

Unsupervised Learning 2001

Lecture 8: Gaussian Processes

Zoubin Ghahramani

`zoubin@gatsby.ucl.ac.uk`

Carl Edward Rasmussen

`edward@gatsby.ucl.ac.uk`

**Gatsby Computational Neuroscience Unit
MSc Intelligent Systems, Computer Science**

Historical Notes

Gaussian Process models are a very natural way to do regression, and go back a long way:

- Blight and Ott (1975), “A Bayesian approach to model inadequacy for polynomial regression” *Biometrika*.
- O’Hagan (1978), “Curve fitting and Optimal Design for Prediction”, *JRSSB*.
- Journel and Huijbregts (1978), “Mining Geostatistics”, Academic Press.
- Sacks, Welch, Mitchell and Wynn (1989), “Design and Analysis of Computer Experiments”, *Statistical Science*.

However, GP models have never been very popular, and has only recently received renewed attention.

Reasons: people didn’t like Bayesian analysis; didn’t have computers that could invert large matrices, couldn’t figure out how to set the priors ...

Bayesian Inference in the Standard Linear Model

Given a set of data containing n input/target pairs $\mathcal{D} = \{x^{(c)}, t^{(c)} : c = 1 \dots n\}$ we may hypothesize a linear model with Gaussian observation noise:

$$y = x^\top w, \quad t = y + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma_n^2) \quad \Rightarrow \quad \mathbf{t}|\mathbf{x}, w \sim \mathcal{N}(\mathbf{x}^\top w, \sigma_n^2 I), \quad (1)$$

the **likelihood** (the probability of the targets given the inputs and parameters) is Gaussian.

Assuming a Gaussian **prior** on the weights, $w \sim \mathcal{N}(0, \sigma_p^2 I)$, the **posterior** (computed by Bayes rule) is also Gaussian:

$$p(w|\mathbf{x}, \mathbf{t}) = \frac{p(\mathbf{t}|\mathbf{x}, w)p(w)}{p(\mathbf{t}|\mathbf{x})}, \quad w|\mathbf{x}, \mathbf{t} \sim \mathcal{N}(\sigma_p^2 A^{-1} \mathbf{x}^\top \mathbf{t}, \sigma_p^2 \sigma_n^2 A^{-1}), \quad (2)$$

where we have defined $A = \mathbf{x}\mathbf{x}^\top \sigma_p^2 + I\sigma_n^2$. This result for the mean is known as *ridge regression* (ridge term $\alpha = \sigma_n^2/\sigma_p^2$). The predictive distribution for t^* (and also for y^*) given a novel input (or set of inputs), x^* , is Gaussian:

$$p(t^*|\mathbf{x}, \mathbf{t}, x^*) = \int p(t^*|x^*, w)p(w|\mathbf{x}, \mathbf{t})dw, \quad (3)$$
$$t^*|\mathbf{x}, \mathbf{t}, x^* \sim \mathcal{N}(\sigma_p^2 x^{*\top} A^{-1} \mathbf{x}^\top \mathbf{t}, I\sigma_n^2 + \sigma_p^2 \sigma_n^2 x^{*\top} A^{-1} x^*).$$

Generalised Linear Models

Instead of using the inputs x themselves in the linear model, it is straight forward to generalise to a set of m *fixed* basis functions:

$$y = \phi w, \quad \phi = (\phi_1(x), \phi_2(x), \dots, \phi_m(x)), \quad t = y + \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, \sigma_n^2) \quad (4)$$

where ϕ is a (row) vector of m *features*. Assuming Gaussian noise and Gaussian weight **prior**, the **likelihood** and **posterior** are again Gaussian:

$$w \sim \mathcal{N}(0, \sigma_p^2), \quad \mathbf{t}|\mathbf{x}, w \sim \mathcal{N}(\Phi w, I\sigma_n^2), \quad w|\mathbf{x}, \mathbf{t} \sim \mathcal{N}(\sigma_p^2 A^{-1} \Phi^\top \mathbf{t}, \sigma_n^2 \sigma_p^2 A^{-1}), \quad (5)$$

where $\Phi_{ci} = \phi_i(x^{(c)})$, are the inputs projected into feature space, and $A = \Phi^\top \Phi \sigma_p^2 + I\sigma_n^2$. Similarly, defining $K = \Phi \Phi^\top \sigma_p^2 + I\sigma_n^2$, the predictive distribution is¹:

$$\begin{aligned} t^*|x^*, \mathbf{x}, \mathbf{t} &\sim \mathcal{N}(\sigma_p^2 \Phi^* A^{-1} \Phi^\top \mathbf{t}, I\sigma_n^2 + \sigma_n^2 \sigma_p^2 \Phi^* A^{-1} \Phi^{*\top}) \\ &= \mathcal{N}(\sigma_p^2 \Phi^* \Phi^\top K^{-1} \mathbf{t}, \Phi^* \Phi^{*\top} \sigma_p^2 + I\sigma_n^2 - \Phi^* \Phi^\top K^{-1} \Phi \Phi^{*\top} \sigma_p^4). \end{aligned} \quad (6)$$

Here Φ appears only in inner products; if we know how to evaluate inner products we need never explicitly work in the high (possibly infinite!) dim. feature space (*kernel trick*).

