CSE-571
Robotics

Gaussian Distributions
Regression
Gaussian Processes

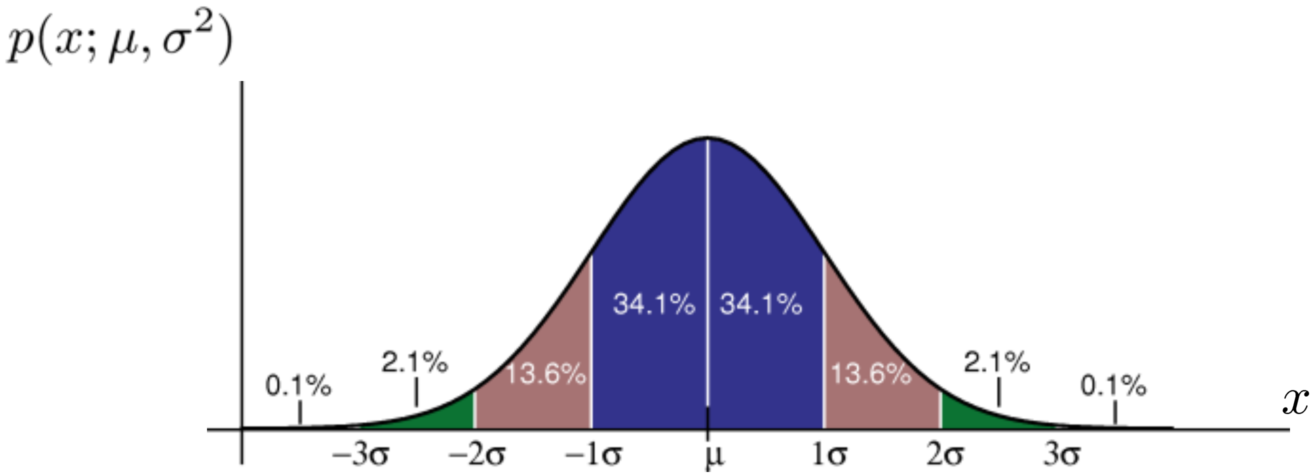
Slide from Pieter Abbeel

Gaussians (1D)

- Gaussian with mean (μ) and standard deviation (σ)

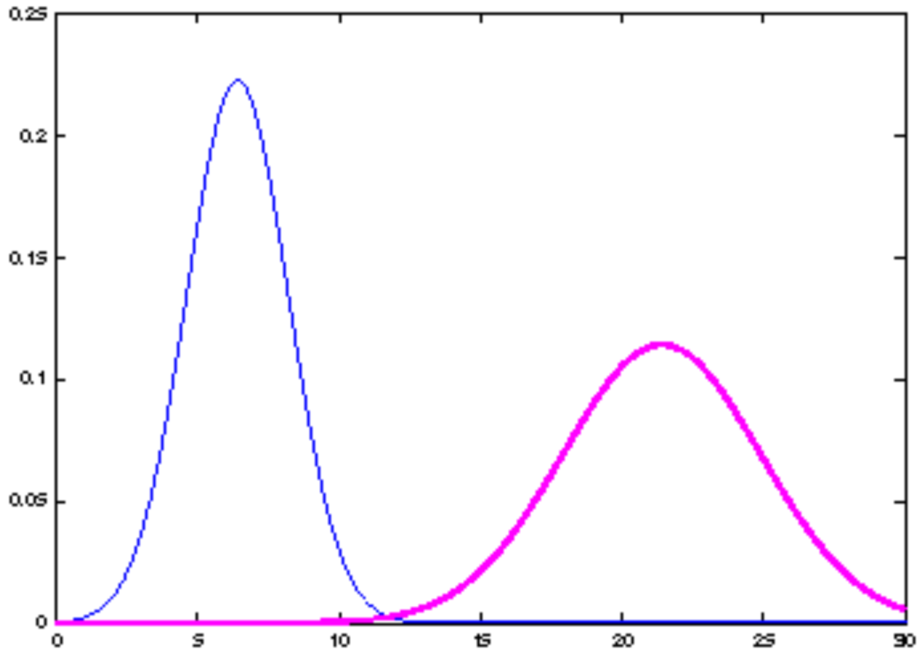
$$X \sim \mathcal{N}(\mu, \sigma^2)$$

$$p(x; \mu, \sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} \exp(-\frac{(x-\mu)^2}{2\sigma^2})$$



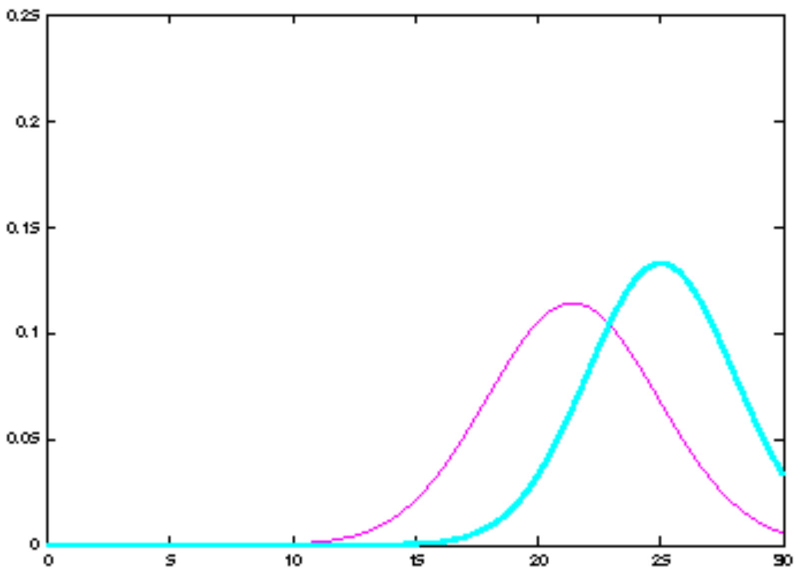
Properties of Gaussians

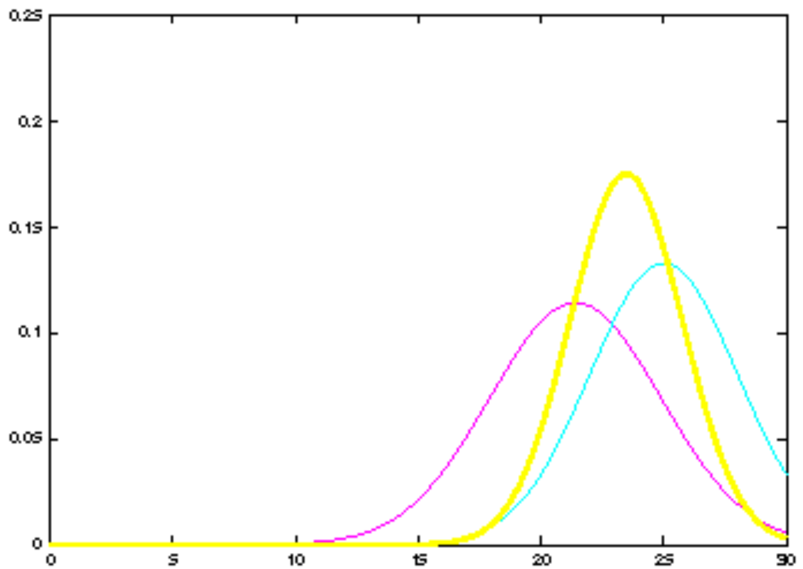
$$\left. \begin{array}{l} X \sim N(\mu, \sigma^2) \\ Y = aX + b \end{array} \right\} \Rightarrow \quad Y \sim N(a\mu + b, a^2\sigma^2)$$



Properties of Gaussians

$$\left.\begin{array}{l} X_1 \sim N(\mu_1, \sigma_1^{\ 2}) \\ X_2 \sim N(\mu_2, \sigma_2^{\ 2}) \end{array}\right\} \Rightarrow p(X_1)\cdot p(X_2) \sim N\left(\frac{\sigma_2^{\ 2}}{\sigma_1^{\ 2}+\sigma_2^{\ 2}}\mu_1 + \frac{\sigma_1^{\ 2}}{\sigma_1^{\ 2}+\sigma_2^{\ 2}}\mu_2, \quad \frac{1}{\sigma_1^{\ -2}+\sigma_2^{\ -2}} \right)$$





Picture from [Bishop: Pattern Recognition and Machine Learning, 2006]

