Active Learning and Optimized Information Gathering

Lecture 6 – Gaussian Process Optimization

CS 101.2
Andreas Krause

Announcements

- Homework 1: out tomorrow
  - Due Thu Jan 29
- Project
  - Proposal due Tue Jan 27
- Office hours
  - Come to office hours before your presentation!
  - Andreas: **Friday 1:30-3pm, 260 Jorgensen**
  - Ryan: Wednesday 4:00-6:00pm, 109 Moore

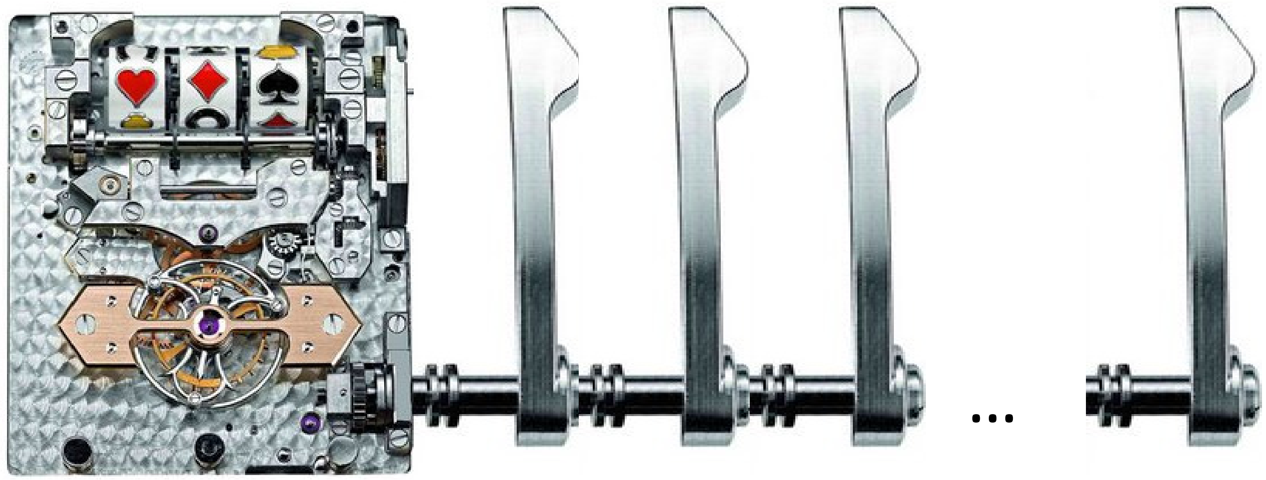
Course outline

1. $\boxed{\text{Online decision making}}$

2. Statistical active learning

3. Combinatorial approaches

Recap Bandit problems



p_1 p_2 p_3 p_k

- K-arms
  - ε_n greedy, UCB1 have regret O(log(T) **K**)
- What about infinite arms ($K=\infty$)
  - Have to make assumptions!

Bandits = Noisy function optimization

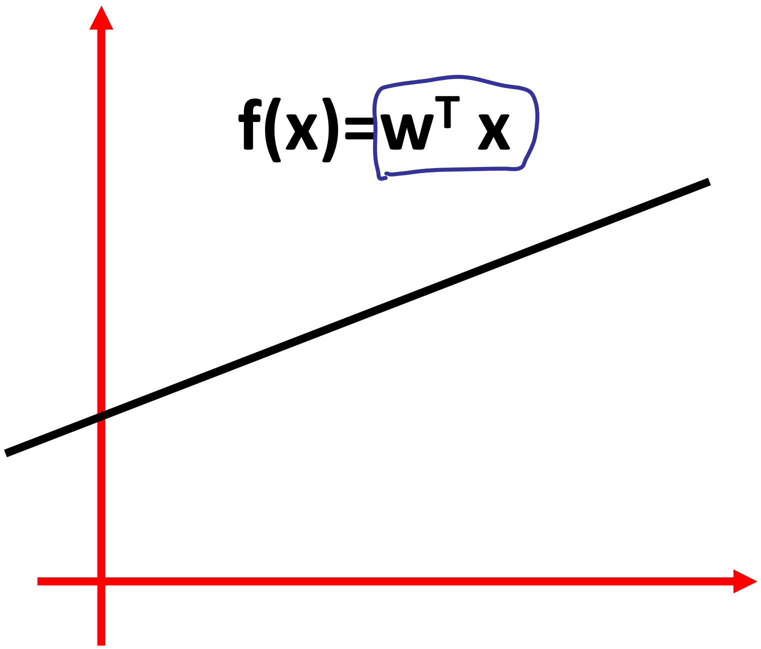
- We are given black box access to function f
  f(x) = mean payoff for arm x

$$x \longrightarrow \boxed{f} \longrightarrow y = f(x) + \text{noise}$$

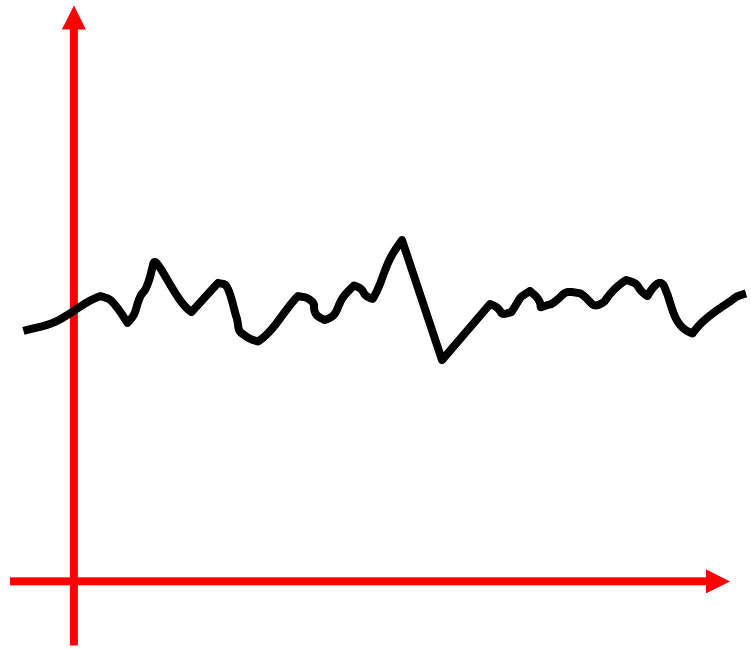
Arm Payoff

- Evaluating f is very expensive
- Want to (quickly) find $x^* = \text{argmax}_x\, f(x)$

Bandits with ∞-many arms



Linear



Lipschitz-continuous (bounded slope)

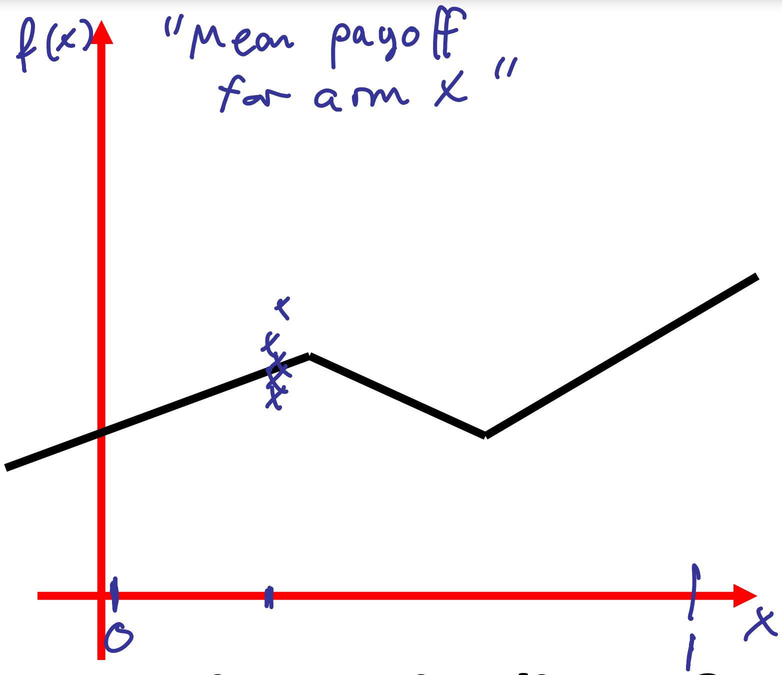
- Can only hope to perform well if we make some assumptions

Regret depends on complexity

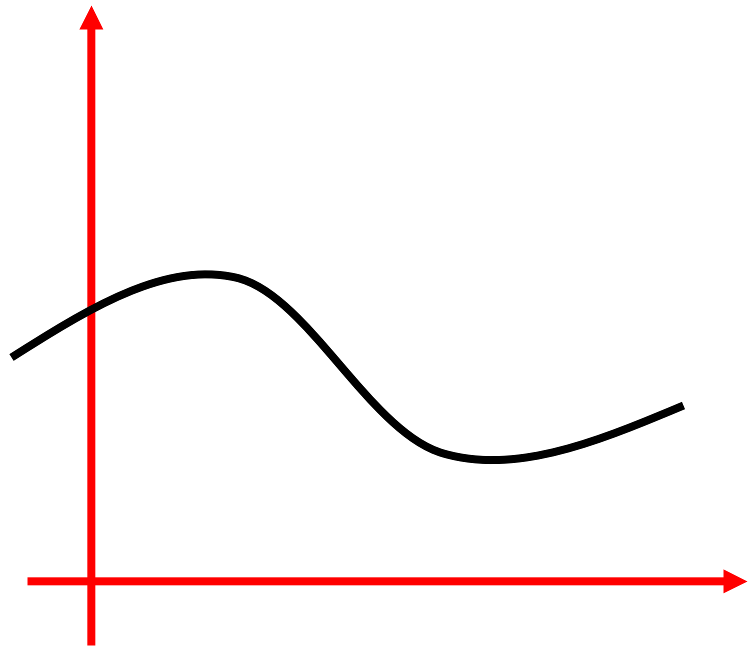
- Bandit linear optimization over R^n
  - "strong" assumptions
  - Regret $O(T^{2/3} n)$
- Bandit problems for optimizing Lipschitz functions
  - "weak" assumptions
  - **Regret $O(C(n) T^{n/(n+1)})$**
  - Curse of dimensionality!