¹Use the matrix inversion lemma, $(Z^{-1} + UV^\top)^{-1} = Z - ZU(I - V^\top ZU)^{-1}V^\top Z$, to derive this.

What is a Gaussian Process?

A Gaussian Process (GP) is a collection of Random Variables (any finite number of) which have a joint Gaussian distribution.

A GP can define a distribution over functions: Consider a set of input-output pairs $\mathcal{D} = \{x^{(c)}, y^{(c)} | c = 1, \dots, n\}$ and the following (example) *covariance function*:

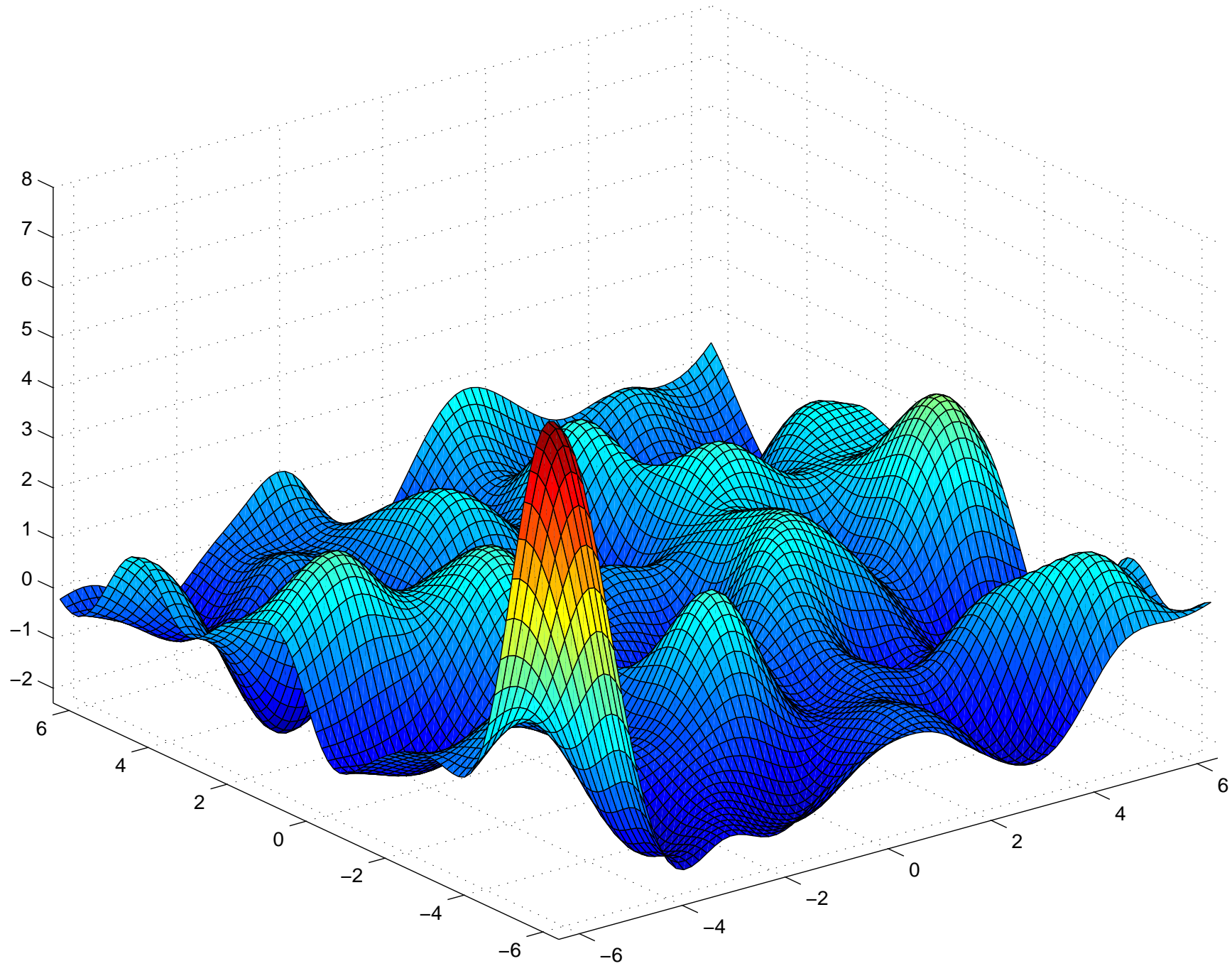
$$\Sigma_{pq} = \text{Cov}(y^{(p)}, y^{(q)}) = K(x^{(p)}, x^{(q)}) = \exp\left(-\frac{1}{2} \sum_{d=1}^D (x_d^{(p)} - x_d^{(q)})^2\right). \quad (7)$$

Given a set of input points $x^{(1)}, \dots, x^{(n)}$ we can compute a covariance matrix Σ , defining a joint Gaussian distribution over the outputs $y^{(1)}, \dots, y^{(n)}$ (or noisy targets, $t^{(1)}, \dots, t^{(n)}$ with added iid Gaussian noise):

$$\mathbf{y} = (y^{(1)}, \dots, y^{(n)}) \sim \mathcal{N}(0, \Sigma), \quad \mathbf{t} = (t^{(1)}, \dots, t^{(n)}) \sim \mathcal{N}(0, \Sigma + \sigma_n^2 I). \quad (8)$$

We can randomly pick a sample from this joint Gaussian distribution, and plot a function based on the input-output pairs. Thus, we have used a GP with a specified covariance function to define a distribution over functions.

