Markov Chain Monte Carlo Algorithms for Gaussian Processes

Michalis K. Titsias, Neil Lawrence and Magnus Rattray
School of Computer Science
University of Manchester

20 June 2008

Outline

Gaussian Processes

- A Gaussian process (GP) is a distribution over a real-valued function $f(\mathbf{x})$. It is defined by
  - a mean function

$$\mu(\mathbf{x}) = E(f(\mathbf{x}))$$

  - and a covariance or kernel function

$$k(\mathbf{x}_n, \mathbf{x}_m) = E(f(\mathbf{x}_n) f(\mathbf{x}_m))$$

  E.g. this can be the RBF (or squared exponential) kernel

$$k(\mathbf{x}_n, \mathbf{x}_m) = \alpha \exp\left(-\frac{||\mathbf{x}_n - \mathbf{x}_m||^2}{2\ell^2}\right)$$

Gaussian Processes

- We evaluate a function in a set of inputs $(\mathbf{x}_i)_{i=1}^N$:

$$f_i = f(\mathbf{x}_i)$$

- A Gaussian process reduces to a multivariate Gaussian distribution over $\mathbf{f} = (f_i)_{i=1}^N$

$$p(\mathbf{f}) = N(\mathbf{x}|\mathbf{0}, K) = \frac{1}{(2\pi)^{\frac{N}{2}}|K|^{\frac{1}{2}}} \exp\left(-\frac{\mathbf{f}^T K^{-1}\mathbf{f}}{2}\right)$$

  where the covariance K is defined by the kernel function
- $p(\mathbf{f})$ is a conditional distribution (a precise notation is $p(\mathbf{f}|X)$)

Gaussian Processes for Bayesian learning

Many problems involve inference over unobserved/latent functions

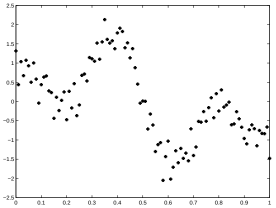
- A Gaussian process can place a prior on a latent function
- Bayesian inference:
  - Data $\mathbf{y} = (y_i)_{i=1}^N$ (associated with inputs $(\mathbf{x}_i)_{i=1}^N$)
  - Likelihood model $p(\mathbf{y}|\mathbf{f})$
  - GP prior $p(\mathbf{f})$ for the latent function $\mathbf{f}$
  - Bayes rule

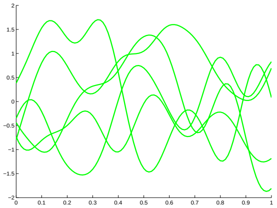
$$\begin{array}{rcl} p(\mathbf{f}|\mathbf{y}) & \propto & p(\mathbf{y}|\mathbf{f}) \times p(\mathbf{f}) \\ \text{Posterior} & \propto & \text{Likelihood} \times \text{Prior} \end{array}$$

- For regression, where the likelihood is Gaussian, this computation is analytically obtained

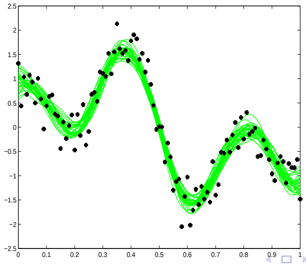
Gaussian Processes for Bayesian Regression

- Data and the GP prior (rbf kernel function)





- Posterior GP process



Gaussian Processes for non-Gaussian Likelihoods

- When the likelihood $p(\mathbf{y}|\mathbf{f})$ is non-Gaussian computations are analytically intractable
- Non-Gaussian likelihoods:
  - Classification problems
  - Spatio-temporal models and geostatistics
  - Non-linear differential equations with latent functions
- Approximations need to be considered
- MCMC is a powerful framework that offers:
  - Arbitrarily precise approximation in the limit of long runs
  - General applicability (independent from the functional form of the likelihood)

MCMC for Gaussian Processes

The Metropolis-Hastings (MH) algorithm

- Initialize $\mathbf{f}^{(0)}$
- Form a Markov chain. Use a proposal distribution $Q(\mathbf{f}^{(t+1)}|\mathbf{f}^{(t)})$ and accept with the MH step

$$\min\left(1, \frac{p(\mathbf{y}|\mathbf{f}^{(t+1)})p(\mathbf{f}^{(t+1)})}{p(\mathbf{y}|\mathbf{f}^{(t)})p(\mathbf{f}^{(t)})}\frac{Q(\mathbf{f}^{(t)}|\mathbf{f}^{(t+1)})}{Q(\mathbf{f}^{(t+1)}|\mathbf{f}^{(t)})}\right)$$

- The posterior is highly-correlated and $\mathbf{f}$ is high dimensional
- How do we choose the proposal $Q(\mathbf{f}^{(t+1)}|\mathbf{f}^{(t)})$?

MCMC for Gaussian Processes

Use the GP prior as the proposal distribution

- Proposal: $Q(\mathbf{f}^{(t+1)}|\mathbf{f}^{(t)}) = p(\mathbf{f}^{(t+1)})$
- MH probability

$$\min\left(1, \frac{p(\mathbf{y}|\mathbf{f}^{(t+1)})}{p(\mathbf{y}|\mathbf{f}^{(t)})}\right)$$

- Nice property: The prior samples functions with the appropriate smoothing requirement
- Bad property: We get almost zero acceptance rate. The chain will get stuck in the same state for thousands of iterations