- Today: Flexible (Bayesian) approach for encoding assumptions about function complexity

What if we believe, the function looks like:



Piece-wise linear?



Analytic?
(∞-diff.'able)

Want flexible way to encode assumptions about functions!

Bayesian inference

- Two Bernoulli variables A(larm), B(urglar)
- P(B=1) = 0.1; P(A=1 | B=1)=0.9; P(A=1 | B=0)=0.1
- What is P(B | A)?

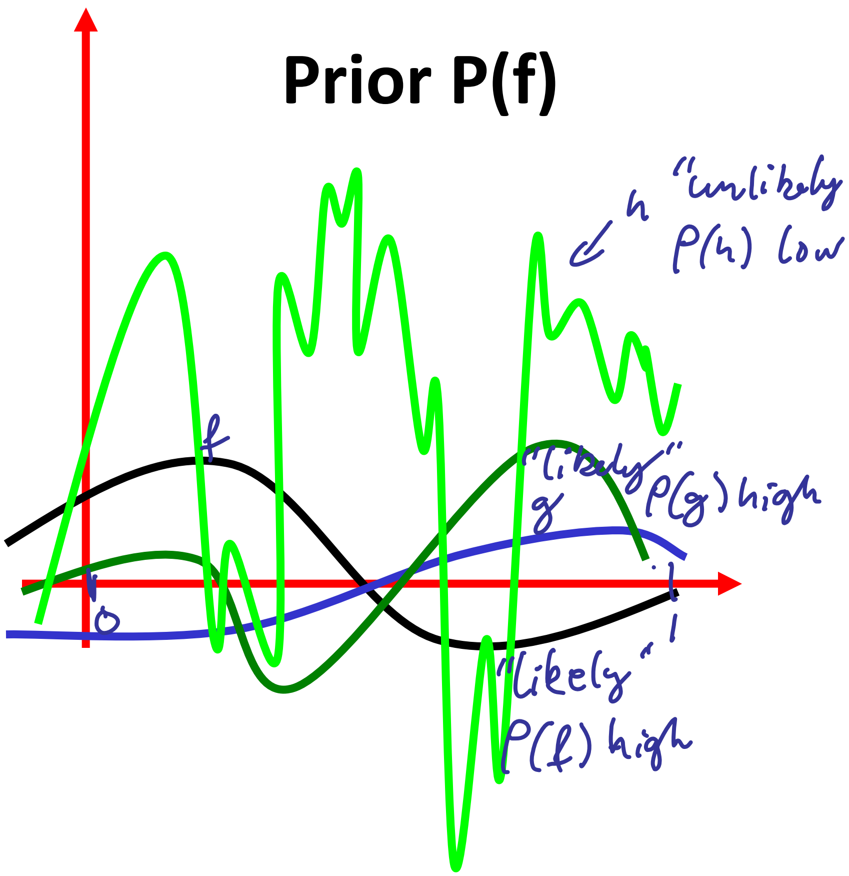
$$P(B=1 \mid A=1) = \frac{P(B=1, A=1)}{\underbrace{P(A=1)}_{?}} = \frac{P(A=1 \mid B=1) \cdot P(B=1)}{P(A=1 \mid B=1)\,P(B=1) + P(A=1 \mid B=0)\,P(B=0)}$$

- P(B) “prior”
- P(A | B) “likelihood”
- P(B | A) “posterior”

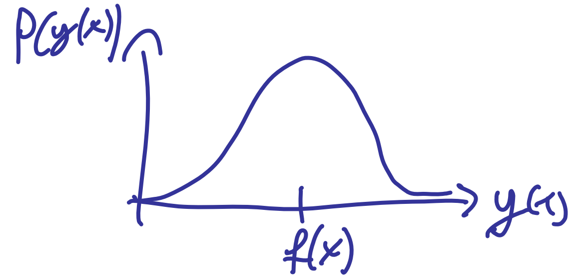
A Bayesian approach

- Bayesian models for **functions**

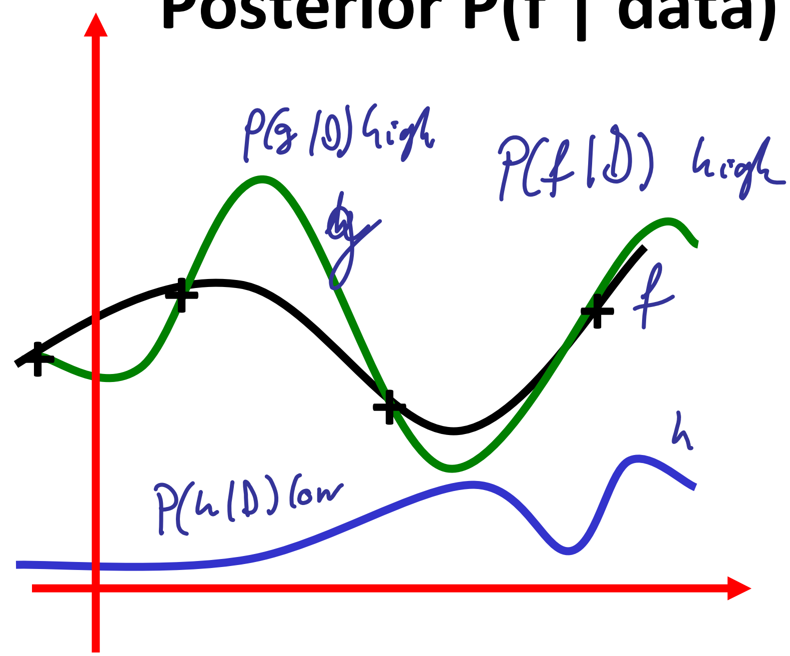
Prior P(f)



Likelihood P(data | f)



Posterior P(f | data)



- Uff… Why is this useful?

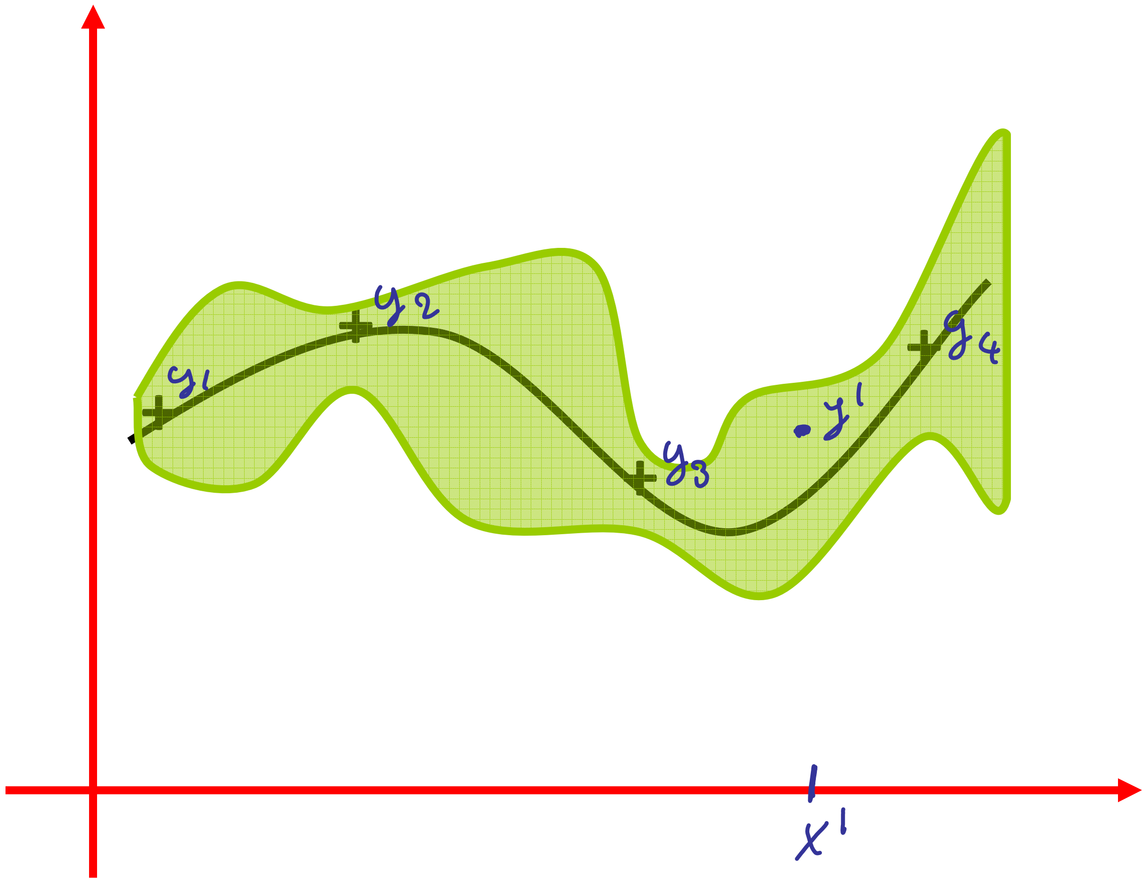
Probability of data

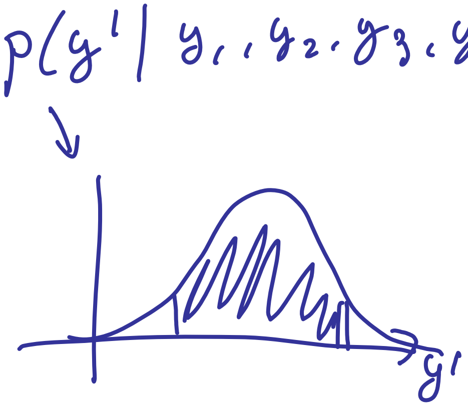
- $P(y_1,\ldots,y_k) =$ $\int p(f, y_1, \ldots, y_k)\, dp(f) = \int p(f)\, p(y_1 \ldots y_k \mid f)\, dp(f)$

- Can compute

$P(y' \mid y_1,\ldots,y_k) =$ $\dfrac{P(y', y_1 \ldots y_k)}{P(y_1 \ldots y_k)}$

Regression with uncertainty about predictions!