Function drawn at random from a Gaussian Process with Gaussian covariance



Predictions from GPs & the Equivalence to Bayesian Linear Models

Given a covariance function for the function values y and a noise level σ_n^2 the joint distribution of the *augmented* targets is:

$$\mathbf{t}_{\text{aug}} = (t^{(1)}, \dots, t^{(n)}, t^*) \sim \mathcal{N}(0, \Sigma + \sigma_n^2 I). \quad (9)$$

Now we *condition*² on the observed targets, to obtain:

$$y^* | x^*, \mathbf{x}, \mathbf{t} \sim \mathcal{N}(K(x^*, \mathbf{x})^\top K(\mathbf{x}, \mathbf{x})^{-1} \mathbf{t}, K(x^*, x^*) - K(x^*, \mathbf{x})K(\mathbf{x}, \mathbf{x})^{-1}K(\mathbf{x}, x^*)). \quad (10)$$

Notice, that identifying K in this equation with K in eq. (6), we get *exactly identical* expressions. This result holds irrespective of whether a finite dimensional decomposition of the Covariance function exists. Mercer's theorem:

$$K(x, x') = \sum_{i=1}^{\infty} \lambda_i \phi_i(x) \phi_i(x'), \quad \int K(x, x') \phi(x) dx = \lambda \phi(x'). \quad (11)$$

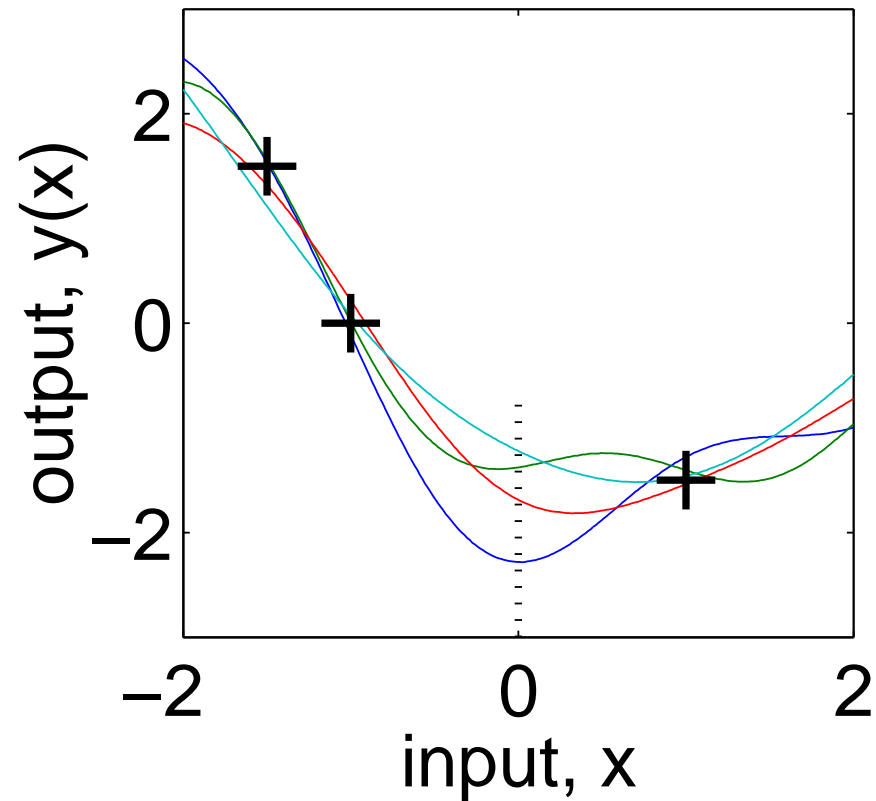
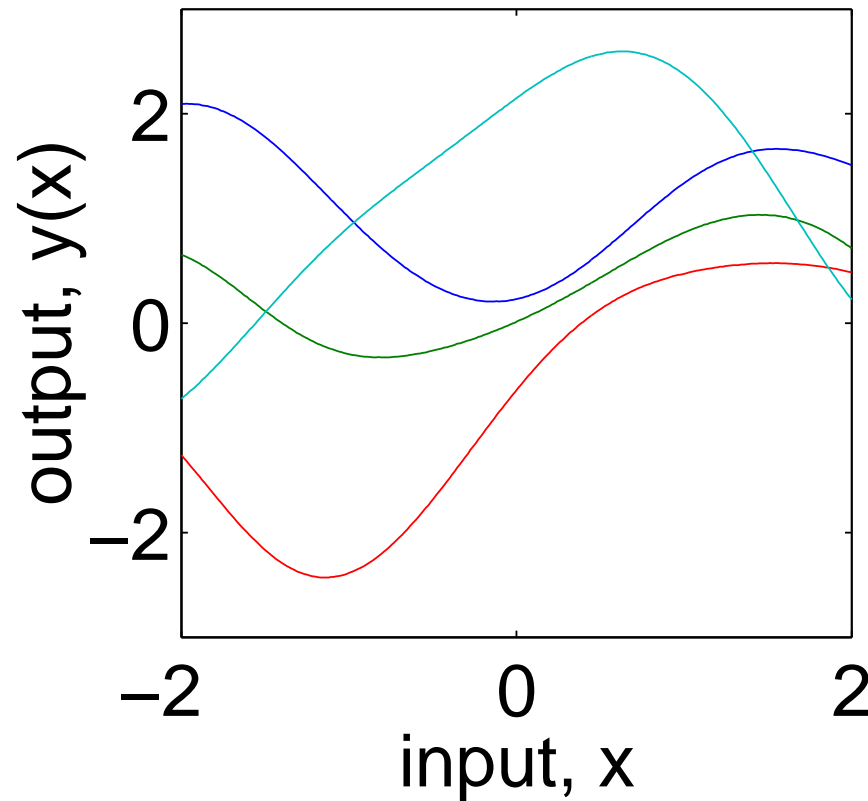
²using: $\begin{bmatrix} x \\ y \end{bmatrix} \sim \mathcal{N}\left(0, \begin{bmatrix} A & C^\top \\ C & B \end{bmatrix}\right) \implies x|y \sim \mathcal{N}(C^\top B^{-1}y, A - C^\top B^{-1}C).$

