ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Data Mining
Learning from Large Data Sets

Lecture 10 – Multi-armed bandits

263-5200-00L

Andreas Krause

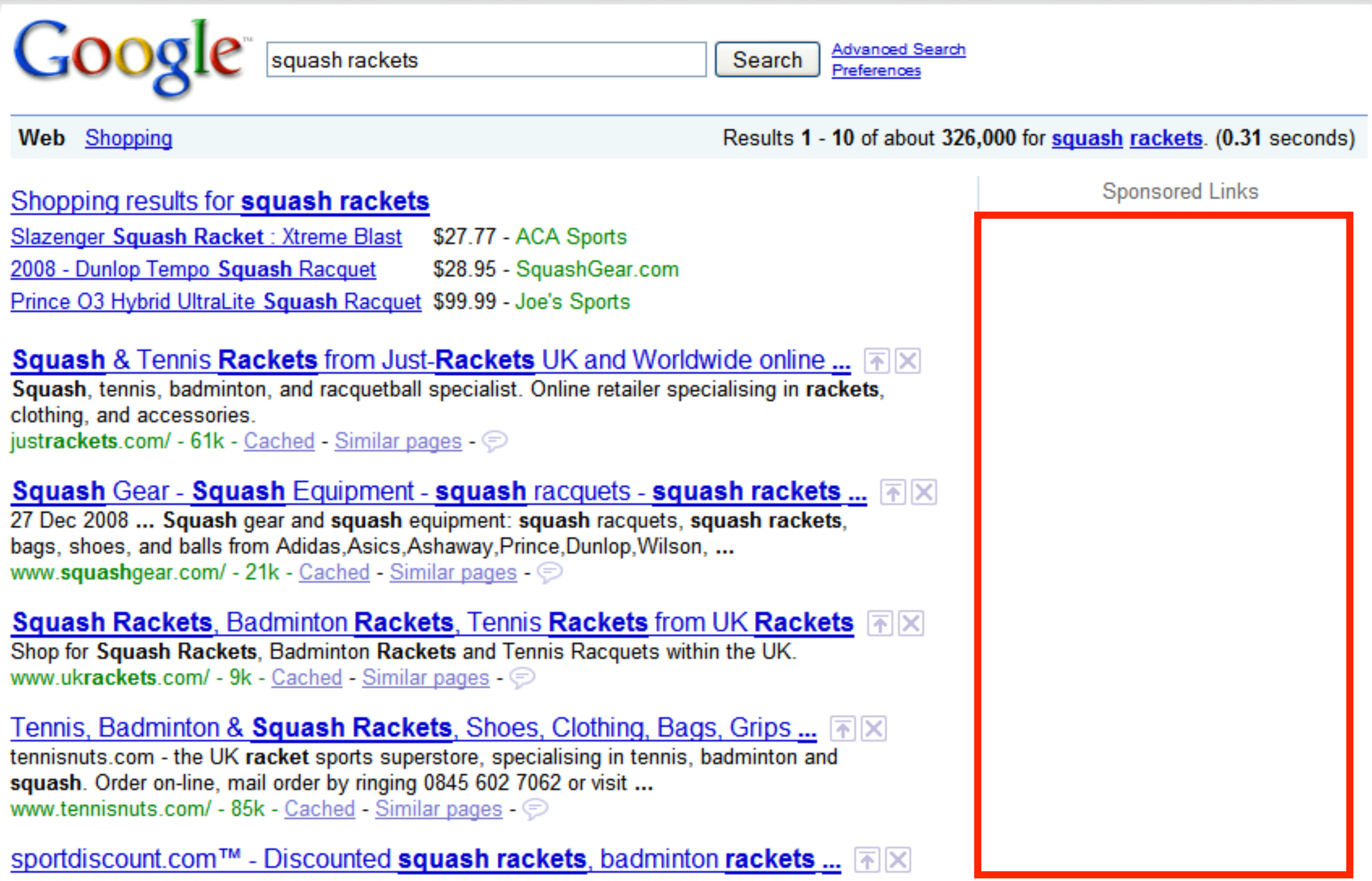
Announcements

- Homework 5 out tomorrow

Course organization

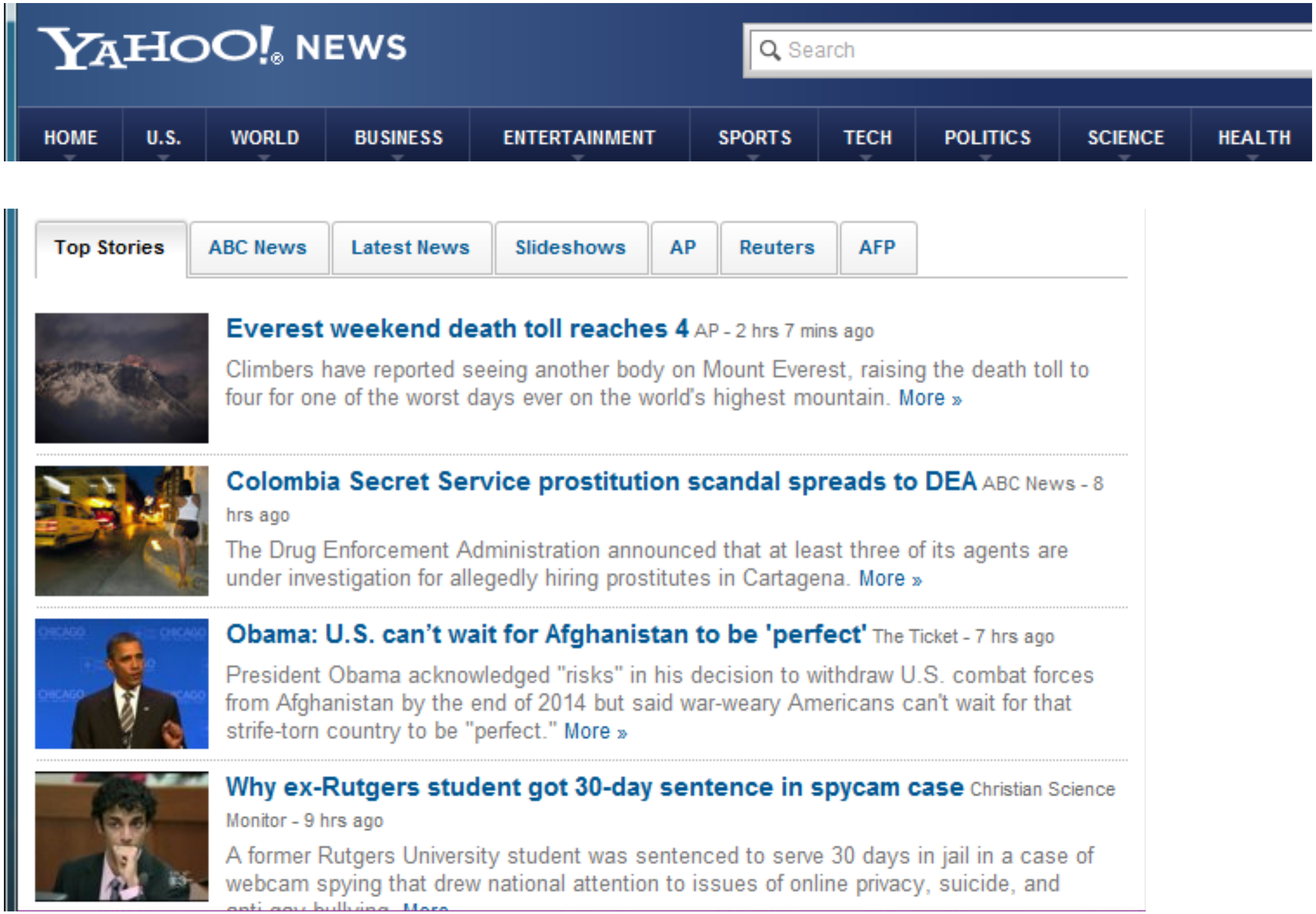
- **Retrieval**
  - Given a query, find "most similar" item in a large data set
  - Determine relevance of search results
  - *Applications*: GoogleGoggles, Shazam, …
- **Supervised learning** (Classification, Regression)
  - Learn a concept (function mapping queries to labels)
  - *Applications*: Spam filtering, predicting price changes, …
- **Unsupervised learning** (Clustering, dimension reduction)
  - Identify clusters, "common patterns"; anomaly detection
  - *Applications*: Recommender systems, fraud detection, …
- **Interactive data mining**
  - Learning through experimentation / from limited feedback
  - *Applications*: Online advertising, opt. UI, learning rankings, …

Sponsored search



Which ads should be displayed to maximize revenue?

Which news should we display?