How can we do this?

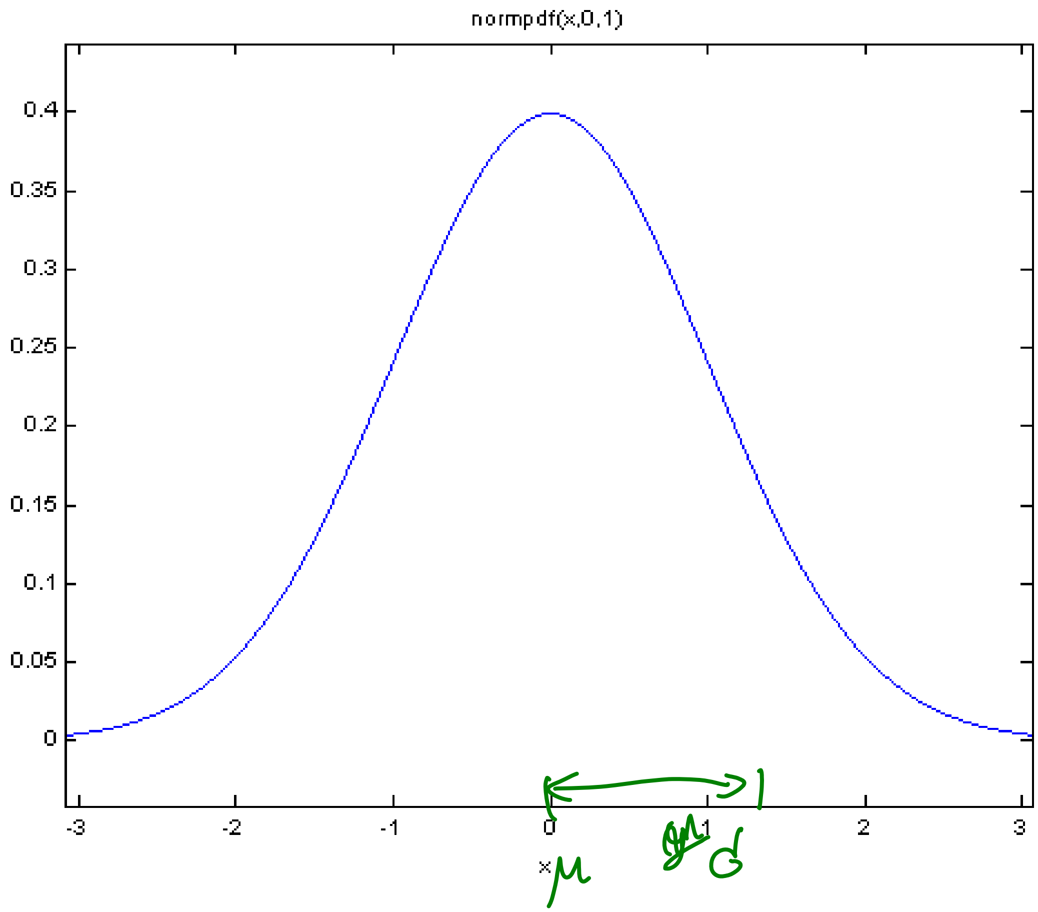
- Want to compute $P(y' \mid y_1,\ldots,y_k)$

$$P(y_1,\ldots,y_k) = \int P(f, y_1,\ldots,y_k)\, df$$

- Horribly complicated integral?? ☹
- Will see how we can compute this
  - (more or less) efficiently
  - In closed form!

… if P(f) is a **Gaussian Process**

Gaussian distribution



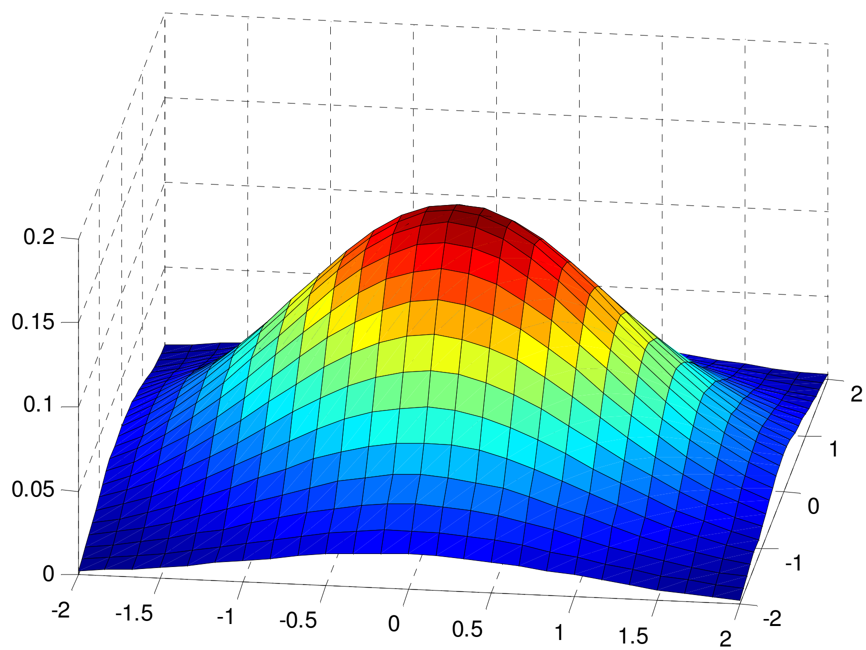
- σ = Standard deviation
- μ = mean

$$\frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)$$

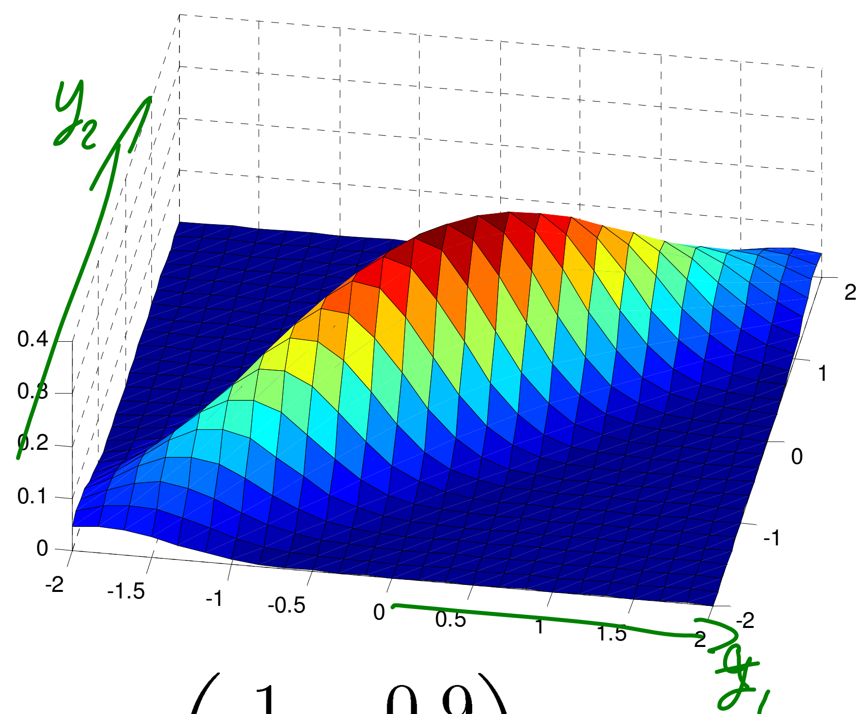
Bivariate Gaussian distribution

$$\frac{1}{2\pi\sqrt{|\Sigma|}}\exp\left(-\frac{1}{2}(y-\mu)^T\Sigma^{-1}(y-\mu)\right) \qquad \Sigma = \begin{pmatrix}\sigma_1^2 & \sigma_{12}\\ \sigma_{21} & \sigma_2^2\end{pmatrix} \qquad \mu = \begin{pmatrix}\mu_1\\ \mu_2\end{pmatrix}$$

σ^2



$$\Sigma = \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}$$



$$\Sigma = \begin{pmatrix}1 & 0.9\\ 0.9 & 1\end{pmatrix}$$

Multivariate Gaussian distribution

$$\mathcal{N}(y; \Sigma, \mu) = \frac{1}{(2\pi)^{n/2}\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(y-\mu)^T \Sigma^{-1} (y-\mu)\right)$$

$$\Sigma = \underbrace{\begin{pmatrix} \sigma_1^2 & \sigma_{12} & \dots & \sigma_{1n} \\ \vdots & & & \vdots \\ \sigma_{n1} & \sigma_{n2} & \dots & \sigma_n^2 \end{pmatrix}}_{\Sigma \in \mathbb{R}^{n \times n}} \qquad \mu = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{pmatrix} \qquad y \in \mathbb{R}^n$$

- Joint distribution over n random variables $P(Y_1, \ldots Y_n)$
- $\sigma_{jk} = \underline{E[\,(Y_j - \mu_j)\,(Y_k - \mu_k)\,]}$
- Y_j and Y_k independent $\Leftrightarrow$ $\sigma_{jk} = 0$

$\sigma_{jk} > 0$: high values of Y_j $\Rightarrow$ high values of Y_k

only true for Gaussians!