Thus we have shown that, predictions with a Gaussian Process model is equivalent to linear regression with a potentially infinite number of basis functions.

Tiny Example

Samples from the Prior

Samples from the Posterior



Predictive distribution at test input x^* is Gaussian:

$$y^* | x^*, \mathbf{x}, \mathbf{t} \sim \mathcal{N}(\mu, \sigma^2), \quad \begin{cases} \mu = K(x^*, \mathbf{x})^\top K(\mathbf{x}, \mathbf{x})^{-1} \mathbf{t} \\ \sigma^2 = K(x^*, x^*) - K(x^*, \mathbf{x}) K(\mathbf{x}, \mathbf{x})^{-1} K(\mathbf{x}, x^*) \end{cases} \quad (12)$$

How do we chose good covariance or basis functions?

We can parameterise covariance functions (as long as we respect positive definiteness) using hyper-parameters. For example, the Gaussian covariance function:

$$K(x^{(p)}, x^{(q)}) = v_1 \exp\left(-\sum_d w_d (x_d^{(p)} - x_d^{(q)})^2\right) + \delta_{p,q} \sigma_n^2, \quad (13)$$

expresses that we have high covariance between points that have nearby inputs, and low covariance for points with distant inputs; the exact notion of distance is parameterised by w . Other possible covariance functions include additive models, and non-stationary covariance functions.

How do we adjust the hyper-parameters, assuming that we don't know much about them apriori? Since the model is linear, we can compute the log evidence:

$$\log p(\mathbf{t}|\mathbf{x}, \theta) = -\frac{1}{2} \log |K(\mathbf{x}, \mathbf{x})| - \frac{1}{2} \mathbf{t}^\top K(\mathbf{x}, \mathbf{x})^{-1} \mathbf{t} - \frac{n}{2} \log(2\pi), \quad (14)$$

whose derivatives:

$$\frac{\partial \log p(\mathbf{t}|\mathbf{x}, \theta)}{\partial \theta_i} = -\frac{1}{2} \text{trace}\left(K^{-1} \frac{\partial K}{\partial \theta_i}\right) + \frac{1}{2} \mathbf{t}^\top K^{-1} \frac{\partial K}{\partial \theta_i} K^{-1} \mathbf{t}, \quad (15)$$

can conveniently be evaluated and allow optimization (or integration!) of the evidence.

On the equivalence of GPs to Neural Networks

Consider networks with independent Gaussian weight priors and input $x^{(p)}$. Under the prior the hidden unit activities are independent, identically distributed and bounded with variance $V_h(x^{(p)})$. Thus

$$E[f(x^{(p)})] = E\left[\sum_{i=1}^H v_i h_i(x^{(p)})\right] = 0, \quad E[(f(x^{(p)}))^2] = H\sigma_v^2 V_h(x^{(p)}) + \sigma_b^2. \quad (16)$$

Central limit theorem: for an infinite network the output is Gaussian with variance $H\sigma_v^2 V_h(x^{(1)}) + \sigma_b^2$; therefore, scale the prior σ_v^2 as H^{-1} .

As $H \rightarrow \infty$ the joint distribution of outputs corresponding to different inputs become Gaussian with zero mean and covariance

$$E[f(x^{(p)})f(x^{(q)})] = \sigma_b^2 + \sum_{i=1}^H \sigma_v^2 E[h_i(x^{(p)})h_i(x^{(q)})] = \sigma_b^2 + C(x^{(p)}, x^{(q)}) \quad (17)$$

Gaussian Process: the joint distribution of any finite number of points is multivariate Gaussian.

However, it is probably much easier to apply a GP model in practice than a finite neural network (difficult to select number of hidden units or integrate over parameters).

