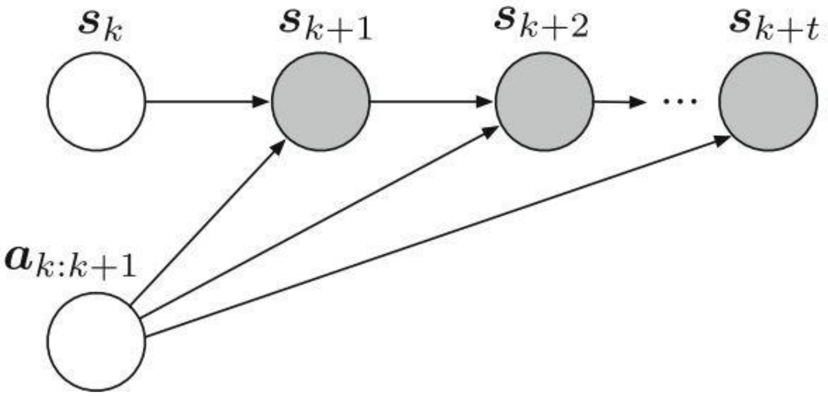
- Forward model gives us "latent" state $\boldsymbol{z}_k$
- $\boldsymbol{z}_k$ helps narrow down the best action $\boldsymbol{a}_k$ to get to $\boldsymbol{s}_{k+1}$
- Example of *distal teacher learning*



Operator Models

- We can often get more efficient policies if we can predict more steps into the future and employ actions based on them
- But lack of measurements can cause errors to accumulate
- Presence of noise or nonlinear dynamics

- Alternative solutions:
- Autoregressive models
- Nonparametric operator models



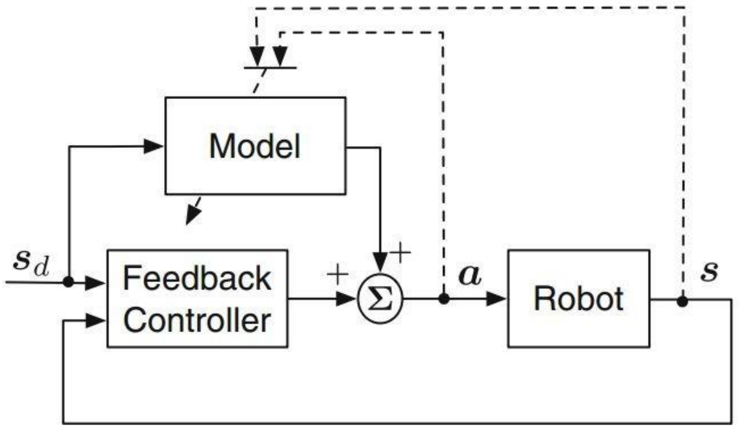
Learning Architectures

Model Type	Learning Architecture	Example Applications
Forward Model	Direct Modeling	Prediction, Filtering, Learning simulations, Optimization
Inverse Model	Direct Modeling, Indirect Modeling	Inverse dynamics control, Computed torque control, Feedback linearization control
Mixed Model	Direct Modeling (if invertible), Indirect Modeling, Distal-Teacher	Inverse kinematics, Operational space control, Multiple-model control
Multi-step Prediction Model	Direct Modeling	Planning, Optimization, Model predictive control, Delay compensation

Direct Modeling

- Learn observed inputs/outputs in a supervised fashion
- Easy to implement using regression techniques or neural nets
- Suitable for offline learning; if online, a feedback controller guides the robot

- Considerations:
- Parametric or non-parametric?
- How to generate sufficiently rich datasets?
- Can also learn a bunch of small local models

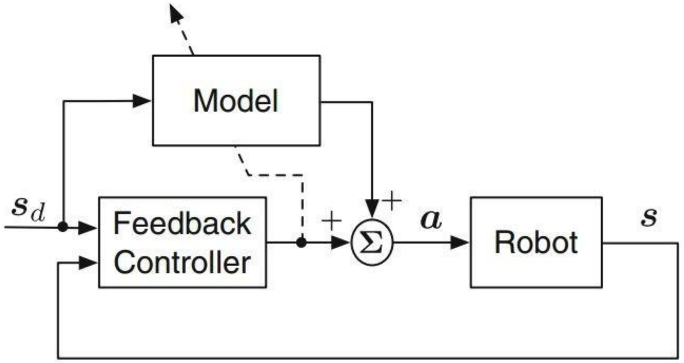


Indirect Modeling

- Direct modeling requires a well-define functional relationship
- If not (e.g. inverse kinematics), use feedback error to inform learning
- Idea: If model were learned perfectly, then controller output would be zero