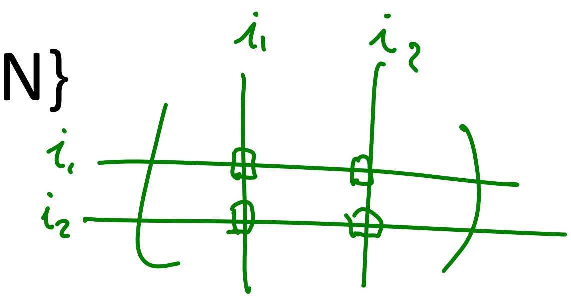
Marginalization

- Suppose $(Y_1,\ldots,Y_n) \sim N(\mu, \Sigma)$
- What is $P(Y_1)$??

$$\Sigma = \begin{pmatrix} \sigma_1^2 & \sigma_{12} & \ldots & \sigma_{1n} \\ \vdots & & & \vdots \\ \sigma_{n1} & \sigma_{n2} & \ldots & \sigma_n^2 \end{pmatrix} \quad \mu = \begin{pmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_n \end{pmatrix}$$

$Y_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$

- More generally: Let $A=\{i_1,\ldots,i_k\} \subseteq \{1,\ldots,N\}$
- Write $Y_A = (Y_{i_1},\ldots,Y_{i_k})$



- $Y_A \sim N(\mu_A, \Sigma_{AA})$

$$\Sigma_{AA} = \begin{pmatrix} \sigma_{i_1}^2 & \sigma_{i_1 i_2} & \ldots & \sigma_{i_1 i_k} \\ \vdots & & & \vdots \\ \sigma_{i_k i_1} & \sigma_{i_k i_2} & \ldots & \sigma_{i_k}^2 \end{pmatrix} \qquad \mu_A = \begin{pmatrix} \mu_{i_1} \\ \mu_{i_2} \\ \vdots \\ \mu_{i_k} \end{pmatrix}$$

Conditioning

- Suppose $(Y_1,\ldots,Y_n) \sim N(\mu, \Sigma)$
- Decompose as (Y_A, Y_B)
- What is $P(Y_A \mid Y_B)$??

$$\mu = \begin{pmatrix} \mu_A \\ \mu_B \end{pmatrix} \quad \Sigma = \begin{pmatrix} \Sigma_{AA} & \Sigma_{AB} \\ \Sigma_{BA} & \Sigma_{BB} \end{pmatrix}$$

- $P(Y_A = y_A \mid Y_B = y_B) = N(y_A; \mu_{A|B}, \Sigma_{A|B})$ where

posterior mean

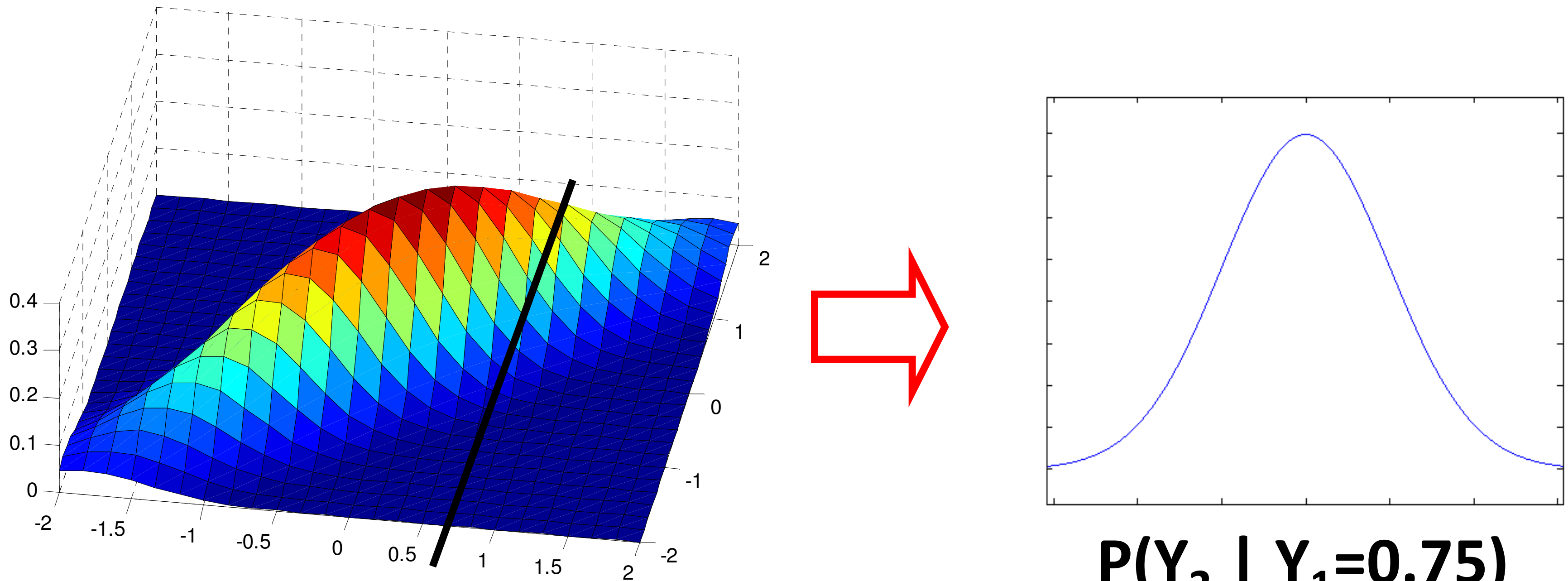
prior mean

$$\mu_{A|B} = \boxed{\mu_A} + \boxed{\Sigma_{AB}\Sigma_{BB}^{-1}}(y_B - \mu_B)$$

$$\Sigma_{A|B} = \Sigma_{AA} - \Sigma_{AB}\Sigma_{BB}^{-1}\Sigma_{BA}$$

- Computable using linear algebra!

Conditioning

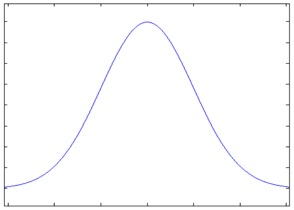


Y_1=0.75

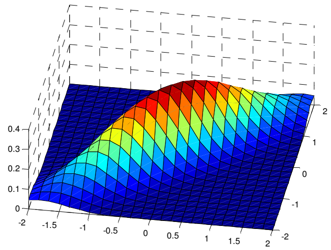
P(Y_2 | Y_1=0.75)

High dimensional Gaussians

- Gaussian



- Bivariate Gaussian



- Multivariate Gaussian

$$\mathcal{N}(y;\Sigma,\mu) = \frac{1}{(2\pi)^{n/2}\sqrt{|\Sigma|}} \exp\left(-\frac{1}{2}(y-\mu)^T\Sigma^{-1}(y-\mu)\right)$$

- Gaussian Process = “∞-variate Gaussian”

Gaussian process

- A Gaussian Process (GP) is a
  - (infinite) set of random variables, indexed by some set V
    i.e., for each $x \in V$ there's a RV Y_x
  - Let $A \subseteq V$, $|A| = \{x_1,\ldots,x_k\} < \infty$
    Then