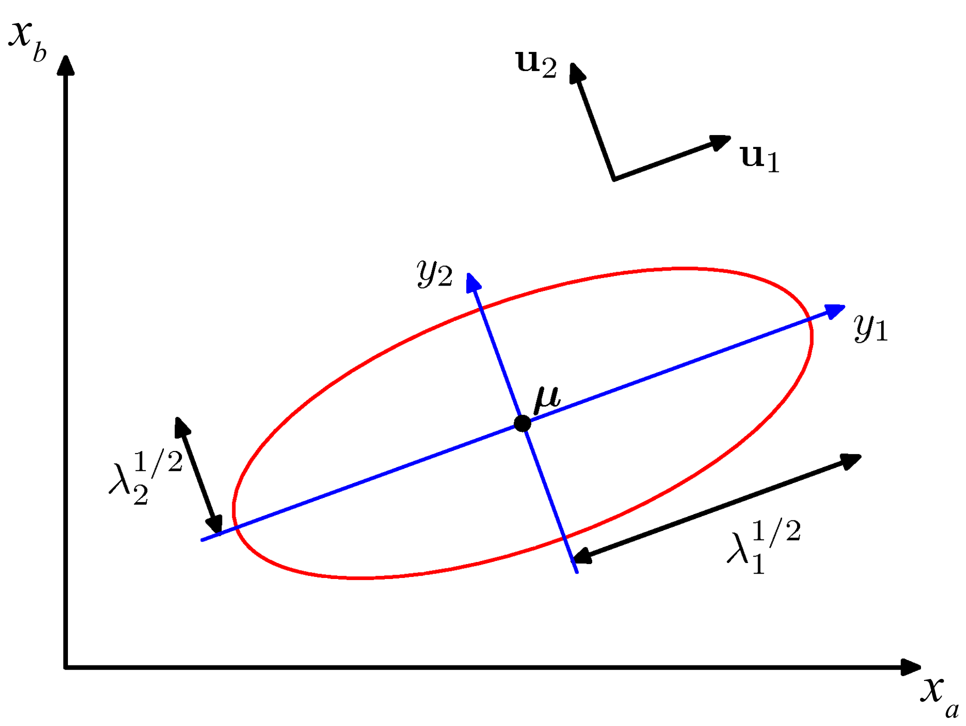
Gaussians (2D)

$$p(\mathbf{x}) = \mathrm{N}(\mu, \Sigma)$$

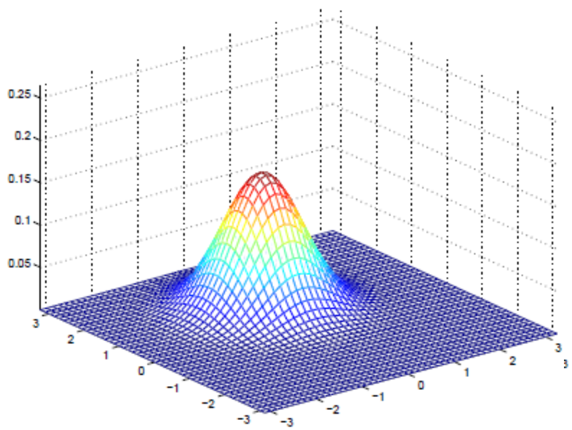
$$\mathbf{x} = \begin{pmatrix} x_a \\ x_b \end{pmatrix}, \quad \mu = \begin{pmatrix} \mu_a \\ \mu_b \end{pmatrix}$$

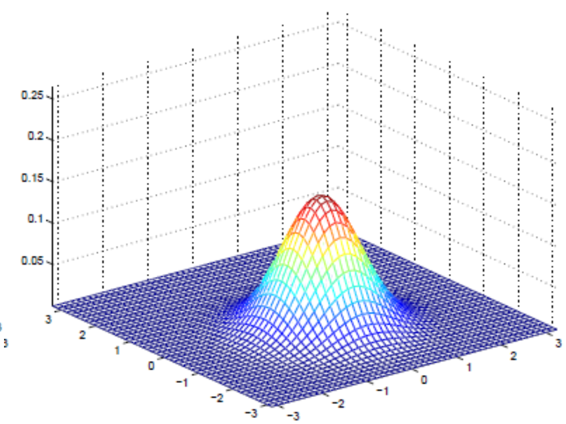
$$\Sigma = \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix}$$

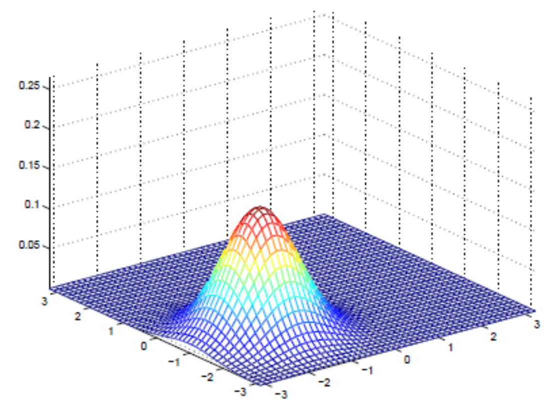
$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2} \left|\Sigma\right|^{1/2}} e^{-\frac{1}{2}(\mathbf{x}-\mu)^T \Sigma^{-1} (\mathbf{x}-\mu)}$$



2D examples





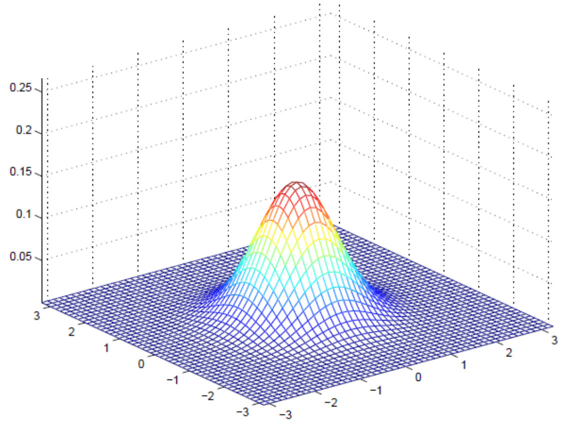


- μ = [1; 0]
- Σ = [1 0; 0 1]

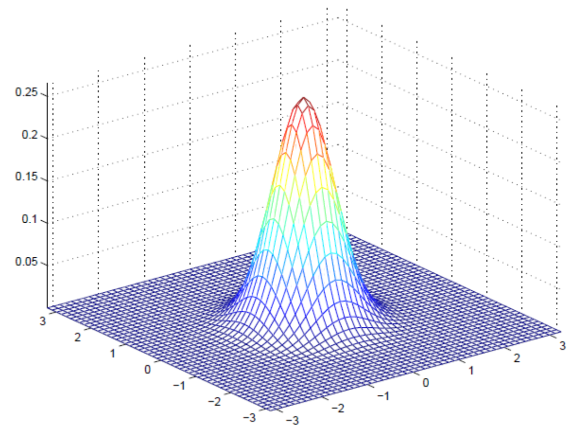
- μ = [-.5; 0]
- Σ = [1 0; 0 1]

- μ = [-1; -1.5]
- Σ = [1 0; 0 1]

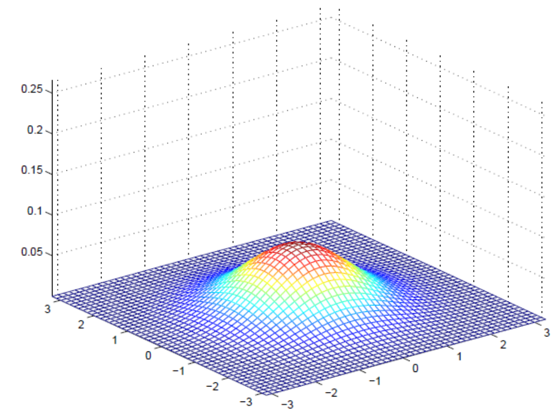
2D examples



- μ = [0; 0]
- Σ = [1 0 ; 0 1]



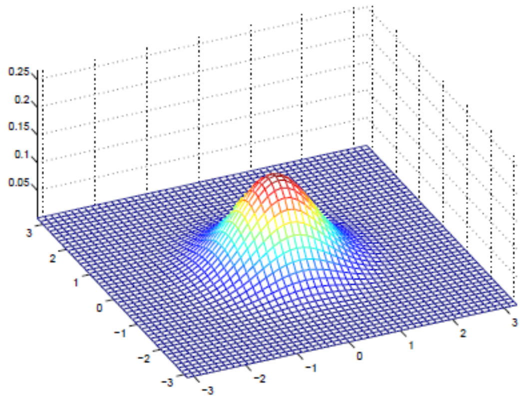
- μ = [0; 0]
- Σ = [.6 0 ; 0 .6]