MCMC for Gaussian Processes

Use Gibbs sampling

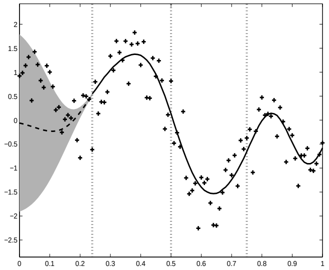
- Proposal: Iteratively sample from the conditional posterior $p(f_i|\mathbf{f}_{-i}, \mathbf{y})$ where $\mathbf{f}_{-i} = \mathbf{f} \setminus f_i$
- Nice property: All samples are accepted and the prior smoothing requirement is satisfied
- Bad property: The Markov chain will move extremely slowly for densely sampled functions:
  - The variance of $p(f_i|\mathbf{f}_{-i}, \mathbf{y})$ is smaller or equal to the variance of the conditional prior $p(f_i|\mathbf{f}_{-i})$
  - But $p(f_i|\mathbf{f}_{-i})$ may already have a tiny variance

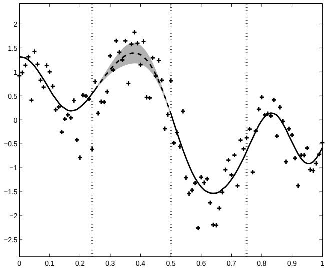
Gibbs-like schemes

- Gibbs-like algorithm: Instead of $p(f_i|\mathbf{f}_{-i}, \mathbf{y})$ use the conditional prior $p(f_i|\mathbf{f}_{-i})$ and accept with the MH step (it has been used in geostatistics, Diggle and Tawn, 1998)
- Gibbs-like algorithm is still inefficient to sample from highly correlated functions
- Block or region sampling:
    - Cluster the function values $\mathbf{f}$ into regions/blocks $\{\mathbf{f}_k\}_{k=1}^{M}$
    - Sample each block $\mathbf{f}_k$ from the conditional GP prior $p(\mathbf{f}_k^{(t+1)}|\mathbf{f}_{-k}^{(t)})$, where $\mathbf{f}_{-k} = \mathbf{f} \setminus \mathbf{f}_k$ and accept with the MH step
    - This scheme can work better
    - But it does not solve the problem of sampling highly correlated functions since the variance of the proposal can be very small in the boundaries between regions

Gibbs-like schemes

- Region sampling with 4 regions (2 of the proposals are shown below)





- Note that the variance of the conditional priors is small close to the boundaries between regions

Sampling using control variables

- Let $\mathbf{f}_c$ be a set of auxiliary function values. We call them control variables
- The control variables provide a low dimensional representation of $\mathbf{f}$ (analogously to the inducing/active variables in sparse GP models)
- Using $\mathbf{f}_c$, we can write the posterior

$$p(\mathbf{f}|\mathbf{y}) = \int_{\mathbf{f}_c} p(\mathbf{f}|\mathbf{f}_c, \mathbf{y}) p(\mathbf{f}_c|\mathbf{y}) d\mathbf{f}_c$$

  When $\mathbf{f}_c$ is highly informative about $\mathbf{f}$, ie. $p(\mathbf{f}|\mathbf{f}_c, \mathbf{y}) \simeq p(\mathbf{f}|\mathbf{f}_c)$, we can approximately sample from $p(\mathbf{f}|\mathbf{y})$:

  - Sample the control variables from $p(\mathbf{f}_c|\mathbf{y})$
  - Generate $\mathbf{f}$ from the conditional prior $p(\mathbf{f}|\mathbf{f}_c)$

Sampling using control variables

- Idea: Sample the control variables from $p(\mathbf{f}_c|\mathbf{y})$ and generate $\mathbf{f}$ from the conditional prior $p(\mathbf{f}|\mathbf{f}_c)$
- Make this a MH algorithm: We only need to specify the proposal $q(\mathbf{f}_c^{(t+1)}|\mathbf{f}_c^{(t)})$, that will mimic sampling from $p(\mathbf{f}_c|\mathbf{y})$
- The whole proposal is

$$Q(\mathbf{f}^{(t+1)}, \mathbf{f}_c^{(t+1)}|\mathbf{f}^{(t)}, \mathbf{f}_c^{(t)}) = p(\mathbf{f}^{(t+1)}|\mathbf{f}_c^{(t+1)})q(\mathbf{f}_c^{(t+1)}|\mathbf{f}_c^{(t)})$$

- Each $(\mathbf{f}^{(t+1)}, \mathbf{f}_c^{(t+1)})$ is accepted using the MH step

$$A = \frac{p(\mathbf{y}|\mathbf{f}^{(t+1)})p(\mathbf{f}_c^{(t+1)})}{p(\mathbf{y}|\mathbf{f}^{(t)})p(\mathbf{f}_c^{(t)})} \frac{q(\mathbf{f}_c^{(t)}|\mathbf{f}_c^{(t+1)})}{q(\mathbf{f}_c^{(t+1)}|\mathbf{f}_c^{(t)})}$$

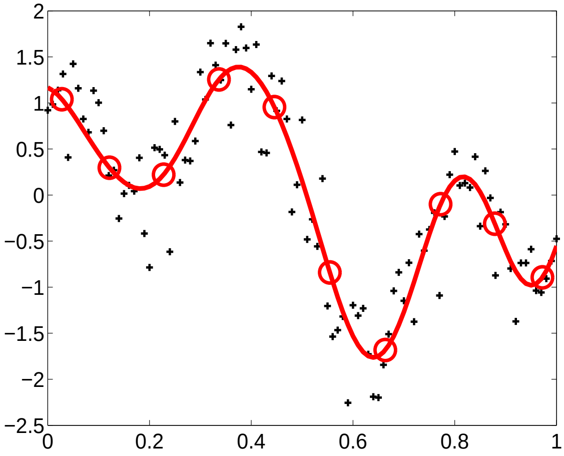
Sampling using control variables: Specification of $q(\mathbf{f}_c^{(t+1)}|\mathbf{f}_c^{(t)})$