- As model converges over time, inverse model will describe the model fully, and controller will have minimal effect

- Must be done online
- Model learns output solution for a specific goal

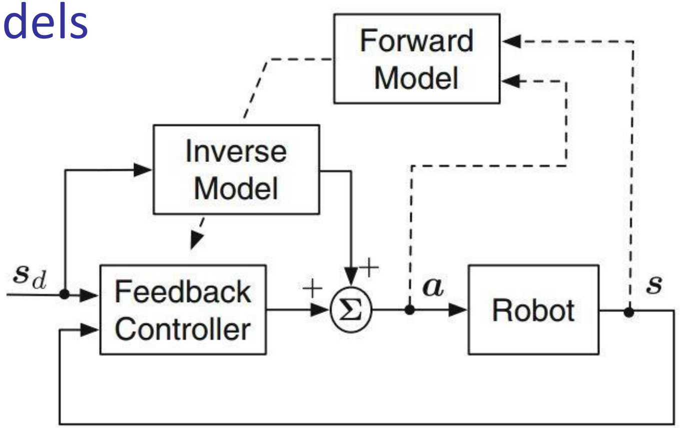


Distal-Teacher Learning

- Recall mixed models use both forward and inverse models
- Idea: Learn an inverse model and combine with controller, but simultaneously learn a forward model (*teacher*) that predicts the errors

- Helpful for learning global, rather than local, models
- Inverse model is still learned for a specific goal

- If learning is perfect:
- Composition of the models should be identity



Model Learning Difficulties

Data Challenges	Algorithmic Constraints	Real-World Challenges
High-dimensionality, Smoothness, Richness of data, Noise, Outliers, Redundant data, Missing data	Incremental updates, Real-time, Online learning, Efficiency, Large data sets, Prior knowledge, Sparse data	Safety, Robustness, Generalization, Interaction, Stability, Uncertainty in the environment

Data Challenges

- Robots are very high-dimensional!
- Usually can only explore parts of the data space associated with given task
- Need rich data, lots of exploration, injected artificial noise
- Dimensionality reduction of data is a common pre-processing step

- Robotics contains many non-smooth models (e.g. friction)
- Kernel methods for approximation, or switching between local models

- Redundant data over time, noise, outliers—need selective filtering, regularization

Algorithmic Constraints

- Often have massive amounts of data coming through sensors
- Need to be able to discern what to use or what is useful

- Need fast, real-time computations, or methods that can reduce data size
- Online learning makes robots more autonomous, but we have less complete models to work with

- May also want to incorporate prior knowledge, active learning by interacting with humans for data labeling

Real-World Challenges

- Learned models must be robust and reliable for usage in real situations
- Good feature selection is often key to removing unnecessary data

- Missing data due to difficult environments or imperfect sensors
- Probabilistic learning methods can help robot to infer missing information
- Predictions can also have associated uncertainties

- Non-stationary systems: Time-dependent dynamics, changing environments

Model Learning Methods

Method	Type	Mode	Online	Complexity	Learning Applications
Locally Weighted Projection Regression [172]	Local	Incremental	Yes	$\mathcal{O}(n)$	Inverse dynamics [134], Foothold quality model [60]
Local Gaussian Process Regression [105]	Local	Incremental	Yes	$\mathcal{O}(m^2)$	Inverse dynamics [105]
Gaussian Mixture Model [55]	Semi-Local	Batch	No	$\mathcal{O}(Mn)$	Human motion model [19]
Bayesian Comittee Machine [165]	Semi-Local	Batch	No	$\mathcal{O}(m^2n)$	Inverse dynamics [122]
Sparse Gaussian Process Regression [30]	Global	Incremental	Yes	$\mathcal{O}(n^2)$	Transition dynamics [128], Task model [46]
Gaussian Process Regression [142]	Global	Batch	No	$\mathcal{O}(n^3)$	Terrain model [117], State estimation model [67]
Support Vector Regression [138]	Global	Batch	No	$\mathcal{O}(n^2)$	ZMP control model [38], Grasp stability model [112]
Incremental Support Vector Machine [81]	Global	Incremental	Yes	$\mathcal{O}(n^2)$	Inverse dynamics [24]

Model Learning Methods

- Supervised or unsupervised?
- *Supervised*: Fast and robust model approximation
- Requires labeled training data as ground truth

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{\epsilon}$$

- *Unsupervised*: Only requires input data; outputs inferred from observations
- Often means that we require more exploration for data richness

- *Global* regression: Use all available data to construct global model
- *Local* regression: Estimate model around local query points