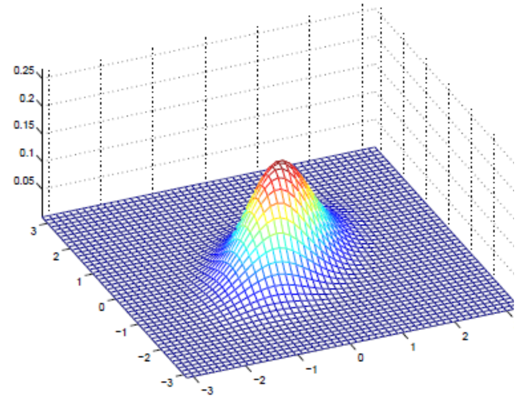


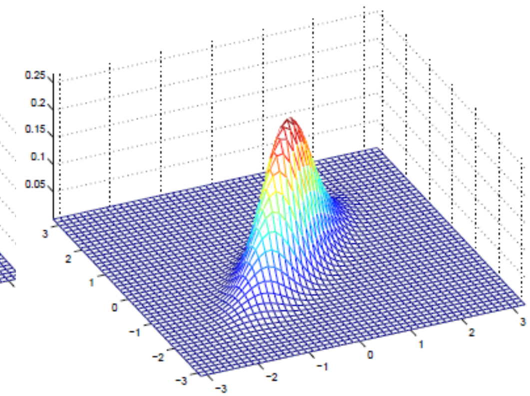
- μ = [0; 0]
- Σ = [2 0 ; 0 2]

Slide from Pieter Abbeel

2D examples



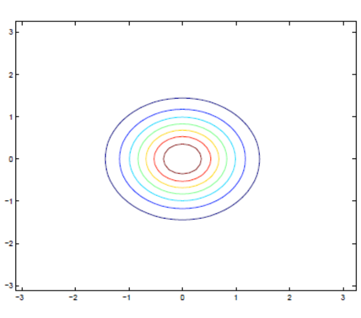


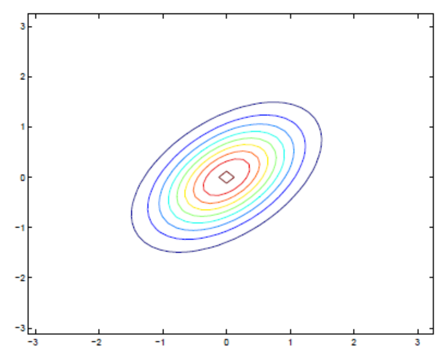


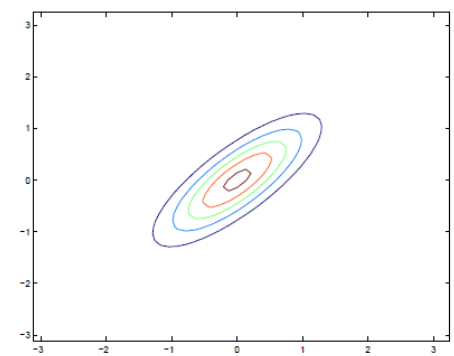
- $\mu = [0; 0]$
- $\Sigma = [1 \;\; 0; 0 \;\; 1]$

- $\mu = [0; 0]$
- $\Sigma = [1 \;\; 0.5; 0.5 \;\; 1]$

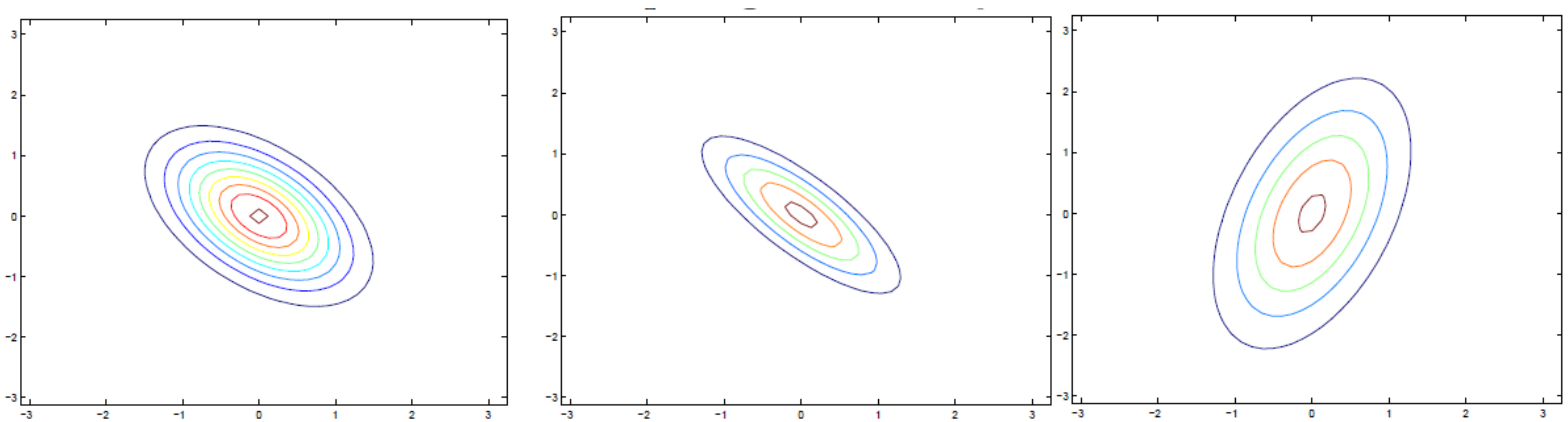
- $\mu = [0; 0]$
- $\Sigma = [1 \;\; 0.8; 0.8 \;\; 1]$







2D examples



- $\mu = [0; 0]$
- $\Sigma = [1\ \ -0.5\ ;\ -0.5\ \ 1]$

- $\mu = [0; 0]$
- $\Sigma = [1\ \ -0.8\ ;\ -0.8\ \ 1]$

- $\mu = [0; 0]$
- $\Sigma = [\,1\ \ 0.8\ ;\ 0.8\ \ 3]$

Pictures from [Bishop: PRML, 2006]

Marginalization / Conditioning

- Marginalizing joint distribution results in a Gaussian

$$p\left(\begin{bmatrix} x_a \\ x_b \end{bmatrix}\right) = N\left(\begin{bmatrix} \mu_a \\ \mu_b \end{bmatrix}, \begin{bmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{bmatrix}\right) \quad p(x_a) = \int p(x_a, x_b)\, dx_b$$

$$p(x_a) = N(\mu_a, \Sigma_{aa})$$

- Conditioning also leads to a Gaussian

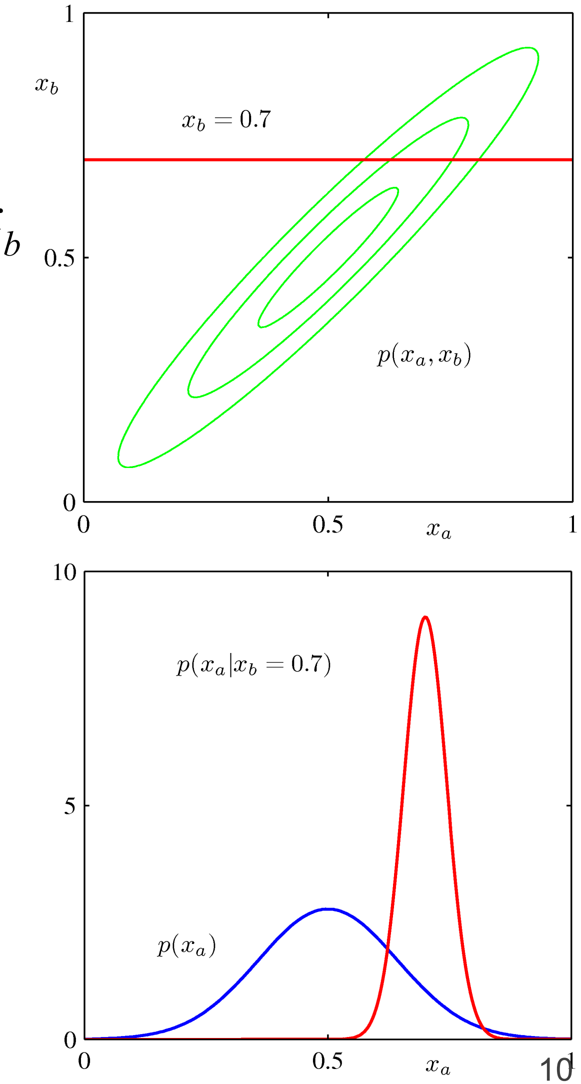
$$p(x_a \mid x_b) = N(\mu_{a|b}, \Sigma_{a|b})$$

