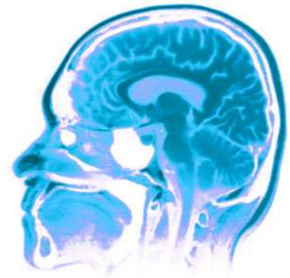
CPSC540



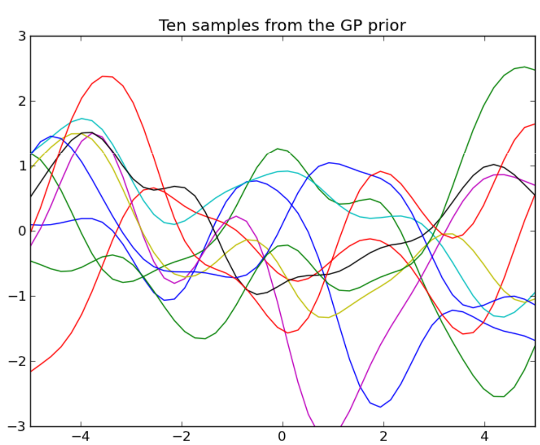
Gaussian Processes, Active Learning,
Bandits and Bayesian Optimization

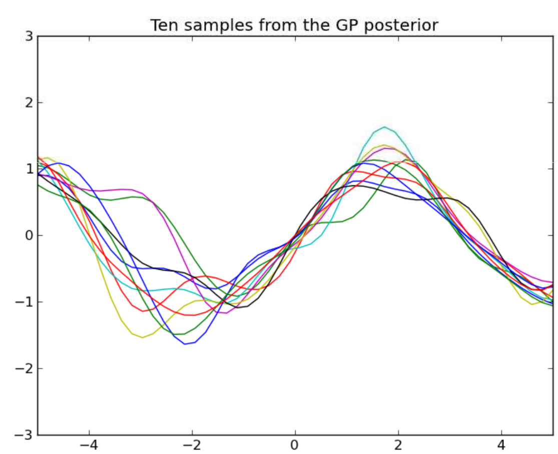
Nando de Freitas
November, 2011
University of British Columbia

Functional regression with GPs

$$f \sim \mathcal{GP}(m,\ k)$$

$$k(x, x') = \exp(-\tfrac{1}{2}(x - x')^2) \qquad p(f|\mathcal{D}) = \frac{p(\mathcal{D}|f)p(f)}{p(\mathcal{D})}$$





$$\mathbf{f} \sim \mathcal{N}(\mu, \Sigma)$$

$$\Sigma_{ij} = k(x_i, x_j) = \exp(-\tfrac{1}{2}(x_i - x_j)^2), \quad i, j = 1, \ldots, n$$

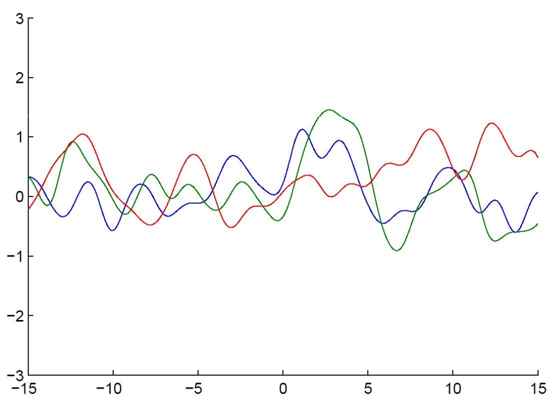
Sampling from prior *P(f)*

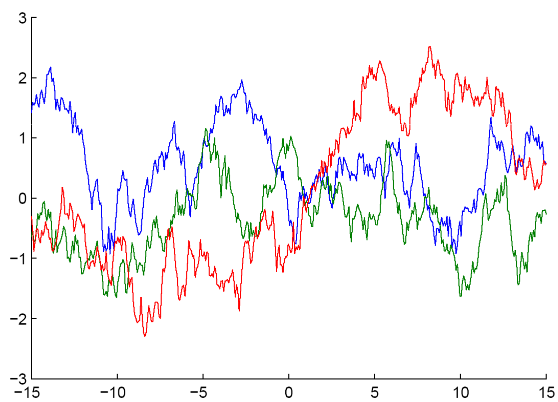
```
from __future__ import division
import numpy as np
import matplotlib.pyplot as pl

f = lambda x: np.sin(0.9*x).flatten() # The true function we're trying to approximate.

def kernel(a, b):
    """Squared exponential kernel."""
    sqdist = np.sum(a**2,1).reshape(-1,1) + np.sum(b**2,1) - 2*np.dot(a, b.T)
    return np.exp(-.5 * sqdist)

N = 15      # Number of training points.
n = 50      #  Number of test points.
s = 0.05    # Noise variance, assumed to be known.

X = np.random.uniform(-5, 5, size=(N,1))   #  Random points at which we sample the function.
K = kernel(X, X)                           # Form the kernel matrix.

# draw samples from the prior
L = np.linalg.cholesky(K + 1e-6*np.eye(N))         # L = sqrt (K)
f_prior = np.dot(L, np.random.normal(size=(N,10)))
```

