Gaussian Processes
A method for rapidly emulating the behaviour of expensive computational codes

C Coleman-Smith

Duke Physics,
cec24@phy.duke.edu

July 20, 2009
Outline

- Introduction
- Method
- A simple example:
  - fitting: $Asin(Bx) + Cx$
- A challenge:
  - fitting output from a model with a phase transition
- Conclusions
Physical problems are frequently so nonlinear and so complicated that analytic solutions are either too painful or plain impossible.

- Turn to numerical simulations of the situation.

Computers will never be as fast as we’d like.

- Typical runs for UrQMD can take several days at high energy.
- Large finite element models, such as climate simulations, can take months on supercomputers

We need a method to run the expensive simulation (the model) as few times as possible while extracting the most information from these runs.

We can create an emulator for the model using a gaussian process, this provides an inexpensive way to interpolate the behaviour of the model at new points in the parameter space.

- An emulator is a statistical approximation to a function.
- It can also act as a first step towards statistical verification of a computational model against real world data.
Emulation Process Overview

Emulating a model

- Given a computational simulation of interest the first step is to run the code at a series of points in its parameter space.
  - The distribution of model-points is determined by prior knowledge of the simulation or by an efficient sampling of the entire parameter space.
- The input parameters and scalar results are then used to train a gaussian process such that it reproduces the model, our emulator. This is essentially regression using gaussian functions.
- The resulting emulator can then be used to estimate the output of the model at points outside of the original training set, interpolation works better than extrapolation.
  - Also provides a variance associated with each new point, this combined with prior knowlege of the model can be used to drive further investigation.
- As more data points are collected, the emulator can be refined.
  - Important for models with non-smooth parameter spaces, or which produce output with many inflections.
Bayesian Modelling

Bayes Theorem:

\[ P(A|B) = \frac{P(B|A)P(A)}{P(B)} \]

- Expresses probability of observing A, given that we have observed B: the posterior \( P(A|B) \)
- The prior probability of observing A: \( P(A) \) and B: \( P(B) \) are known or estimated beforehand.

- The Bayesian approach is quite different from familiar frequentist approach, a method for updating one’s degree of belief in a hypothesis given some additional observations, the posterior.
- It allows us to talk about the probability of the true value of a physical quantity, in the frequency frame we can only talk about the probability of the outcome of an experiment.
- Eg: Estimating some set of parameters \( \theta \) given some data \( y \) from a distribution characterised by them.

\[ P(\theta|y) = \frac{f(y|\theta)\pi(\theta)}{Z} \]
Bayesian Modelling, application to our problem

Formal statement of the problem

Given a set of observations $D$ of $N$ pairs of input vectors and scalar outputs: $\{x_n, t_n\}$ from our model. We desire to predict the output $t_{n+1}$ of the model at a new point $x_{n+1}$.

We can posit a prior distribution for our model itself $f$, over the space of all possible models $P(f|\Theta)$ where $\Theta$ is some set of parameters which determine $f$.

$$P(t_N|\{x_n\}, \Theta) = \int dfP(t_N|\{x_n\}, f, \Theta)P(f|\Theta)$$

Where $t_N = (t_1, ..., t_N)$ is a vector of all the model results, if we define $t_{N+1} = (t_1, ..., t_N, t_{N+1})$ then we can write the conditional distribution for $t_{N+1}$, the value of the model at $x_{N+1}$.

$$P(t_{N+1}|D, \Theta) = \frac{P(t_{N+1}|\{x_N\}, \Theta)}{P(t_N|\{x_N\}, \Theta)}$$

As it stands integrating over the set of possible model’s is quite a challenge! We circumvent this by introducing a Gaussian process to act as our model.
A Gaussian process is a natural extension of a gaussian distribution, we have a set of variables $t = (t(x_1), t(x_2), ... t(x_N))$ which are jointly distributed:

$$P(t|C, \{x_n\}) = \frac{1}{Z} \exp\left(-\frac{1}{2}(t - \mu)^T C^{-1}(t - \mu)\right)$$

Where $C$ is the covariance matrix of the process, $\mu$ is the mean vector and $Z$ is a normalising constant. Entries in $C$ are given by a covariance function: $C(x_m, x_n; \Theta)$.

We suppose that our model output vector $t_N$ is a gaussian process, with covariance matrix; $C_N$ and a zero mean vector.