- $q(\mathbf{f}_c^{(t+1)}|\mathbf{f}_c^{(t)})$ must mimic sampling from $p(\mathbf{f}_c|\mathbf{y})$
- The control points are meant to be almost independent, thus Gibbs can be efficient
    - Sample each f_{c_i} from the conditional posterior $p(f_{c_i}|\mathbf{f}_{c_{-i}}, \mathbf{y})$
- Unfortunately computing $p(f_{c_i}|\mathbf{f}_{c_{-i}}, \mathbf{y})$ is intractable
- But we can use the Gibbs-like algorithm: Iterate between different control variables i:
    - Sample $f_{c_i}^{(t+1)}$ from $p(f_{c_i}^{(t+1)}|\mathbf{f}_{c_{-i}}^{(t)})$ and $\mathbf{f}^{(t+1)}$ from $p(\mathbf{f}^{(t+1)}|f_{c_i}^{(t+1)}, \mathbf{f}_{c_{-i}}^{(t)})$. Accept with the MH step
    - The proposal for $\mathbf{f}$ is the leave-one-out conditional prior

$$p(\mathbf{f}^{t+1}|\mathbf{f}_{c_{-i}}^{(t)}) = \int_{f_{c_i}^{(t+1)}} p(\mathbf{f}^{t+1}|f_{c_i}^{(t+1)}, \mathbf{f}_{c_{-i}}^{(t)}) p(f_{c_i}^{(t+1)}|\mathbf{f}_{c_{-i}}^{(t)}) df_{c_i}^{(t+1)}$$

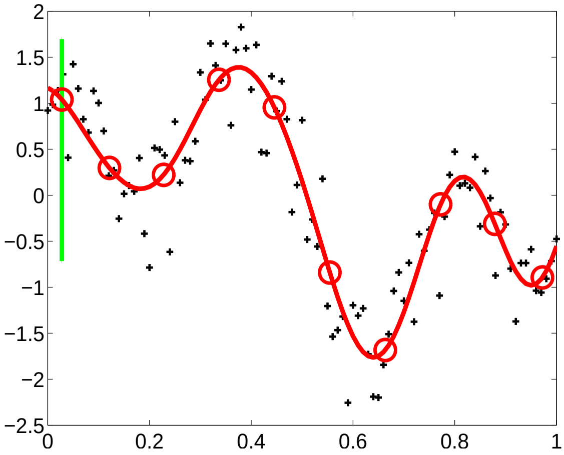
Sampling using control variables: Demonstration

Data, current $\mathbf{f}^{(t)}$ (red line) and current control variables $\mathbf{f}_c^{(t)}$ (red circles)



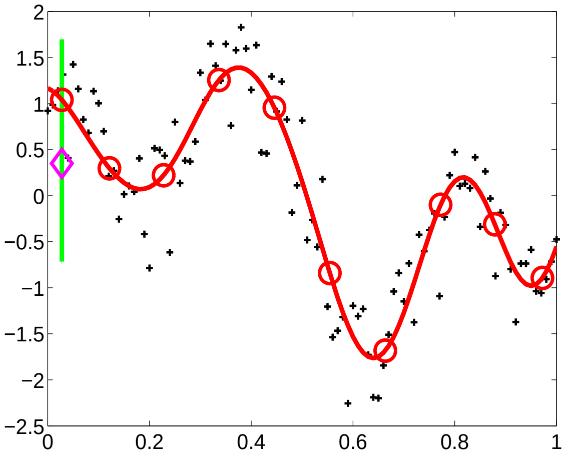
Sampling using control variables: Demonstration

First control variable: The proposal $p(f_{c_1}^{(t+1)}|\mathbf{f}_{c_{-1}}^{(t)})$ (green bar)



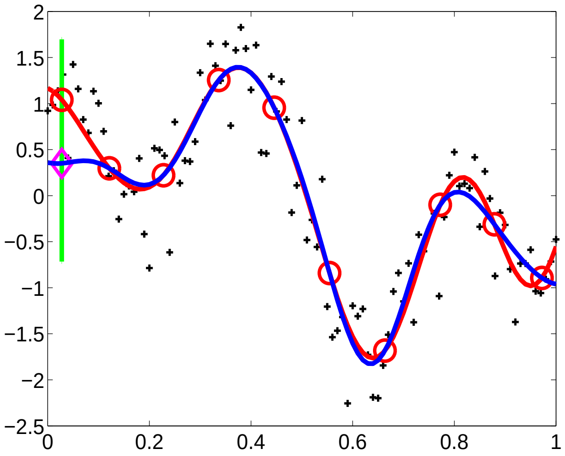
Sampling using control variables: Demonstration

First control variable: The proposed $f_{c_1}^{(t+1)}$ (diamond in magenta)



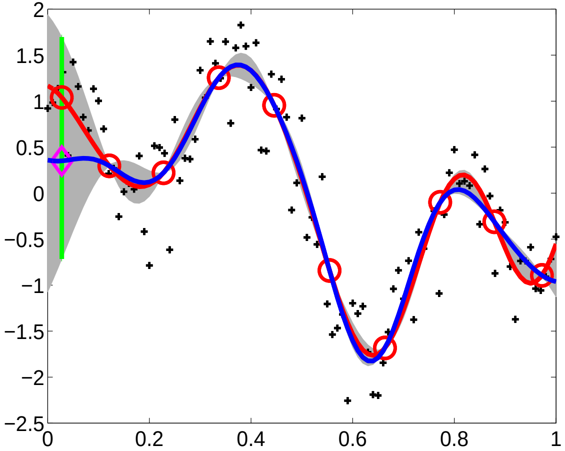
Sampling using control variables: Demonstration

First control variable: The proposed function $\mathbf{f}^{(t+1)}$ (blue line)