Other choices of kernels

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

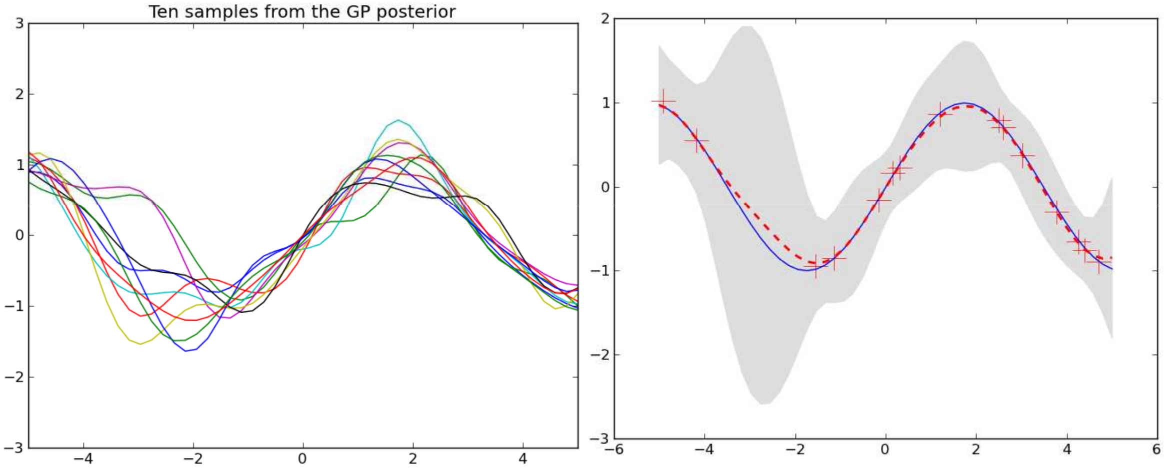
$$K_{\mathrm{SE}}(r) = a^2 \exp\left(-\frac{r^2}{2\lambda^2}\right) \qquad K_{\mathrm{Mat}}(r) = a^2 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu} r}{\lambda}\right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu} r}{\lambda}\right)$$



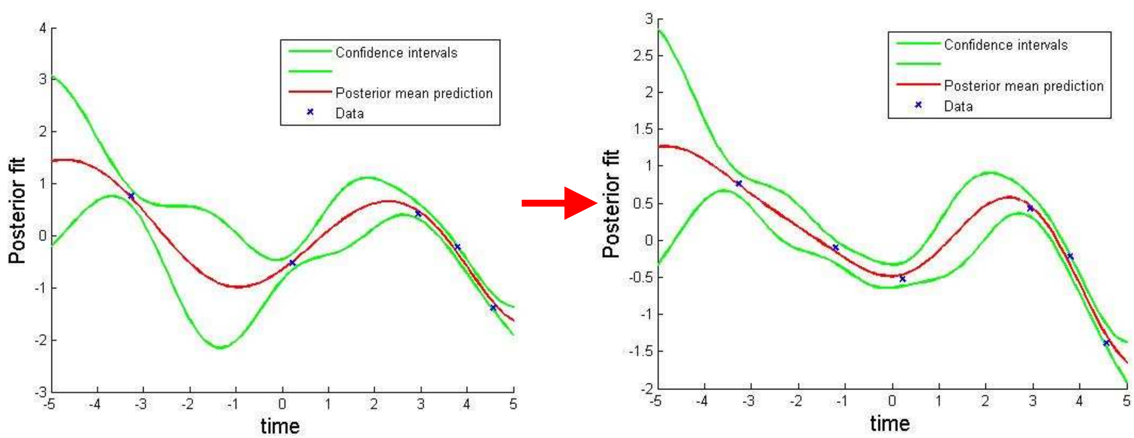


[Snelson, 2007]

