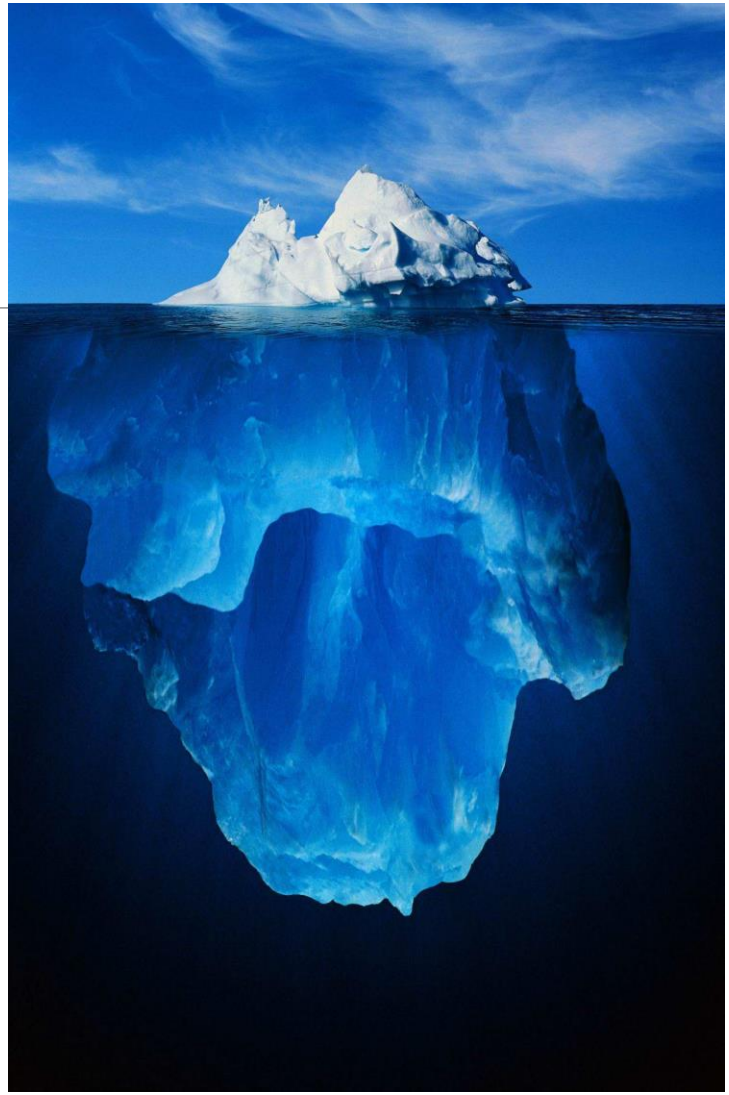
Introduction to Gaussian Process

CS 778

Chris Tensmeyer

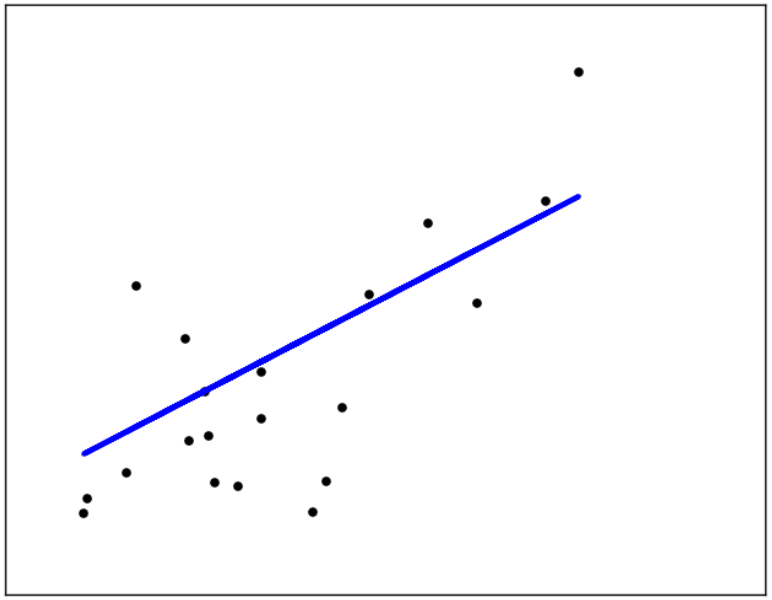
What Topic?



- Machine Learning
- Regression
- Bayesian ML
- Bayesian Regression
- Bayesian Non-parametric
- Gaussian Process (GP)
- GP – Regression
- GP – Regression with Squared Exponential Kernel

Regression

- Predict real value as output
- Learn underlying target function
- Some data samples of target function (with noise)
- E.g. Predict Credit Score
- Estimate model parameters from data points.
  - $y = \boldsymbol{w} \cdot \boldsymbol{x}$, estimate $\boldsymbol{w}$, to minimize loss



Paper Titles

- Gaussian process classification for segmenting and annotating sequences
- Discovering hidden features with Gaussian processes regression
- Active learning with Gaussian processes for object categorization
- Semi-supervised learning via Gaussian processes
- Discriminative Gaussian process latent variable model for classification
- Computation with infinite neural networks
- Fast sparse Gaussian process methods: The informative vector machine
- Approximate Dynamic Programming with Gaussian Processes

Paper Titles

- Learning to control an octopus arm with Gaussian process temporal difference measure
- Gaussian Processes and Reinforcement Learning for Identification and Control of an Autonomous Blimp
- Gaussian process latent variable models for visualization of high dimensional data
- Bayesian unsupervised signal classification by Dirichlet process mixtures of Gaussian processes

Bayesian ML

- Probability Theory
  - What makes function a Prob. Dist?
- Estimate *distribution* over model parameters, $\boldsymbol{\theta}$, that generated $\boldsymbol{D}$
  - AKA function or hypothesis
- Important distributions
  - Prior: $\mathrm{p}(\boldsymbol{\theta})$
  - Data Likelihood: $\mathrm{p}(\boldsymbol{D}|\boldsymbol{\theta})$
  - Posterior: $\mathrm{p}(\boldsymbol{\theta}|\mathbf{D}) = \frac{\mathbf{1}}{\boldsymbol{p}(\boldsymbol{D})} * \mathrm{p}(\boldsymbol{D}|\boldsymbol{\theta}) * \mathrm{p}(\boldsymbol{\theta})$
  - Posterior Predictive: $p(y|\boldsymbol{x}) = \int p(y|\boldsymbol{x}, \boldsymbol{\theta})\mathrm{p}(\boldsymbol{\theta}|\mathbf{D})\boldsymbol{d\theta}$
    - Let all possible functions (one per $\boldsymbol{\theta}$) vote for the correct value

Running Example

- Predict weight given height
  - Lbs & inches
- Let's use a linear model with Gaussian Noise
  - 2 parameters: m, σ^2
  - Let's assume that σ^2 = 5
- $Weight \sim N(m * height, \sigma^2)$
  - $\mathrm{p}(x;\ \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{\sigma^2}}$
- Let's use a prior for m

Prior

GENERAL

- $\mathrm{p}(\boldsymbol{\theta})$
- Your own bias about what makes a good solution
- Score solutions without seeing data
- Encoded as a distribution over parameters

EXAMPLE

- Let's get 5 good guesses about what *m* is
  - Units are lbs/inch
  - $p(m = guess_i) = 0.2$
- Other possible *m* get 0 prob. under our prior
  - Normally use a prior which assigns everything a non-zero probability.

Data Likelihood

GENERAL

- $\mathrm{p}(\boldsymbol{D}|\boldsymbol{\theta})$
- $\boldsymbol{\theta}$ describes the process of creating $\boldsymbol{D}$
- Assuming $\boldsymbol{\theta}$ reflects reality, out of all the $\boldsymbol{D}$', how likely is the $\boldsymbol{D}$ we got?
- Maximum Likelihood Estimation

EXAMPLE

- Data: Height -> Weight $(h_i \rightarrow w_i)$
  - 70in -> 150lbs
  - 80in -> 200lbs
  - 60in -> 130lbs
- $\mathrm{p}(\boldsymbol{D}|\mathrm{m}, \sigma^2) = \prod_{i=1}^{|D|} N(w_i; h_i * m, \sigma^2)$
- The *m* that maximizes ^ is the MLE solution.
  - Is it one of our 5 guesses?
  - No. What is it?
  - $m = \frac{686}{298} = 2.302$

Posterior

GENERAL

- $\mathrm{p}(\boldsymbol{\theta}|\mathbf{D}) = \frac{\mathrm{p}(\boldsymbol{D}|\boldsymbol{\theta}) * \mathrm{p}(\boldsymbol{\theta})}{\boldsymbol{p}(\boldsymbol{D})}$
- Bayes Theorem
- Combines Prior with Data Likelihood
- Often find $\boldsymbol{\theta}$ that maximizes $\mathrm{p}(\boldsymbol{\theta}|\mathbf{D})$
  - Maximum a posteriori (MAP) estimate
  - Can ignore $\boldsymbol{p}(\boldsymbol{D})$ in this case