More about Covariance functions

Modelling task: select covariance function.

Sums of covariance functions are covariance functions (for additive models).

The following family of *stationary* covariance functions:

$$K(x^{(p)}, x^{(q)}) = \exp\left(-\sum_d (x_d^{(p)} - x_d^{(q)})^\nu\right), \quad (18)$$

parameterised by, $0 < \nu \leq 2$, giving the roughness has the (rough) Ornstein-Uhlenbeck process $\nu = 1$ and (smooth) process $\nu = 2$ as special cases.

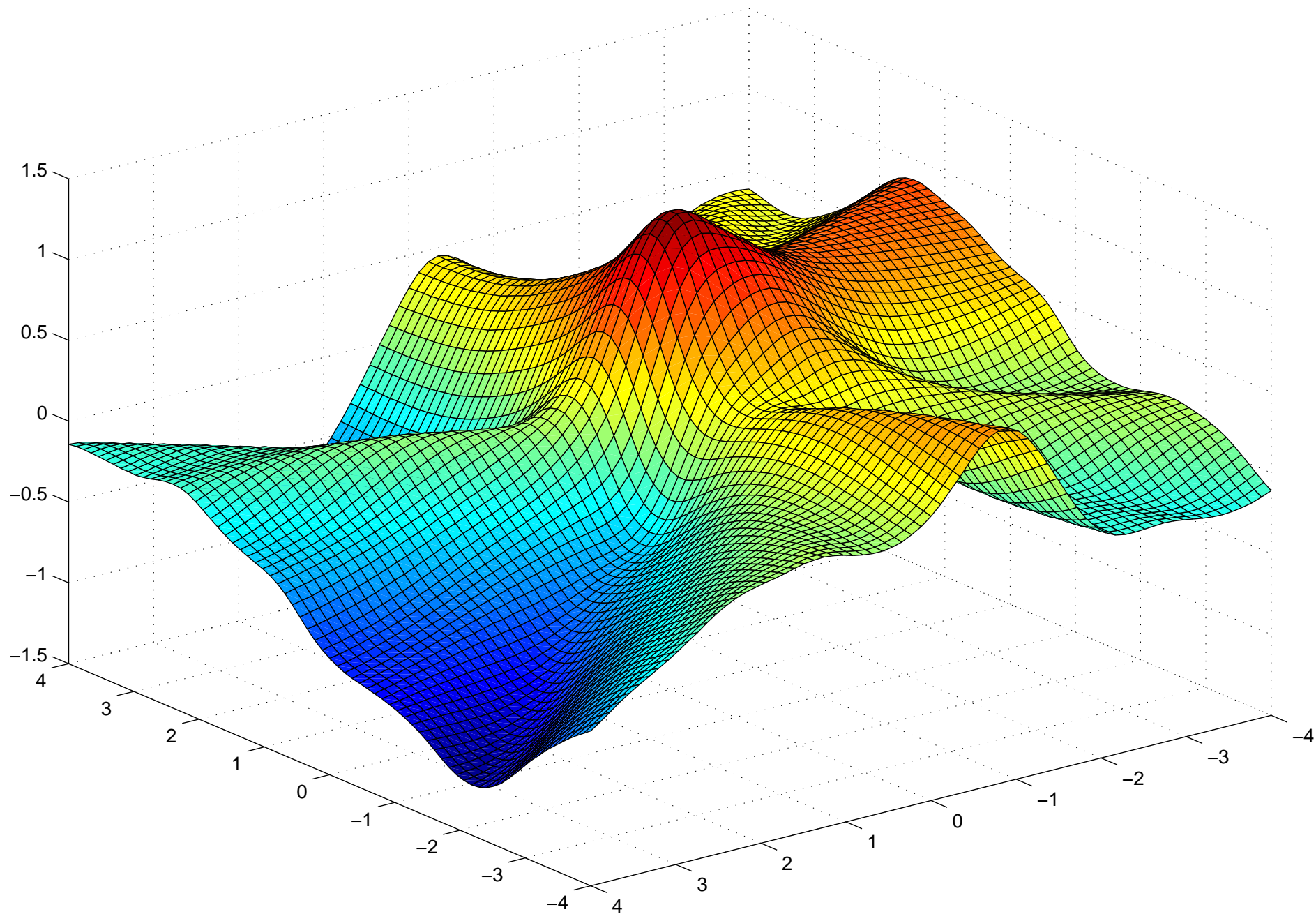
The covariance function for a neural network (with erf hidden units) is given by:

$$K(x, x') = \frac{2}{\pi} \arcsin\left(\frac{2x^\top \Sigma x'}{\sqrt{(1 + x^\top \Sigma x)(1 + 2x'^\top \Sigma x')}}\right), \quad (19)$$

where Σ is the covariance for the (zero mean) Gaussian (ARD) prior on the weights.

It may be helpful to understand properties of the covariance functions by examining functions drawn at random from the priors. For the neural network prior, these functions saturate, as you move away from the origin; a property which may be useful to control applications.