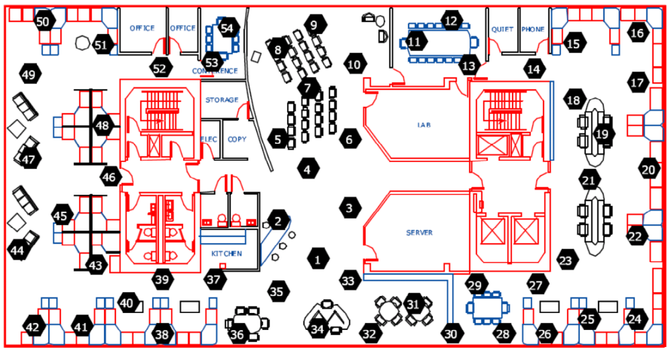
Posterior mean and variance

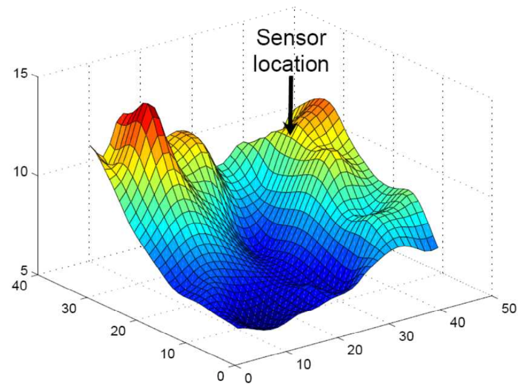


Active learning with GPs



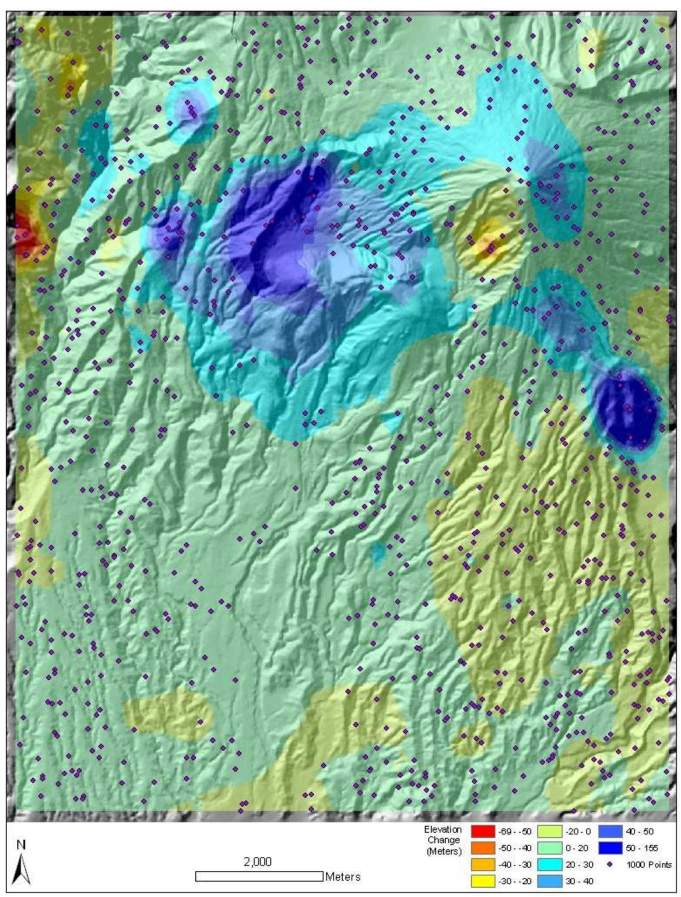
Sensor placement



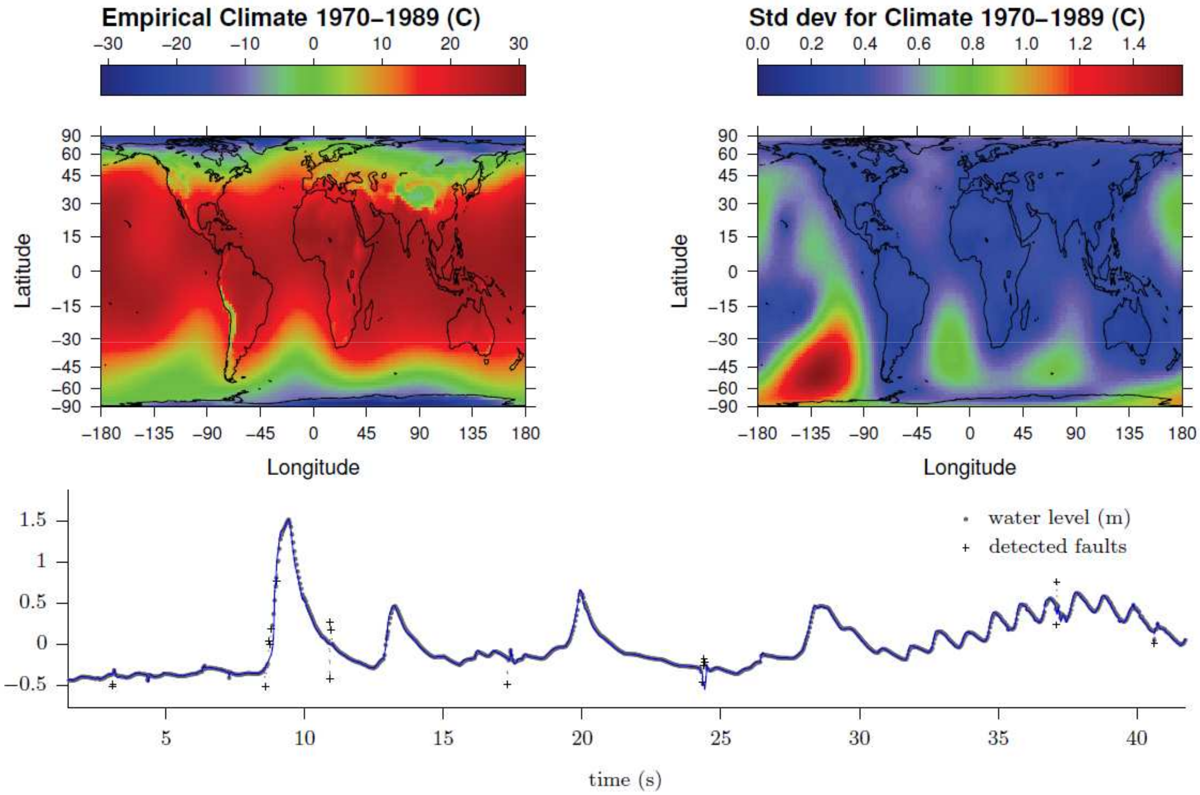


[Andreas Krause & Carlos Guestrin]