Global Regression

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{\epsilon} \quad \Longrightarrow \quad \boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{\theta}^T \boldsymbol{\phi}(\boldsymbol{x})$$

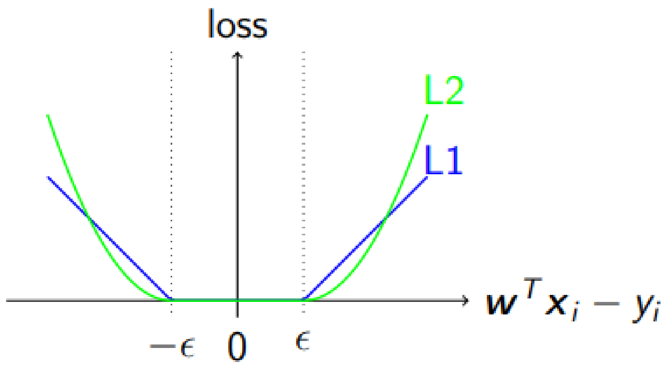
- Model the unknown function as an inner product between weights $\boldsymbol{\theta}$ and a nonlinear projection $\boldsymbol{\phi}$ of the input $\boldsymbol{x}$

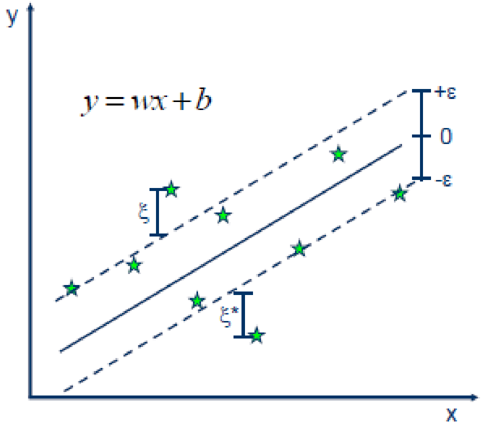
- Parametric models: Size of $\boldsymbol{\theta}$ fixed (e.g. neural net architecture)
- Nonparametric models: Weights can increase with training data

- Tradeoff in model complexity: Want a simple but also good-fitting model
- $\boldsymbol{\theta}$ can be expanded in training data but also regularized; we can also place probabilistic distributions on $\boldsymbol{\theta}$

Support Vector Regression

- **Support vector machine** (SVM): Binary linear classifier via separating margin
- **Support vector regression** (SVR): Similar but useful for continuous contexts
- Idea: Only errors outside margins contribute to overall loss
- More robust model, less overfitting





- Minimize:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right)$$

- Constraints:

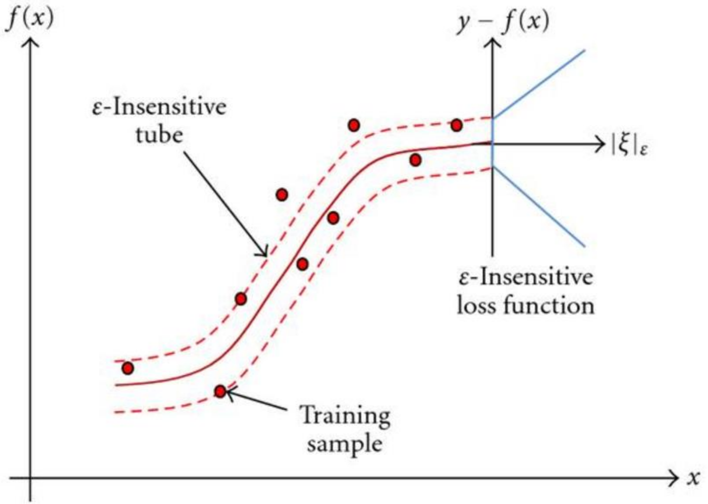
$$y_i - wx_i - b \le \varepsilon + \xi_i$$

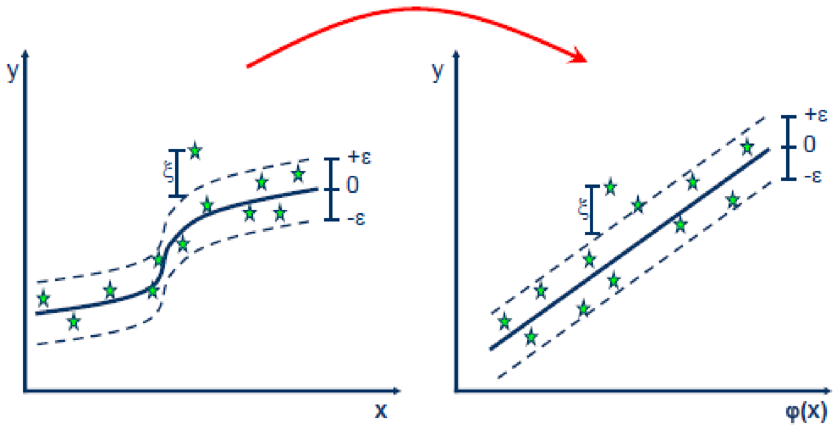
$$wx_i + b - y_i \le \varepsilon + \xi_i^*$$