Sponsored search

- *Earlier approaches*: Pay by impression
  Go with highest bidder

  $\max_i q_i$

  ignores "effectiveness" of ads

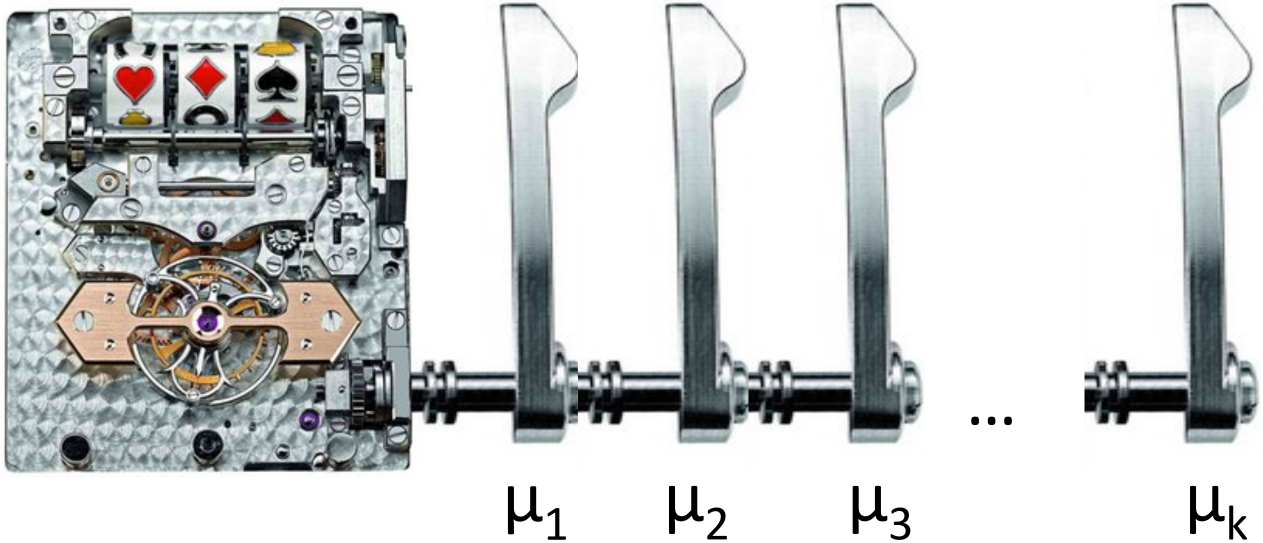
- *Key idea*: Pay per click!
  Maximize revenue over all ads i

  $E[\text{revenue}_i] = P(\text{click}_i \mid \text{query})\, q_i$

Don't know! Need to gather information about effectiveness! (arrow to $P(\text{click}_i \mid \text{query})$)

Bid for ad i (pay per click, known) (arrow to q_i)

k-armed bandits



- Each arm i
  - wins (reward = 1) with fixed (unknown) probability μ_i
  - wins (reward = 0) with fixed (unknown) probability $1-\mu_i$
- All draws are independent given $\mu_1, \ldots, \mu_k$
- How should we pull arms to maximize total reward?

Stochastic k-armed bandits

- Discrete set of k choices
- Each choice (arm) i associated with **unknown** probability distribution P_i supported in [0,1]
- Play game for T rounds
- In each round t, we pick an arm i, and obtain an random sample X_t from P_i independent of previous samples
- Our goal is to maximize $\sum_{t=1}^{T} X_t$

Online optimization with limited feedback

Choices	X_1				
a_1					
a_2	0				
…					
a_n					

Reward → Time

Total: $\sum_t X_t \rightarrow \max$

- Like in online (supervised) learning:
  - Have a make a choice each time
- Unlike online learning:
  - Only receive information about chosen action

Solving the bandit problem

- Optimal policy can be found for k independent arms with known prior distribution [Gittins '79]
  - Terribly hard to analyze any more complex settings
- Modern view: "No-regret" instead of optimality
  - Often easier to analyze!

Performance metric: *Regret*

- Let μ_i be the mean of P_i
- Payoff of best arm: $\mu^* = \max_i \mu_i$
- Let $i_1, \ldots, i_T$ be the sequence of arms pulled
- Instantaneous regret at time t: $r_t = \mu^* - \mu_{i_t}$
- Total regret: $$R_T = \sum_{t=1}^{T} r_t$$

- Typical goal: Want allocation strategy that guarantees

$R_T/T \rightarrow 0$ as $T \rightarrow \infty$

Allocation strategies

- If we knew the mean payoffs, which arm would we pull?

Pick $\arg\max_i \mu_i$

- What if we only care about estimating the payoffs?

Pick each choice equally often, $\frac{T}{k}$

Estimate $\hat{\mu}_i = \frac{k}{T} \sum_{j=1}^{T/k} X_{i,j}$

Regret: $R_T = T \frac{1}{k} \sum_{i=1}^{k} (\mu^* - \mu_i)$

Exploration—Exploitation Tradeoff

- Need to trade off exploration (gathering data about payoffs) and exploitation (making choices based on data already gathered)

Exploration—Exploitation Tradeoff

- **For t=1:T**
  - Set $\varepsilon_t = \mathcal{O}(1/t)$
  - With probability ε_t: **Explore** by picking arm uniformly at random
  - With probability $1 - \varepsilon_t$: **Exploit** by picking arm with highest empirical mean payoff

- **Theorem** [Auer et al '02]
  For suitable choice of ε_t it holds that

$$R_T = O(k \log T) \quad \Rightarrow \frac{R_T}{T} = O\left(\frac{k \log T}{T}\right)$$

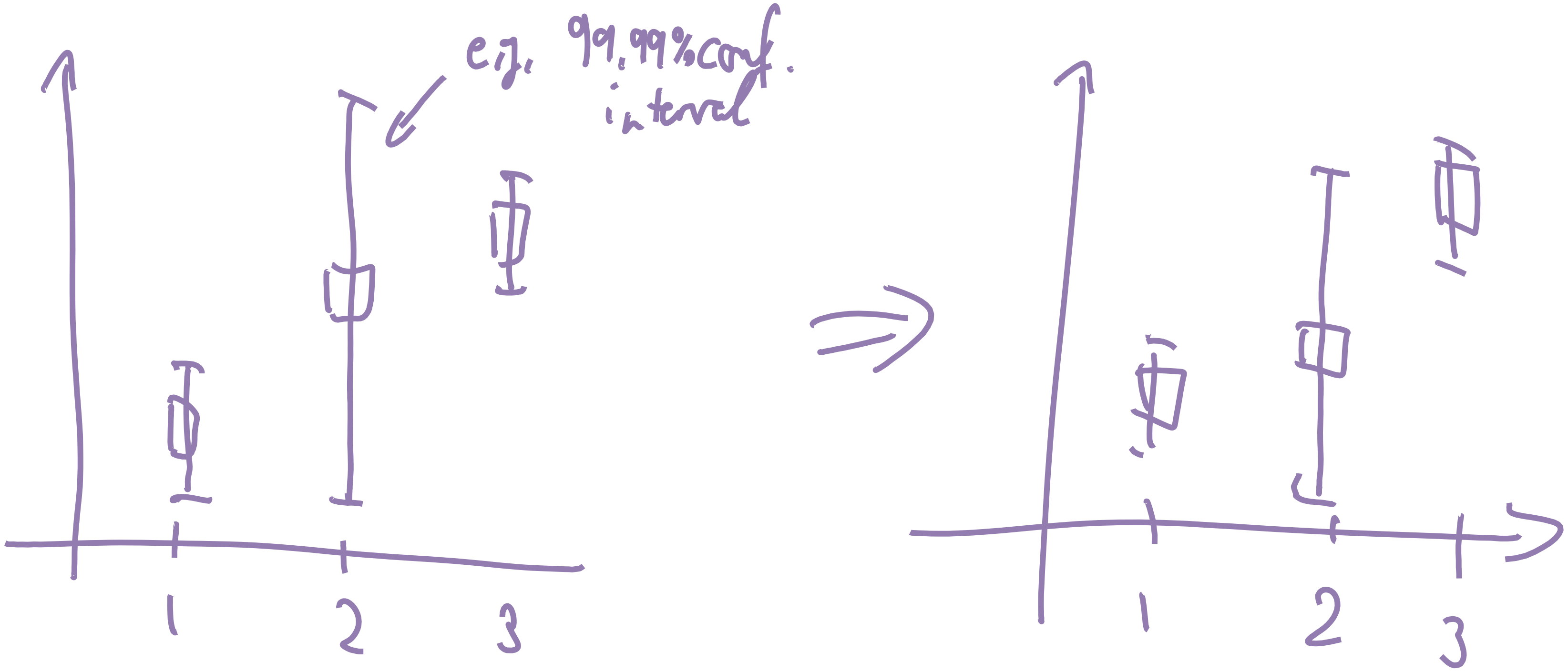
Issues with epsilon greedy

- "Not elegant": Algorithm explicitly distinguishes between exploration and exploitation
- More importantly: Exploration chooses clearly suboptimal choices with equal probability

Comparing arms

- Suppose have done some experiments
  - Arm 1: .1 .2 .1 .3 0 .2 .1 .2
  - Arm 2: .6
  - Arm 3: .7 .8 .6 .8 .7 .9 .8 .7
- Means:
  - Arm 1: .15, Arm 2: .6, Arm 3: .75
- Which arm would you pick next?
- **Idea**: Not just look at mean, but also confidence!

Upper confidence based selection



Calculating confidence bounds

- Suppose we fix arm i
- Let $Y_1,\ldots,Y_m$ be the payoffs of arm i in the first m trials
  - By assumption, they are indepenent trials with distribution $P_i(Y)$
- Mean payoff: $\mu = \mathbb{E}[Y]$
- Our estimate:

$$\hat{\mu}_m = \frac{1}{m}\sum_{\ell=1}^{m} Y_\ell$$

- Want to obtain b such that w.h.p. $|\mu - \hat{\mu}_m| \leq b$
- Also want and b to be as small as possible (why?)