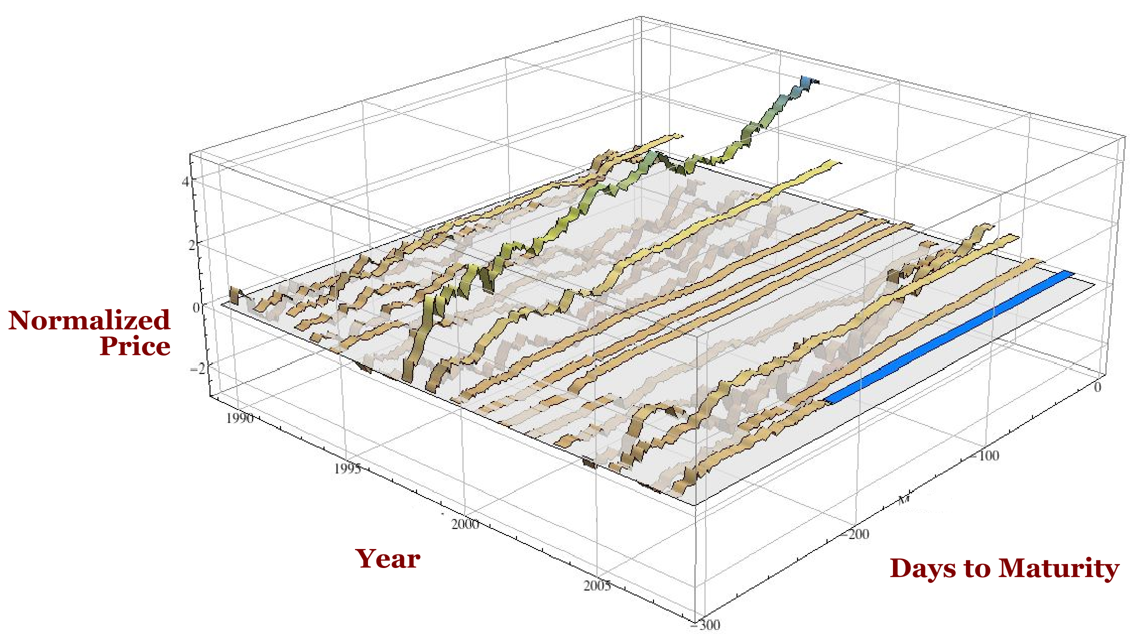
Kriging



GPs for environmental data



Forecasting commodities



[Nicolas Chapados & Yoshua Bengio]

GP regression

Zero-mean GP prior

$$p(\mathbf{f}) = \mathcal{N}(\mathbf{0}, \mathbf{K})$$

Gaussian noise / likelihood

$$y = f + \epsilon \ , \qquad \mathcal{E}[\epsilon(\mathbf{x})\epsilon(\mathbf{x}')] = \sigma^2 \delta_{\mathbf{x}\mathbf{x}'} \quad p(\mathbf{y}|\mathbf{f}) = \mathcal{N}(\mathbf{f}, \sigma^2 \mathbf{I})$$

The marginal likelihood (evidence) is Gaussian:

$$\begin{aligned} p(\mathbf{y}) &= \int \mathrm{d}\mathbf{f}\, p(\mathbf{y}|\mathbf{f}) p(\mathbf{f}) \\ &= \mathcal{N}(\mathbf{0}, \mathbf{K} + \sigma^2 \mathbf{I}) \end{aligned}$$

GP regression

$$(\mathbf{X}, \mathbf{f}, \mathbf{y}) = (\{\mathbf{x}_n\}, \{f_n\}, \{y_n\})_{n=1}^{N} \quad \text{Train set}$$

$$(\mathbf{X}_T, \mathbf{f}_T, \mathbf{y}_T) = (\{\mathbf{x}_t\}, \{f_t\}, \{y_t\})_{t=1}^{T} \quad \text{Test set}$$

Both sets are, by definition, jointly Gaussian:

$$p(\mathbf{f}, \mathbf{f}_T) = \mathcal{N}(\mathbf{0}, \mathbf{K}_{N+T})$$

$$\mathbf{K}_{N+T} = \begin{bmatrix} \mathbf{K}_N & \mathbf{K}_{NT} \\ \mathbf{K}_{TN} & \mathbf{K}_T \end{bmatrix}$$

The joint distribution of the measurements is:

$$p(\mathbf{y}, \mathbf{y}_T) = \mathcal{N}(\mathbf{0}, \mathbf{K}_{N+T} + \sigma^2 \mathbf{I})$$

GP regression

Using the Schuur complement, the predictive conditional distribution is Gaussian too:

$$p(\mathbf{y}_T|\mathbf{y}) = \mathcal{N}(\boldsymbol{\mu}_T, \boldsymbol{\Sigma}_T) \ ,$$

$$\boldsymbol{\mu}_T = \mathbf{K}_{TN}[\mathbf{K}_N + \sigma^2\mathbf{I}]^{-1}\mathbf{y}$$

$$\boldsymbol{\Sigma}_T = \mathbf{K}_T - \mathbf{K}_{TN}[\mathbf{K}_N + \sigma^2\mathbf{I}]^{-1}\mathbf{K}_{NT} + \sigma^2\mathbf{I}$$

Posterior predictions

```
y = f(X) + s*np.random.randn(N)  # Obtain noisy evaluations of f at training points X.
Xtest = np.linspace(-5, 5, n).reshape(-1,1) # Points we're going to make predictions at.

# compute the mean at our test points.
L = np.linalg.cholesky(K + s*np.eye(N))
Lk = np.linalg.solve(L, kernel(X, Xtest))
mu = np.dot(Lk.T, np.linalg.solve(L, y))

# compute the variance at our test points.
K_ = kernel(Xtest, Xtest)
s2 = np.diag(K_) - np.sum(Lk**2, axis=0)
s = np.sqrt(s2)

pl.figure(1)
pl.clf()
pl.plot(X, y, 'r+', ms=20)
pl.plot(Xtest, f(Xtest), 'b-')
pl.gca().fill_between(Xtest.flat, mu-3*s, mu+3*s, color="#dddddd")
pl.plot(Xtest, mu, 'r--', lw=2)
pl.savefig('predictive.png', bbox_inches='tight')
pl.title('Mean predictions plus 3 st.deviations')
pl.axis([-5, 5, -3, 3])
```