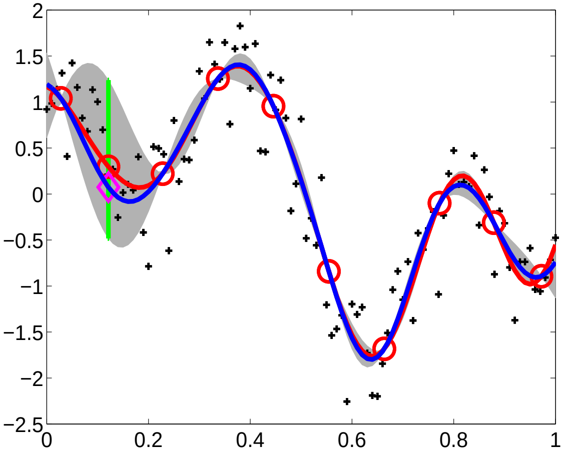
Sampling using control variables: Demonstration

First control variable: **Shaded area** is the overall effective proposal $p(\mathbf{f}^{(t+1)}|\mathbf{f}_{c_{-1}}^{(t)})$



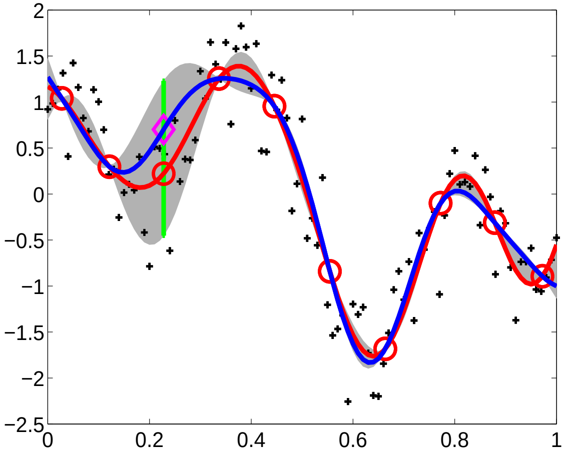
Sampling using control variables: Demonstration

Iteration between control variables: Allows **f** to be drawn with considerable variance everywhere in the input space.



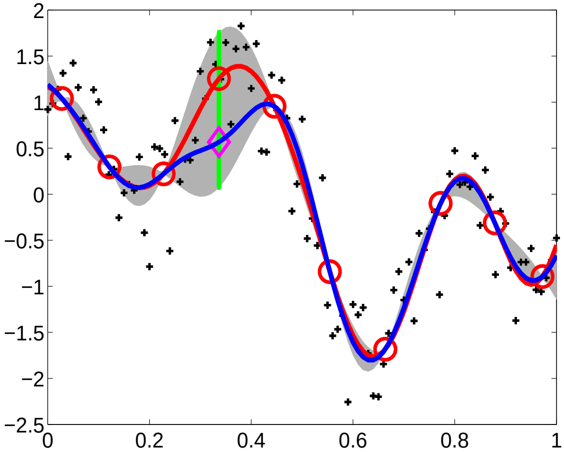
Sampling using control variables: Demonstration

Iteration between control variables: Allows **f** to be drawn with considerable variance everywhere in the input space.



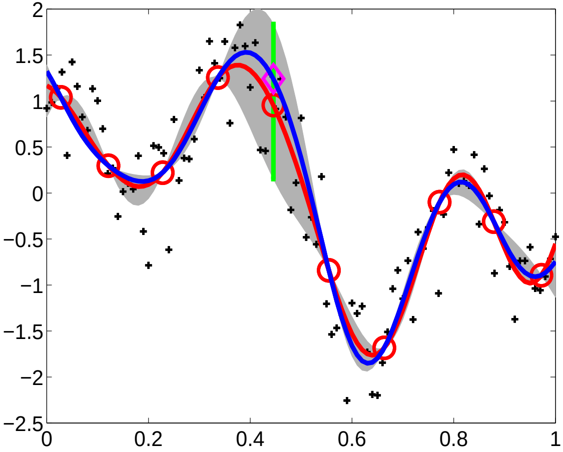
Sampling using control variables: Demonstration

Iteration between control variables: Allows **f** to be drawn with considerable variance everywhere in the input space.



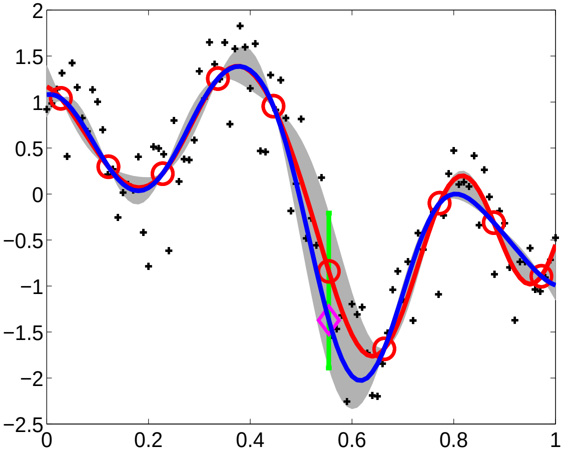
Sampling using control variables: Demonstration

Iteration between control variables: Allows **f** to be drawn with considerable variance everywhere in the input space.



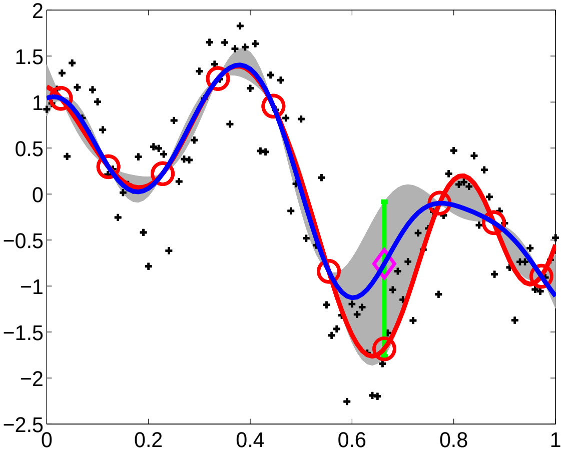
Sampling using control variables: Demonstration

Iteration between control variables: Allows **f** to be drawn with considerable variance everywhere in the input space.