EXAMPLE

- Plug in each of our guesses for *m* into the numerator
  - Get denominator by $\sum p(m = guess_i|\boldsymbol{D}) = 1$
- Guess with highest probability is the MAP estimate

Posterior Predictive

GENERAL

- $p(y|\boldsymbol{x}) = \int p(y|\boldsymbol{x}, \boldsymbol{\theta}) \mathrm{p}(\boldsymbol{\theta}|\mathbf{D}) d\boldsymbol{\theta}$
- Let all $\boldsymbol{\theta}$ vote, weighted by posterior probability
  - Bayes Optimal Ensemble
- Rarely tractable integral
  - MLE or MAP is easier
- Really want: $\mathrm{argmax}_{\mathrm{y}}\, p(y|\boldsymbol{x})$
  - Can also do confidence intervals

EXAMPLE

- Given a test point height = 75in, predict weight
  - Using MLE?
  - Using MAP?
- Posterior Predictive
  - Guess a weight and score it
  - Score weight using each of 5 *m*
  - Weight scores by *p(m)*
  - Iterate until $\mathrm{argmax}_{\mathrm{w}}\, p(w|h)$

Other Regression Methods

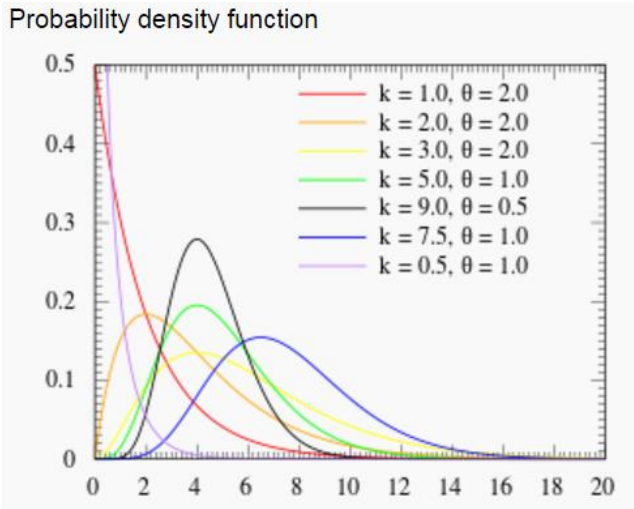
- Simple Linear Regression (LR)
  - $y = \boldsymbol{w} \cdot \boldsymbol{x}$ or in Bayesian Land: $y \sim N(\boldsymbol{w} \cdot \boldsymbol{x}, \sigma^2)$
  - Ordinary Least Squares: find $\boldsymbol{w}$ to minimize $SSE = \sum(\boldsymbol{w} \cdot \boldsymbol{x} - y_t)^2$
    - Equivalent to maximizing $p(\boldsymbol{D}) = \prod p(y_t|\boldsymbol{x}, \boldsymbol{w}) = \prod N(y_t|\boldsymbol{w} \cdot \boldsymbol{x}; \sigma^2)$
    - Closed form solution: $\boldsymbol{w} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$
  - Kernelized Linear Regression (KLR)
  - In LR, replace $\boldsymbol{x}$ with $\varphi(\boldsymbol{x})$, where φ is a non-linear feature transform
  - Kernel trick allows you to do implicit $\varphi(\boldsymbol{x})$ for efficiency
  - E.g. Learn coefficients of order N polynomial

Other Regression Methods

- Ridge Regression (RR)
  - Like LR, but with L2 weight penalty to encourage small weights
  - More robust to outliers (less overfit)
  - L2 weight penalty = Gaussian Prior on weights
  - $w_i \sim N(0, \sigma^2)$ => $\mathrm{p}(\mathbf{w}) \propto \prod e^{\frac{-w_i^2}{\sigma^2}}$ => $-\log\big(\mathrm{p}(\mathbf{w})\big) \propto \sum w_i^2$
  - Find $\boldsymbol{w}$ to minimize $\mathrm{Obj}(\mathbf{w}) = \sum(\boldsymbol{w} \cdot \boldsymbol{x} - y_t)^2 + \lambda \sum w_i^2$
- Lasso Regression (Lasso)
  - LR with L1 penalty
  - Laplacian Distribution Prior: $\mathrm{p}(w_i) \propto e^{-|w_i|}$
  - Fewer non-zero weights (sparse)

Other Regression Methods

- Bayesian Regression (BR)
  - Like RR, but estimate λ from data
  - Prior over the Prior
  - $w_i \sim N(0, \sigma^2)$ but $\sigma^2 \sim \Gamma(\alpha, \beta)$
    - α, β are user set



- Automatic Relevance Determination (ARD)
  - Like BR, but $w_i \sim N(0, \sigma_i)$ and $\sigma_i \sim \Gamma(\alpha, \beta)$
  - Each weight comes from different prior distribution
  - Good for feature selection

Gaussian Process

- GP = Stochastic Process where Random Variables are Gaussian Distributed

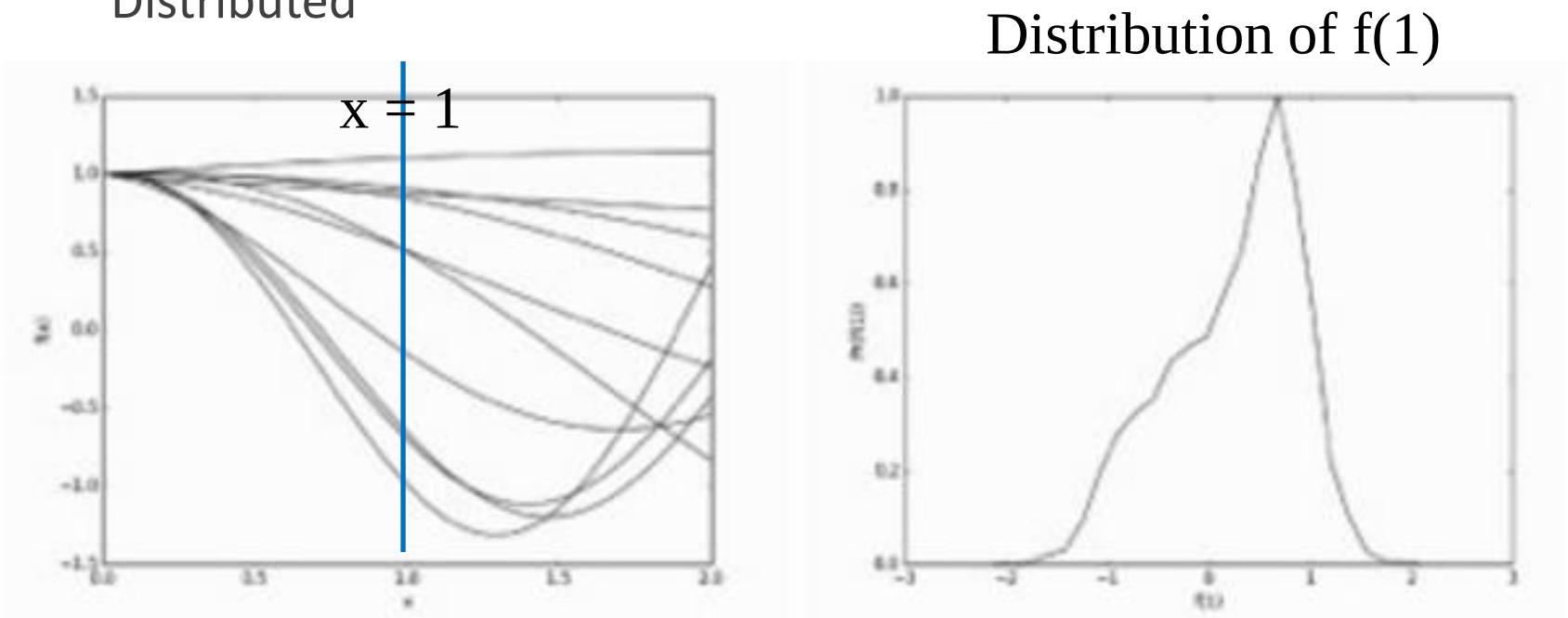
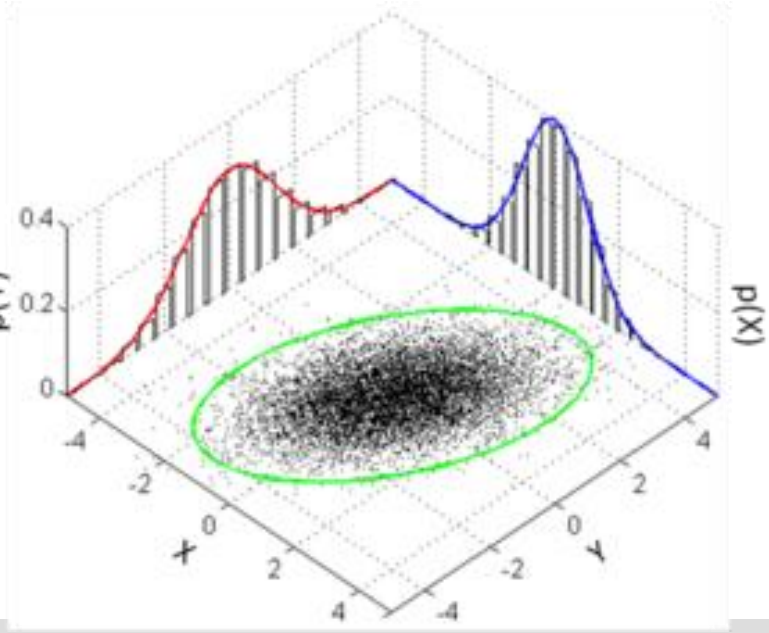


FIGURE 18.2 *Left*: 10 samples from the stochastic process $f(x) = \exp(ax)\cos(bx)$ with a and b drawn from Gaussian distributions. *Right*: The probability distribution of $f(1)$ based on 10,000 samples of $f(x)$.