- How can we bound P($|\mu - \hat{\mu}_m| \leq b$)?

Hoeffding's inequality

- Let $X_1,\ldots,X_m$ be i.i.d. random variables taking values in [0,1]

$$\mu = \mathbb{E}[X] \qquad\qquad \hat{\mu}_m = \frac{1}{m}\sum_{\ell=1}^{m} X_\ell$$

- Then $$P(|\mu - \hat{\mu}_m| \geq b) \leq 2\exp\left(-2b^2 m\right) = \delta$$

$b = \frac{c}{\sqrt{m}}$ $\qquad$ $-2b^2m = -2c^2$

How large should c be? $\qquad e^{-2c^2} \leq \delta/2$

$\Rightarrow -2c^2 \leq \ln \delta/2$

$\Rightarrow c^2 \geq \frac{1}{2}\ln\frac{2}{\delta}$

The UCB1 algorithm [Auer et al '02]

- Set $\hat{\mu}_1 = \cdots = \hat{\mu}_k = 0$ $\quad n_1 = \cdots = n_k = 0$
- For t = 1:T
  - For each arm i calculate $UCB(i) = \hat{\mu}_i + \sqrt{\frac{2 \ln t}{n_i}}$
  - Pick arm $j = \arg\max_i UCB(i)$ and observe y_t
  - Set $n_j \leftarrow n_j + 1$ and $\hat{\mu}_j \leftarrow \hat{\mu}_j + \frac{1}{n_j}(y_t - \hat{\mu}_j)$
- "Optimism in the face of uncertainty"

Performance of UCB

- Theorem [Auer et al 2002]
  - Suppose the optimal mean payoff is $\mu^* = \max_i \mu_i$

    and for each arm let $\Delta_i = \mu^* - \mu_i$
  - Then it holds that

$$\mathbb{E}[R_T] = \left[8 \sum_{i:\mu_i < \mu^*} \left(\frac{\ln T}{\Delta_i} \right) \right] + \left(1 + \frac{\pi^2}{3} \right) \left(\sum_{i=1}^{k} \Delta_i \right)$$

$O(k \ln T)$ $\qquad$ $O(k)$

$$\Rightarrow O\left(\frac{R_T}{T}\right) = \left(\frac{k \ln T}{T}\right)$$

Summary so far

- k-armed bandit problem as a formalization of the exploration-exploitation tradeoff
- Analog of online optimization (e.g., online SVM), but with limited feedback
- Simple algorithms are able to achieve no regret
  - Epsilon-greedy
  - Upper confidence sampling

Applications of bandit algorithms

- Clinical trials
- Matching markets
- Asset pricing
- Adaptive routing
- Computer Go

- Data mining:
  - Online advertising
  - Scheduling web crawlers
  - Optimizing user interfaces
  - Learning to optimize relevance
  - …

Extensions

- Infinite-armed bandits
- Dueling bandits
- Contextual bandits
- Bandits in metric spaces
- Mortal bandits
- Restless bandits
- Bandit slates
- …

Challenges in recommendation

- Number of recommendations *k* to choose from large
  - Similar ads → similar click-through rates!

- Performance depends on query / context
  - Similar queries → similar click-through rates!

- Need to compile sets of *k* recs. (instead of only one)
  - Similar sets → similar click-through rates!

- **Key question: How do we model and exploit "similarity"??**

Infinite-armed bandits

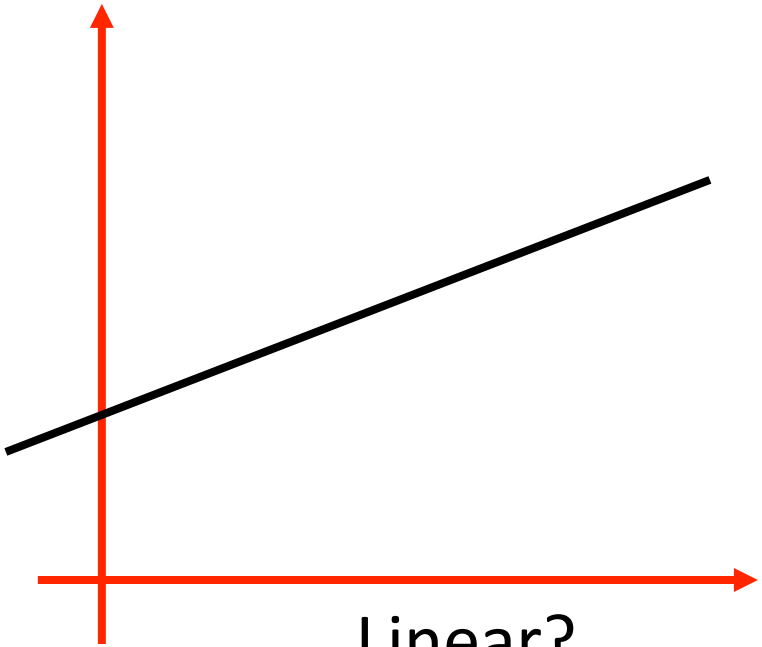
μ_1 μ_2 μ_3 … μ_k μ_∞

- In many applications, number of arms is **huge** (sponsored search, parameter optimization, learning relevance of web pages)
- May not be able to try each arm even once
- Need assumptions on how payoffs are related!

Stochastic ∞-armed bandits

- (Possibly infinite) Set X of choices
- Class *F* of functions on X
- Each choice x in X associated with (unknown) probability distribution P_x supported in [0,1] with means $\mu_x = f(x)$ for some $f \in F$
- Play game for T rounds
- In each round t, we pick an arm x, and obtain an random sample Y_t from P_x independent of previous samples
- Our goal is to maximize $\sum_{t=1}^{T} Y_t$

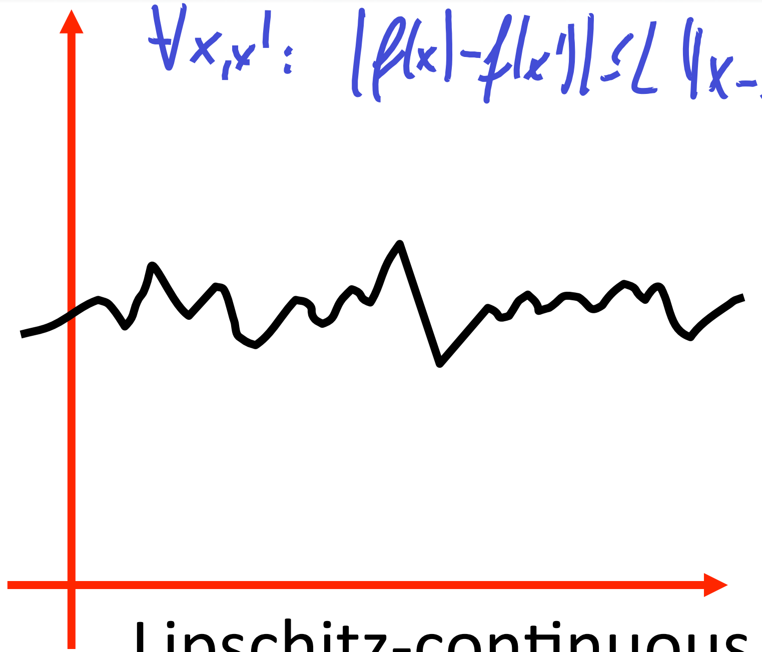
Assumptions on *f*



Linear?
[Dani et al, '07]

Fast convergence;
$$R_T = \mathcal{O}^*(d\sqrt{T})$$
But strong assumption

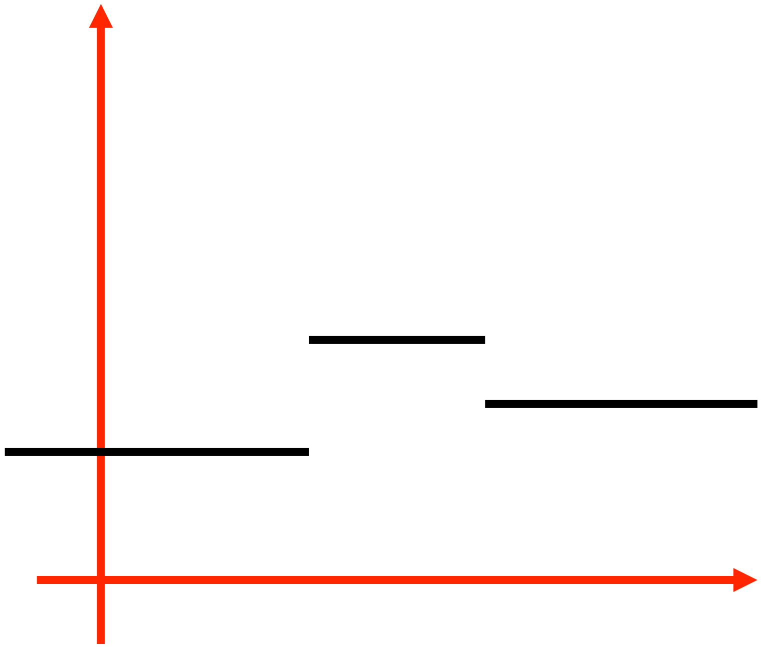
$\forall x, x': |f(x) - f(x')| \leq L \|x - x'\|$