Parameter learning for GPs: maximum likelihood

$$
\begin{aligned}
L &= \log p(\mathbf{y}|\mathbf{x}, \theta) \\
&= -\tfrac{1}{2}\log|\Sigma| - \tfrac{1}{2}(\mathbf{y}-\boldsymbol{\mu})^\top \Sigma^{-1}(\mathbf{y}-\boldsymbol{\mu}) - \tfrac{n}{2}\log(2\pi)
\end{aligned}
$$

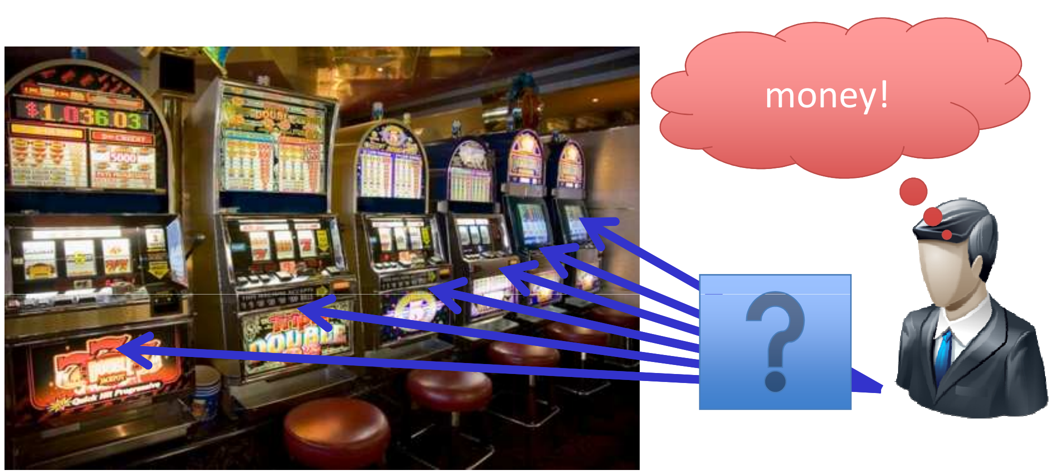
For example, we can parameterize the mean and covariance:

$$f \sim \mathcal{GP}(m,\ k),$$

$$
\begin{aligned}
m(x) &= ax^2 + bx + c, \\
k(x, x') &= \sigma_y^2 \exp\left(-\frac{(x-x')^2}{2\ell^2}\right) + \sigma_n^2 \delta_{ii'}
\end{aligned}
$$

$$\theta = \{a, b, c, \sigma_y, \sigma_n, \ell\}$$

Multi-armed bandit problem



- What ad to deliver on a webpage?
- What items is the user more likely to click on?
- What dose will make us learn the effects of a drug?

Multi-armed bandit problem

Actions
$\{1, 2, \ldots, K\}$

Reward r(t)

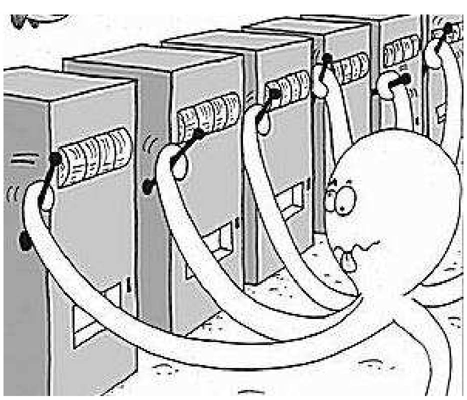
$t = 1, 2, \ldots, T$
Sequence of trials

- Trade-off between Exploration and Exploitation
- Regret = Player reward – Reward of best action

The full information game

Initialization: Choose a real number $\eta > 0$. Set $G_i(0) = 0$ for $i = 1, \ldots, K$.
Repeat for $t = 1, 2, \ldots$:

1. Choose action i_t according to the distribution: $p_i(t) = \dfrac{\exp(\eta G_i(t-1))}{\sum_{j=1}^{K} \exp(\eta G_j(t-1))}$
2. Receive the reward vector $r(t)$ and score the gain $r_{i_t}(t)$.
3. Set $G_i(t) = G_i(t-1) + r_i(t)$ for $i = 1, \ldots, K$.



Cumulative Regret Bound:
$R_{\text{Hedge}} \leq \sqrt{2T \ln K}$

[Freund & Shapire '95]

Problem: Must observe reward for each action!