Gaussian Process - MVN

- Generalization of Multi-Variate Normal to ∞ dimensions

$$f_{\mathbf{x}}(x_1, \ldots, x_k) = \frac{1}{\sqrt{(2\pi)^k |\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x}-\boldsymbol{\mu})\right),$$



Gaussian Process - Background

- First theorized in 1940s
- Used in geostatistic and meterology in 1970s for time series prediction
  - Under the name *kriging*

Smooth Functions

- How would you measure the smoothness of a function?
  - "Smooth" functions is a common prior or bias
- Smoothness: function values close in input space should be similar
  - How similar is f(1) to f(1.1)?
  - Compared to f(1) to f(10)?
- For any finite # of points, $\{\boldsymbol{x_i}\}$, can fit a MVN based on training set composed of smooth functions to learn $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
  - $Smoothness(f) \coloneqq p_{MVN}(f(\boldsymbol{x_0}), \dots f(\boldsymbol{x_i})|\boldsymbol{\mu}, \boldsymbol{\Sigma})$

Dist. Over Functions

- Functions == infinite lookup table == infinite vector of values
  - Vector entries are value of function at some input
- Problem with our *MVN smoothness* at finite # of points
  - Functions score well if not smooth between those points
- GP solves this by being non-parametric
  - $\boldsymbol{\Sigma}$ replaced by $k(\boldsymbol{x}, \boldsymbol{x}')$
  - $\boldsymbol{\mu}$ replaced by $m(\boldsymbol{x})$
- GP is ∞ dimensions, but any finite subset of dimension is MVN

$$f(\mathbf{x}) \sim \mathcal{GP}\big(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\big).$$

Covariance Function

$$m(\mathbf{x}) = \mathbb{E}[f(\mathbf{x})],$$
$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))],$$

- $k(\boldsymbol{x_i}, \boldsymbol{x_j})$ yields $\boldsymbol{\Sigma}$ for MVN over any finite subset $\{f(\boldsymbol{x_i})\}$
- $\boldsymbol{\Sigma_{ij}} = k(\boldsymbol{x_i}, \boldsymbol{x_j})$
- $\forall\{\boldsymbol{x_i}\}, k(\boldsymbol{x_i}, \boldsymbol{x_j})$ must make $\boldsymbol{\Sigma}$ Positive Definite
  - Otherwise MVN is not defined
- Common to subtract out the mean function from train/test points
  - Less common to learn mean function from data

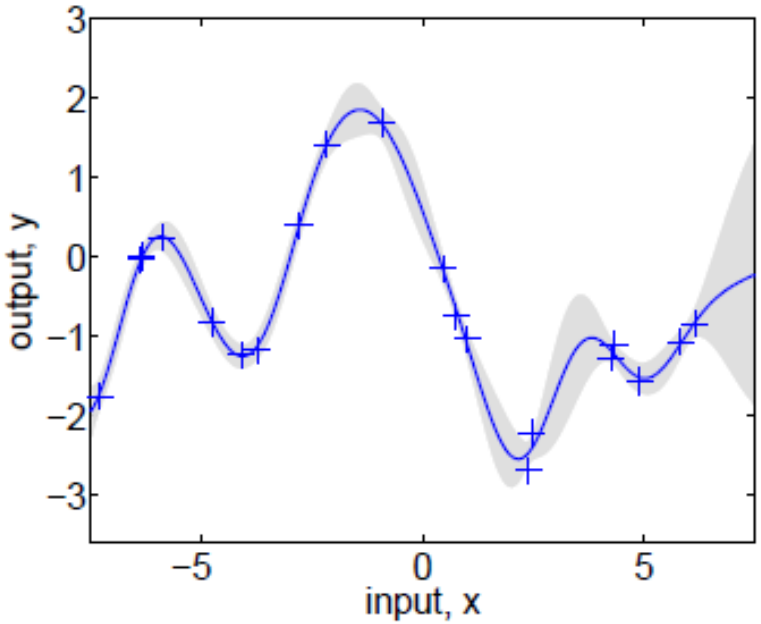
Covariance Functions

- Constant : $K_{\mathrm{C}}(x, x') = C$
- Linear: $K_{\mathrm{L}}(x, x') = x^T x'$
- Gaussian Noise: $K_{\mathrm{GN}}(x, x') = \sigma^2 \delta_{x,x'}$
- Squared Exponential: $K_{\mathrm{SE}}(x, x') = \exp\left(-\frac{||d||^2}{2l^2}\right)$
- Ornstein–Uhlenbeck: $K_{\mathrm{OU}}(x, x') = \exp\left(-\frac{|d|}{l}\right)$
- Matérn: $K_{\mathrm{Matern}}(x, x') = \frac{2^{1-\nu}}{\Gamma(\nu)}\left(\frac{\sqrt{2\nu}|d|}{l}\right)^{\nu} K_{\nu}\left(\frac{\sqrt{2\nu}|d|}{l}\right)$
- Periodic: $K_{\mathrm{P}}(x, x') = \exp\left(-\frac{2\sin^2(\frac{d}{2})}{l^2}\right)$
- Rational Quadratic: $K_{\mathrm{RQ}}(x, x') = (1 + |d|^2)^{-\alpha}, \quad \alpha \geq 0$

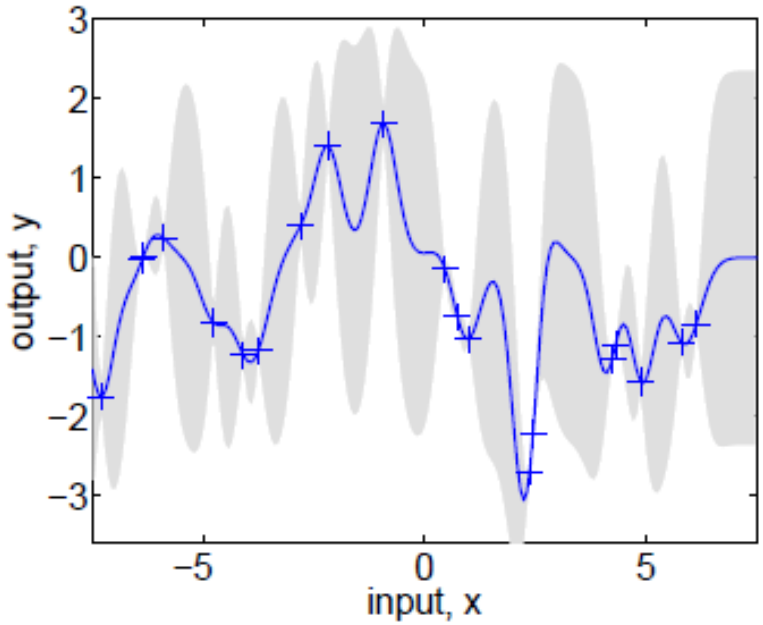
Covariance Functions

- Learning in GP is finding hyperparameters that fit data well!
  - Many design choices: kernel, fixed, global, local, per-dimension
  - Math is messy
  - Do SGD on hyperparameters to maximize log probability of training data
- Observation Noise
  - Universal hyperparameter σ_n is our estimate of how bad we measured the training targets
  - Similar to Bayesian LR
  - Helps prevent overfit

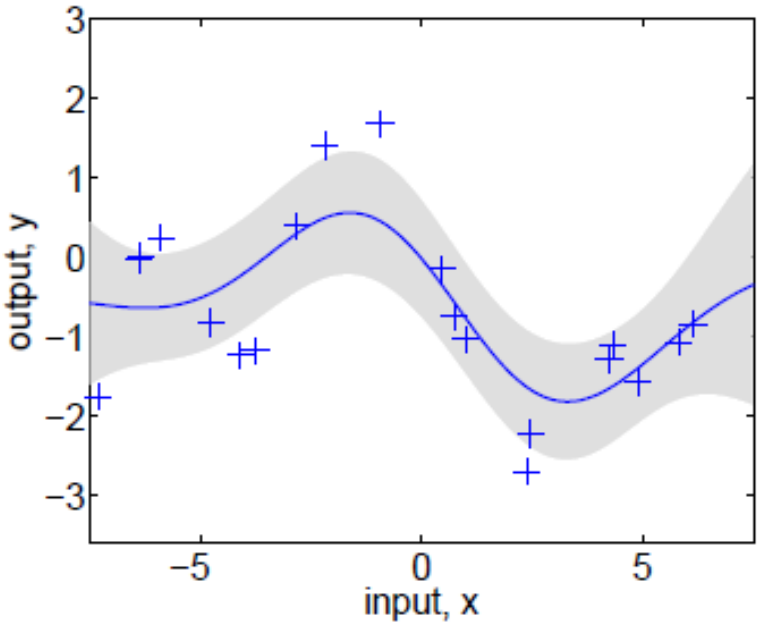
Hyperparameters Length-Scale



(a), $\ell = 1$



(b), $\ell = 0.3$



(c), $\ell = 3$

Inference/Prediction

- The joint distribution of training targets and test target is MVN

$$P(t^*|\mathbf{t}_N) = P(t^*, \mathbf{t}_N)/P(\mathbf{t}_N).$$

$$\mathbf{K}_{N+1} = \left(\begin{bmatrix} \mathbf{K}_N \\ \mathbf{k}^{*T} \end{bmatrix} \quad \begin{bmatrix} \mathbf{k}^* \\ [k^{**}] \end{bmatrix} \right)$$