$$\mu_{a|b} = \mu_a + \Sigma_{ab}\Sigma_{bb}^{-1}(x_b - \mu_b)$$

Cross co-variance **Prior Variance (b)** **Observed value** **Prior mean (b)**

$$\Sigma_{a|b} = \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}$$

Prior Variance (a) **Shrink term (>= 0)**



Regression

Regression

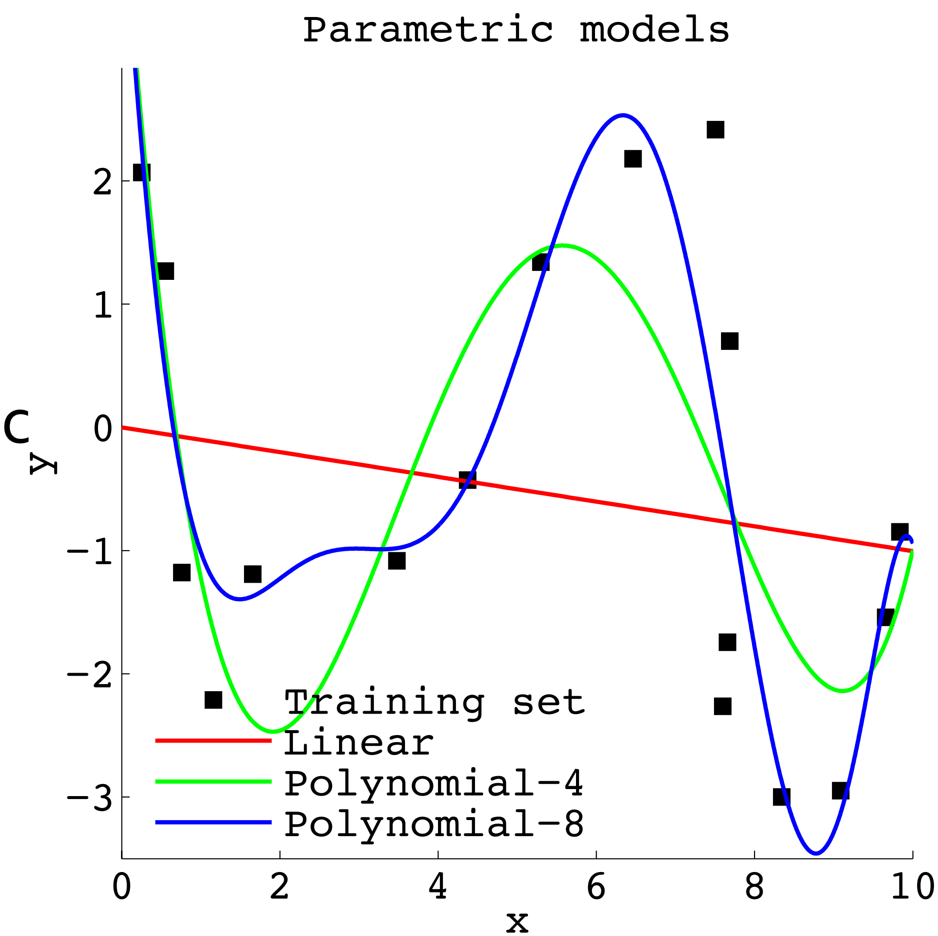
- Modeling the relationship between real-valued variables in data
  - Sensor models, dynamics models, stock market etc

- Two broad classes of models:
  - **Parametric:**
    - Learn a model of the data, use model to make new predictions
    - *Eg:* Linear, Non-linear, Neural Networks etc.
  - **Non-Parametric:**
    - Keep the data around and use it to make new predictions
    - *Eg:* Nearest Neighbor methods, Locally Weighted Regression, Gaussian Processes etc.

Example - Parametric models

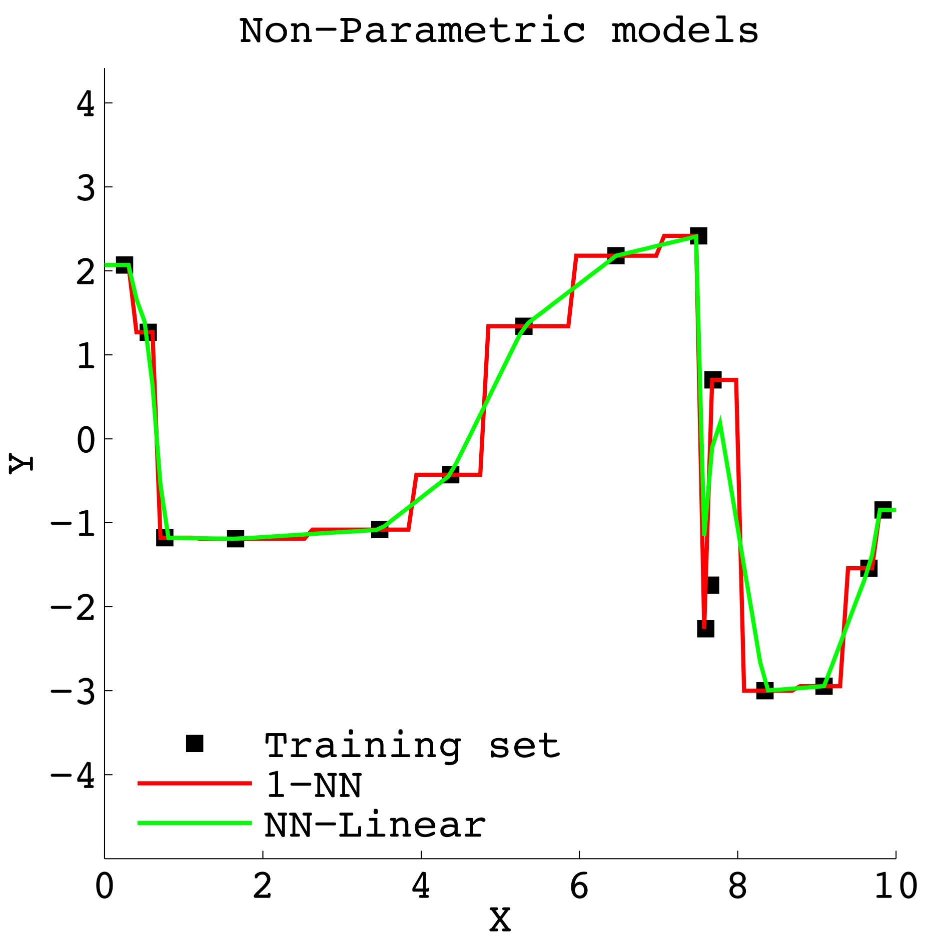
- Idea: Summarize data using a learned model:
  - Linear, Polynomial
  - Neural Networks etc

- Computationally efficient, tradeoff complexity vs generalization



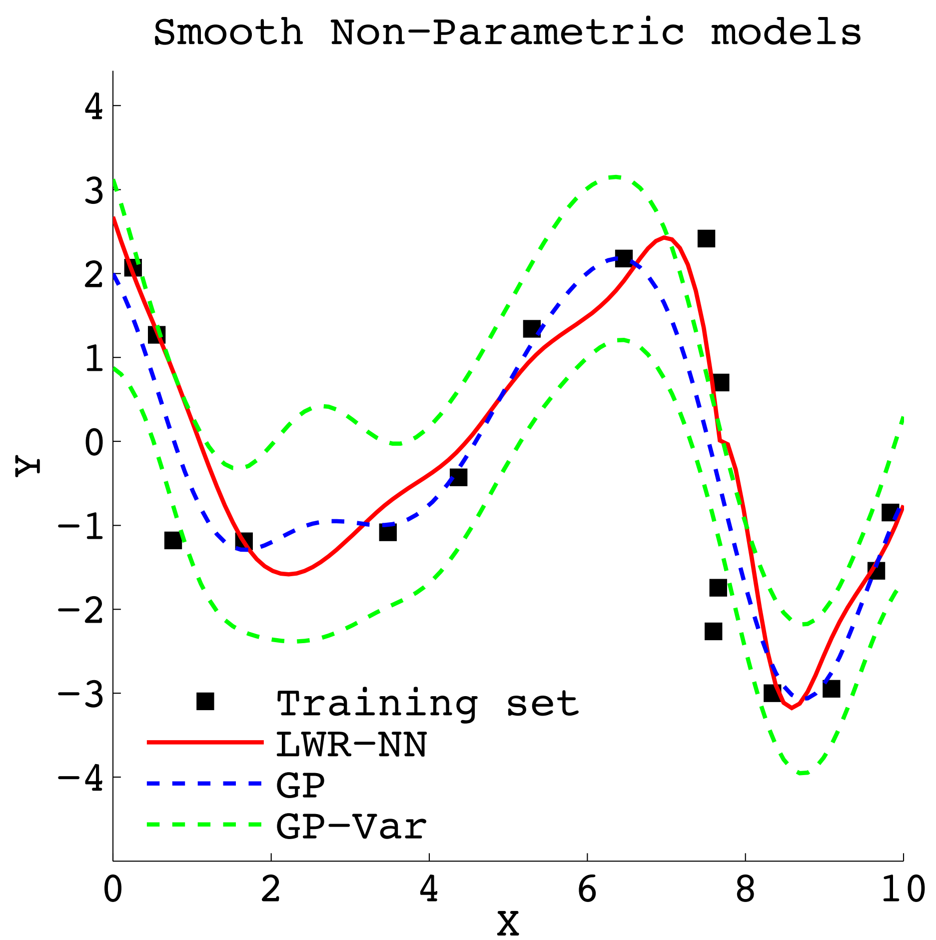
Example – Nearest Neighbor methods

- Idea: Use nearest neighbor's prediction (with some interpolation)
  - Non-parametric, keeps all data
  - Ex: 1-NN, NN with linear interpolation
- Easy. Needs lot of data
  - Best you can do in limit of infinite data
- Computationally expensive in high dimensions