Partial information game (EXP3)

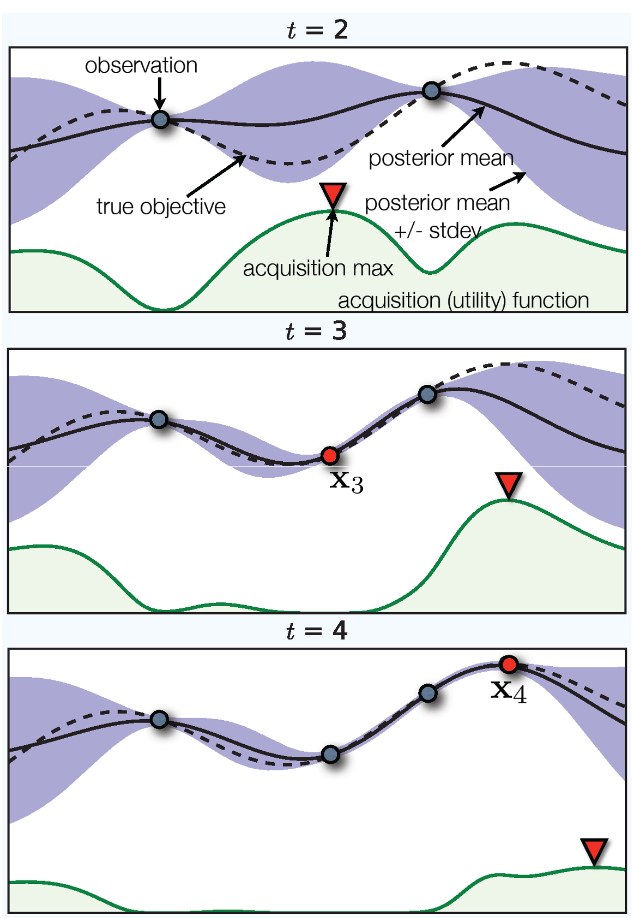
[Auer et al. '95]

Initialization: Choose a real number $\eta > 0$. Set $G_i(0) = 0$ for $i = 1, \ldots, K$.
Repeat for $t = 1, 2, \ldots$:

1. Choose action i_t according to the distribution: $p_i(t) = \dfrac{\exp(\eta G_i(t-1))}{\sum_{j=1}^{K} \exp(\eta G_j(t-1))}$.
2. Receive the reward vector $r(t)$ and score the gain $r_{i_t}(t)$.
3. Set $G_i(t) = G_i(t-1) + r_i(t)$ for $i = 1, \ldots, K$.

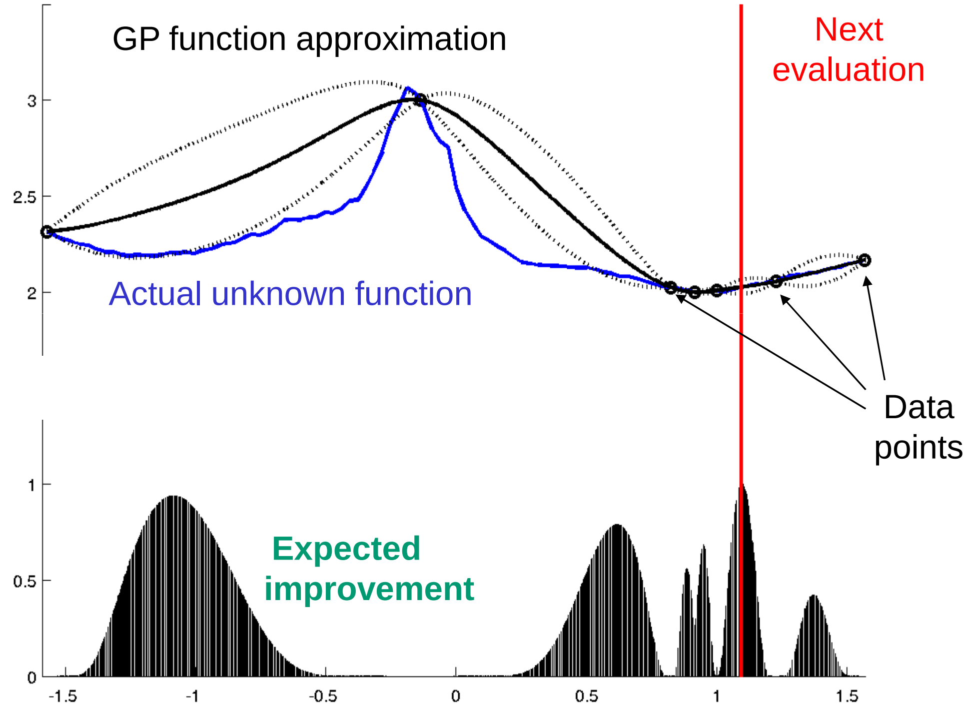
Initialization: Choose $\gamma \in (0, 1]$. Initialize Hedge(η).
Repeat for $t = 1, 2, \ldots$:

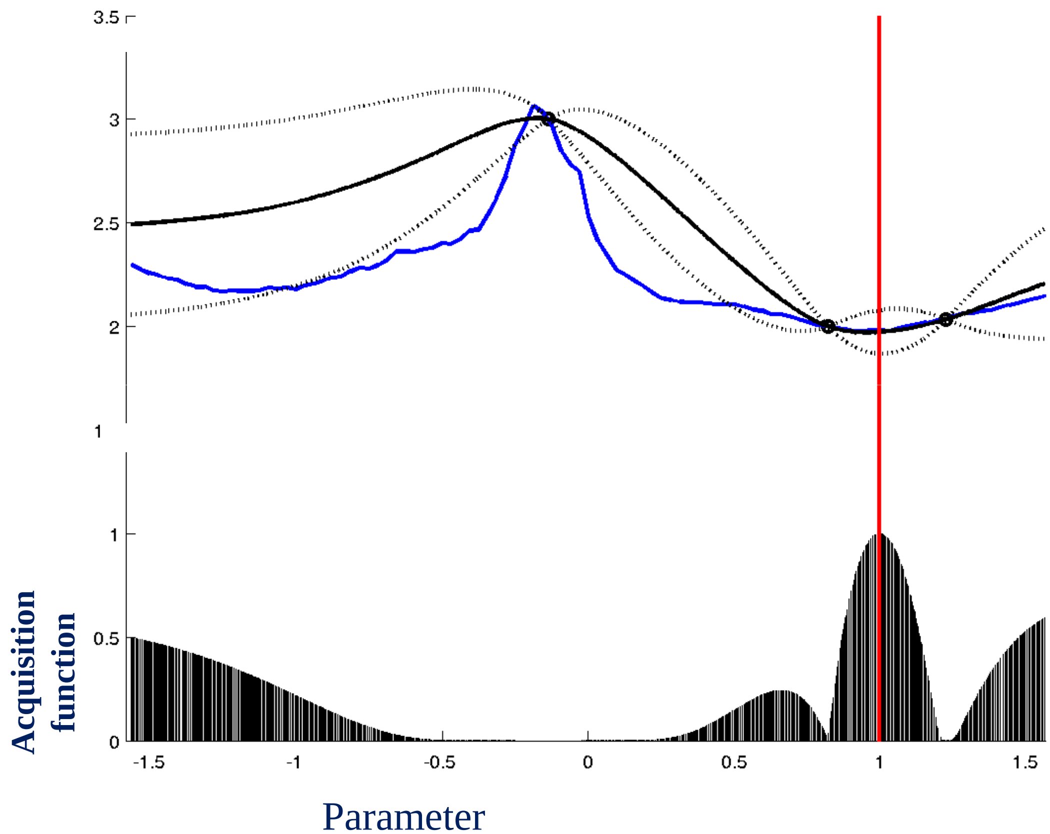
1. Get the distribution $p(t)$ from Hedge.
2. Select action i_t to be j with probability $\hat{p}_j(t) = (1-\gamma)p_j(t) + \dfrac{\gamma}{K}$.
3. Receive reward $r_{i_t}(t) \in [0, 1]$.
4. Feed the simulated reward $\hat{r}(t)$ back to Hedge, where $\hat{r}_j(t) = \begin{cases} \dfrac{r_{i_t}(t)}{\hat{p}_{i_t}(t)} & \text{if } j = i_t \\ 0 & \text{otherwise} \end{cases}$



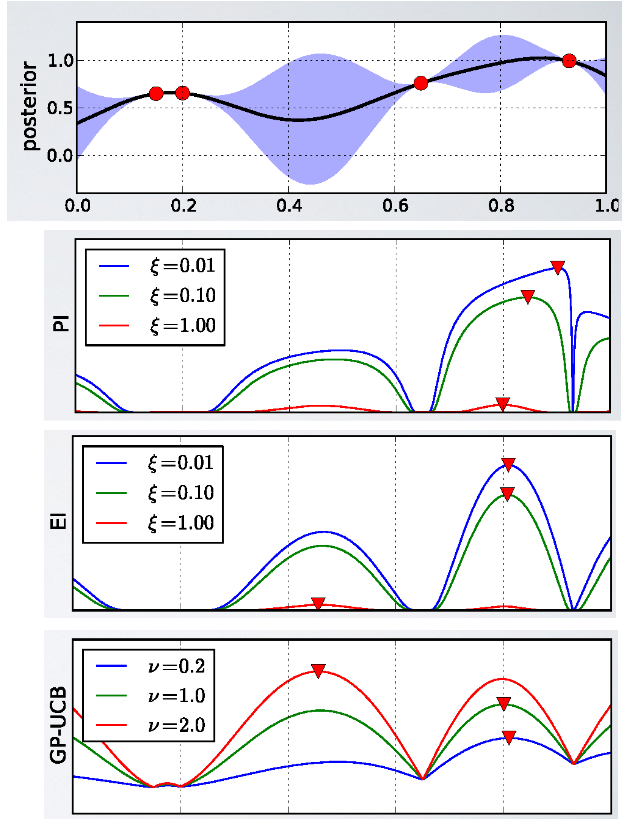
Bayesian optimization

1: **for** $t = 1, 2, \ldots$ **do**
2: Find $\mathbf{x}_t$ by combining attributes of the posterior distribution in a utility function u and maximizing: $\mathbf{x}_t = \text{argmax}_{\mathbf{x}}\, u(\mathbf{x}|\mathcal{D}_{1:t-1})$.
3: Sample the objective function: $y_t = f(\mathbf{x}_t) + \varepsilon_t$.
4: Augment the data $\mathcal{D}_{1:t} = \{\mathcal{D}_{1:t-1}, (\mathbf{x}_t, y_t)\}$ and update the GP.
5: **end for**


GP function approximation
Next
evaluation
Actual unknown function
Data
points
Expected
improvement
3
2.5
2
1
0.5
0
-1.5
-1
-0.5
0
0.5
1
1.5