- We condition the MVN on the observed training targets
- Resulting conditional distribution is Gaussian
- Use distribution mean/covariance for prediction

$$P(t^*|\mathbf{t}, \mathbf{x}, \mathbf{x}^*) \propto \mathcal{N}\left(\mathbf{k}^{*T}\mathbf{K}^{-1}\mathbf{t}, k^{**} - \mathbf{k}^{*T}\mathbf{K}^{-1}\mathbf{k}^*\right),$$

Magic Math From Wikipedia

Conditional distributions [edit]

If N-dimensional **x** is partitioned as follows

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \end{bmatrix} \text{ with sizes } \begin{bmatrix} q \times 1 \\ (N-q) \times 1 \end{bmatrix}$$

and accordingly **μ** and **Σ** are partitioned as follows

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix} \text{ with sizes } \begin{bmatrix} q \times 1 \\ (N-q) \times 1 \end{bmatrix}$$

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{bmatrix} \text{ with sizes } \begin{bmatrix} q \times q & q \times (N-q) \\ (N-q) \times q & (N-q) \times (N-q) \end{bmatrix}$$

then, the distribution of $\mathbf{x}_1$ conditional on $\mathbf{x}_2 = a$ is multivariate normal $(\mathbf{x}_1 | \mathbf{x}_2 = \mathbf{a}) \sim N(\bar{\boldsymbol{\mu}}, \bar{\boldsymbol{\Sigma}})$ where

$$\bar{\boldsymbol{\mu}} = \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\mathbf{a} - \boldsymbol{\mu}_2)$$

and covariance matrix

$$\overline{\boldsymbol{\Sigma}} = \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21}.$$ [12]

Discussion

PROS

- Learned function is smooth but not constrained by form
- Confidence intervals
- Good prior for other models
  - E.g. Bias N-degree polynomials to be smooth
- Can do multiple test points that co-vary appropriately

CONS

- Math/Optimization is messy
  - Use lots of approximations
- NxN Matrix inversion
  - Numerical Stability issues
- Scalability in # instances issues like SVMs
- High dimensional data is challenging

Implementation

- For given training data $(\mathbf{X}, \mathbf{t})$, test data $\mathbf{x}^*$, covariance function $k()$, and hyperparameters $\boldsymbol{\theta} = (\sigma_f^2, l\sigma_n^2)$:
  - compute the covariance matrix $\mathbf{K} = k(\mathbf{X}, \mathbf{X}) + \sigma_n \mathbf{I}$ for hyperparameters $\boldsymbol{\theta}$
  - compute the covariance matrix $\mathbf{k}^* = k(\mathbf{X}, \mathbf{x}^*)$
  - compute the covariance matrix $k^{**} = k(\mathbf{x}^*, \mathbf{x}^*)$
  - the mean of the process is $\mathbf{k}^{*T}\mathbf{K}^{-1}\mathbf{t}$
  - the covariance is $k^{**} - \mathbf{k}^{*T}\mathbf{K}^{-1}\mathbf{k}^*$

- Only compute $\boldsymbol{K^{-1}}$ once using numerical stable methods
  - Cholesky decomposition: $\boldsymbol{K = LL^T}$, where $\boldsymbol{L}$ is lower triangular
- For each test instance, compute covariance vector **k***
- $\sigma_n \boldsymbol{I}$ is our observation noise added to raw *k(**X**,**X**)*

Example

- Using a squared exponential kernel
- $k(\boldsymbol{x}, \boldsymbol{x}') = \exp(-0.5 * \|\boldsymbol{x} - \boldsymbol{x}'\|^2)$
- Data to the right
- Task is to predict f(2) and f(4)
  - $\sigma_n = 0.25$

X	f(X)
0	-5
1	0
2	?
3	5
4	?

Example – Covariance Matrix

- Begin by calculating covariance matrix
  - Use f(0), f(1), f(3) as dimensions of the MVN
- $k(0,1) = \exp(-0.5\ (1\ -0)^2) = e^{-.5}$
- $k(0,3) = \exp(-0.5\ (3\ -0)^2) = e^{-4.5}$
- $k(1,3) = \exp(-0.5\ (3\ -1)^2) = e^{-2}$
- $k(x,x) = \exp(-0.5\ (x\ -x)^2) = 1$

X	f(X)
0	-5
1	0
2	?
3	5
4	?

$$\boldsymbol{K} = \begin{matrix} 1 & e^{-0.5} & e^{-4.5} \\ e^{-0.5} & 1 & e^{-2} \\ e^{-4.5} & e^{-2} & 1 \end{matrix} + \sigma_n \boldsymbol{I}$$

Example – Matrix Inversion

$$\boldsymbol{K} = \begin{matrix} 1 & e^{-0.5} & e^{-4.5} \\ e^{-0.5} & 1 & e^{-2} \\ e^{-4.5} & e^{-2} & 1 \end{matrix} + \sigma_n \boldsymbol{I} \quad \text{or} \quad \boldsymbol{K} = \begin{matrix} 1.25 & 0.607 & 0.011 \\ 0.607 & 1.25 & 0.135 \\ 0.011 & 0.135 & 1.25 \end{matrix}$$

Next step, invert **K**

$$\boldsymbol{K^{-1}} = \begin{matrix} 1.049 & -0.514 & 0.046 \\ -0.514 & 1.061 & -0.11 \\ 0.046 & -0.11 & 0.812 \end{matrix}$$

X	f(X)
0	-5
1	0
2	?
3	5
4	?

Example – Multiply by Targets

$$P(t^*|\mathbf{t}, \mathbf{x}, \mathbf{x}^*) \propto \mathcal{N}\left(\mathbf{k}^{*T}\mathbf{K}^{-1}\mathbf{t}, k^{**} - \mathbf{k}^{*T}\mathbf{K}^{-1}\mathbf{k}^*\right),$$

Need $\boldsymbol{K^{-1}t}$

$$\boldsymbol{K^{-1}} = \begin{matrix} 1.049 & -0.514 & 0.046 \\ -0.514 & 1.061 & -0.11 \\ 0.046 & -0.11 & 0.812 \end{matrix}$$

$$\boldsymbol{t} = \begin{matrix} \mathbf{-5} \\ \mathbf{0} \\ \mathbf{5} \end{matrix}$$

$$\boldsymbol{K^{-1}t} = \begin{matrix} -5.013 \\ 2.018 \\ 3.826 \end{matrix}$$

X	f(X)
0	-5
1	0
2	?
3	5
4	?

Example – Predicting f(2)

$$P(t^*|\mathbf{t},\mathbf{x},\mathbf{x}^*) \propto \mathcal{N}\left(\mathbf{k}^{*T}\mathbf{K}^{-1}\mathbf{t}, k^{**} - \mathbf{k}^{*T}\mathbf{K}^{-1}\mathbf{k}^*\right),$$

Now need **k*** for f(2)

$$\boldsymbol{K^{-1}t} = \begin{matrix} -5.013 \\ 2.018 \\ 3.826 \end{matrix}$$

- $k(2,0) = \exp(-0.5\,(2\ -0)^2) = e^{-2}$
- $k(2,1) = k(2,3) = e^{-0.5}$

$$\boldsymbol{k}^* = \begin{matrix} 0.135 \\ 0.607 \\ 0.607 \end{matrix}$$

k** = 1 (no obs noise)

X	f(X)
0	-5
1	0
2	?
3	5
4	?