The conditional distribution for the new point $\{x_{N+1}, t_{N+1}\}$ is:

$$P(t_{N+1}|D, C, x_{N+1}, \Theta) = \frac{P(t_{N+1}|C, \Theta, x_{N+1}, \{x_n\})}{P(t_N|C, \Theta, \{x_n\})}$$

$$= \frac{1}{Z^*} \exp\left(-\frac{1}{2}(t_{N+1}^T C_{N+1}^{-1} t_{N+1} - t_N^T C_N^{-1} t_N)\right)$$

Where $Z^*$ is some overall normalization function.
The new covariance matrix $C_{N+1}$ is a direct extension of the original covariance matrix of the training data $C_N$.

We can express $C_{N+1}$ in terms of $C_N$, resulting in a probability distribution for the new point $t_{N+1}$ at $x_{N+1}$.

\[
P(t_{N+1}|D, C, x_{N+1}, \Theta) = \frac{1}{Z^*} \exp \left( - \frac{(t_{N+1} - \hat{t}_{N+1})^2}{2\hat{\sigma}^2_{t_{N+1}}} \right)
\]

Where we have defined the mean and variance of the above gaussian to be:

\[
\hat{t}_{N+1} = k_{N+1}^T C_N^{-1} t_N
\]
\[
\sigma^2_{\hat{t}_{N+1}} = \kappa - k_{N+1}^T C_N^{-1} k_{N+1}
\]

Where $k_{N} = (C(x_1, x_{N+1}), C(x_2, x_{N+1}), \ldots, C(x_N, x_{N+1}))$ and $\kappa = C(x_{N+1}, x_{N+1})$.

This probability distribution is our emulator. It is a statistical approximation for the action of the model.
The mathematical details

- The matrices $C_{N+1}$ and $C_N$ are related as follows:

\[ \hat{t}_{N+1} = k^T_{N+1}C_N^{-1}t_N \]
\[ \sigma^2_{\hat{t}_{N+1}} = \kappa - k^T_{N+1}C_N^{-1}k_{N+1} \]

- The inverse of a nonsingular partitioned matrix $M$, is given by:

\[ M = \begin{pmatrix} A & B \\ C & D \end{pmatrix}^{-1} = \begin{pmatrix} \hat{\alpha} & \hat{\beta} \\ \hat{\gamma} & \hat{\delta} \end{pmatrix} \]
\[ \hat{\alpha} = (A - BD^{-1}A) \]
\[ \hat{\beta} = -\hat{\alpha}^{-1}BD^{-1} \]
\[ \hat{\gamma} = -D^{-1}C\hat{\alpha}^{-1} \]
\[ \hat{\delta} = D^{-1} + D^{-1}C\hat{\alpha}^{-1}BD^{-1} \]
The Covariance Function

- The form of the covariance matrix largely specifies the behaviour of the emulator, we are constrained to covariance functions which lead to $C$ being nonsingular. It is efficient to separate the covariance function into model and noise components.

$$C(x_m, x_n; \Theta) = C_f(x_m, x_n; \Theta) + \delta_{mn}N(x_m; \Theta)$$

- We choose the model part to be gaussian in $(x_m - x_n)$ this reflects our prior belief in the correlation of local points.

$$C_f(x_m, x_n; \Theta) = \theta_1 \exp \left( -\frac{1}{2} \sum_{l=1}^{L} \frac{(x_m^{(l)} - x_n^{(l)})^2}{\theta_3} \right) + \theta_2$$

The various $\theta$ parameters belong to $\Theta$ and need to be determined for the emulator to work. They specify the amplitude, offset and scale-length of the correlation function.

- The noise term could be a function of $x_m$, to model systems with correlated noise. However currently it is simply:

$$N(x_m; \Theta) = \theta_4$$

Where $\theta_4$ provides a constant additional variance to the process, a “nugget” term.
The Covariance Matrix

- The covariance function $C(x_m, x_n; \Theta)$ is only a direct function of the relative separation of the input parameters $(x_m - x_n)$.
- All of the curve “fitting” as such, comes from the choice of thetas which parameterise the covariance.
- For large data sets the inversion of $C$ can become difficult, special care needs to be taken to ensure the matrix is well conditioned.

Covariance matrix for clustered points.