Example: Smooth Non-Parametric models

- Idea: Interpolate based on "close" training data
  - Closeness defined using a "kernel" function
  - Test output is a weighted interpolation of training outputs
  - Locally Weighted Regression, Gaussian Processes
- Can model arbitrary (smooth) functions
  - Need to keep around some (maybe all) training data



Gaussian Process (GP) Regression

High-level Idea of GPs

- Non-parametric regression model
- Distribution over functions
- Fully specified by training data, mean and covariance functions
- Covariance given by "kernel" which measures distance of inputs in kernel space

Formal definition

- Given, inputs (x) and targets(y):

$$D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\} = (\mathbf{X}, \mathbf{y})$$

- GPs model the targets as a noisy function of the inputs:

$$y_i = f(\mathbf{x}_i) + \varepsilon;\ \varepsilon \sim N(0, \sigma_n^2)$$

- Formally, a GP is a collection of random variables, any finite number of which have a ***joint Gaussian*** distribution:

$$f(x) \sim GP(m(x), k(x, x'))$$

$$m(x) = E[f(x)]$$

$$k(x, x') = E[(f(x) - m(x))(f(x') - m(x'))]$$

Formal definition

- Given a (finite) set of inputs (X), GP models the outputs (y) as jointly Gaussian:

Noise

$$P(y \mid X) = N(m(X), K(X,X) + \sigma_n^2 I)$$

$$m = \begin{pmatrix} m(x_1) \\ m(x_2) \\ \vdots \\ \\ m(x_n) \end{pmatrix} \mathbf{K} = \begin{pmatrix} k(x_1,x_1) & \dots & k(x_1,x_n) \\ k(x_2,x_1) & & \\ \vdots & k(x_i,x_i) & \vdots \\ & & \\ k(x_n,x_1) & \dots & k(x_n,x_n) \end{pmatrix}$$

- Usually, we assume zero-mean prior
  - Can define other mean functions (constant, polynomials etc)

Covariance matrix - Kernel

- Covariance matrix (K) is defined through the "kernel" function:
  - Specifies covariance of the outputs as the function of inputs

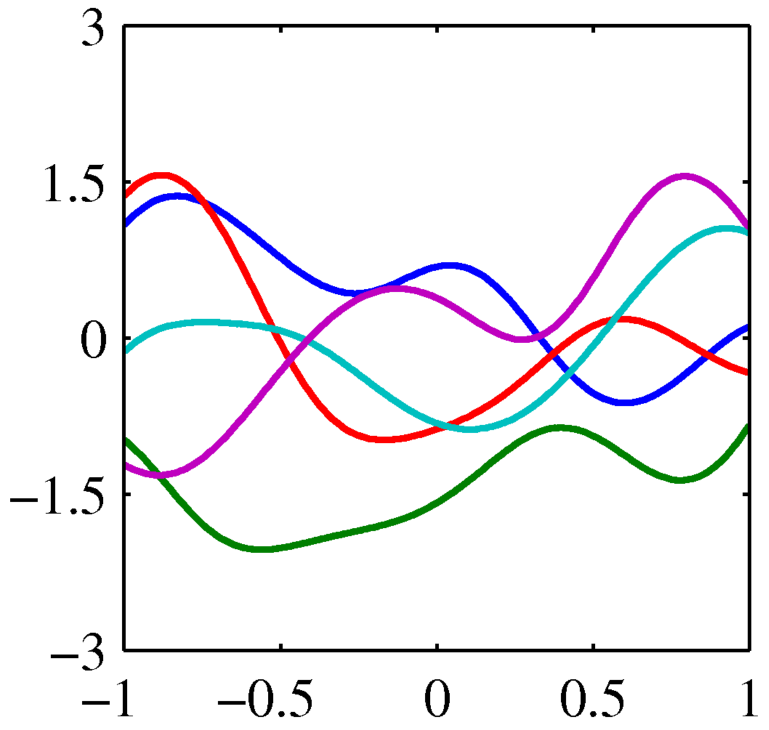
- Example: Squared Exponential Kernel
  - Covariance proportional to distance in input space
  - Similar input points will have similar outputs

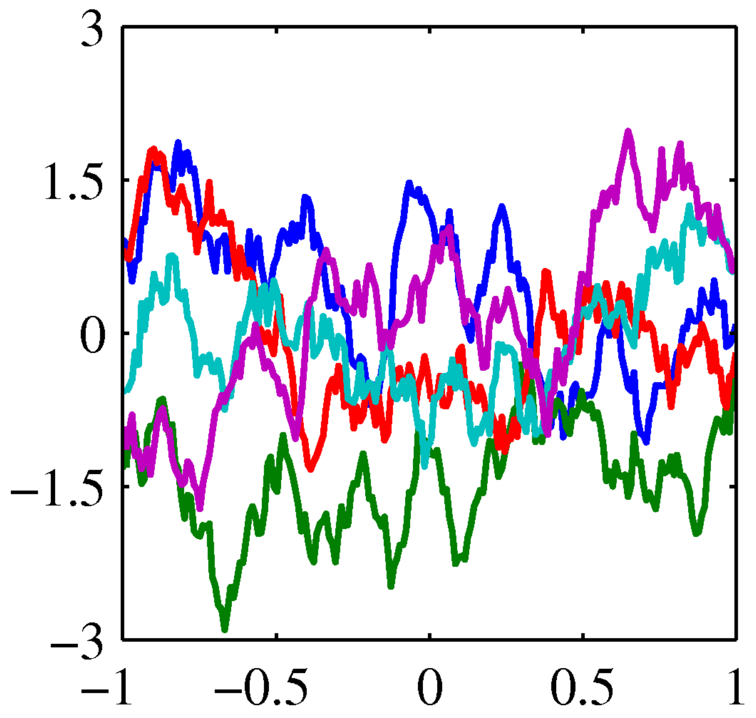
$$k(x,x') = \sigma_f^2 \, e^{-\frac{1}{2}(x-x')W(x-x')^T}$$

Functions Sampled from Prior

- GP prior: Outputs jointly zero-mean Gaussian:

$$P(\mathbf{y} \mid \mathbf{X}) = \mathrm{N}(\mathbf{0}, \mathbf{K} + \sigma_n^2 \mathbf{I})$$





GP Prediction – Gaussian Conditioning

- Training data: $D = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\} = (\mathbf{X}, \mathbf{y})$
- Test pair (y unknown): $\{x_*, y_*\}$
- GP outputs are jointly Gaussian:

$$P(y, y_* \mid X, x_*) = N(\mu, \Sigma); \quad P(y \mid X) = N(0, \mathrm{K} + \sigma_n^2 I)$$

- Conditioning on y:

$$P(y_* \mid \mathbf{x}_*, \mathbf{y}, \mathbf{X}) = N\left(\mu_*, \sigma_*^2\right)$$

$$\mu_* = k_*^T \left(\mathbf{K} + \sigma_n^2 \mathbf{I}\right)^{-1} \mathbf{y}$$

$$\sigma_*^2 = k_{**} - k_*^T \left(\mathbf{K} + \sigma_n^2 \mathbf{I}\right)^{-1} k_*$$

$$k_*[i] = k(\mathbf{x}_*, \mathbf{x}_i); \; k_{**} = k(\mathbf{x}_*, \mathbf{x}_*)$$