Example – Predicting f(2)

$$P(t^*|\mathbf{t},\mathbf{x},\mathbf{x}^*) \propto \mathcal{N}\left(\mathbf{k}^{*T}\mathbf{K}^{-1}\mathbf{t}, k^{**} - \mathbf{k}^{*T}\mathbf{K}^{-1}\mathbf{k}^*\right),$$

$$\boldsymbol{k}^* = \begin{matrix} 0.135 \\ 0.607 \\ 0.607 \end{matrix} \qquad \boldsymbol{K}^{-1} = \begin{matrix} 1.049 & -0.514 & 0.046 \\ -0.514 & 1.061 & -0.11 \\ 0.046 & -0.11 & 0.812 \end{matrix}$$

$$\boldsymbol{K}^{-1}\boldsymbol{t} = \begin{matrix} -5.013 \\ 2.018 \\ 3.826 \end{matrix} \Rightarrow \mu = \boldsymbol{k}^{*T}\boldsymbol{K}^{-1}\boldsymbol{t} = 2.866$$

$$\boldsymbol{k}^{*T}\boldsymbol{K}^{-1}\boldsymbol{k}^* = \begin{matrix} -0.142 \\ 0.507 \\ 0.432 \end{matrix} \boldsymbol{k}^* = 0.550$$

$$\sigma = k^{**} - \boldsymbol{k}^{*T}\boldsymbol{K}^{-1}\boldsymbol{k}^* = 1 - 0.55 = 0.45$$

Example – Predicting f(4)

$$P(t^*|\mathbf{t},\mathbf{x},\mathbf{x}^*) \propto \mathcal{N}\left(\mathbf{k}^{*T}\mathbf{K}^{-1}\mathbf{t}, k^{**} - \mathbf{k}^{*T}\mathbf{K}^{-1}\mathbf{k}^*\right),$$

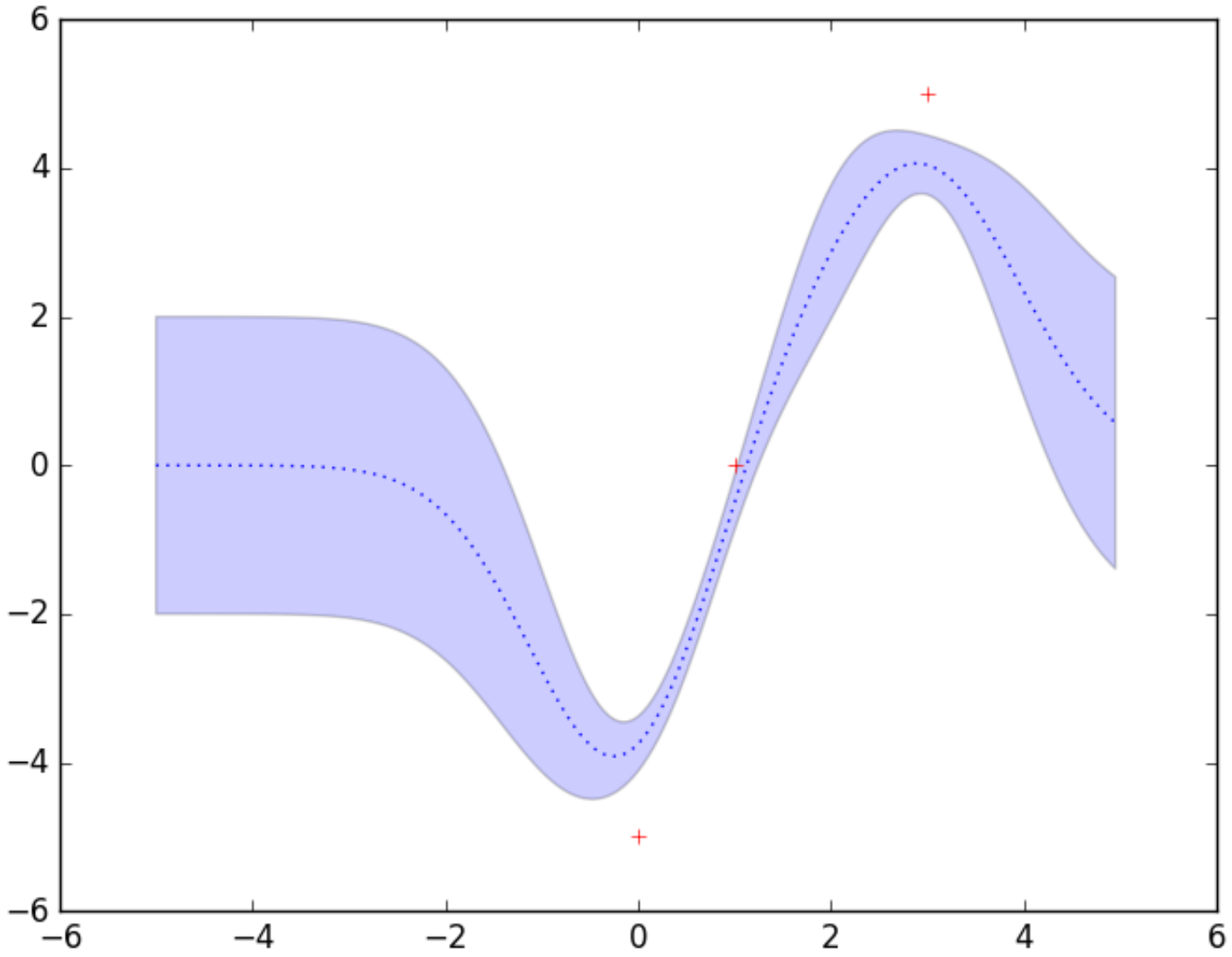
$$\boldsymbol{k}^* = \begin{matrix} 0.00033 \\ 0.011 \\ .607 \end{matrix} \qquad \boldsymbol{K^{-1}} = \begin{matrix} 1.049 & -0.514 & 0.046 \\ -0.514 & 1.061 & -0.11 \\ 0.046 & -0.11 & 0.812 \end{matrix}$$

$$\boldsymbol{K^{-1}t} = \begin{matrix} -5.013 \\ 2.018 \\ 3.826 \end{matrix} \Rightarrow \mu = \boldsymbol{k^{*T}K^{-1}t} = 2.341$$

$$\boldsymbol{k^{*T}K^{-1}k^*} = \begin{matrix} 0.023 \\ -0.055 \\ 0.491 \end{matrix} \boldsymbol{k^*} = 0.297$$

$$\sigma = k^{**} - \boldsymbol{k^{*T}K^{-1}k^*} = 1 - 0.297 = 0.703$$

Example - Plot



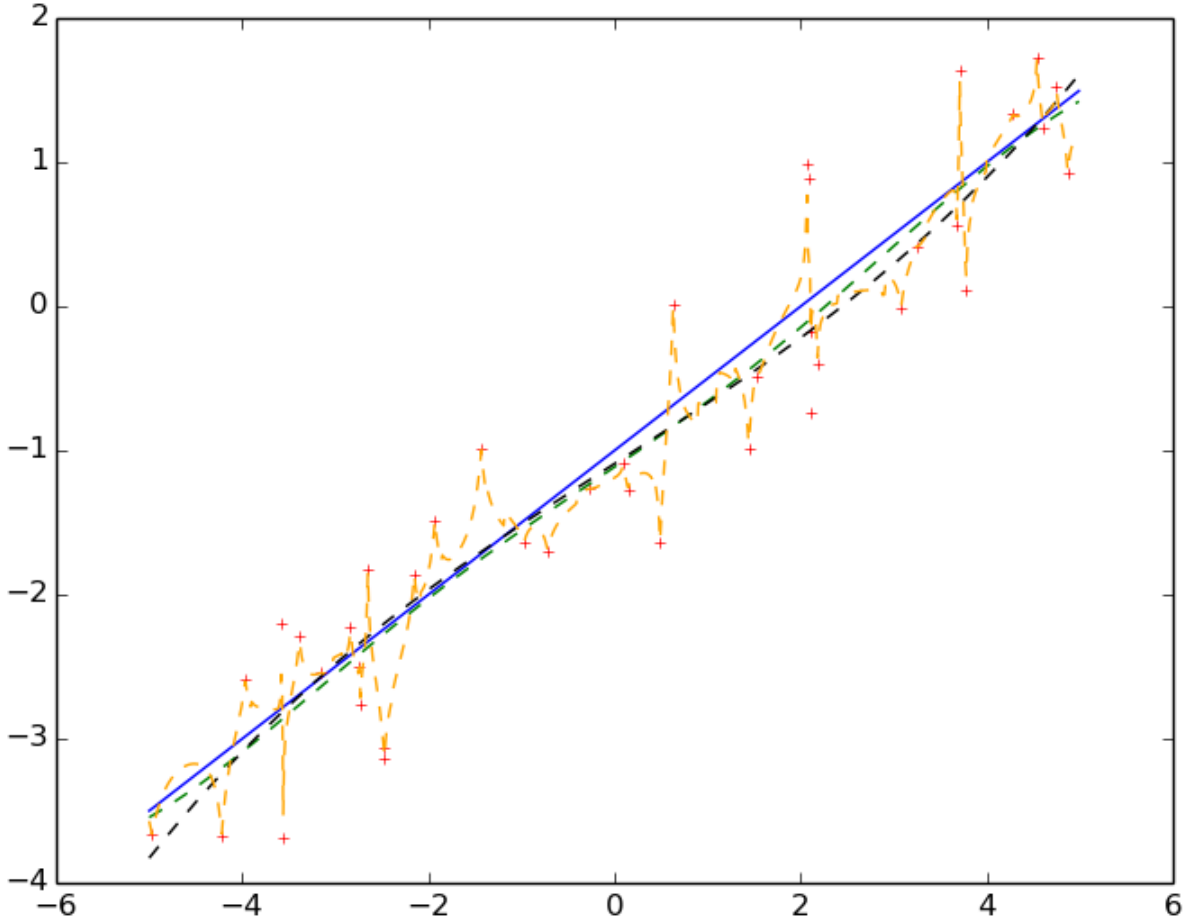
X	f(X)
0	-5
1	0
2	2.8
3	5
4	2.3

My Experiments

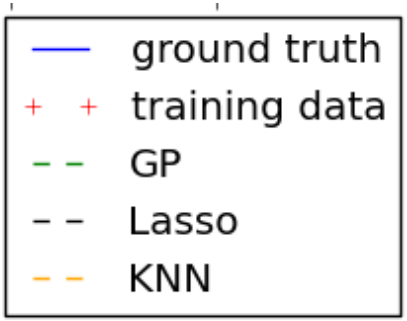
- 1-D synthetic datasets generated with Gaussian noise
  - Linear Cubic
  - Periodic Sigmoid Complex
- 40 instances for training on the interval [-5,5]
  - Additive Gaussian noise with $\sigma = 0.5$
- Test with 1000 evenly spaced instances without noise (MSE)
- Use 3-fold CV on training data to set hyperparameters with grid-search
- Compare with Kernelized Lasso regression and 7-NN with inverse distance weighting
  - Polynomial and Sine function basis