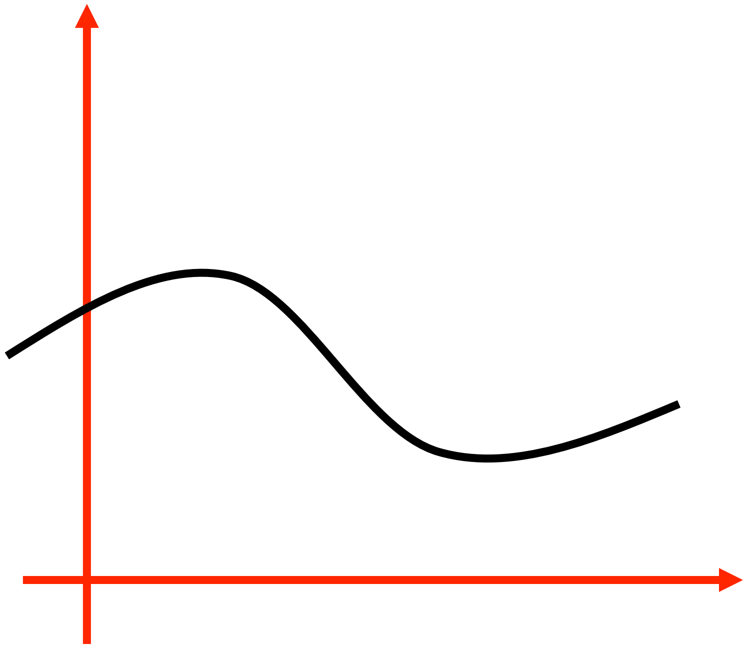
Lipschitz-continuous
(bounded slope)
[Kleinberg '08]

Very flexible, but
$$R_T = \Omega(T^{\frac{d+1}{d+2}})$$

What if we believe, the function looks like:



Piece-wise constant?
(E.g.: data is clustered)



Analytic?
(1-diff.' able)

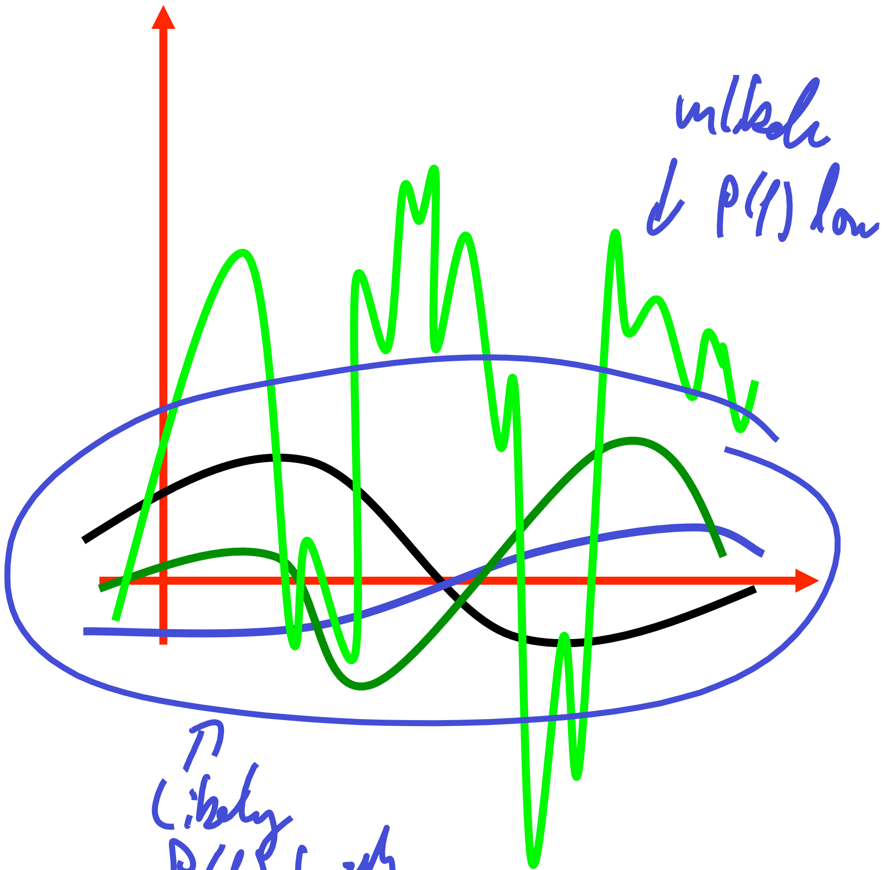
Want flexible way to encode assumptions about functions!

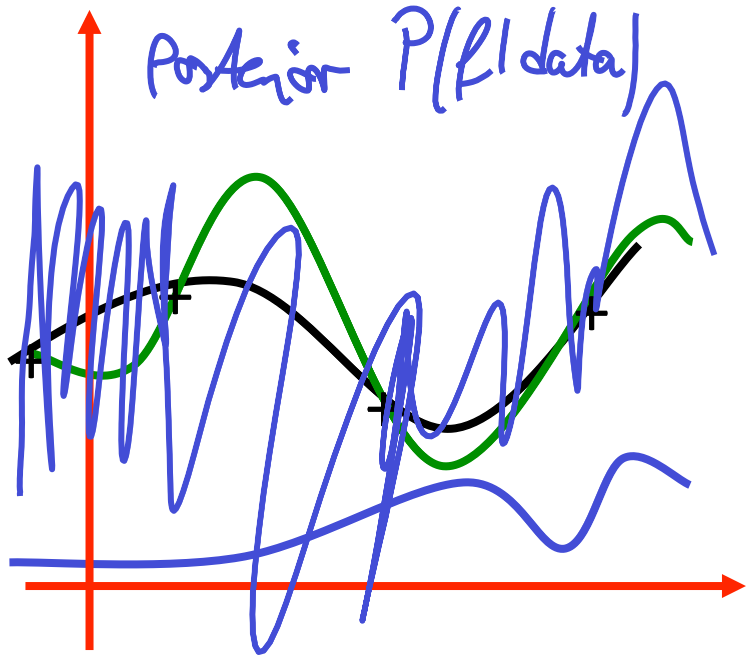
A Bayesian approach

- Bayesian models for **functions**

Prior P(f)

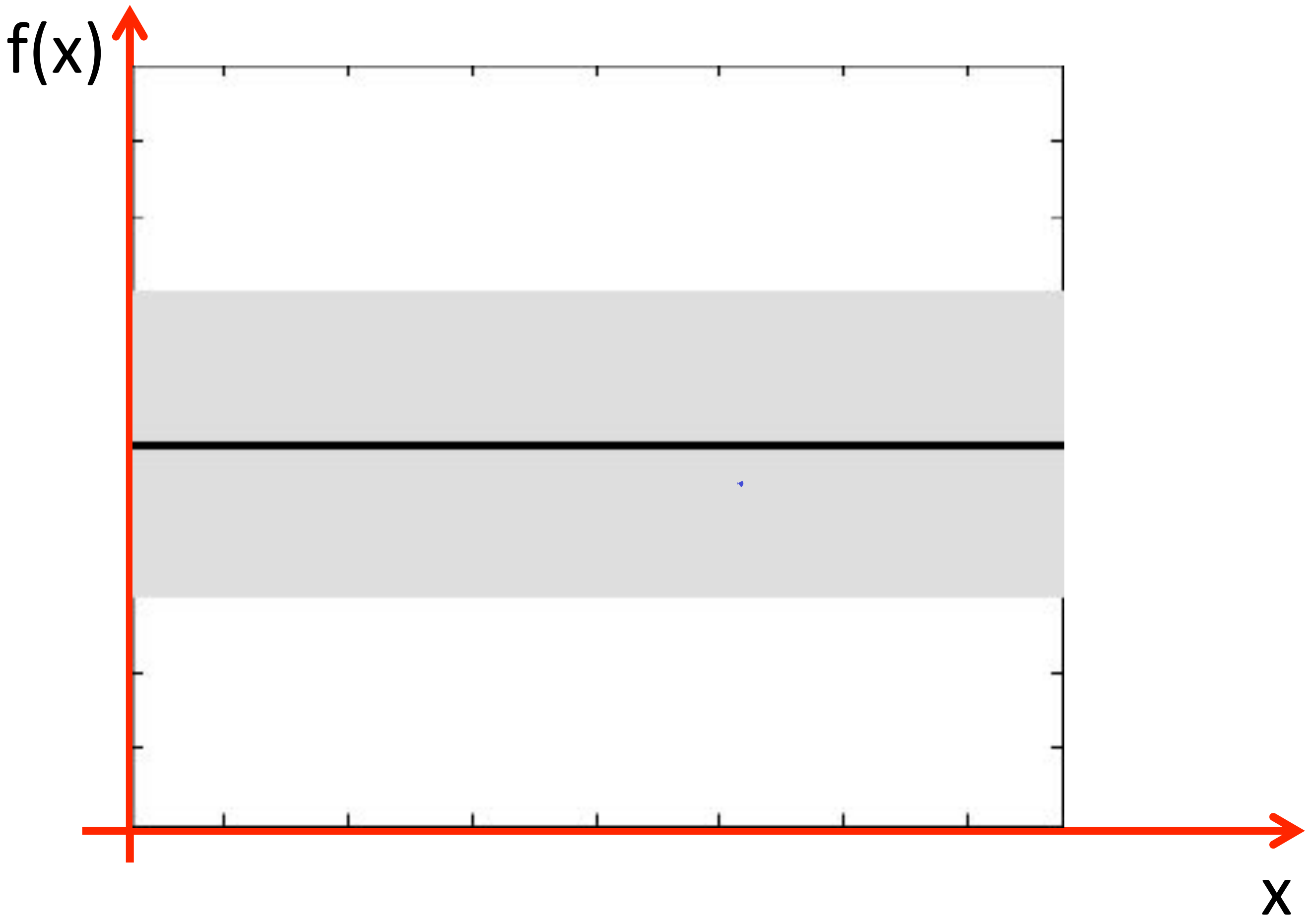
Likelihood P(data | f)