3.5
3
2.5
2
1
1
0.5
0
-1.5
-1
-0.5
0
0.5
1
1.5
Acquisition
function
Parameter

Acquisition functions



- aka <u>infill</u>, <u>figure of merit</u>
- <u>acquisition function</u> guides the optimization by determining which $\mathbf{x}_{t+1}$ to observe next
- uses predictive posterior to combine <u>exploration</u> (high-variance regions) and <u>exploitation</u> (high-mean regions)
- optimize to find sample point (can be done cheaply/approximately)

Acquisition functions

$$\boxed{\mu^+ = \operatorname{argmax}_{\mathbf{x}_i \in \mathbf{x}_{1:t}} \mu(\mathbf{x}_i)}$$

- Probability of Improvement

$$\mathrm{PI}(\mathbf{x}) = \Phi\left(\frac{\mu(\mathbf{x}) - \mu^+ - \xi}{\sigma(\mathbf{x})}\right)$$

Kushner 1964

- Expected Improvement

$$\mathrm{EI}(\mathbf{x}) = (\mu(\mathbf{x}) - \mu^+ - \xi)\Phi(Z) + \sigma(\mathbf{x})\phi(Z)$$

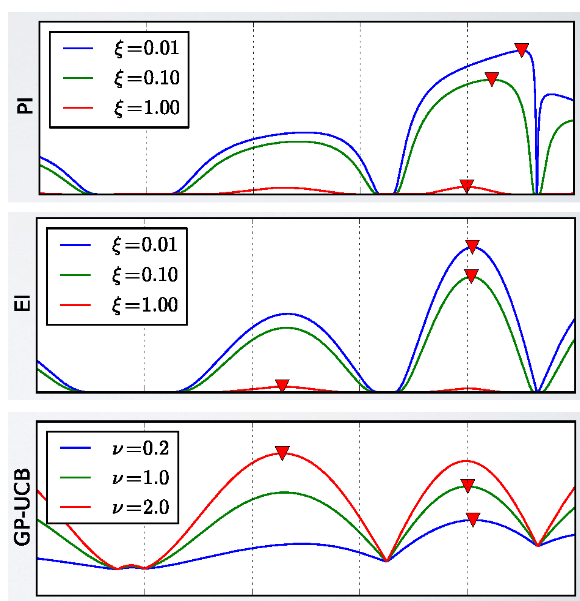
$$Z = \frac{\mu(\mathbf{x}) - \mu^+ - \xi}{\sigma(\mathbf{x})}$$

Mockus 1978

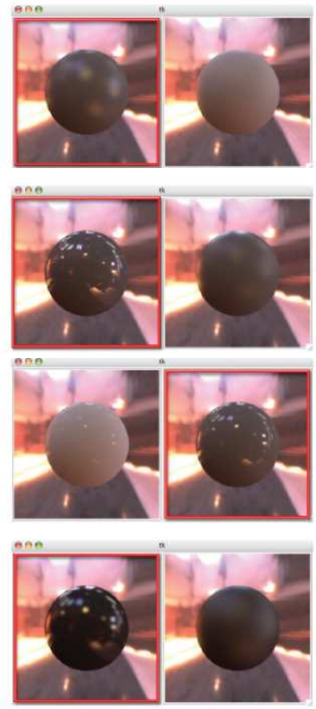
- Upper Confidence Bound

$$\text{GP-UCB}(\mathbf{x}) = \mu(\mathbf{x}) + \sqrt{\nu\tau_t}\sigma(\mathbf{x})$$

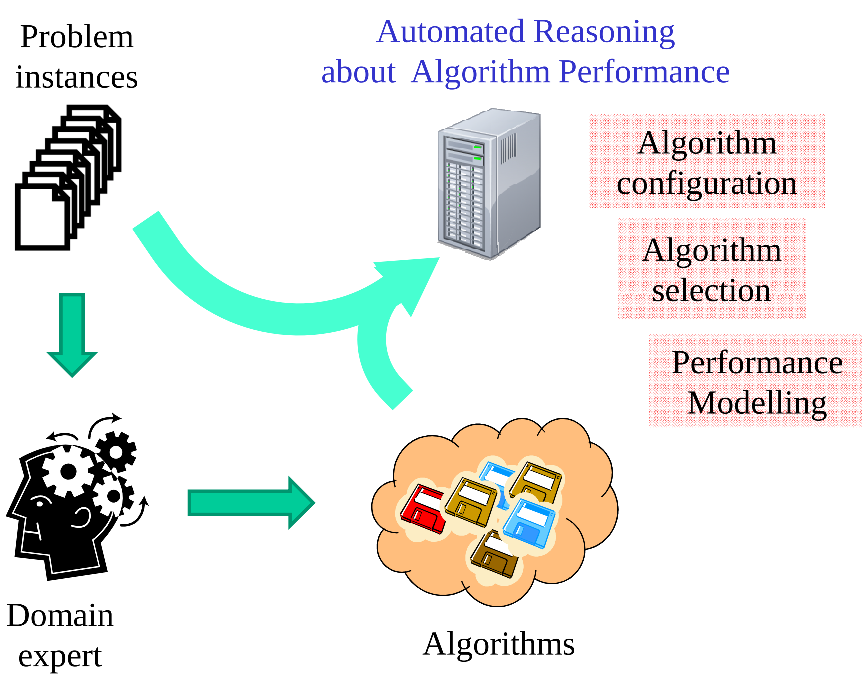
Srinivas *et al.* 2010



Intelligent user interfaces



Automatic algorithm configuration



[Hutter, Hoos & Stützle; AAAI '07]