Results - Linear

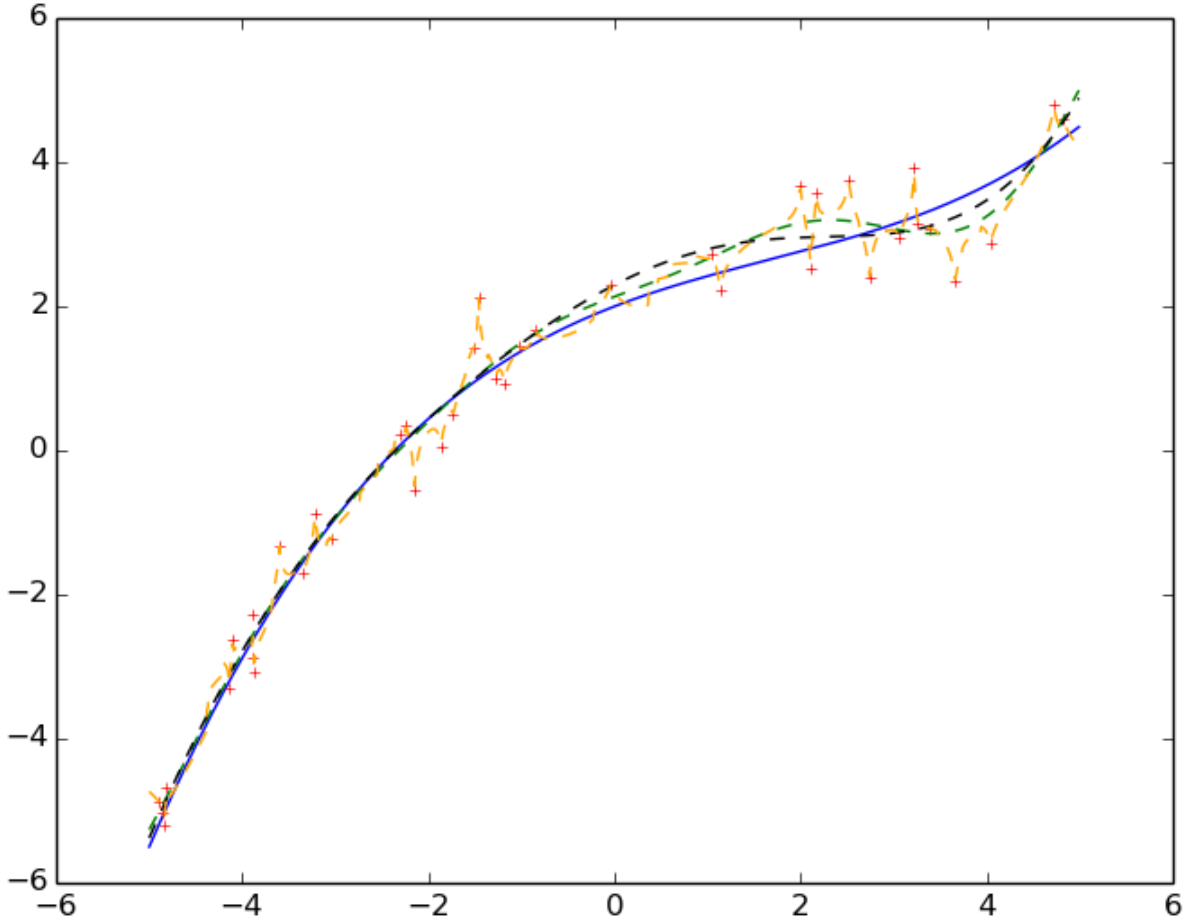
$$y = 0.5x - 1$$



Model	MSE
GP	**0.009**
Lasso	0.019
KNN	0.058

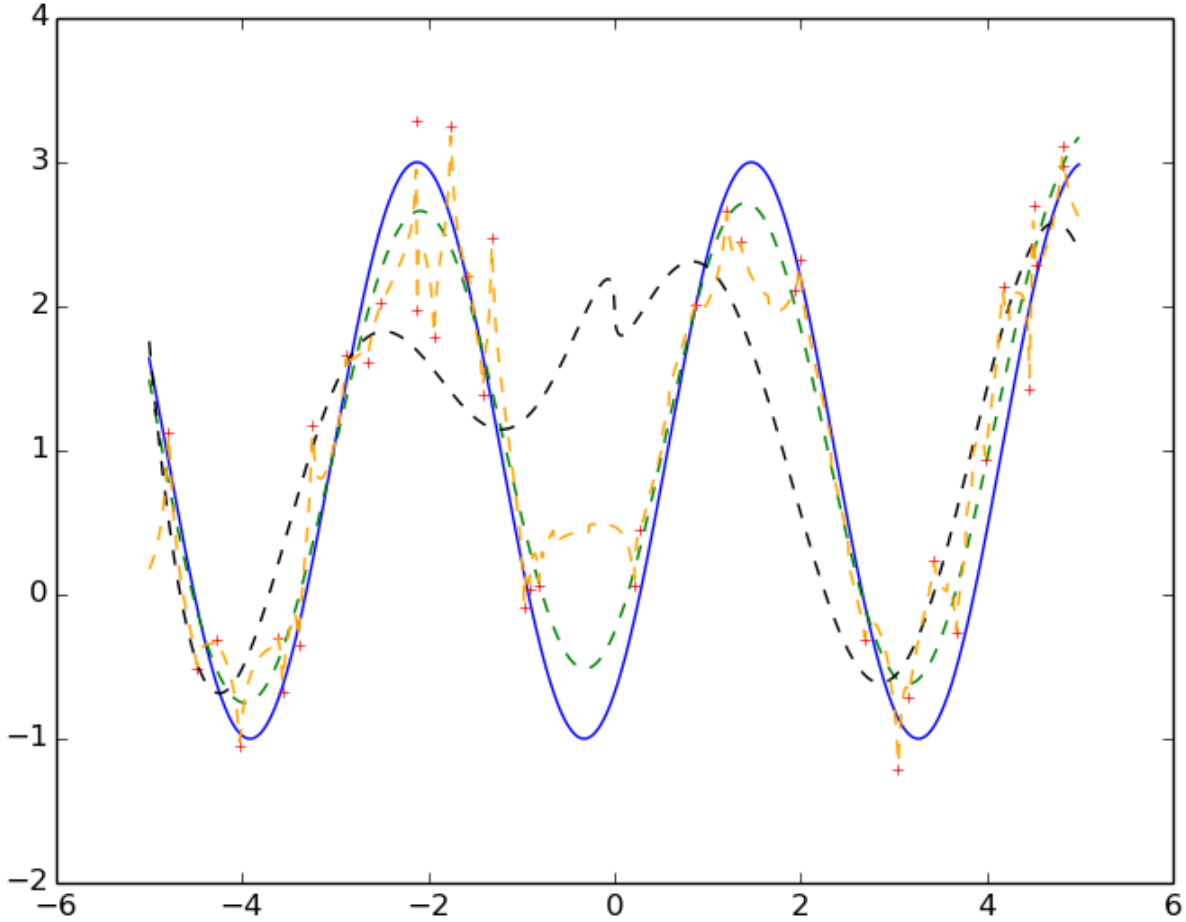


Results - Cubic $y = 0.02x^3 - 0.1x^2 + 0.5x + 2$



Model	MSE
GP	0.046
Lasso	**0.037**
KNN	0.101

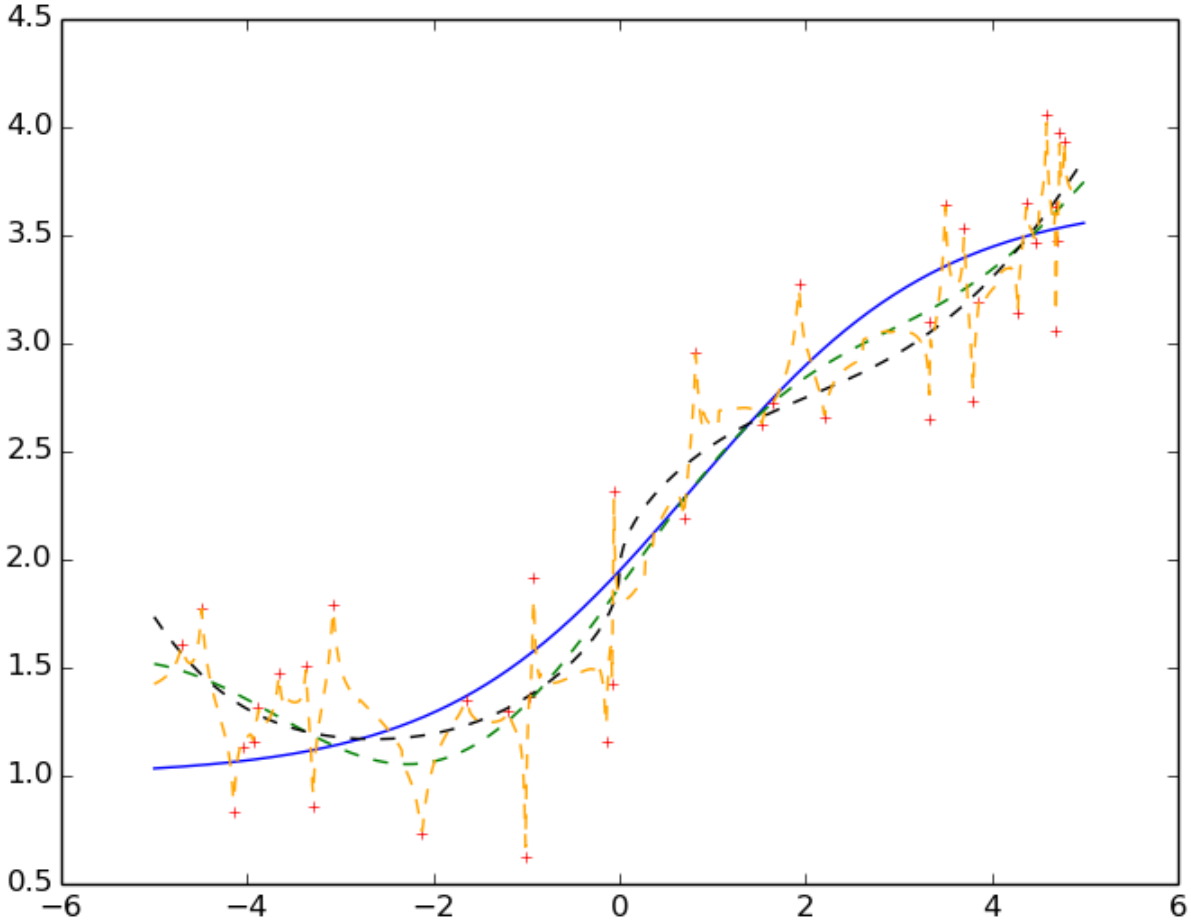
Results - Periodic $y = 2\sin(1.75x - 1) + 1$



Model	MSE
GP	**0.087**
Lasso	1.444
KNN	0.350

Results - Sigmoid

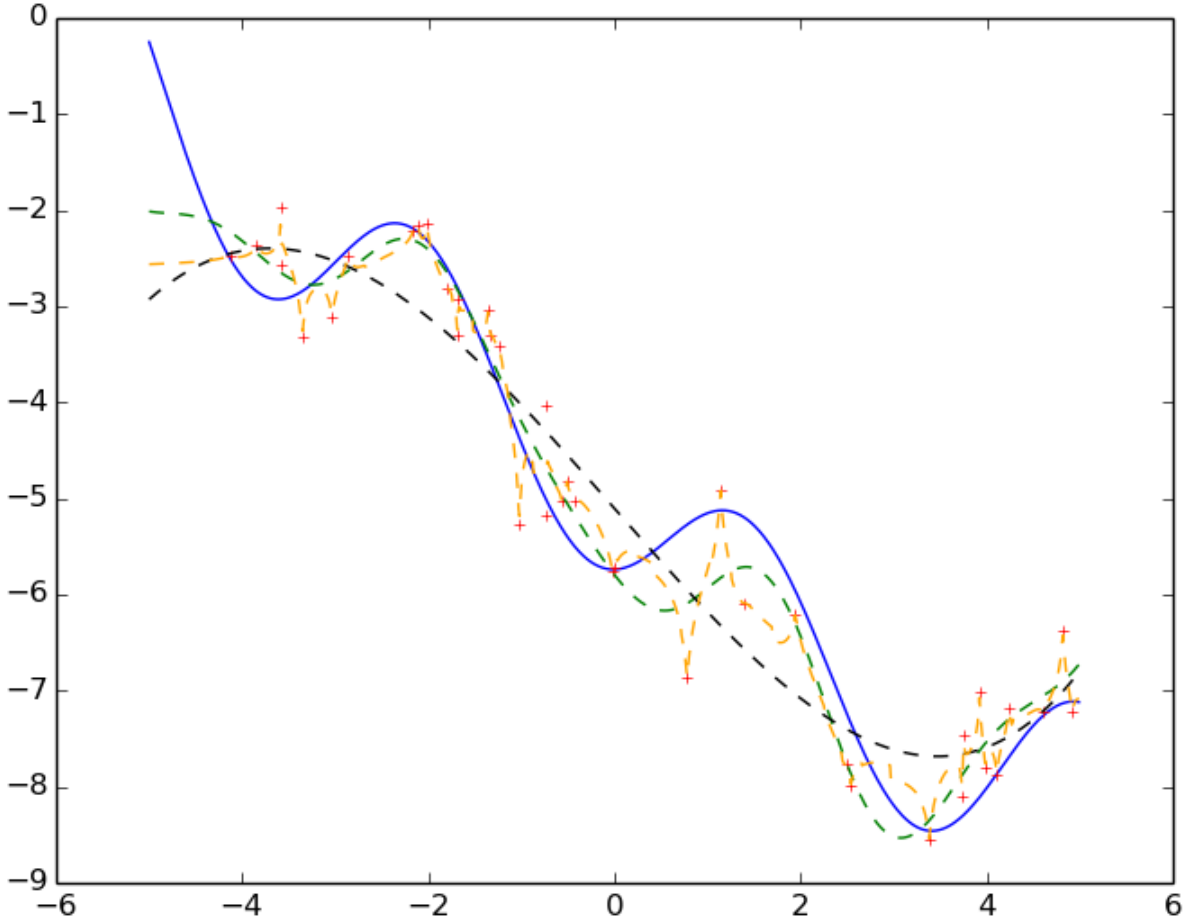
$$y = \frac{4}{1.5 + e^{-(.75x-1)}} + 1$$



Model	MSE
GP	**0.034**
Lasso	0.045
KNN	0.060

Results - Complex

$y =$ linear combination of previous functions



Model	MSE
GP	**0.194**
Lasso	0.574
KNN	0.327

Real Dataset

- Concrete Slump Test
  - How fast does concrete flow given its materials?
  - 103 instances; 7 input features; 3 outputs; all real values
- Input Features are kg per cubic meter of concrete
  - Cement, Slag, Fly Ash, Water, SP, Coarse Agg, Fine Agg
- Output Variables are
  - Slump, Flow, 28-day Compressive Strength
- Ran 10-fold CV with our three models on each output variable

Concrete Results

Slump	
Model	Avg MSE +- Std
GP	0.103 +- 0.037