$$\xi_i, \xi_i^* \ge 0$$

Support Vector Regression

- Kernel trick allows for learning nonlinear models as well
- Choice of kernel allows us to inject prior knowledge into the model





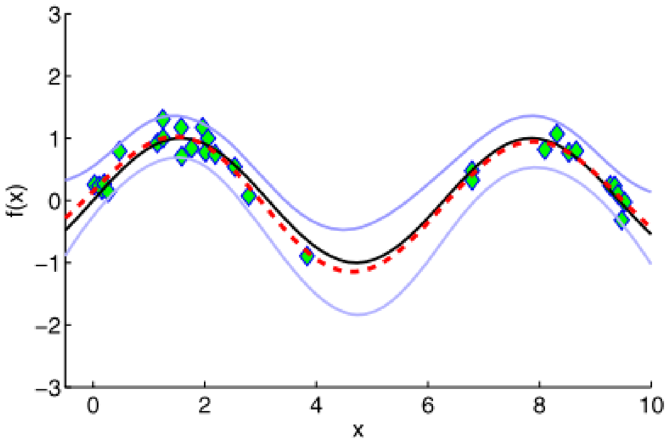
Gaussian Processes

- **Gaussian process**: A collection of random variables such that any subset of them is jointly Gaussian; a distribution over *functions*
- Non-parametric distribution over *all possible functions* consistent with data

- Functions can be shaped with different kernels
- Predictions depend on nearby data points

- Ex: Squared exponential kernel

$$k(\boldsymbol{x}, \boldsymbol{x}') = \sigma^2 \exp\left(\frac{-\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{2\tau^2}\right)$$



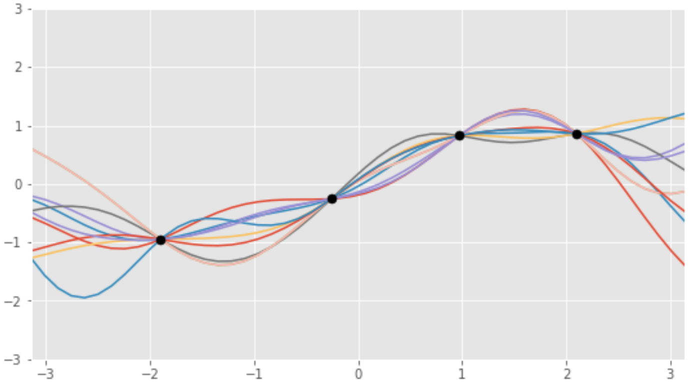
J. Ko and D. Fox (2009)

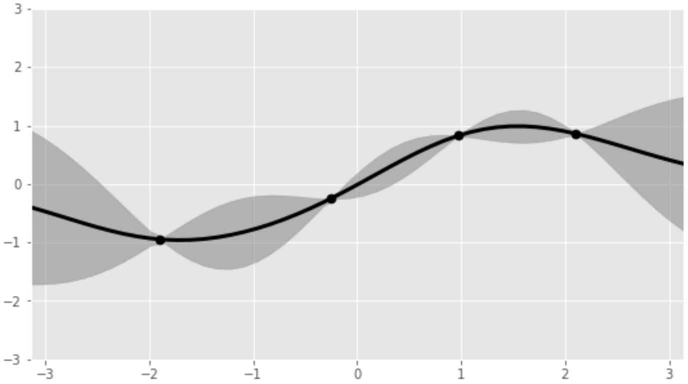
Gaussian Process Regression

- For classification, data is represented as sample from a multivariate Gaussian

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{y}^{*} \end{bmatrix} \sim N\left(\mathbf{0}, \begin{bmatrix} \boldsymbol{K} & \boldsymbol{K}_{*}^{T} \\ \boldsymbol{K}_{*} & \boldsymbol{K}_{**} \end{bmatrix}\right) \Rightarrow \boldsymbol{y}^{*} \sim N(\boldsymbol{\mu}_{*}, \boldsymbol{\Sigma}_{*})$$

- Presence of data generally narrows down distributions; lack of data leads to higher uncertainty