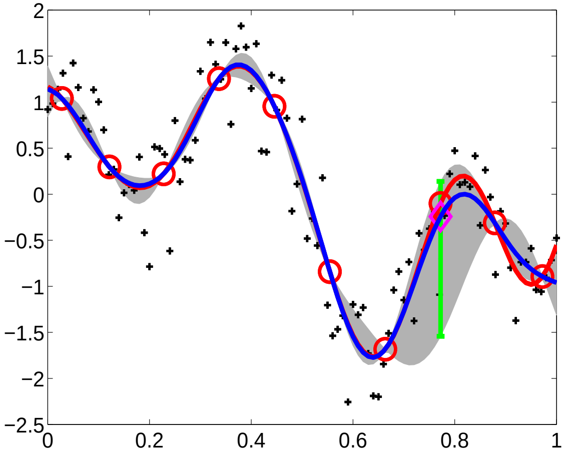
Sampling using control variables: Demonstration

Iteration between control variables: Allows **f** to be drawn with considerable variance everywhere in the input space.



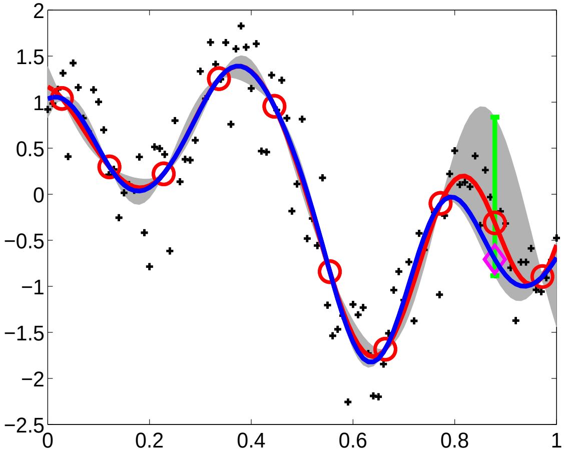
Sampling using control variables: Demonstration

Iteration between control variables: Allows **f** to be drawn with considerable variance everywhere in the input space.



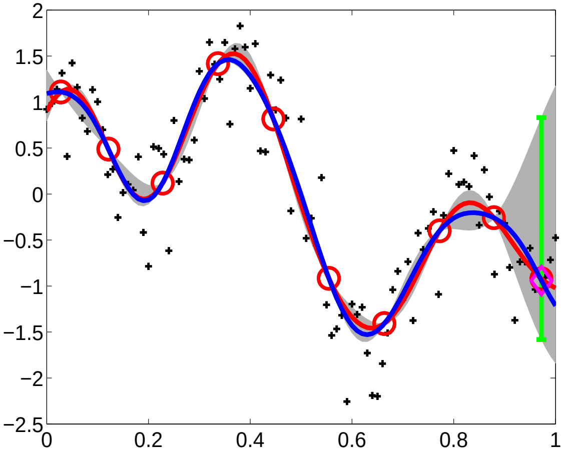
Sampling using control variables: Demonstration

Iteration between control variables: Allows **f** to be drawn with considerable variance everywhere in the input space.



Sampling using control variables: Demonstration

Iteration between control variables: Allows **f** to be drawn with considerable variance everywhere in the input space.



Sampling using control variables: Input control locations

- To apply the algorithm, we need to select the number M of control variables and their input locations X_c
- Choose X_c using a PCA-like approach
  - Knowledge of $\mathbf{f}_c$ must determine $\mathbf{f}$ with small error
  - Given $\mathbf{f}_c$ the prediction of $\mathbf{f}$ is $K_{f,c}K_{c,c}^{-1}\mathbf{f}_c$
  - Mininimize the averaged error $||\mathbf{f} - K_{f,c}K_{c,c}^{-1}\mathbf{f}_c||^2$

$$\begin{aligned} G(X_c) &= \int_{\mathbf{f},\mathbf{f}_c} ||\mathbf{f} - K_{f,c}K_{c,c}^{-1}\mathbf{f}_c||^2 p(\mathbf{f}|\mathbf{f}_c)p(\mathbf{f}_c)d\mathbf{f}d\mathbf{f}_c \\ &= \mathrm{Tr}(K_{f,f} - K_{f,c}K_{c,c}^{-1}K_{f,c}^T) \end{aligned}$$

  - Minimize $G(X_c)$ w.r.t. X_c using gradient-based optimization

Note: $G(X_c)$ is the total variance of the conditional prior $p(\mathbf{f}|\mathbf{f}_c)$

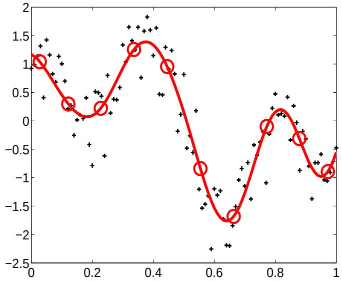
Sampling using control points: Choice of M

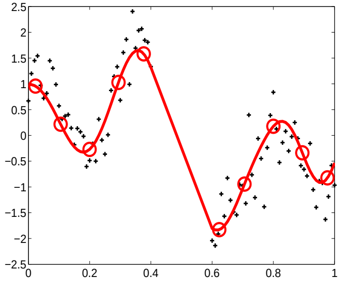
To find the number M of control variables