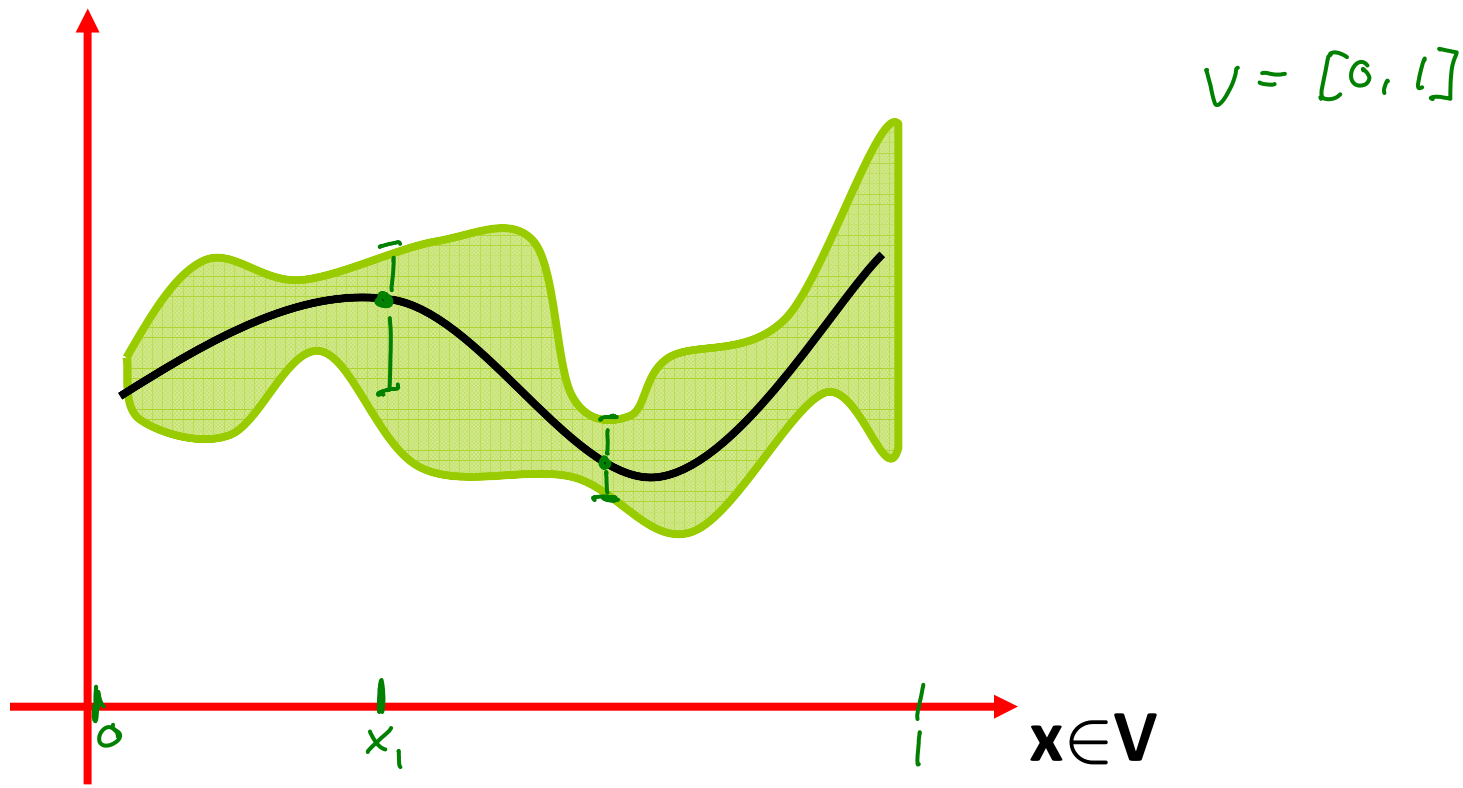
    $$Y_A \sim N(\mu_A, \Sigma_{AA})$$

    where

$$\Sigma_{AA} = \begin{pmatrix} \mathcal{K}(x_1,x_1) & \mathcal{K}(x_1,x_2) & \ldots & \mathcal{K}(x_1,x_n) \\ \vdots & & & \vdots \\ \mathcal{K}(x_k,x_1) & \mathcal{K}(x_k,x_2) & \ldots & \mathcal{K}(x_k,x_k) \end{pmatrix} \qquad \mu_A = \begin{pmatrix} \mu(x_1) \\ \mu(x_2) \\ \vdots \\ \mu(x_k) \end{pmatrix}$$

  - K: $V \times V \to R$ is called **kernel** (covariance) function
    μ: $V \to R$ is called **mean** function

Visualizing GPs



- Typically, only care about "marginals", i.e., P(y) = N(y; μ(x), K(x,x))

Mean functions

- Can encode prior knowledge
- Typically, one simply assumes

$$\mu(x) = 0$$

- Will do that here to simplify notation

Not a strong assumption

Kernel functions

- K must be **symmetric**

  $K(x,x') = K(x',x)$ for all $x, x' \in V$

- K must be **positive definite**

  For all A: Σ_{AA} is positive definite matrix

  Matrix $M \in \mathbb{R}^{n \times n}$ pos. def.

  $\Leftrightarrow \forall x \in \mathbb{R}^n \quad x^T M x \geq 0$

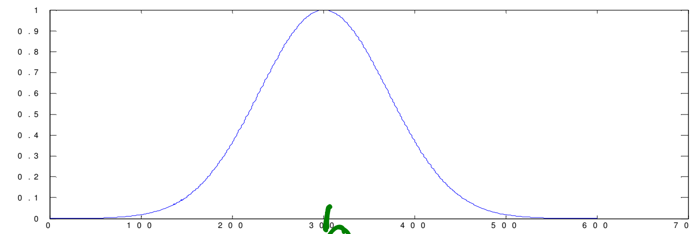
  $\Leftrightarrow$ All eigenvalues > 0

- Kernel function K: assumptions about correlation!

Kernel functions: Examples

- Squared exponential kernel
  $K(x,x') = \exp(-(x-x')^2/h^2)$

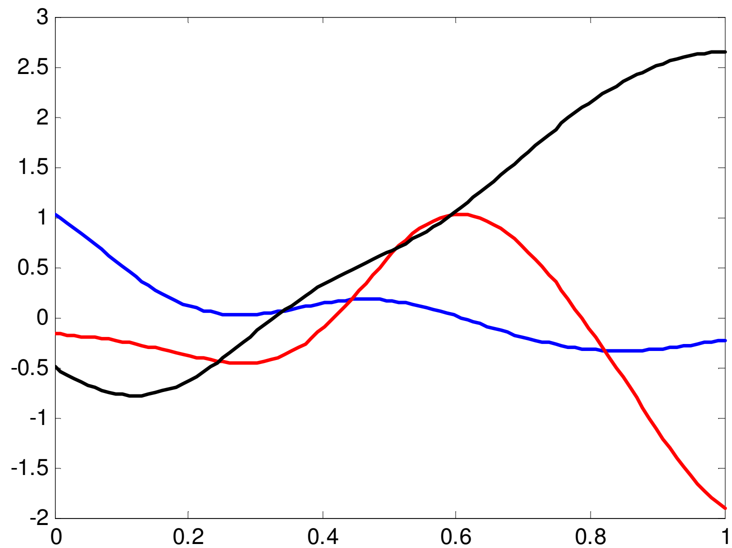
$V = \mathbb{R}$



Distance |x-x'|

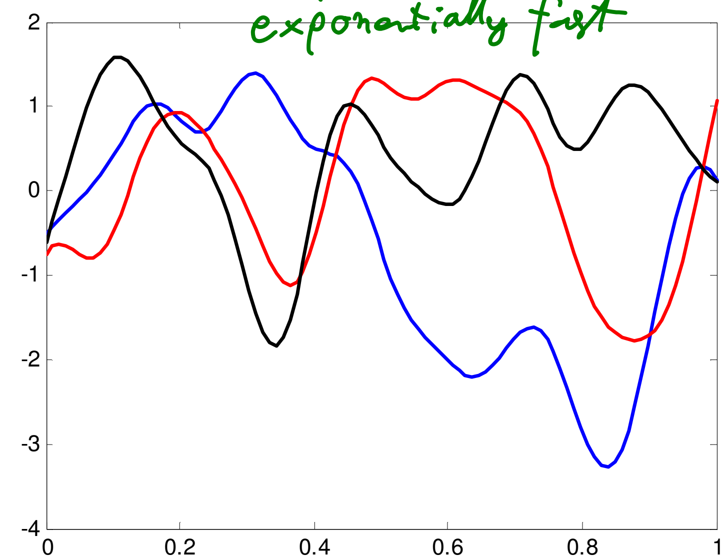
"Correlation decays with distance" exponentially fast

Samples from P(f)



Samples are ∞-differentiable with Prob 1

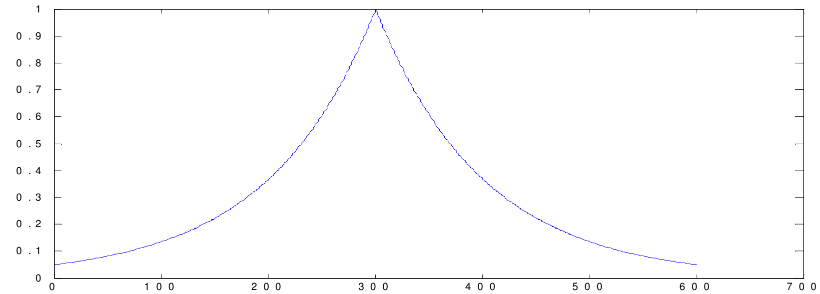
Bandwidth h=.3



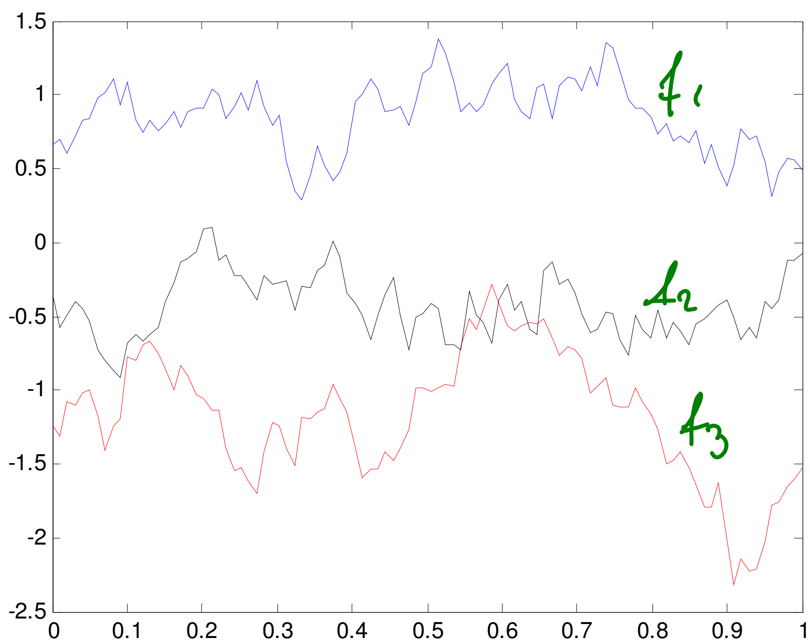
Bandwidth h=.1

Kernel functions: Examples

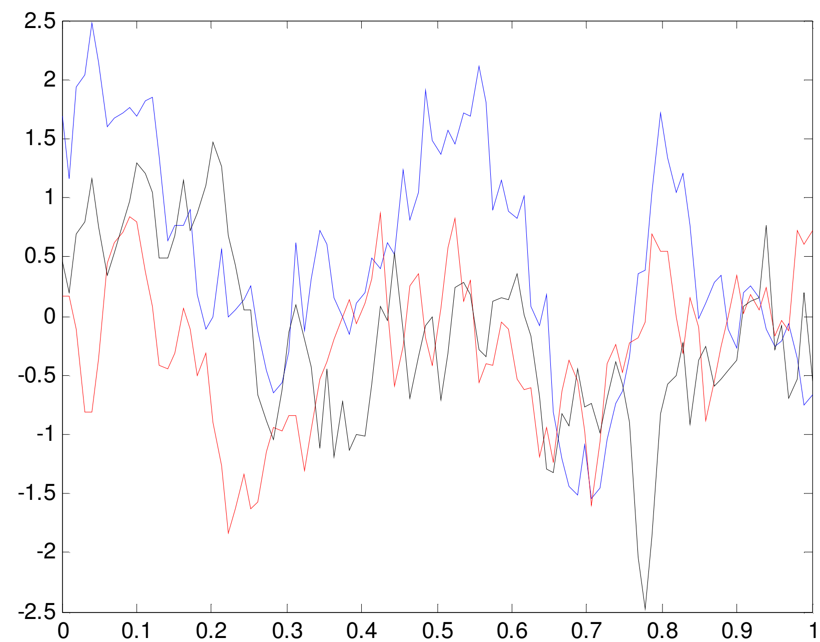
- Exponential kernel
  $K(x,x') = \exp(-|x-x'|/h)$