Function drawn at random from a Neural Network covariance function



Computing (and learning from) derivatives

Since differentiation is a linear operator, it is also easy to compute the predictive distribution of (partial) derivatives:

$$E\left[\frac{\partial y}{\partial x_d^*}\right] = \frac{\partial E[y]}{x_d^*} = \frac{\partial \mu}{x_d^*} = \frac{\partial K(x^*, \mathbf{x})}{\partial x_d^*} K(\mathbf{x}, \mathbf{x})^{-1} \mathbf{t}, \quad (20)$$

and similarly the uncertainty of these derivatives

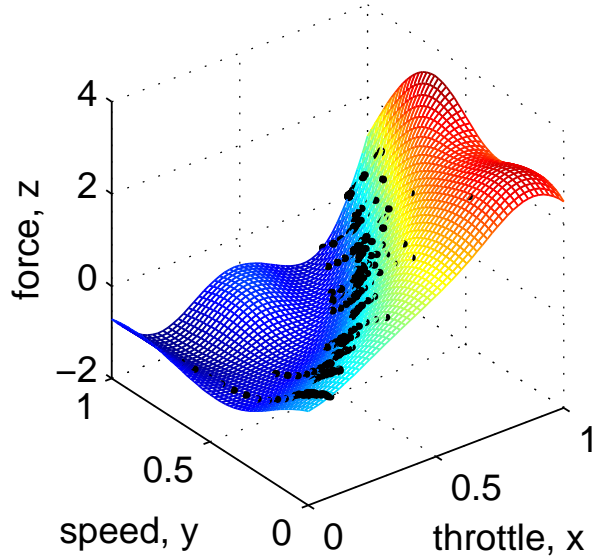
$$E\left[\left(\frac{\partial y}{\partial x_d^*} - E\left[\frac{\partial y}{\partial x_d^*}\right]\right)^2\right] = \frac{\partial^2 K(x^{(p)}, x^{(q)})}{\partial x_d^{(p)} \partial x_d^{(q)}} \Bigg|_{x^{(p)}=x^{(q)}=x^*} - \frac{\partial K(x^*, \mathbf{x})}{\partial x_d^*} K(\mathbf{x}, \mathbf{x})^{-1} \frac{\partial K(\mathbf{x}, x^*)}{\partial x_d^*}. \quad (21)$$

Similarly, if we can obtain (noisy) measurements of derivative information, we can incorporate that in the data on which we base our predictions.

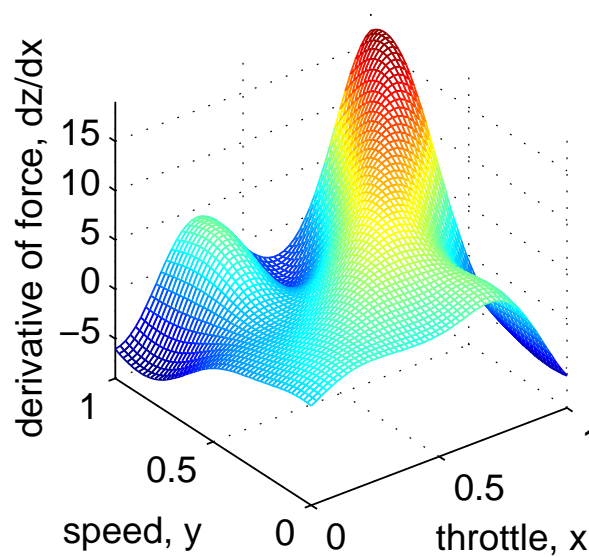
In the examples on the next page, the evidence for the neural network covariance function is 7.9×10^{13} larger than for the Gaussian covariance.