Lasso	**0.072** +- 0.024
KNN	**0.067** +- 0.040

Flow	
Model	Avg MSE +- Std
GP	0.091 +- 0.026
Lasso	**0.051** +- 0.020
KNN	**0.063** +- 0.036

Strength	
Model	Avg MSE +- Std
GP	0.030 +- 0.015
Lasso	**0.004** +- 0.002
KNN	0.008 +- 0.005

Learning Hyperparameters

- Find hyperparameters to maximize (log) probability of data

$$\log P(\mathbf{t}|\mathbf{x}, \boldsymbol{\theta}) = -\frac{1}{2}\mathbf{t}^T(\mathbf{K} + \sigma_n^2\mathbf{I})^{-1}\mathbf{t} - \frac{1}{2}\log|\mathbf{K} + \sigma_n^2\mathbf{I}| - \frac{N}{2}\log 2\pi$$

- Can use Gradient Descent by computing for $\mathbf{Q} = (\mathbf{K} + \sigma_n^2\mathbf{I})$

$$\frac{\partial}{\partial\boldsymbol{\theta}}\log P(\mathbf{t}|\mathbf{x}, \boldsymbol{\theta}) = \frac{1}{2}\mathbf{t}^T\mathbf{Q}^{-1}\frac{\partial Q}{\partial\boldsymbol{\theta}}\mathbf{Q}^{-1}\mathbf{t} - \frac{1}{2}\text{trace}\left(\mathbf{Q}^{-1}\frac{\partial\mathbf{Q}}{\partial\boldsymbol{\theta}}\right)$$

- $\frac{\partial\mathbf{Q}}{\partial\boldsymbol{\theta}}$ is just the elementwise derivative of the covariance matrix which is the derivative of the covariance function