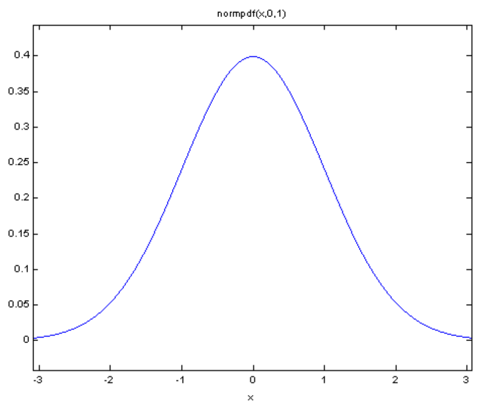


- Why is this useful?

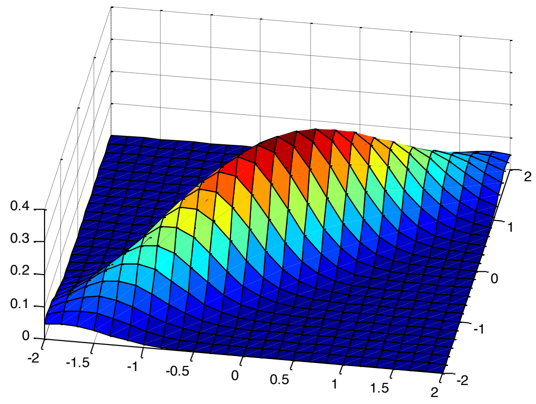
Regression with uncertainty about predictions!



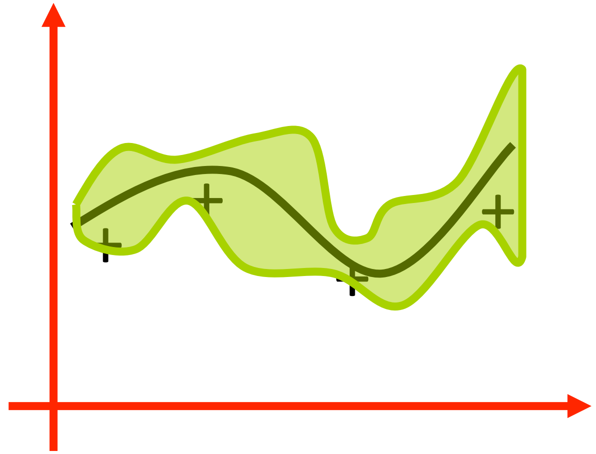
Gaussian Processes to model payoff *f*



Normal dist.
(1-D Gaussian)



Multivariate normal
(n-D Gaussian)



Gaussian process
(∞-D Gaussian)

- Gaussian process (GP) = normal distribution over *functions*
- Finite marginals are multivariate Gaussians $P(f(x)) = N(..)$
- Closed form formulae for Bayesian posterior update exist
- Parameterized by *covariance function K(x,x') = Cov(f(x),f(x'))*

Gaussian process

- A Gaussian Process (GP) is an

  (infinite) set of random variables, indexed by some set X
  i.e., for each x in X there's a random variable Y_x where
  there exists functions $\mu : X \to \mathbb{R} \quad \mathcal{K} : X \times X \to \mathbb{R}$
  such that for all $A \subseteq X \ , \quad A = \{x_1, \ldots, x_k\}$
  it holds that $$Y_A = [Y_{x_1}, \ldots, Y_{x_k}] \sim \mathcal{N}(\mu_A, \Sigma_{AA})$$
  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 is called **kernel** (covariance) function
    μ is called **mean** function

Kernel functions

- K must be symmetric

  $K(x,x') = K(x',x)$ for all x, x'

- K must be positive definite

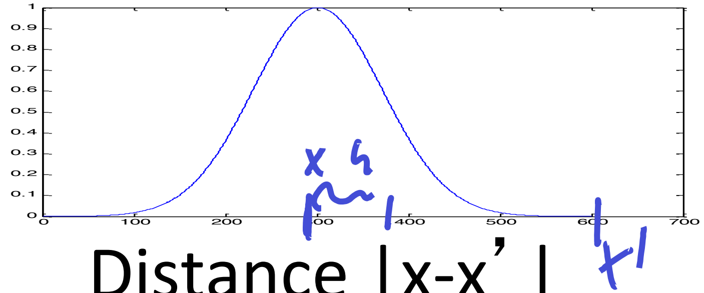
  For all A: Σ_{AA} is positive definite matrix

  $\forall x \in \mathbb{R}^k : x^T \Sigma_{AA} x \geq 0$

- Kernel function K: assumptions about correlation!

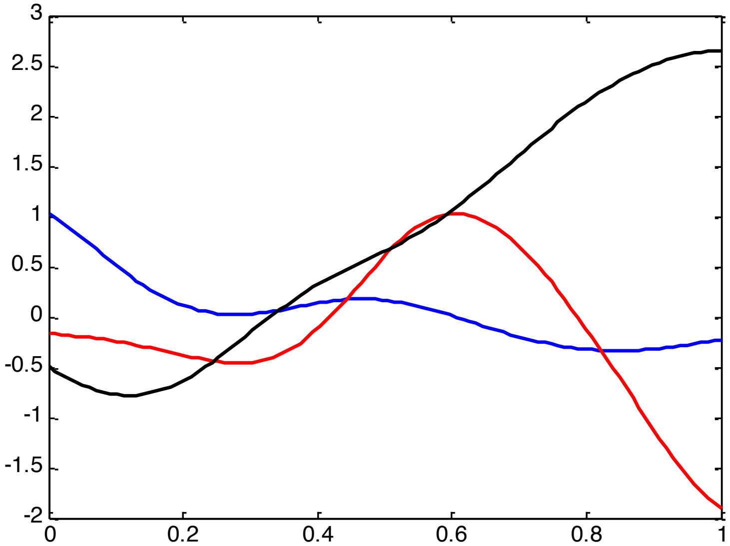
Kernel functions: Examples

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

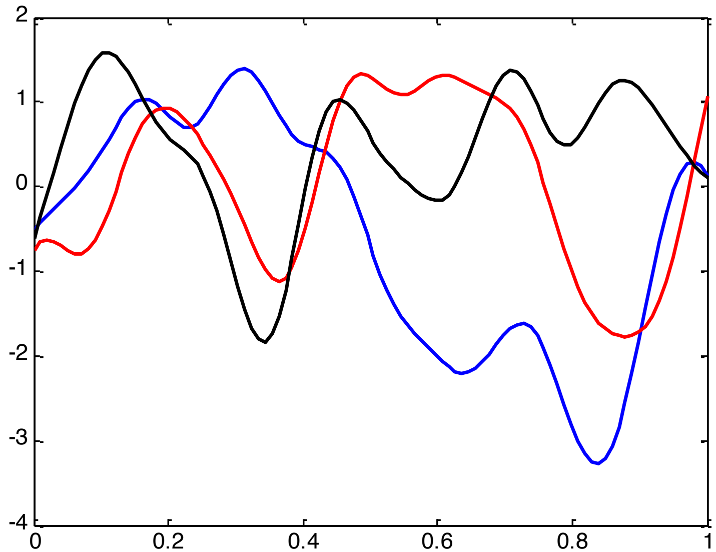


Distance |x-x' |

Samples from P(f)