$f_1 \sim GP(\mu, k)$



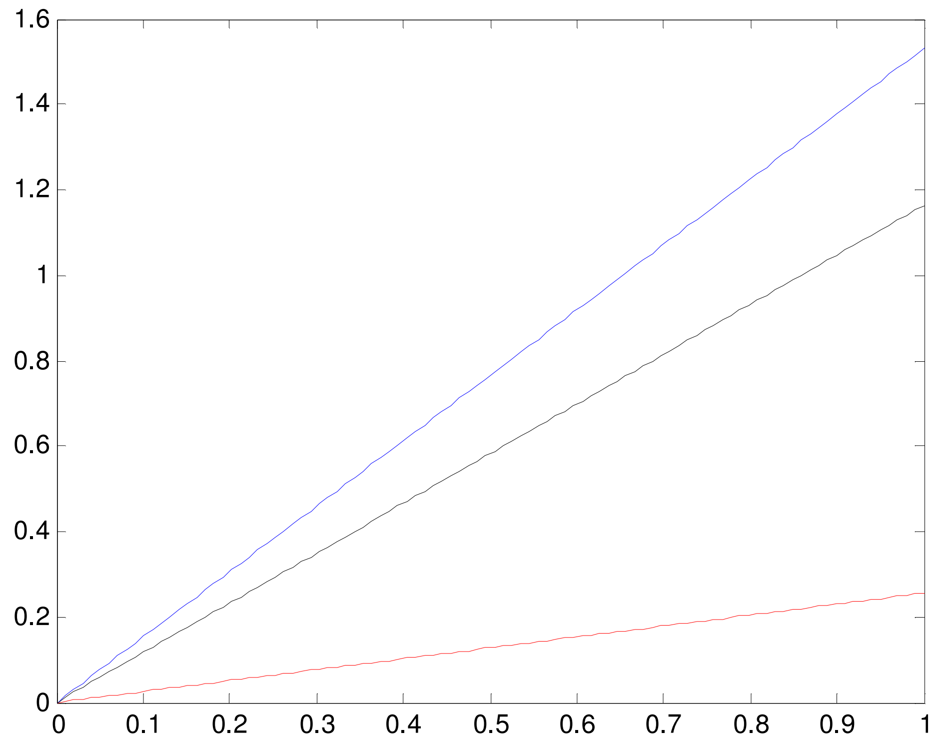
Bandwidth h=1



Bandwidth h=.3

Kernel functions: Examples

- Linear kernel:
  $K(x,x') = x^T x'$

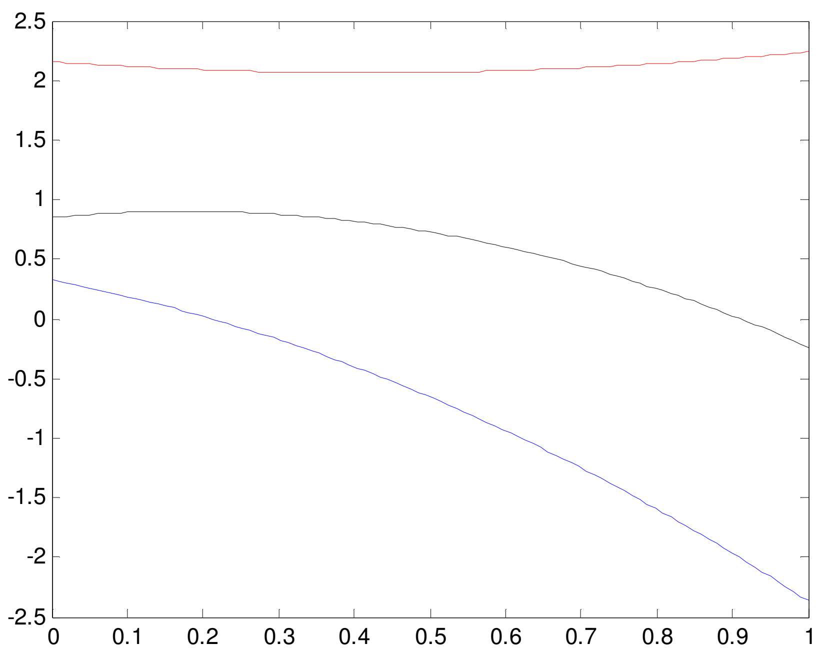


- Corresponds to linear regression!

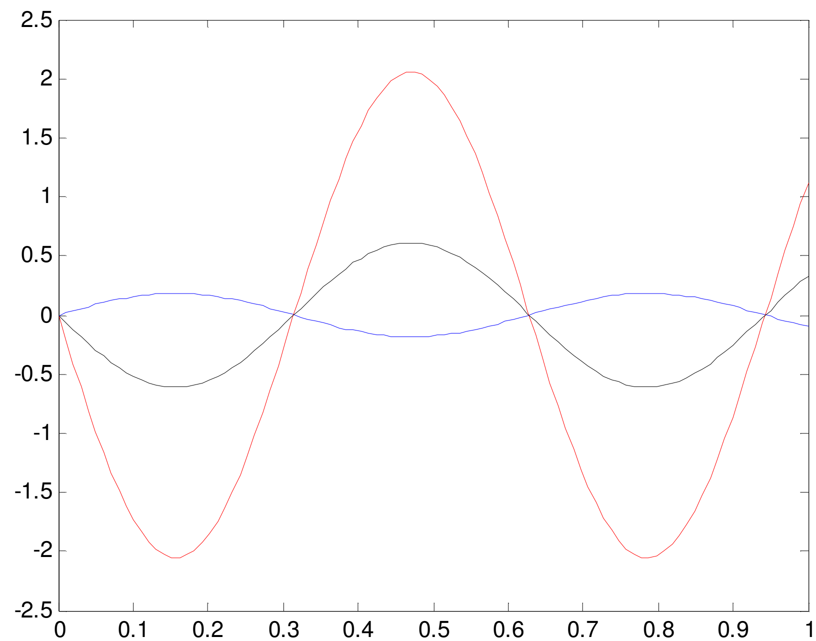
Kernel functions: Examples

- Linear kernel with features:
  $K(x,x') = \Phi(x)^T\Phi(x')$

E.g., $\Phi(x) = [1,x,x^2]$



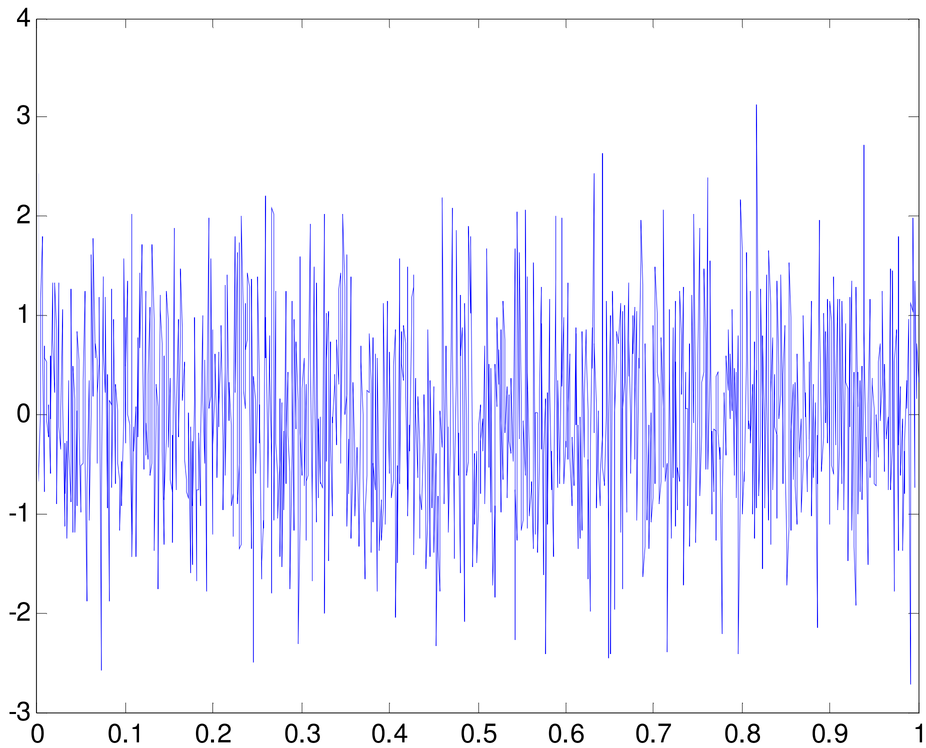
E.g., $\Phi(x) = \sin(x)$



Kernel functions: Examples

- White noise:

  $K(x,x) = 1$; $K(x,x') = 0$ for $x' \neq x$



Constructing kernels from kernels

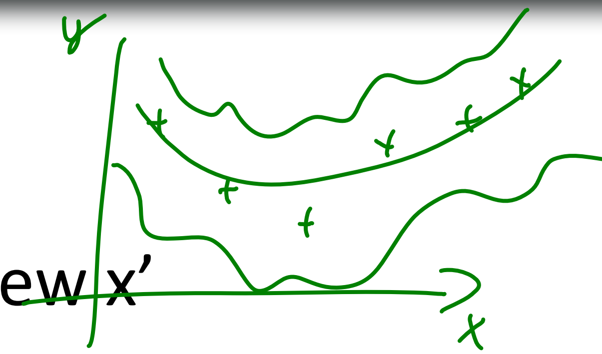
If $K_1(x,x')$ and $K_2(x,x')$ are kernel functions then

$\alpha\, K_1(x,x') + \beta\, K_2(x,x')$ is a kernel for $\alpha,\beta > 0$

$K_1(x,x')*K_2(x,x')$ is a kernel

GP Regression

- Suppose we know kernel function K
- Get data $(x_1,y_1),\ldots,(x_n,y_n)$
- Want to predict $y' = f(x')$ for some new x'



$$P(y' \mid y_1 \ldots y_n)$$

$$\Sigma \in \mathbb{R}^{n+1 \times n+1}$$

1. Construct $P(y', y_1 \ldots y_n) = N(\cdot\,; \mu, \Sigma)$

2. Condition:

$$P(y' \mid y_1 \ldots y_n) = N(y'; \mu_{x'|D}, \sigma^2_{x'|D})$$

$$\mu = \underbrace{\begin{pmatrix} \mu(x') \\ \mu(x_1) \\ \vdots \\ \mu(x_n) \end{pmatrix}}_{=0} \qquad \Sigma = \begin{pmatrix} K(x',x') & K(x',x_1) & \ldots & K(x',x_n) \\ \vdots & & & \\ \vdots & & & \\ K(x_n,x') & \ldots & \ldots & K(x_n,x_n) \end{pmatrix}$$

"Read off" Σ

$$\mu_{x'|D} = \underbrace{\mu_{x'}}_{0} + \Sigma_{x',D}\,\Sigma_{DD}^{-1}(y_D - \mu_D)$$

$$\sigma^2_{x'|D} = \sigma^2_{x'} + \Sigma_{x',D}\,\Sigma_{DD}^{-1}\,\Sigma_{D,x'}$$

Linear prediction

- Posterior mean $\mu_{x`|D} = \underbrace{\Sigma_{x`,D}\Sigma_{D,D}^{-1}}_{\in \mathbb{R}^{1 \times m}} y_D$

  $\Sigma = \begin{pmatrix} \Sigma_{AA} & \Sigma_{AD} \\ \Sigma_{BA} & \Sigma_{BB} \end{pmatrix}$

  $A = \{x'\}, B = D$

- Hence, $\mu_{x`|D} = \sum_{i=1}^{n} w_i \, y_i = w^T y$

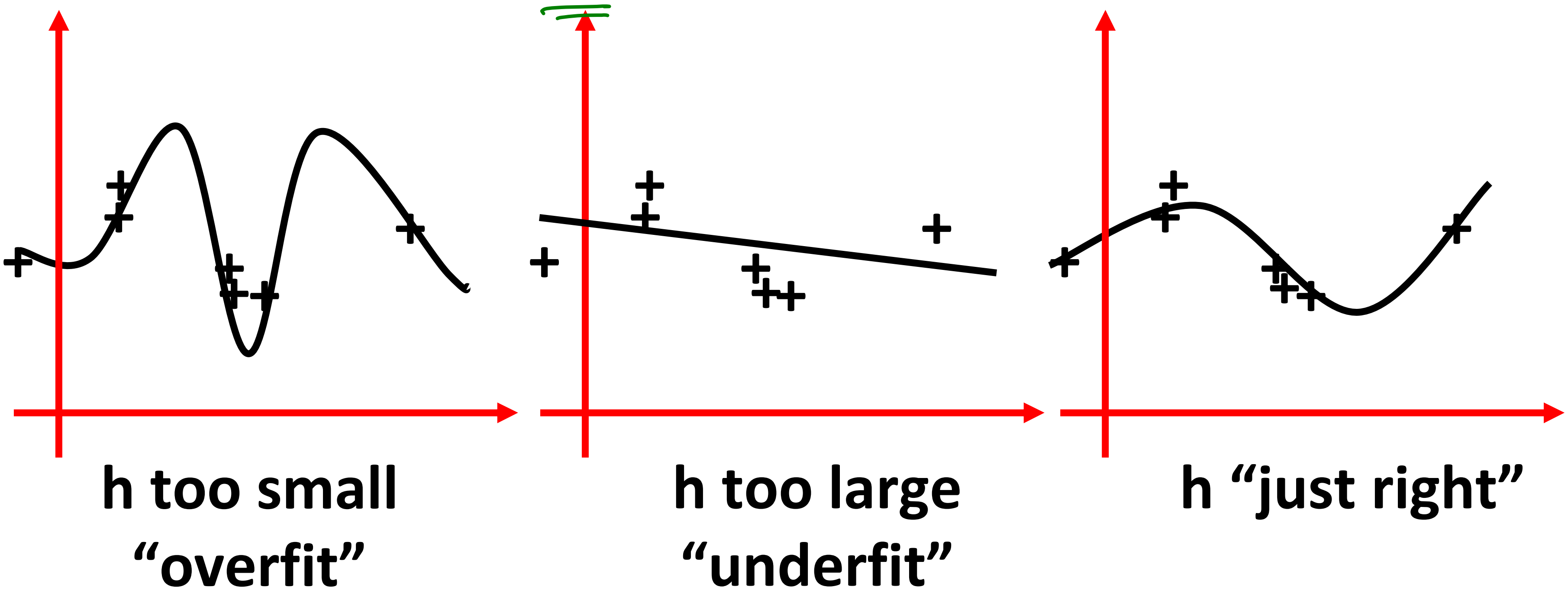
  Regression coefficients

  depends on K

- Prediction $\mu_{x`|D}$ depends **linearly** on inputs y_i!
- For fixed data set $D = \{(x_1,y_1),\ldots,(x_n,y_n)\}$, can *precompute* weights w_i
- Like linear regression, but number of parameters $\boxed{w\_i}$ grows with training data
  - ➔ "Nonparametric regression"
  - ➔ Can fit any data set!! ☺

Learning parameters

- Example: $K(x,x') = \exp(-(x-x')^2/h^2)$
  Need to specify h!



- In general, kernel function has parameters θ
- Want to learn θ from data

Learning parameters

- Pick parameters that make data most likely!

$$P(y; x, \theta) = \frac{1}{(2\pi)^{n/2}\sqrt{|K_\theta|}} \exp\left(-\frac{1}{2} y^T K_\theta^{-1} y\right)$$

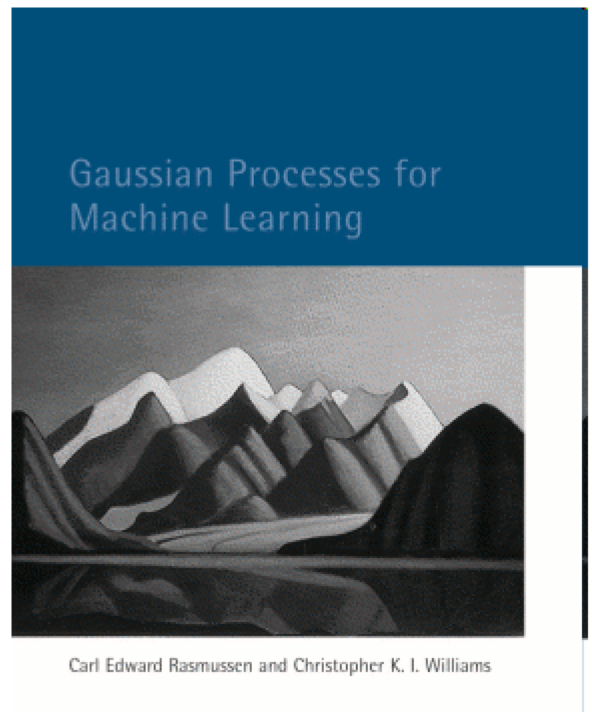
$$\theta^* \operatorname*{argmax}_\theta P(y; x, \theta) = \operatorname*{argmax}_\theta \log P(y; x, \theta)$$

$$= \text{const} \underbrace{-\tfrac{1}{2}\log |K_\theta|}_{\text{complexity}} \underbrace{-\tfrac{1}{2} y^T K_\theta^{-1} y}_{\text{goodness of}}$$

- log P(y | θ) differentiable if K(x,x') is!
  - ➔Can do gradient descent, conjugate gradient, etc.
- Tends to work well (not over- or underfit) in practice!