- Minimize $G(X_c)$ by incrementally adding control variables until $G(X_c)$ becomes smaller than a certain percentage of the total variance of $p(\mathbf{f})$ (5% used in all our experiments)
- Start the simulation and observe the acceptance rate of the chain
- Keep adding control variables until the acceptance rate becomes larger than 25% (following standard heuristics Gelman, Carlin, Stern and Rubin (2004))

Sampling using control variables: $G(X_c)$ function

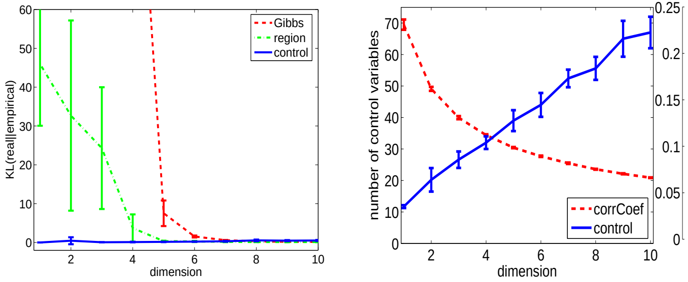
The minimization of G places the control inputs close to the clusters of the input data in such a way that the kernel function is taken into account





Applications: Demonstration on regression

- Regression: Compare Gibbs, local region sampling and control variables in regression (randomly chosen GP functions of varied input-dimensions: $d = 1, \ldots, 10$, with fixed $N = 200$ training points)



- Note: The number of control variables increases as the function values become more independent... this is very intuitive

Applications: Classification

- Classification: Wisconsin Breast Cancer (WBC) and the Pima Indians Diabetes. Hyperparameters fixed to those obtained by Expectation-Propagation

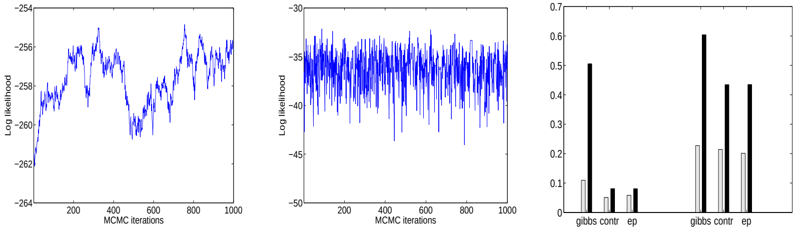


Figure: Log-likelihood for *Gibbs* (left) and *control* (middle) in WBC dataset. (right) shows the test errors (grey bars) and the average negative log likelihoods (black bars) on the WBC (left) and PID (right)

Applications: Transcriptional regulation

- Data: Gene expression levels $\mathbf{y} = (y_{jt})$ of N genes at T times
- Goal: We suspect/know that a certain protein regulates (i.e. is a transcription factor (TF)) these genes and we wish to model this relationship
- Model: Use a differential equation (Barenco et al. [2006]; Rogers et al. [2007]; Lawerence et al. [2007])

$$\frac{dy_j(t)}{dt} = B_j + S_j g(f(t)) - D_j y_j(t)$$

- where

  t - time

  $y_j(t)$ - expression of the jth gene

  $f(t)$ - concentration of the transcription factor protein

  D_j - decay rate

  B_j - basal rate

  S_j - Sensitivity

Transcriptional regulation using Gaussian processes

- Solve the equation

$$y_j(t) = \frac{B_j}{D_j} + A_j \exp(-D_j t) + S_j \exp(-D_j t) \int_0^t g(f(u)) \exp(D_j u) du$$

- Apply numerical integration using a very dense grid $(u_i)_{i=1}^P$ and $\mathbf{f} = (f_i(u_i))_{i=1}^P$

$$y_j(t) \simeq \frac{B_j}{D_j} + A_j \exp(-D_j t) + S_j \exp(-D_j t) \sum_{p=1}^{P_t} w_p g(f_p) \exp(D_j u_p)$$

  Assuming Gaussian noise for the observed gene expressions $\{y_{jt}\}$, the ODE defines the likelihood $p(\mathbf{y}|\mathbf{f})$

- Bayesian inference: Assume a GP prior for the transcription factor $\mathbf{f}$ and apply MCMC to infer $(\mathbf{f}, \{A_j, B_j, D_j, S_j\}_{j=1}^N)$
  - $\mathbf{f}$ is inferred in a continuous manner $(P \gg T)$

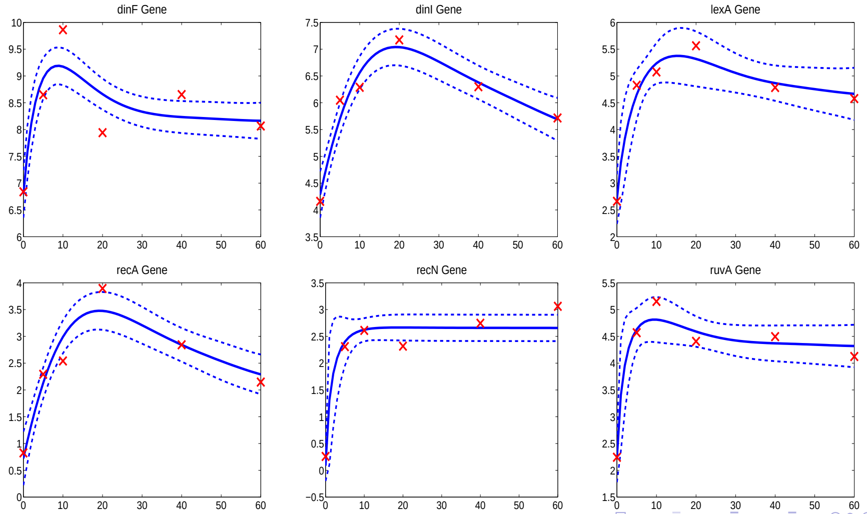
Results in E.coli data: Rogers, Khanin and Girolami (2007)