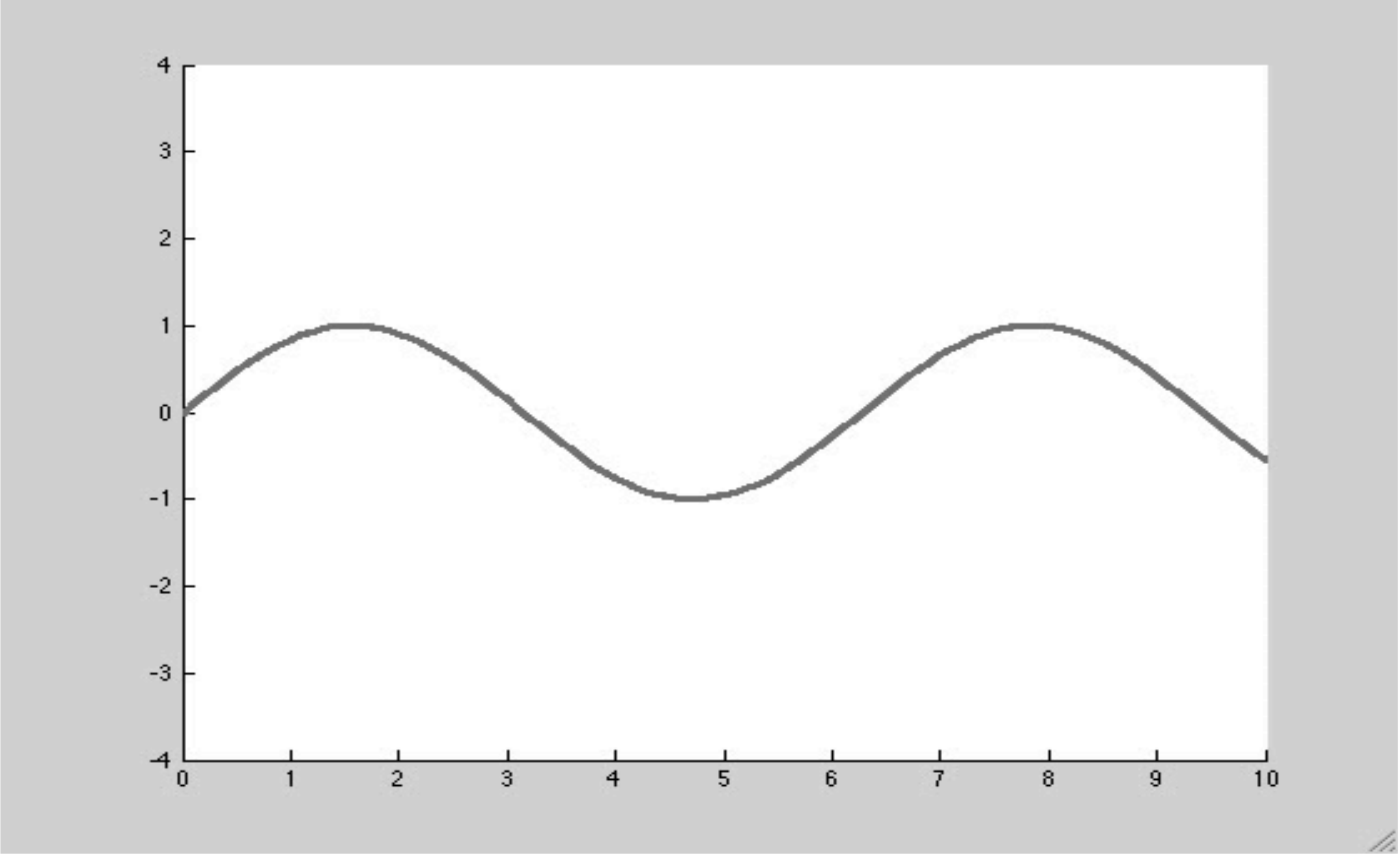
$$p(x_a \mid x_b) = \mathrm{N}\left(\mu_{a|b}, \Sigma_{a|b}\right)$$

$$\mu_{a|b} = \mu_a + \Sigma_{ab} \Sigma_{bb}^{-1} (x_b - \mu_b)$$

$$\Sigma_{a|b} = \Sigma_{aa} - \Sigma_{ab} \Sigma_{bb}^{-1} \Sigma_{ba}$$

Recall conditional

GP Prediction



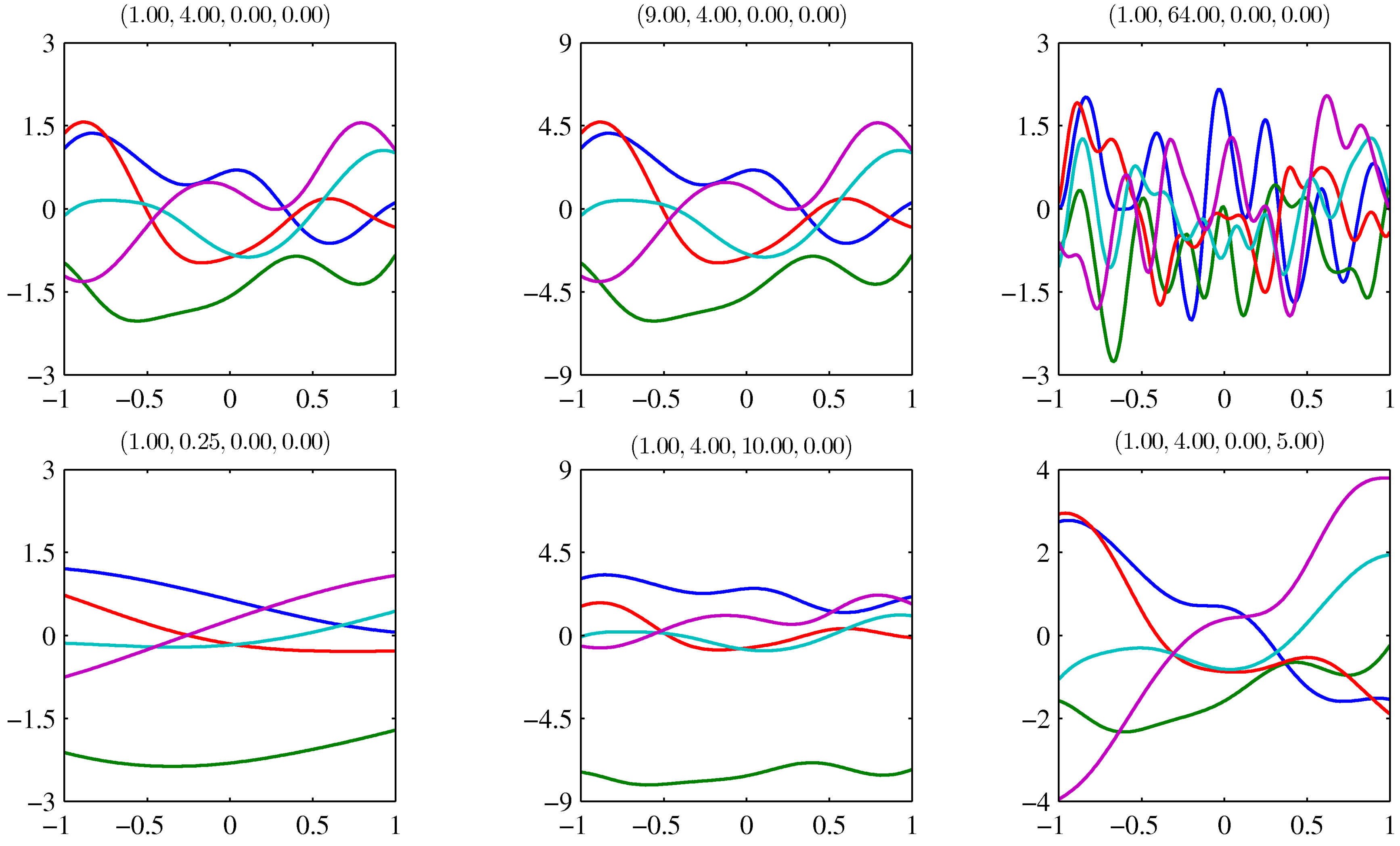
Hyperparameters

- Noise Standard deviation (σ_n^2)
  - Affects how likely a new observation changes predictions (and covariance)
- Kernel (choose based on data)
  - SE, Exponential, Matern etc.
- Kernel hyperparameters:
  - SE kernel: $$k(x,x') = \sigma_f^2 \, e^{-\frac{1}{2}(x-x')W(x-x')^T}$$
    - Length scale (how fast the function changes)
    - Scale factor (how large the function variance is)

Pictures from [Bishop: PRML, 2006]

Hyperparameters

$$k(x,x') = \theta_0 \exp\left(-\frac{\theta_1}{2}|x-x'|^2\right) + \theta_2 + \theta_3 x^T x'$$