Matlab demo

- [Rasmussen & Williams, Gaussian Processes for Machine Learning]
- http://www.gaussianprocess.org/gpml/



Gaussian process

- A Gaussian Process (GP) is a
  - (infinite) set of random variables, indexed by some set V
    i.e., for each $x \in V$ there's a RV Y_x
  - Let $A \subseteq V$, $|A| = \{x_1,\ldots,x_k\} < \infty$
    Then
    $$Y_A \sim N(\mu_A, \Sigma_{AA})$$
    where

$$\Sigma_{AA} = \begin{pmatrix} \mathcal{K}(x_1,x_1) & \mathcal{K}(x_1,x_2) & \ldots & \mathcal{K}(x_1,x_n) \\ \vdots & & & \vdots \\ \mathcal{K}(x_k,x_1) & \mathcal{K}(x_k,x_2) & \ldots & \mathcal{K}(x_k,x_k) \end{pmatrix} \quad \mu_A = \begin{pmatrix} \mu(x_1) \\ \mu(x_2) \\ \vdots \\ \mu(x_k) \end{pmatrix}$$

  - K: $V \times V \to R$ is called **kernel** (covariance) function
    μ: $V \to R$ is called **mean** function

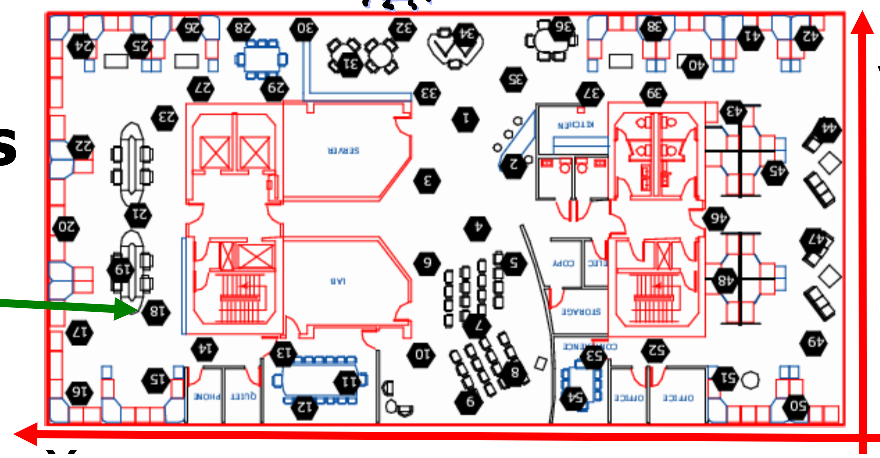
GPs over other sets

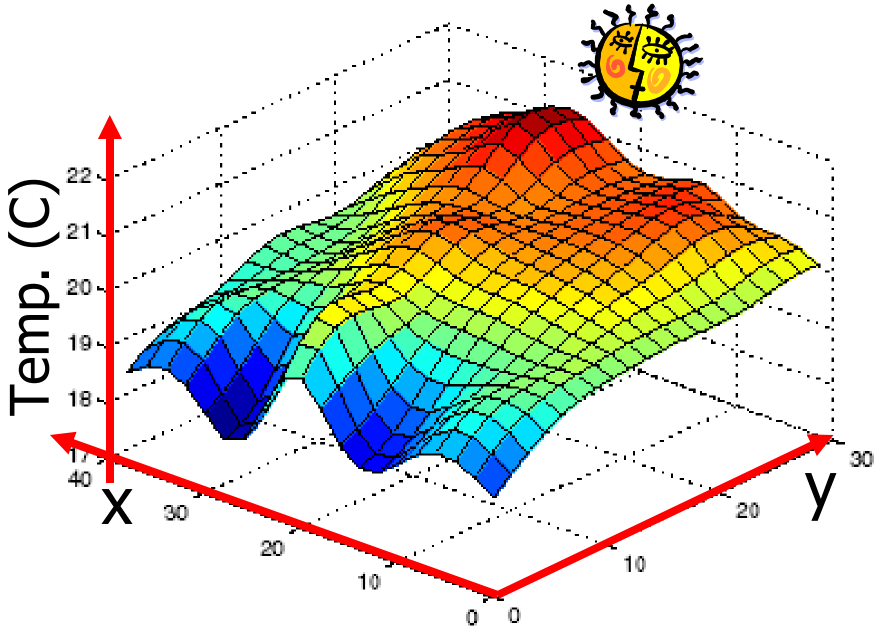
- GP is collection of random variables, indexed by set V
- So far: Have seen GPs over V = R
- Can define GPs over
  - Text (strings)
  - Graphs
  - Sets
  - …
- Only need to choose appropriate kernel function

Example: Using GPs to model spatial phenomena

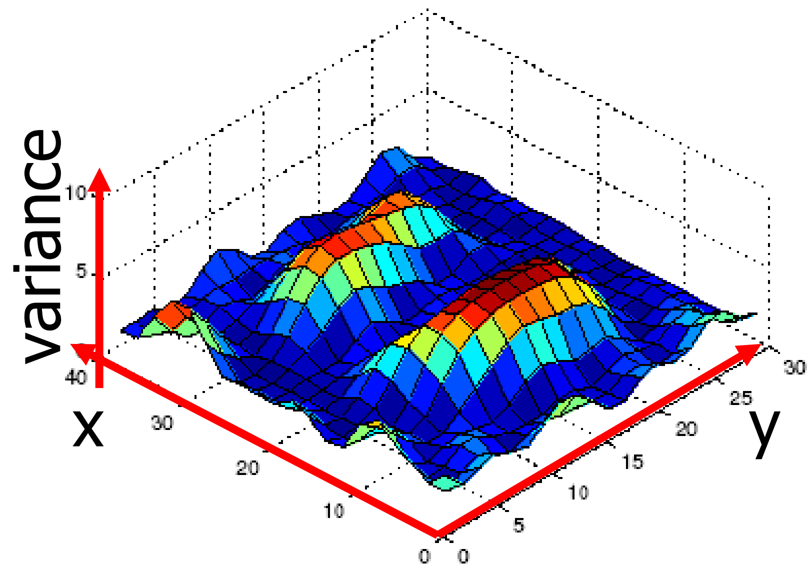
Real deployment of temperature sensors

measurements from 52 sensors (black dots)





Predicted temperature throughout the space



Other extensions (won't cover here)

- GPs for classification
  - Nonparametric generalization of logistic regression
  - Like SVMs (but give confidence on predicted labels!)
- GPs for modeling non-Gaussian phenomena
  - Model count data over space, …
- Active set methods for fast inference
- …

Still active research area in machine learning

Bandits = Noisy function optimization

- We are given black box access to function f

$$\mathbf{x} \longrightarrow \boxed{\;\mathbf{f}\;} \longrightarrow \mathbf{y = f(x) + noise}$$

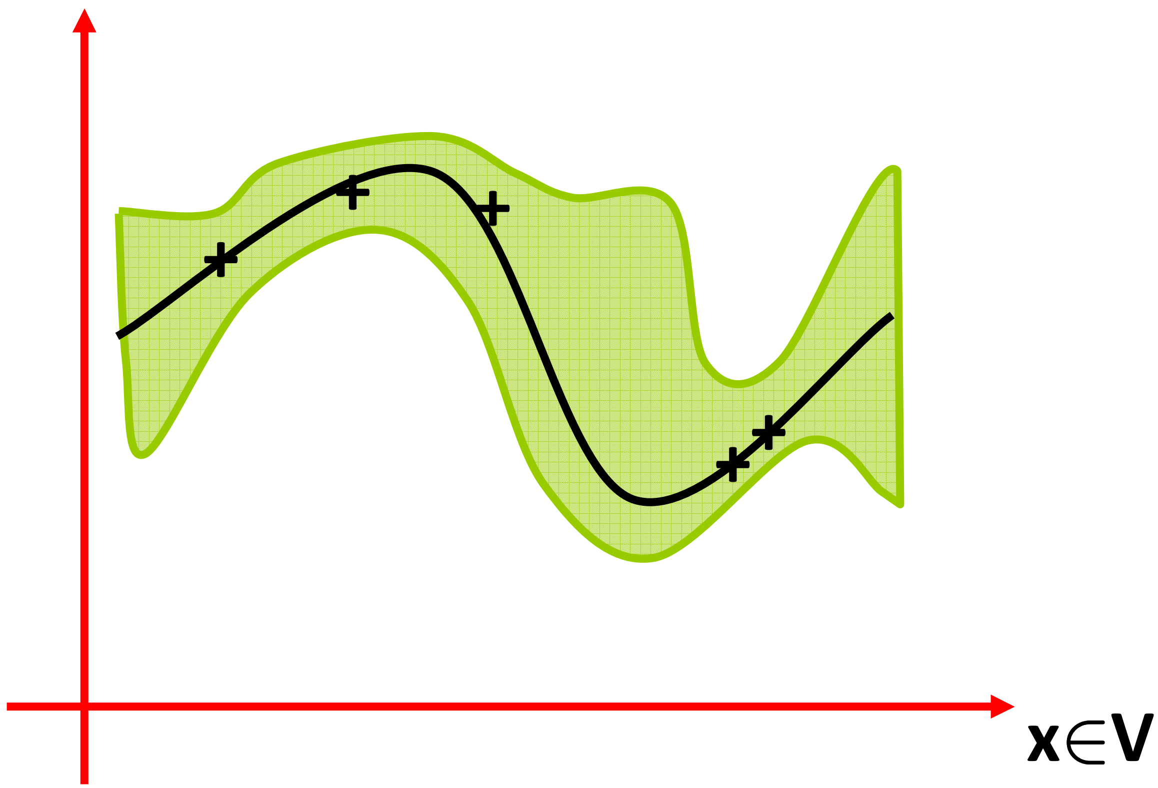
- Evaluating f is very expensive
- Want to (quickly) find $x^* = \text{argmax}_x\, f(x)$
- Idea: Assume f is a sample from a Gaussian Process!

  ➔ Gaussian Process optimization

  (a.k.a.: Response surface optimization)

Upper confidence bound approach

- $UCB(x \mid D) = \underbrace{\mu(x \mid D)} + \underbrace{2 * \sigma(x \mid D)}$
- Pick point $x^* = \text{argmax}_x \; UCB(x \mid D)$



Matlab demo

Properties

- Implicitly trades off exploration and exploitation
- Exploits prior knowledge about function
- Can converge to optimal solution very quickly! ☺
- Seems to work well in many applications

- Can perform poorly if our prior assumptions are wrong ☹

What you need to know

- GPs =
  - Nonparametric generalization of linear regression
  - Flexible ways to encode prior assumptions about mean payoffs
- Definition of GPs
- Properties of multivariate Gaussians (marginalization, conditioning)
- Gaussian Process optimization
  - Combination of regression and optimization
  - Use confidence bands for selecting samples