- One transcription factor (lexA) that acts as a repressor. We consider the Michaelis-Menten kinetic equation

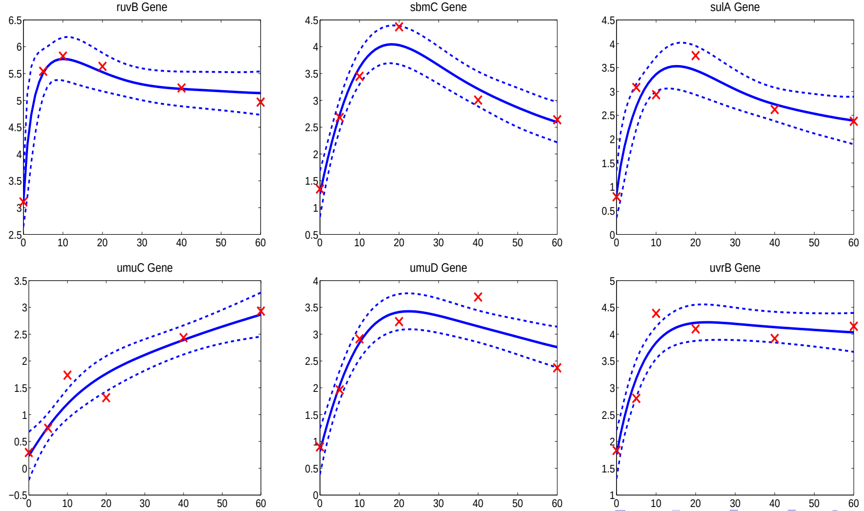
$$\frac{dy_j(t)}{dt} = B_j + S_j \frac{1}{\exp(f(t)) + \gamma_j} - D_j y_j(t)$$

- We have 14 genes (5 kinetic parameters each)
- Gene expressions are available for $T = 6$ time slots
- TF ($\mathbf{f}$) is discretized using 121 points
- MCMC details:
    - 6 control points are used
    - Running time was 5 hours for 5×10^5 iterations plus burn in

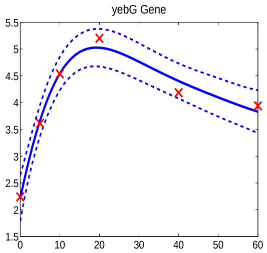
Results in E.coli data: Predicted gene expressions

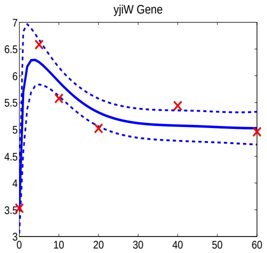


Results in E.coli data: Predicted gene expressions

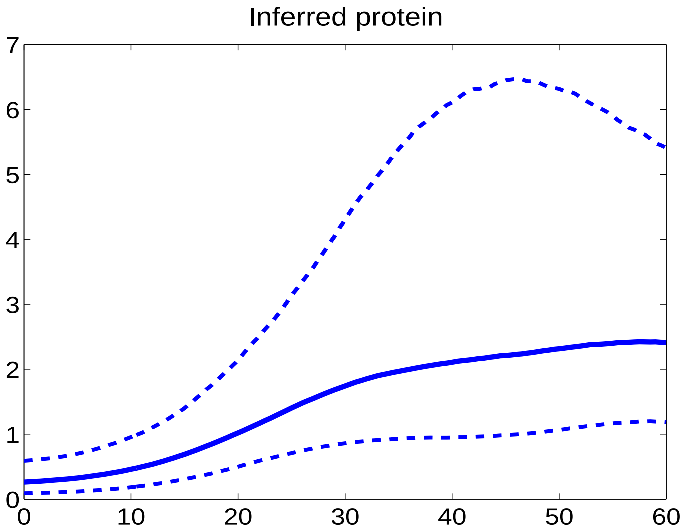


Results in E.coli data: Predicted gene expressions

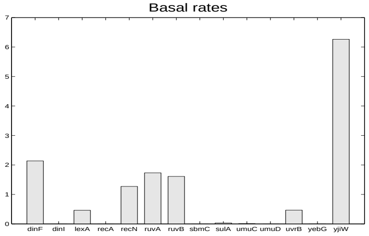


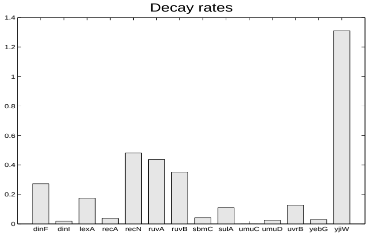


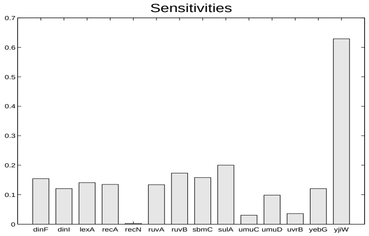
Results in E.coli data: Protein concentration