Engine Example: Gaussian Covariance Function

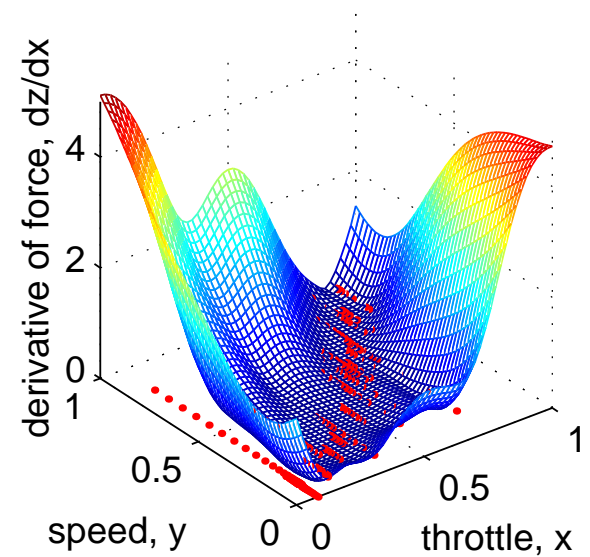
mean predicted force



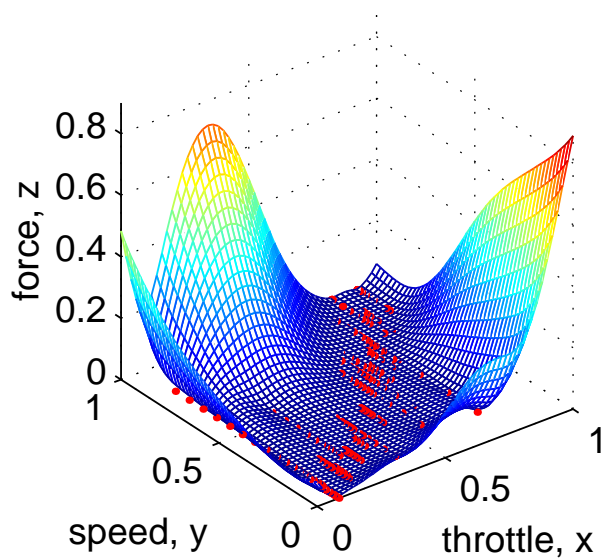
predicted mean partial deriv, dz/dx



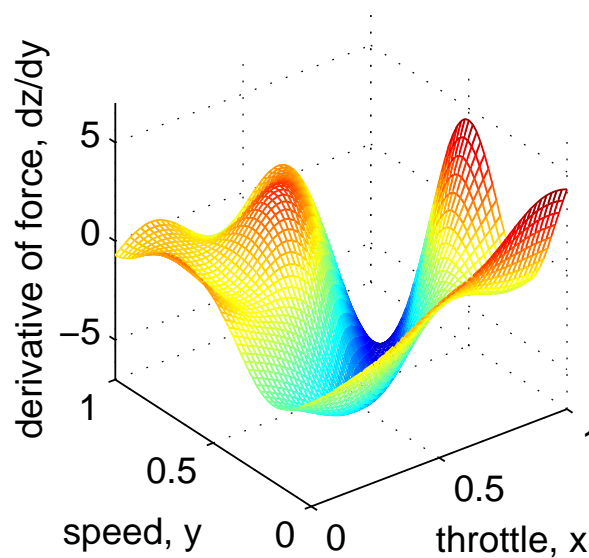
one std error in partial deriv, dz/dx



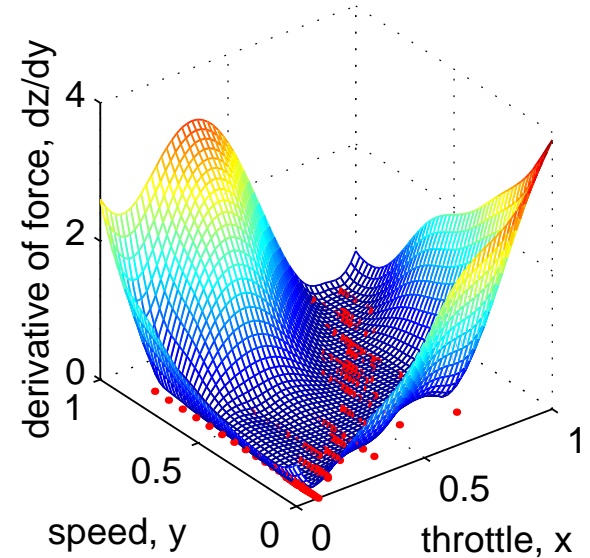
one std error uncertainty



predicted mean partial deriv, dz/dy

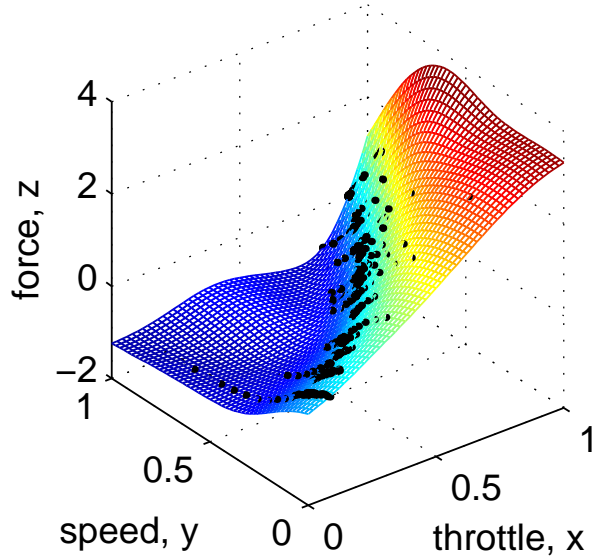


one std error in partial deriv, dz/dy

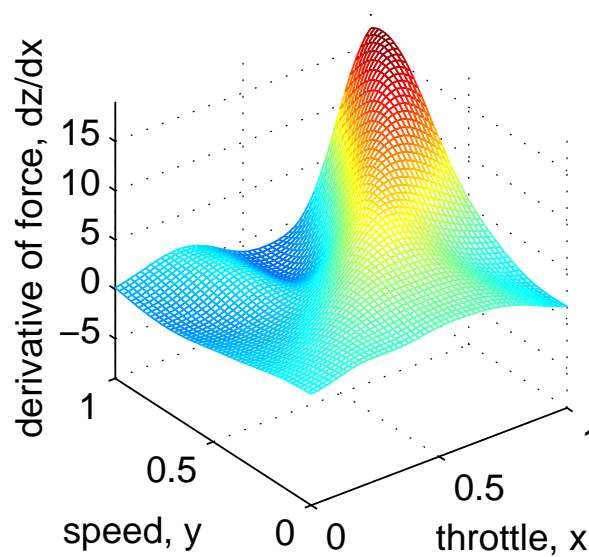


Engine Example: Neural Network Covariance Function

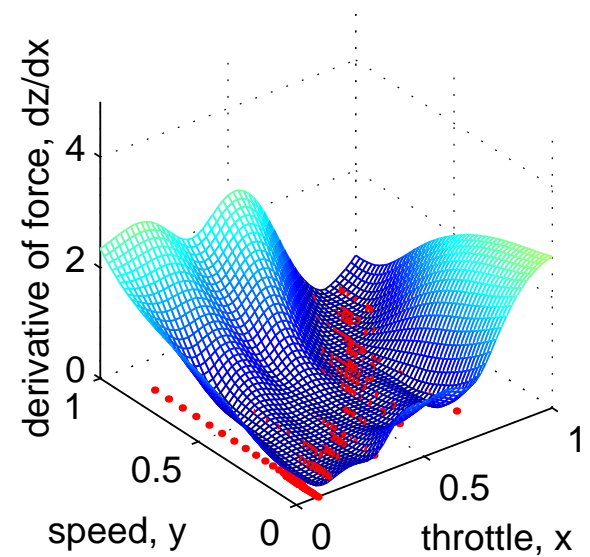
mean predicted force



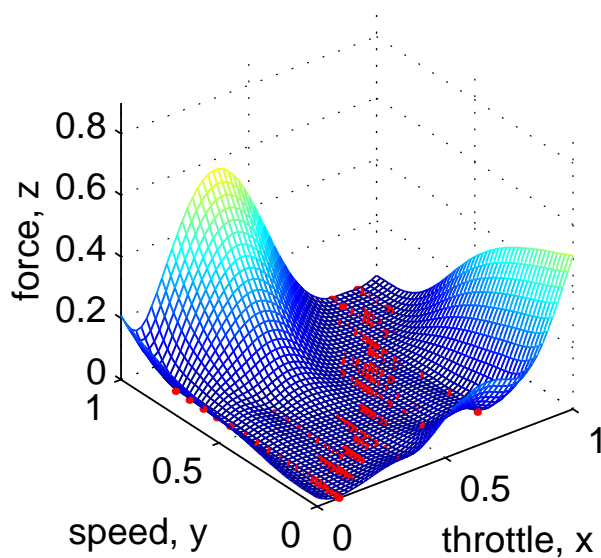
predicted mean partial deriv, dz/dx



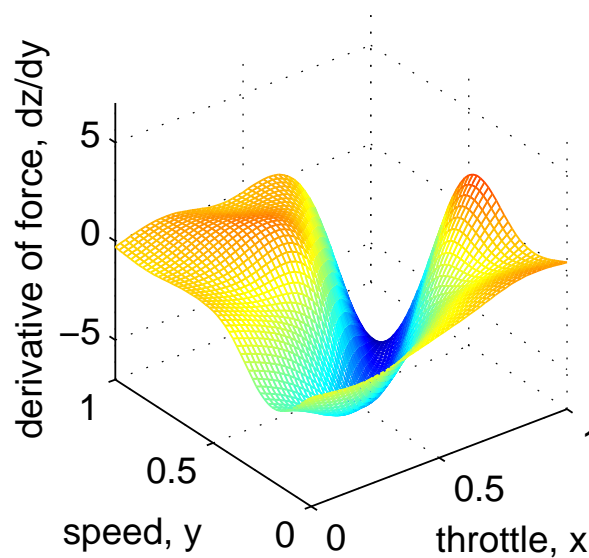