Local Learning

- We only want to consider data in a neighborhood around a query point $\boldsymbol{x}_q$
- An example cost function using n local data points:

$$J = \sum_{k=1}^{n} w\left(\left(\boldsymbol{x}_k - \boldsymbol{x}_q\right)^T \boldsymbol{D}\left(\boldsymbol{x}_k - \boldsymbol{x}_q\right)\right) \left\|\boldsymbol{y}_k - \hat{\boldsymbol{f}}(\boldsymbol{x}_k)\right\|^2$$

- Neighborhood function w controlled by distance metric $\boldsymbol{D}$
- Local model $\hat{\boldsymbol{f}}$ around $\boldsymbol{x}_q$; may be estimated through minimization of the cost
- Different methods for estimating $\boldsymbol{D}$, e.g. cross-validation

- Very common to partition input spaces, fewer requirements on smoothness and regularization than global methods

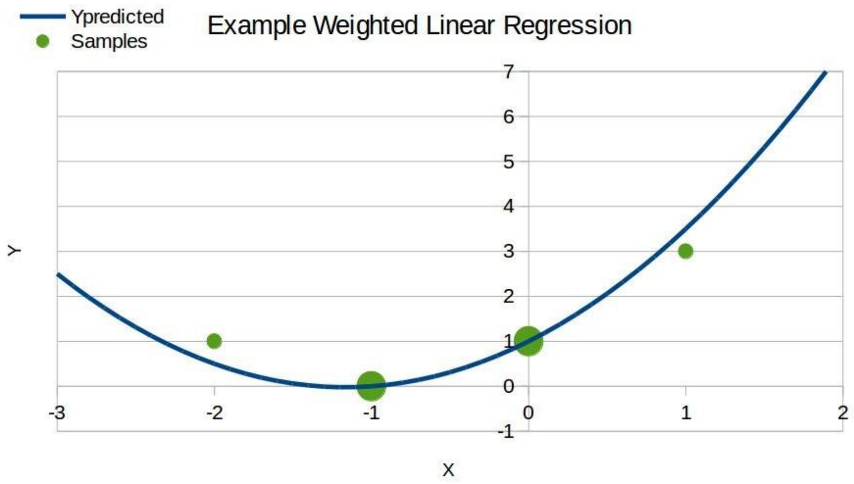
Weighted Linear Regression

- Many familiar regression methods can be augmented with weights to emphasize data around a particular query
- **Weighted linear regression**: Samples all get weights depending on how close they are; loss function pays more attention to them

$$\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{\epsilon}$$

- Loss function: $\sum_{i=0}^{N} w_i (y_i - \hat{y})^2$

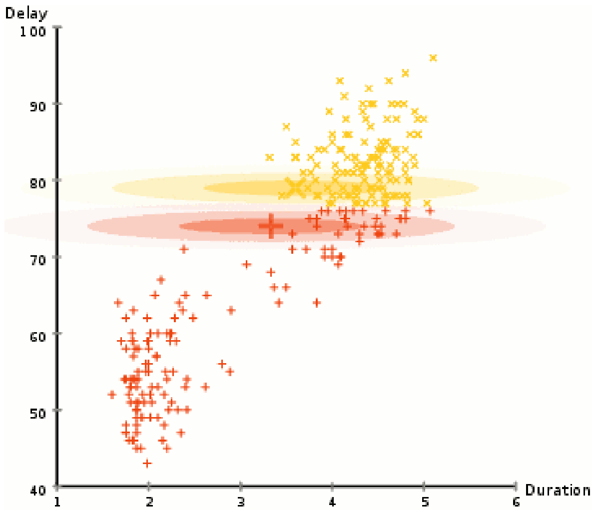
$$\widehat{\boldsymbol{\beta}} = (\boldsymbol{X}^T \boldsymbol{W} \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{W} \boldsymbol{y}$$



Gaussian Mixture Models

- Idea: Multiple Gaussians in different regions of the data space
- Soft version of k-means—each cluster now has an associated mean and covariance, and data is generated depending on each prior

$$p(\boldsymbol{x}) = \sum_{i=1}^{K} \phi_i N(\boldsymbol{x}|\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$$



- As with k-means, no closed-form solution!
- GMMs can be learned using EM algorithm

Summary

- Different types of models and learning paradigms depending on problem

- Forward and some inverse models can be directly fed into supervised learning
- Other inverse models can be learned via feedback controller outputs
- Distal teacher learning for both forward and inverse simultaneously

- Learning method considerations: Global vs local, parametric vs nonparametric, online vs offline, incremental vs batch
- Many can be boiled down to some variation of regression