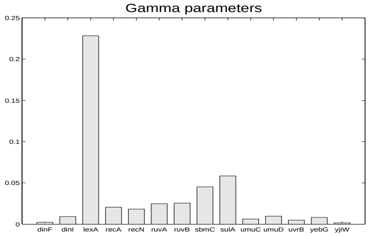


Results in E.coli data: Kinetic parameters

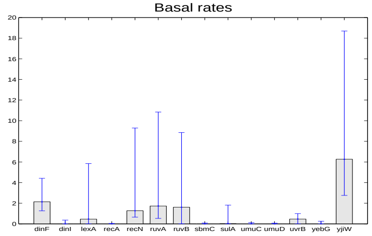


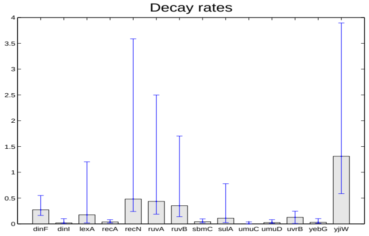


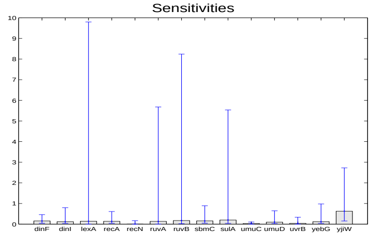


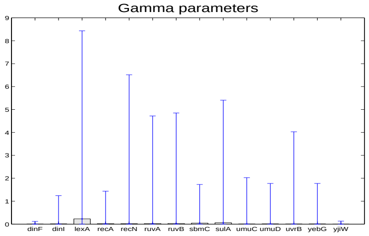


Results in E.coli data: Confidence intervals for the kinetic parameters









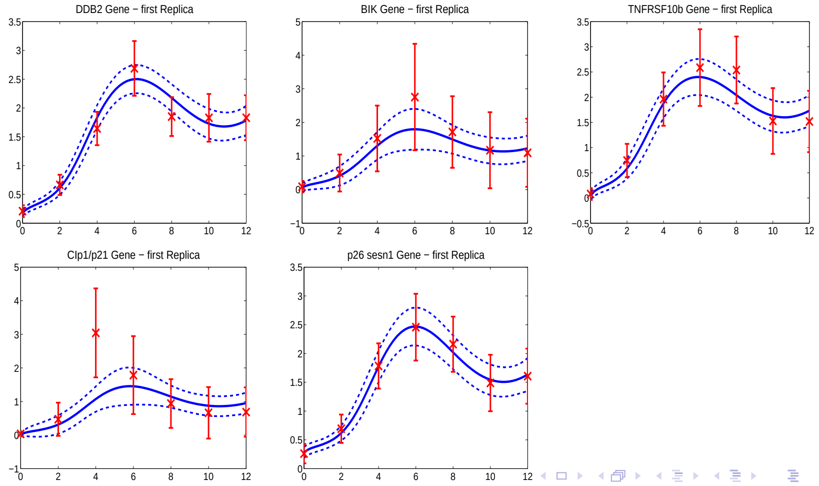
Data used by Barenco et al. [2006]

- One transcription factor (p53) that acts as an activator. We consider the Michaelis-Menten kinetic equation

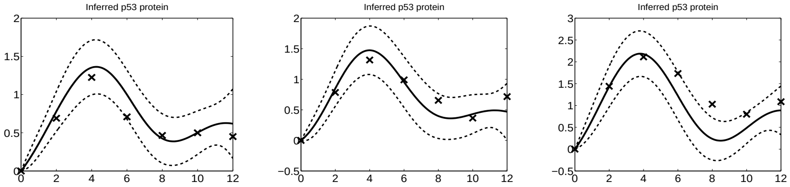
$$\frac{dy_j(t)}{dt} = B_j + S_j \frac{\exp(f(t))}{\exp(f(t)) + \gamma_j} - D_j y_j(t)$$

- We have 5 genes
- Gene expressions are available for $T = 7$ times and there are 3 replicas of the time series data
- TF ($\mathbf{f}$) is discretized using 121 points
- MCMC details:
    - 7 control points are used
    - Running time 4 hours for 5×10^5 iterations plus burn in

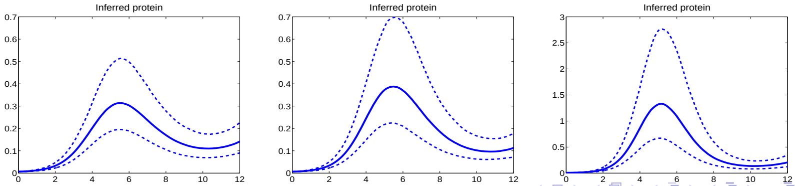
Data used by Barenco et al. [2006]: Predicted gene expressions for the 1st replica



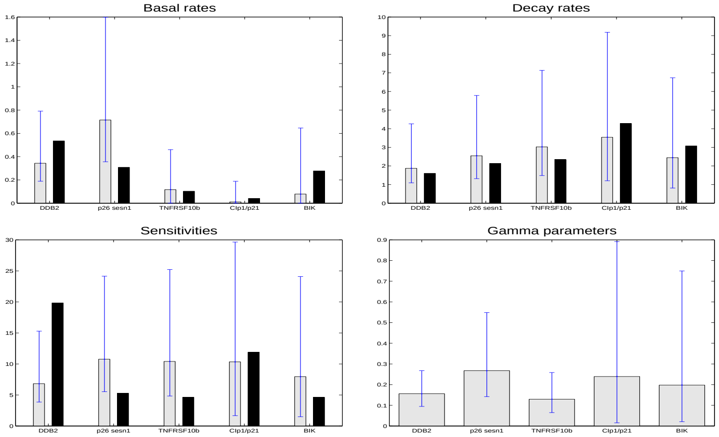
Data used by Barenco et al. [2006]: Protein concentrations



Linear model (Barenco et al. predictions are shown as crosses)



Data used by Barenco et al. [2006]: Kinetic parameters



Our results (grey) compared with Barenco et al. [2006] (black).
Note that Barenco et al. use a linear model

Summary/Future work

Summary:

- A new MCMC algorithm for Gaussian processes using control variables
- It can be generally applicable

Future work:

- Deal with large systems of ODEs for the transcriptional regulation application
- Consider applications in geostatistics
- Use the $G(X_c)$ function to learn sparse GP models in an unsupervised fashion without the outputs $\mathbf{y}$ being involved