Learning Hyperparameters

- For squared exponential covariance function in this form

$$k(\mathbf{x}, \mathbf{x}') = \exp(\sigma_f) \exp\left(-\frac{1}{2}\exp(\sigma_l)|\mathbf{x} - \mathbf{x}'|^2\right) + \exp(\sigma_n)\mathbf{I}$$
$$= k' + \exp(\sigma_n)\mathbf{I}$$

- We get the following derivatives

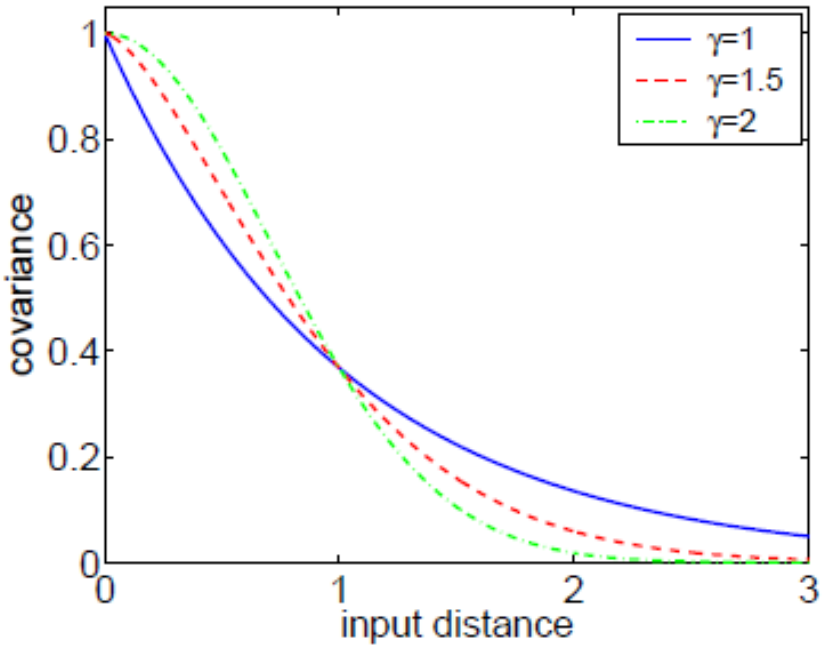
$$\frac{\partial k}{\partial \sigma_f} = k' \qquad \frac{\partial k}{\partial \sigma l} = k' \times \left(-\frac{1}{2}\exp(\sigma_l)|\mathbf{x} - \mathbf{x}'|^2\right)$$

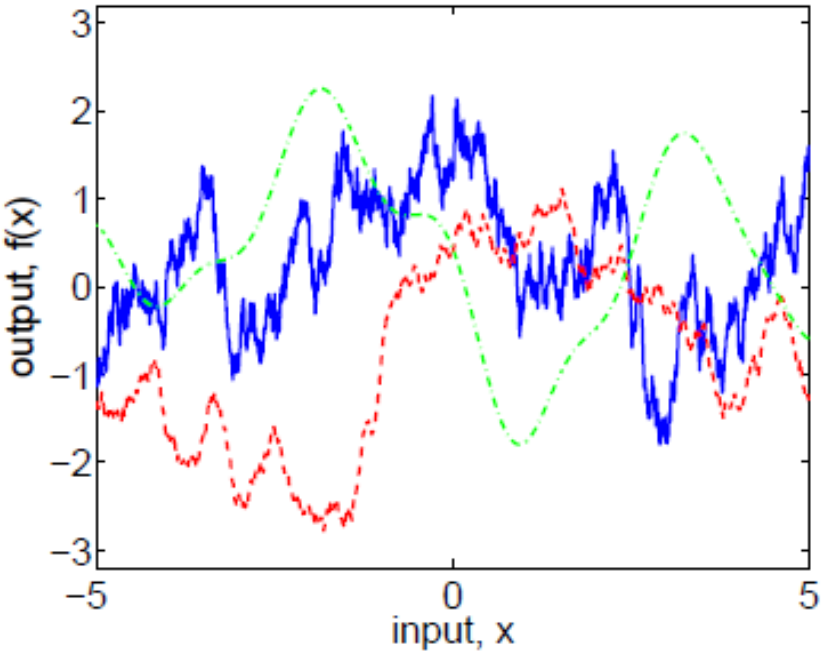
$$\frac{\partial k}{\partial \sigma_n} = \exp(\sigma_n)\mathbf{I}$$

Covariance Function Tradeoffs

$$k(r) = \exp\left(-(r/\ell)^{\gamma}\right) \quad \text{for} \quad 0 < \gamma \leq 2$$

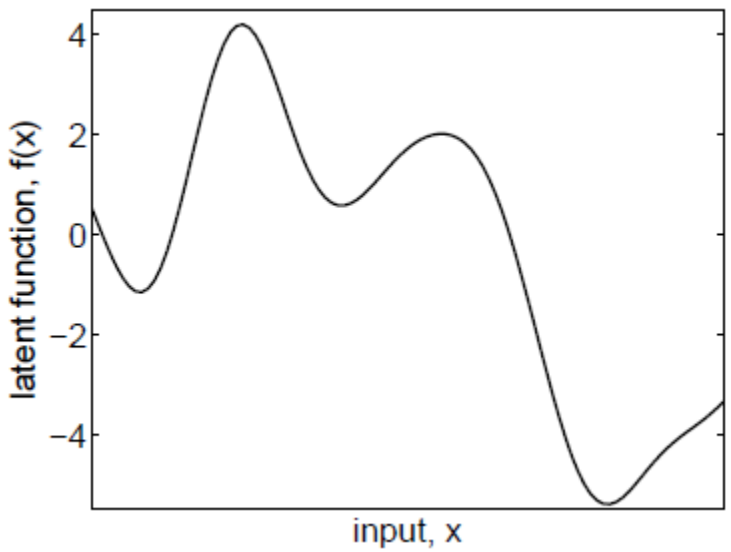
$$r = |\mathbf{x} - \mathbf{x}'|$$

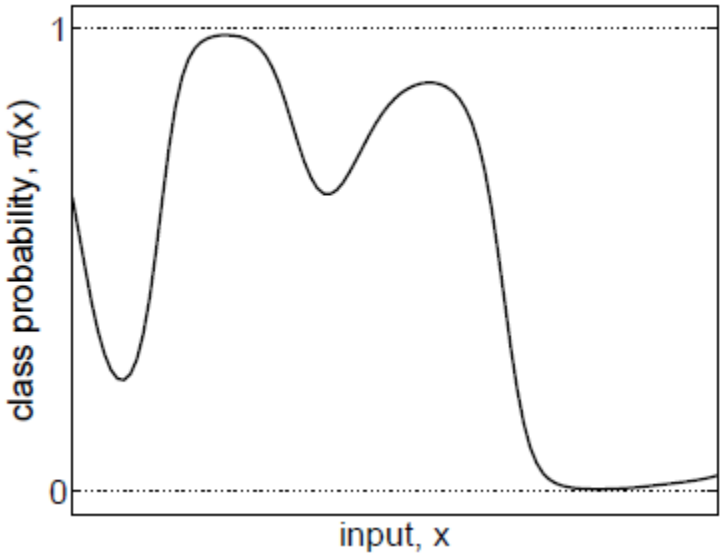




Classification

- Many regression methods become binary classification by passing output through a sigmoid, Softmax, or other threshold
  - MLP, SVM, Logistic Regression
  - Same with GP, but scarier math





Classification Math

Inference is naturally divided into two steps: first computing the distribution of the latent variable corresponding to a test case

$$p(f_*|X,\mathbf{y},\mathbf{x}_*) = \int p(f_*|X,\mathbf{x}_*,\mathbf{f})p(\mathbf{f}|X,\mathbf{y})\,d\mathbf{f}$$

GP Regression

Intractable!

GP prior ignored

where $p(\mathbf{f}|X,\mathbf{y}) = p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|X)/p(\mathbf{y}|X)$

$$p(y_i|f_i) = \sigma(f_i)$$

Vote for class 1

Weight for vote

$$p(y_*=+1|X,\mathbf{y},\mathbf{x}_*) = \int \sigma(f_*)p(f_*|X,\mathbf{y},\mathbf{x}_*)\,df_*$$

Bells and Whistles

- Multiple Regression – Assume correlation between multiple outputs
- Correlated noise – especially useful for time
- Training set values + derivatives (max or min occurs at…)
- Noise in the x-direction
- Mixture (ensemble) of local GP experts
- Integral approximation with GP
- Covariance over structured inputs (strings, trees, etc)

Bibliography

- Wikipedia: Gaussian Process, Multi-variate Normal
- Scikit-Learn Documentation
- www.gaussianprocess.org
- C. E. Rasmussen & C. K. I. Williams, Gaussian Processes for Machine Learning, the MIT Press, 2006, ISBN 026218253X. c 2006 Massachusetts Institute of Technology. www.GaussianProcess.org/gpml
- S. Marsland, Machine Learning An Algorithmic Perspective 2nd Edition, CRC Press, 2015