Covariance matrix for uniformly distributed points.
To create the emulator we need to specify $\Theta$. Theoretically we could evaluate:

$$P(t_{N+1}|x_{N+1}, D, C) = \int P(t_{N+1}|x_{N+1}, D, C, \Theta)P(\Theta|D, C)d\Theta$$

This integration is again challenging, however we can approximate the answer:

If we assume that the posterior distribution over $\Theta$: $P(\Theta|t_{N+1}, x_{N+1}, D, C)$ is sharply peaked around the most probable values of $\Theta_{MP}$, we can write:

$$P(t_{N+1}|x_{N+1}, D, C) \approx P(t_{N+1}|x_{N+1}, D, C, \Theta_{MP})P(\Theta_{MP}|D, C)d\Delta\Theta$$

We need to find the most probable parameters: $\Theta_{MP}$. The posterior distribution for $\Theta$ is:

$$P(\Theta|D, C) = \frac{P(t_{N}|x_{N}, C, \Theta)P(\Theta)}{P(t_{N}|x_{N}, C)}$$

We ignore the prior $P(\Theta)$ and the evidence terms $P(t_{N}|x_{N}, C)$, and seek to maximise the log likelyhood to obtain the best values for the parameters.

$$L = -\frac{1}{2} \log(\text{Det}(C_N)) - \frac{1}{2} t_N^T C_N^{-1} t_N - \frac{N}{2} \log 2\pi$$

This is carried out numerically using a nelder-mead algorithm. 
A summary of the Emulation Process

1. Data Points
2. Model
3. Output
4. Parameter Estimation
   - loop until satisfied
5. Emulator
6. Model Estimation
Our simple model function (green trace) is: $4 + \sin(-8x) + x$
A simple example - 1

If we only sample the model at 2 points we get:

The correlation length here is very short $\approx 0.001$
If we increase the model sampling to 4 points:

The correlation length here is again very short $\approx 0.001$. 
The correlation length here is $\theta_3 = 0.113$, this is long enough to bind the gaussians together.

This example makes it clear that a good knowledge of the model at hand is important in estimating the values of $\Theta$. 
The Ising model represents the spin-coupled interaction of a set of N particles fixed in place on a lattice. It is a simple model for paramagnetism in solids.

\[ H = -\frac{1}{2} \sum_{\langle i,j \rangle} J_{ij} S_i S_j - \sum_i B_i S_i \]

The interactions are taken only over nearest neighbours on the lattice, in the following the magnetic field at each site \( B_i \) is taken to be zero and the coupling constant \( J_{ij} \) is taken to be uniformly 1.

The model is implemented via a Metropolis Monte Carlo algorithm, a lattice is created and spins are flipped either by the spin-interactions or randomly due to the heat-bath in which the system is embedded.

A phase transition in the magnetization, net spin, of the lattice is observed as the heat bath temperature \( T \) passes through it’s critical value \( T_c \).
The Ising Model

We want to fit the magnetisation over the phase transition.

showing the variation of magnetisation against temperature for a 64x64 ising model
Each point is the mean of 5 repeats
No errors were taken into account during emulation

The errors are not included in the following analysis.
- This instance has no nugget term, so it is forced through all of the points.
- The correlation length here is very short: $\theta_3 = 0.0048$
The new (blue) trace has a small nugget $\theta_4 = 0.0023$

- We force the correlation length to be longer $\theta_3 = 0.389$, this *loosens* the emulator.
By applying the emulator to three separate regions we obtain a far better result showing the variation of magnetisation against temperature for a 64x64 Ising model. Each point is the mean of 5 repeats. No errors were taken into account during emulation.
Conclusions

- Emulation of smooth scalar models is quite straightforward, estimating the parameters $\Theta$ is the hard part.
- Choice of covariance function is important if the model is not smooth or is periodic.
- Emulation of models with phase transitions requires some care.
- Automatic splitting of a model into safe to emulate regions seems to be a way to deal with this.
- The emulator is a good statistical representation of a computational model, this can be used to evaluate the accuracy of the model itself against field data.
- To Do:
  - Implement support for non-nugget errors.
  - Implement Automatic region splitting.
  - Implement support for user specified covariance functions.
Suggested References

Thanks to Professors Bass and Mueller for their input.

Thanks to Sebastian Eiser for R based emulator implementation and documents.