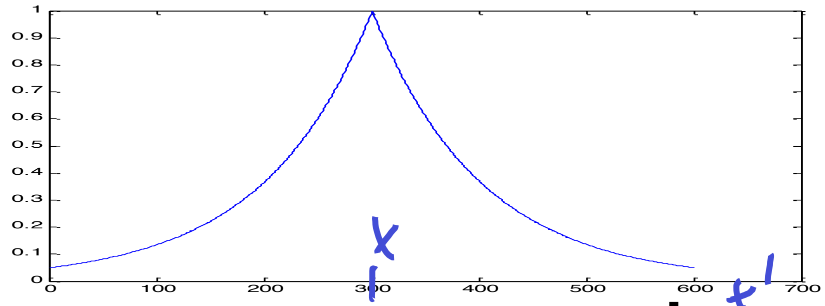
Bandwidth h=.3



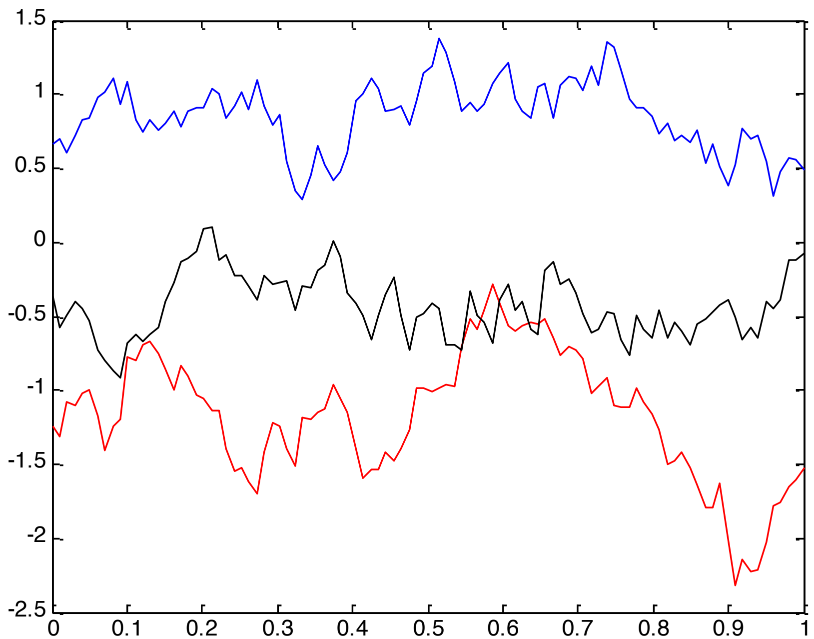
Bandwidth h=.1

Kernel functions: Examples

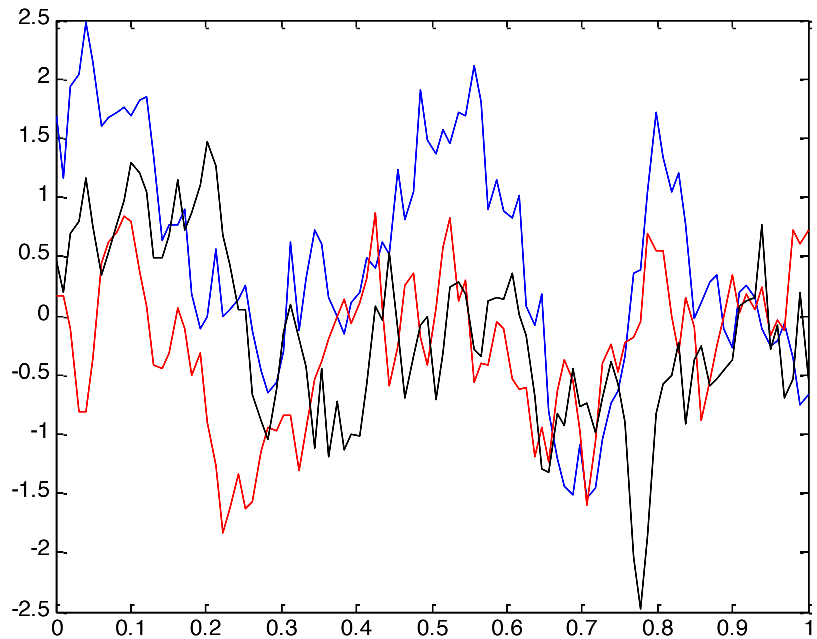
- Exponential kernel
  K(x,x') = exp(-|x-x'|/h)



Distance |x-x'|



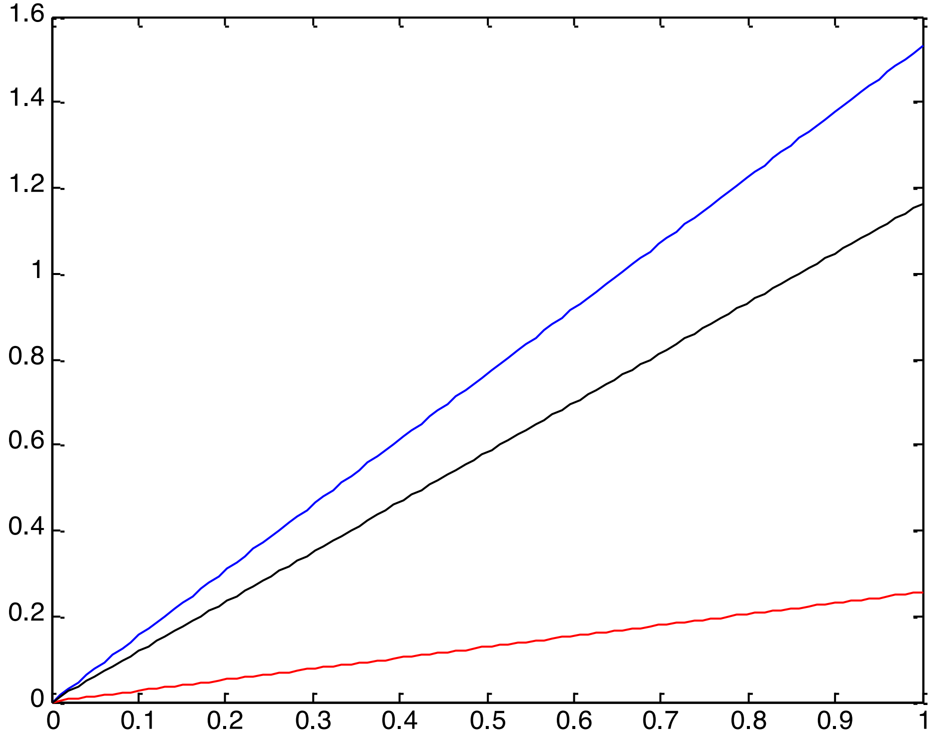
Bandwidth h=1



Bandwidth h=.3

Kernel functions: Examples

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

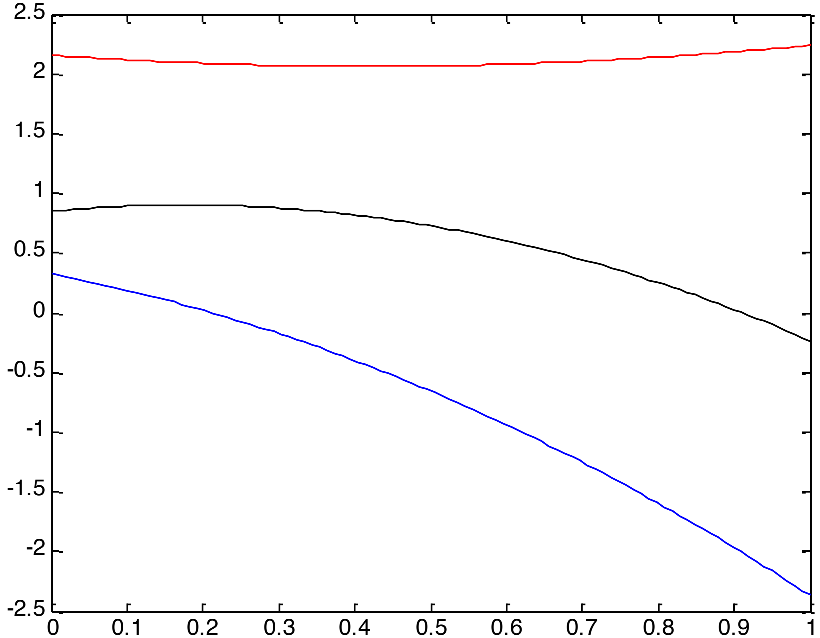


- Corresponds to linear regression!

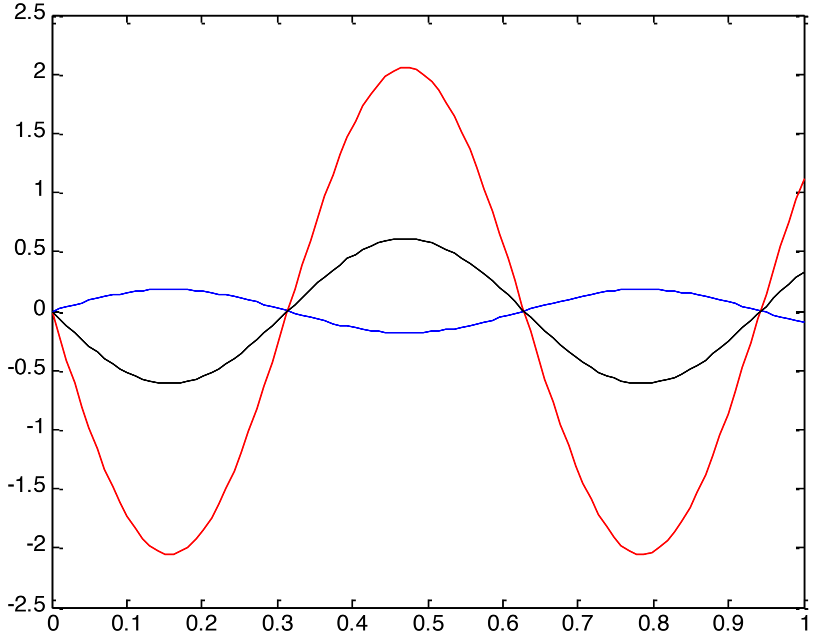
Kernel functions: Examples

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

E.g., $\Phi(x) = [0,x,x^2]$



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



Making predictions with GPs

- Suppose $P(f) = GP(f; \mu, \mathcal{K})$

  and we observe $y_i = f(\mathbf{x}_i) + \epsilon_i \qquad A = \{\mathbf{x}_1, \ldots, \mathbf{x}_k\}$
- Then $P(f \mid \mathbf{x}_1, \ldots, \mathbf{x}_k, y_1, \ldots, y_k) = GP(f; \mu', \mathcal{K}')$
- In particular,

$$P(f(x) \mid \mathbf{x}_1, \ldots, \mathbf{x}_k, y_1, \ldots, y_k) = \mathcal{N}(f(x); \mu_{x|A}, \sigma^2_{x|A})$$