Hyperparameter Estimation

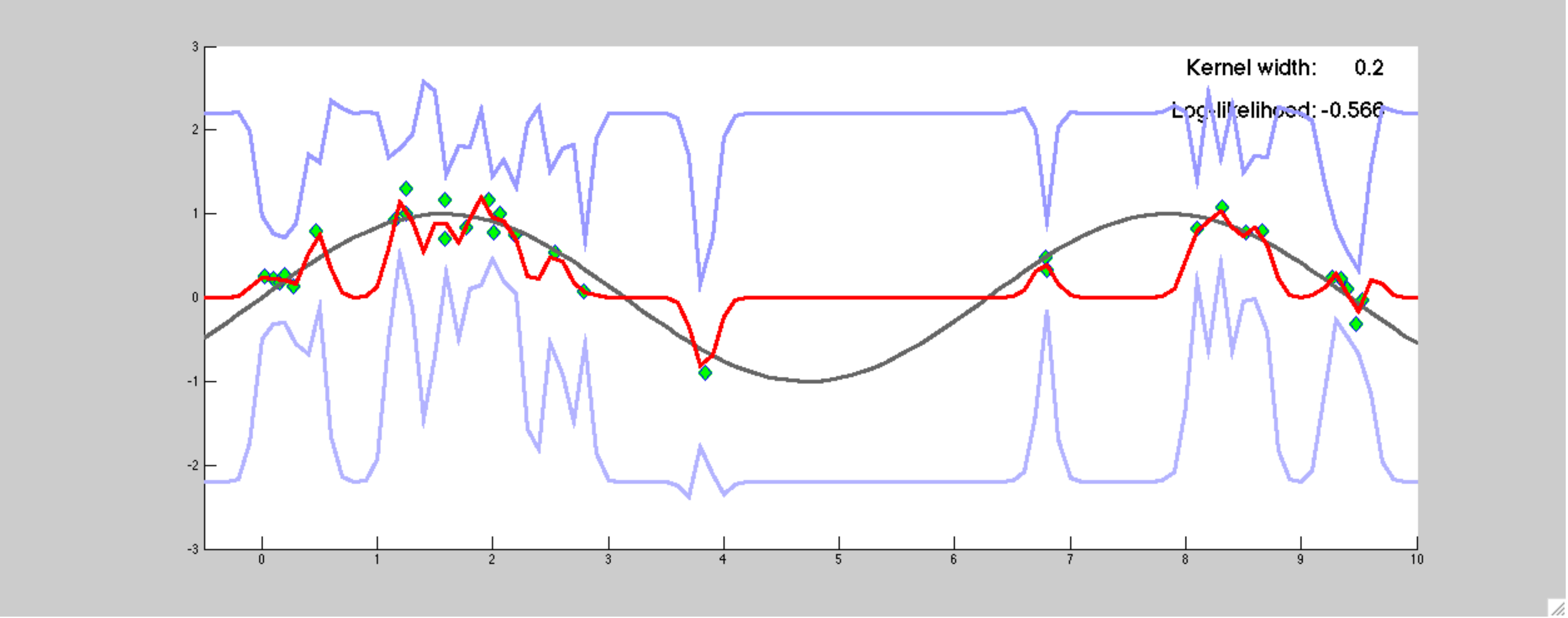
- Maximize data log likelihood:

$$\theta_* = \arg\max_{\theta} p(\mathbf{y} \mid \mathbf{X}, \theta)$$

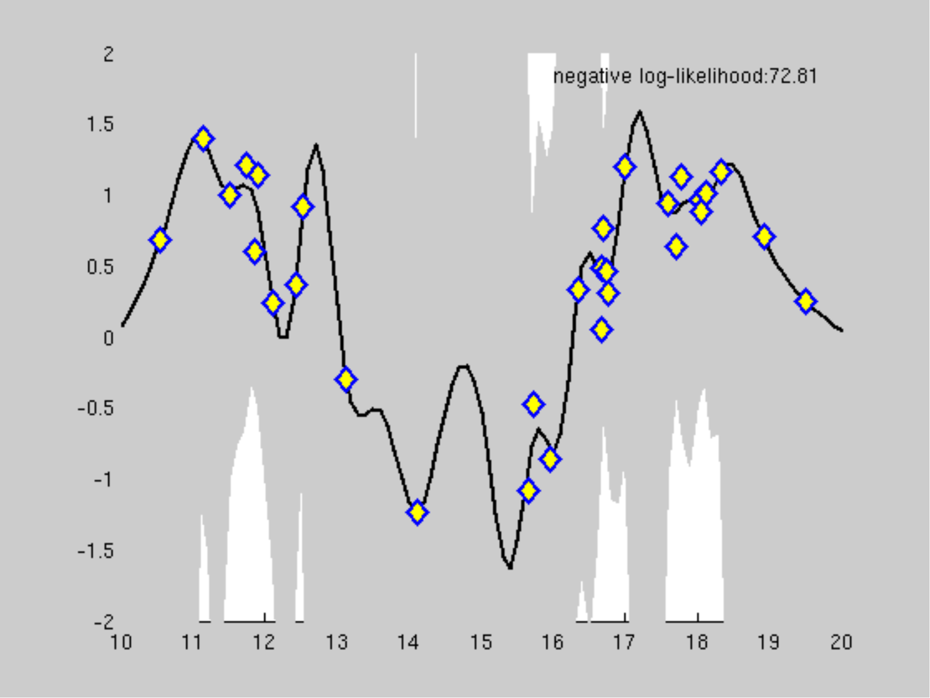
$$\log p(\mathbf{y} \mid \mathbf{X}, \theta) = -\frac{1}{2}\mathbf{y}^T \left(\mathbf{K} + \sigma_n^2 \mathbf{I}\right)^{-1} \mathbf{y} - \frac{1}{2}\log\left(\mathbf{K} + \sigma_n^2 \mathbf{I}\right) - \frac{n}{2}\log 2\pi$$

- Compute derivatives wrt. params $\theta = \langle \sigma_n^2, l, \sigma_f^2 \rangle$
- Optimize using conjugate gradient descent

Kernel Width

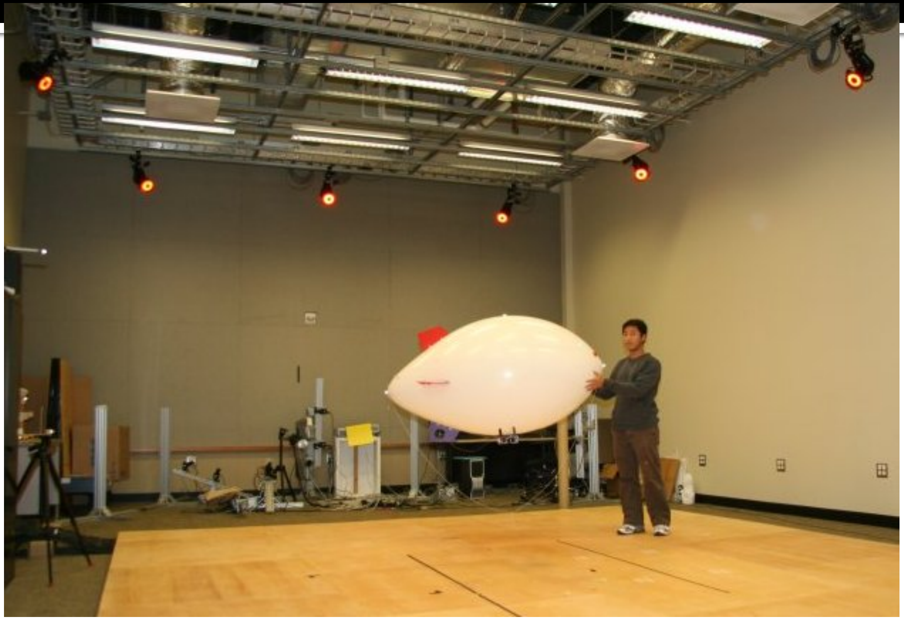


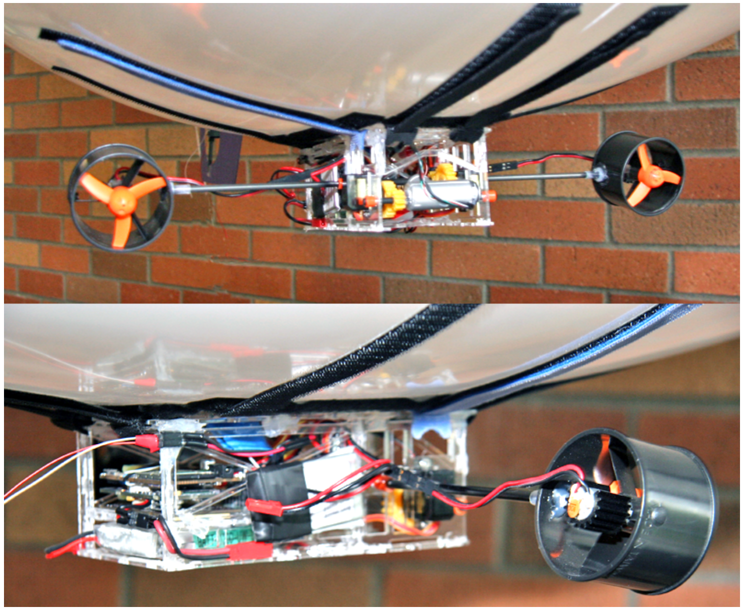
GP Optimization



- Learn hyperparameters via numerical methods
- Learn noise model at the same time