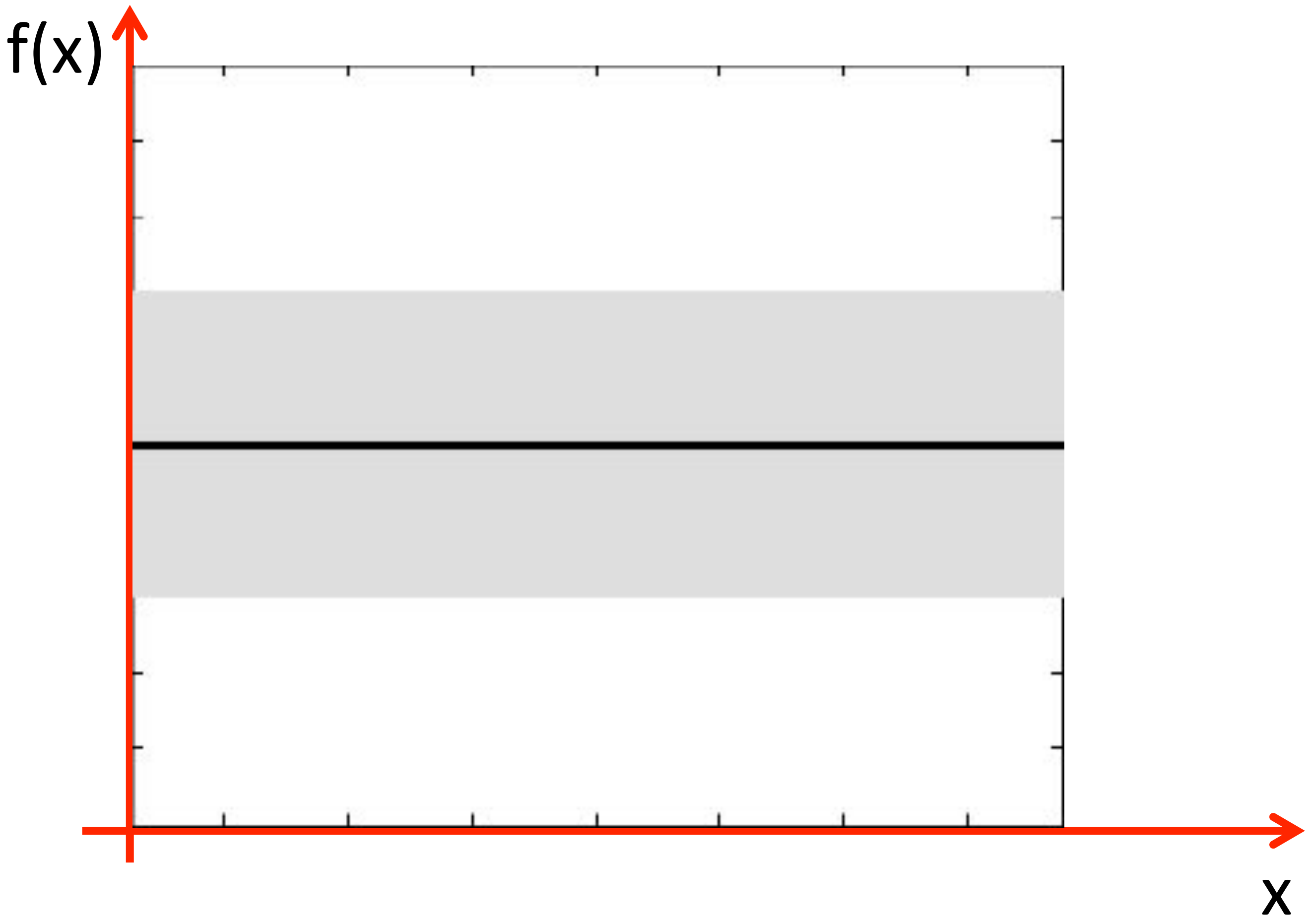
where

$$\mu_{x|A} = \mu(\mathbf{x}) + \mathbf{\Sigma}_{x,A}(\mathbf{\Sigma}_{AA} + \sigma^2 \mathbf{I})^{-1}(\mathbf{y}_A - \mu_A)$$

$$\sigma^2_{x|A} = \mathcal{K}(\mathbf{x}, \mathbf{x}) - \mathbf{\Sigma}_{x,A}(\mathbf{\Sigma}_{AA} + \sigma^2 \mathbf{I})^{-1}\mathbf{\Sigma}^T_{x,A}$$

➔ Closed form formulas for prediction!

Illustrations: Predictions in GPs

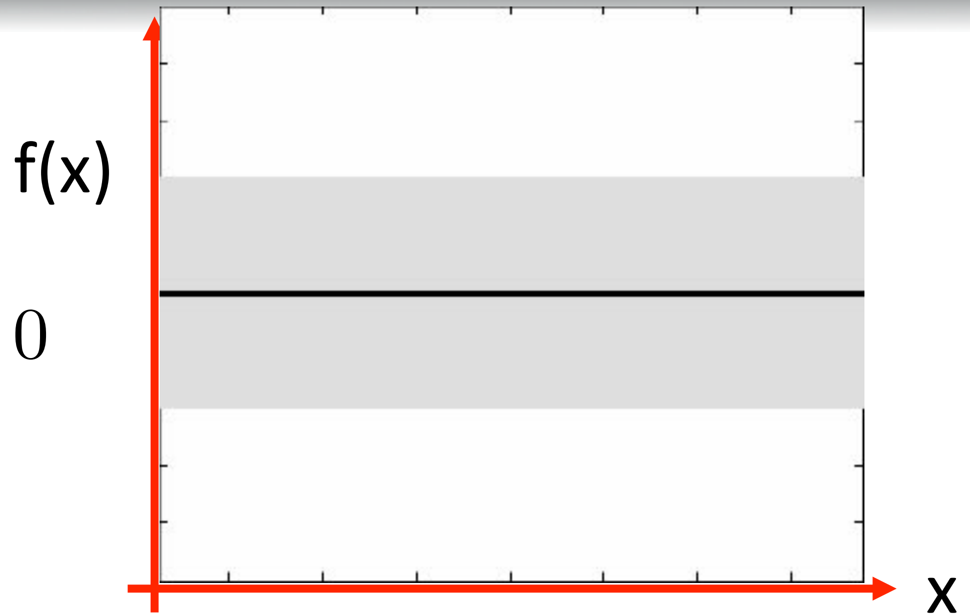


Gaussian process (bandit) optimization

Goal: Adaptively pick inputs $x_1, x_2, \ldots$ such that

$$\frac{1}{T}\sum_{t=1}^{T}[f(x^*) - f(x_t)] \to 0$$

Average regret



Key question: how should we pick samples?

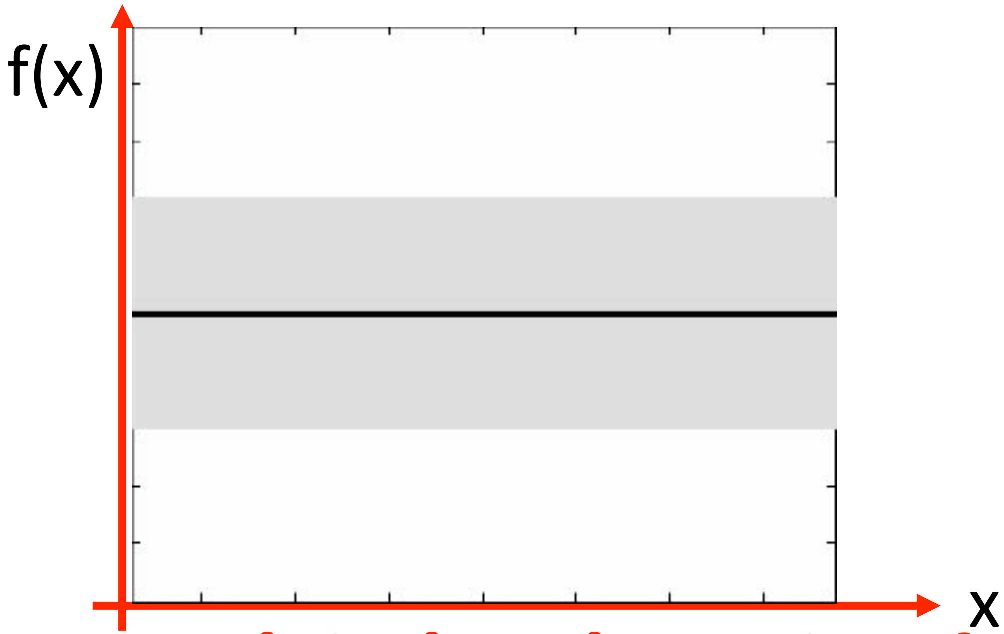
Several commonly used heuristics:

- Expected Improvement [Močkus *et al.* '78]
- Most Probable Improvement [Močkus '89]
- Used successfully in machine learning [Ginsbourger *et al.* '08, Jones '01, Lizotte *et al.* '07]

- Let's get some intuition

Simple algorithm for GP optimization

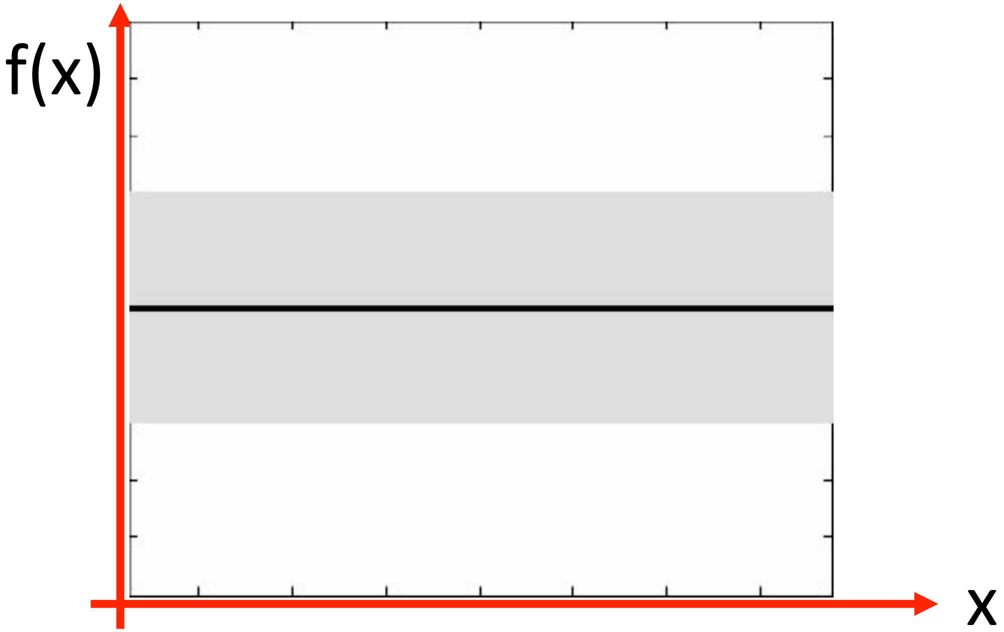
- In each round t do:
  - Pick $x_t = \arg\max_{x \in D} \mu_{t-1}(x)$
  - Observe $y_t = f(x_t) + \epsilon_t$
  - Use Bayes' rule to get posterior mean $\mu_t(\cdot)$



Can get stuck in local maxima!

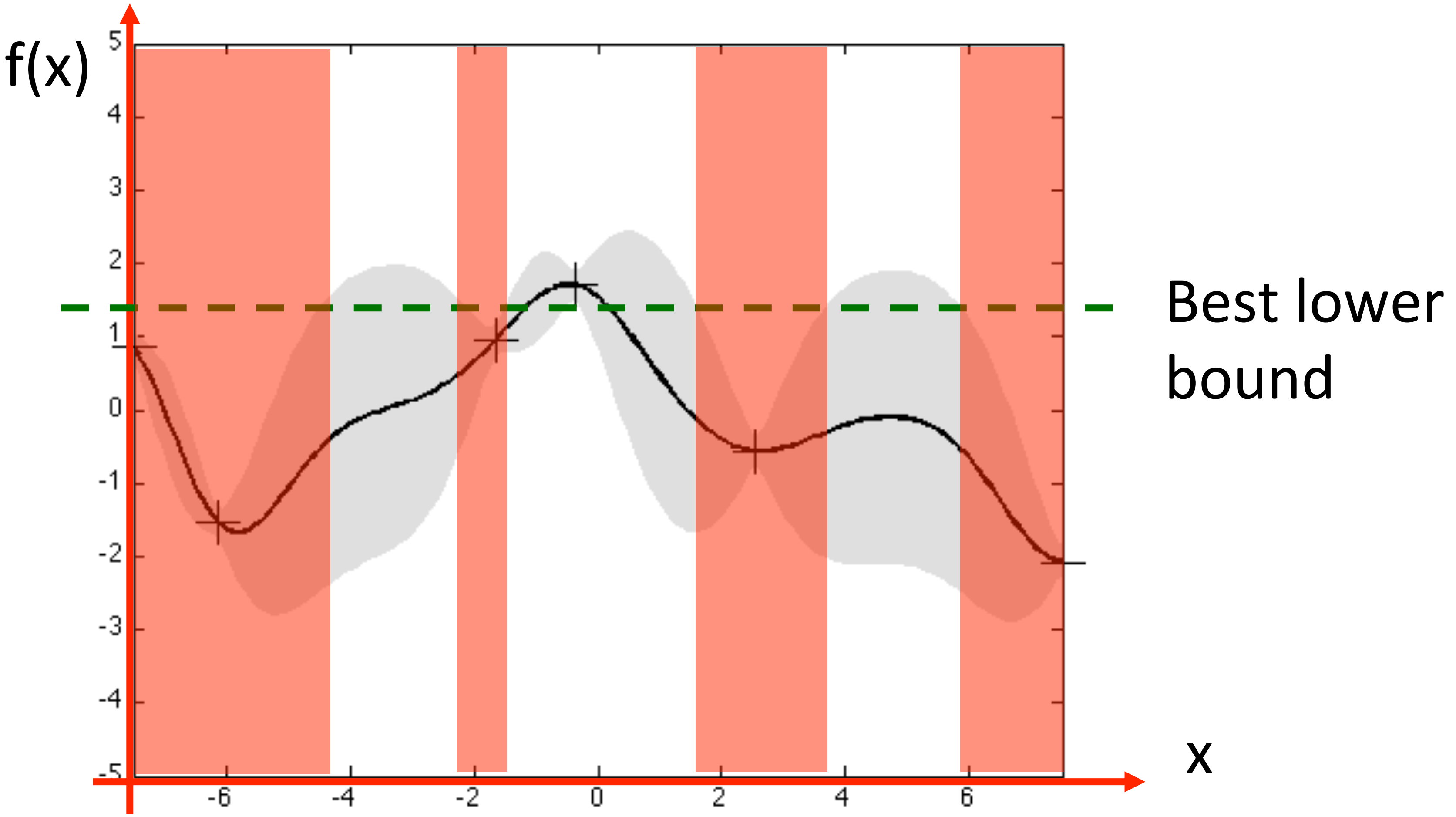
Uncertainty sampling

Pick: $$x_t = \arg\max_{x \in D} \sigma^2_{t-1}(x)$$



Wastes samples by exploring f everywhere!

Avoiding unnecessary samples

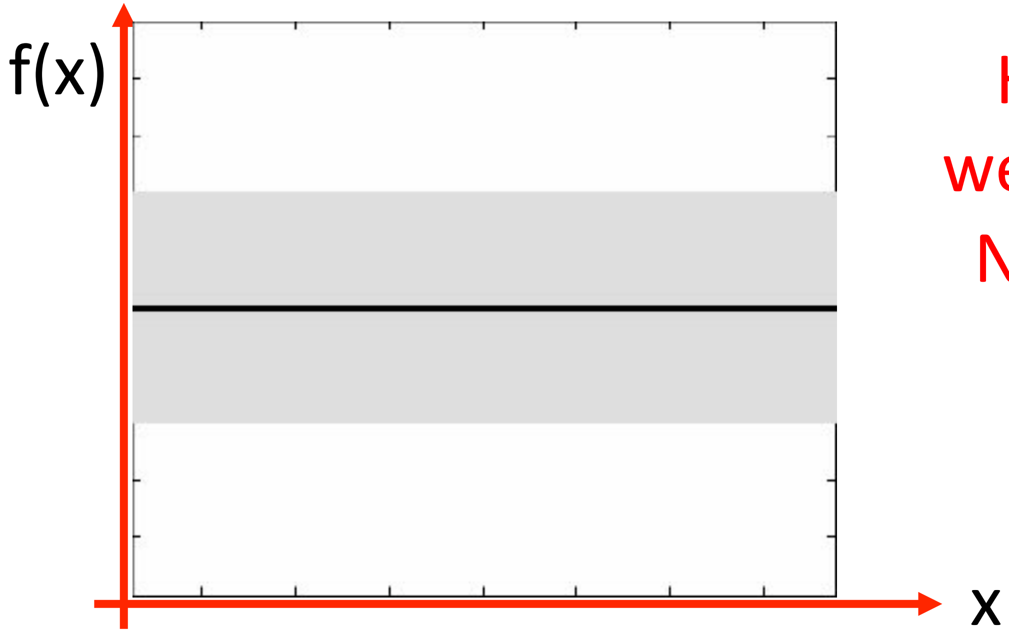


Key insight: Never need to sample where upper confidence limit < best lower bound!

Upper confidence sampling

Pick input that maximizes upper confidence bound:

$$x_t = \arg\max_{x \in D} \mu_{t-1}(x) + \beta_t \sigma_{t-1}(x)$$



How should we choose β_t?? Need theory!

Naturally trades off exploration and exploitation

Does not waste samples (with high prob.)