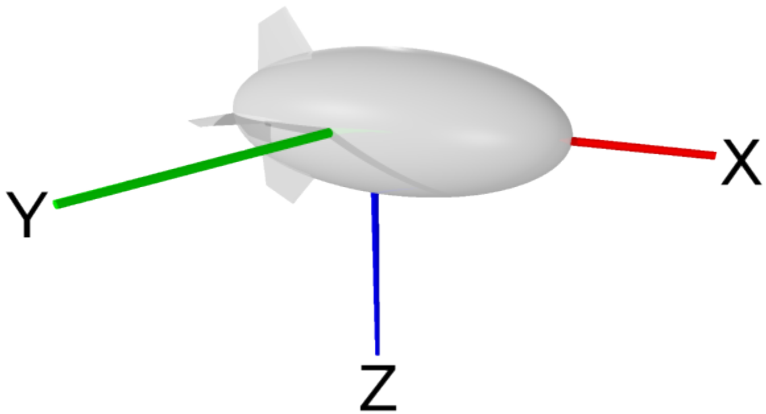
Blimp Platform





- System:
  - Commercial blimp envelope with custom gondola
  - XScale based computer with Bluetooth connectivity
  - Two main motors with tail motor (3D control)
- Ground truth obtained via VICON motion capture system

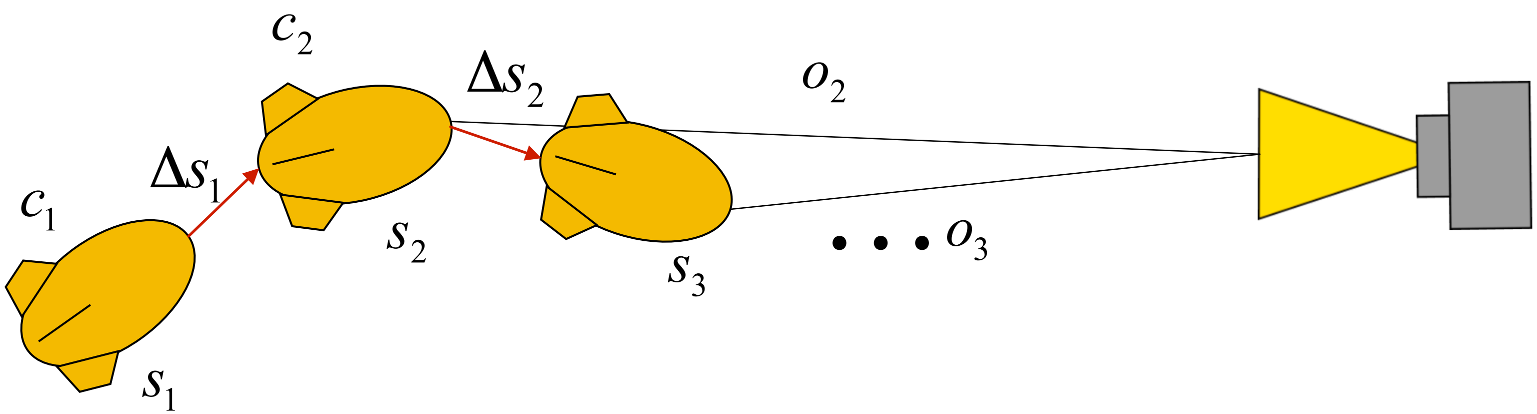
Non-linear Parametric Model



$$\dot{s} = \frac{d}{dt}\begin{bmatrix} p \\ \xi \\ v \\ \omega \end{bmatrix} = \begin{bmatrix} R_b^e v \\ H(\xi) \\ M^{-1}\left(\sum Forces - \omega * Mv\right) \\ J^{-1}\left(\sum Torques - \omega * J\omega\right) \end{bmatrix}$$

- 12-D state=[pos,rot,transvel,rotvel]
- Describes evolution of state as ODE
- Forces / torques considered: buoyancy, gravity, drag, thrust
- 16 parameters are learned by optimization on ground truth motion capture data

Learning GP Dynamics Model

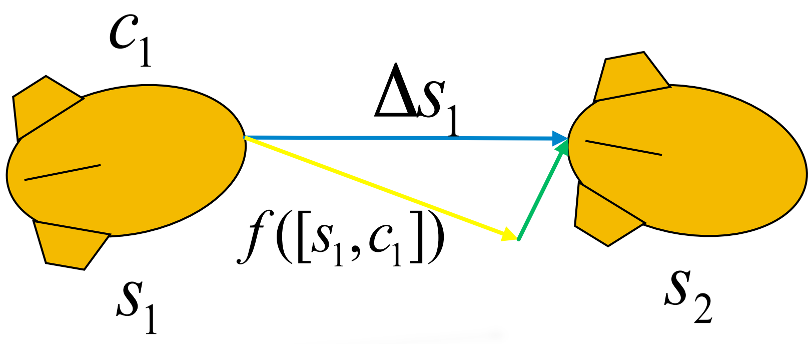


- Use ground truth state to extract:
  - Dynamics data

$$D_S = \langle [s_1, c_1], \Delta s_1 \rangle, \langle [s_2, c_2], \Delta s_2 \rangle \quad \ldots$$

- Learn model using Gaussian process regression
  - Learn process noise inherent in system

Learning Enhanced-GP Models



- Combine GP model with parametric model

$$D_X = \langle [s_1, c_1], \Delta s_1 - f([s_1, c_1]) \rangle$$

- Advantages
  - Captures aspects of system not considered by parametric model
  - Learns noise model in same way as GP-only models
  - Higher accuracy for same amount of training data

GP Modeling Accuracy

Dynamics model error

Propagation method	pos(mm)	rot(deg)	vel(mm/s)	rotvel(deg/s)
Param	3.3	0.5	14.6	1.5
GPonly	1.8	**0.2**	9.8	**1.1**
EGP	**1.6**	**0.2**	**9.6**	1.3

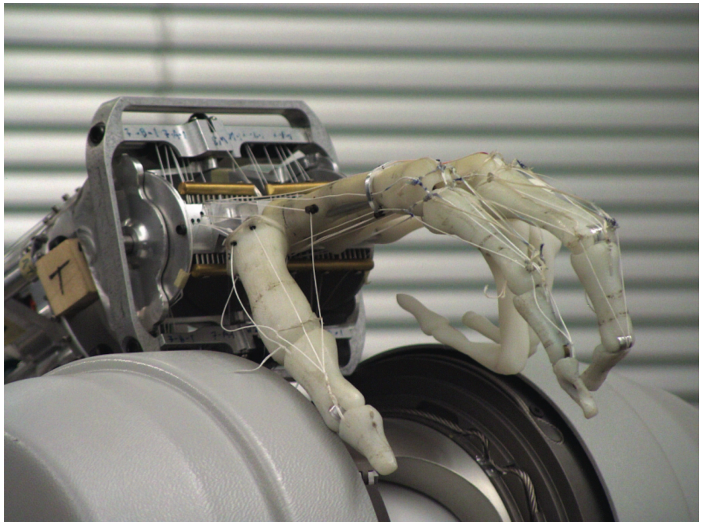
- 1800 training points, mean error over 900 test points
- For dynamics model, 0.25 sec predictions

Related Issues

- Heteroscedastic (state dependent) noise
- Non-stationary GPs
- Coupled outputs
- Sparse GPs
  - Online: Decide whether or not to accept new point
  - Remove points
  - Optimize small set of points
- Classification
  - Laplace approximation
  - No closed-form solution, sampling

Summary

- GPs provide flexible modeling framework
- Take data noise and uncertainty due to data sparsity into account
- Combination with parametric models increases accuracy and reduces need for training data
- Computational complexity is a key problem



Some References

- Website: http://www.gaussianprocess.org/
- GP book: http://www.gaussianprocess.org/gpml/
- GPLVM: http://inverseprobability.com/fgplvm/
- GPDM: http://www.dgp.toronto.edu/~jmwang/gpdm/
- Bishop book: http://research.microsoft.com/en-us/um/people/cmbishop/prml/