one std error in partial deriv, dz/dx



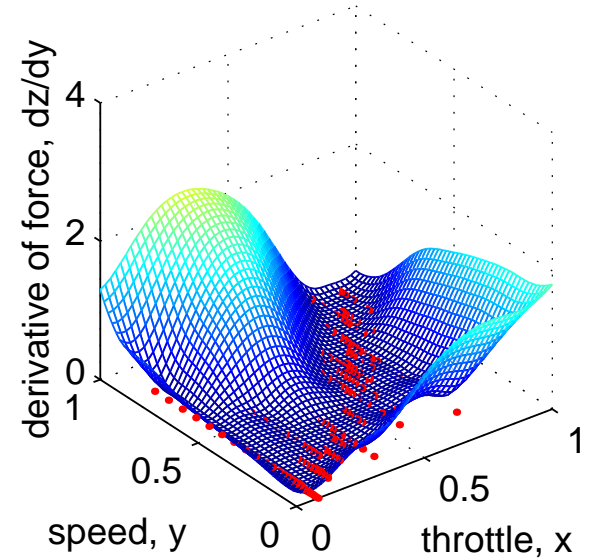
one std error uncertainty



predicted mean partial deriv, dz/dy



one std error in partial deriv, dz/dy



Conclusions and Future Directions

We've presented a clean, Bayesian, hierarchical, fully probabilistic, unifying framework for regression with GPs.

The GP formalism allows us to specify complex priors over functions in a convenient way, and allows evaluation of the evidence.

GPs avoid many of the problems hampering parametric and semiparametric models, such as Neural Networks; GPs are much easier to use and often have better performance.

One big challenge is how to reduce the computational load: project to a smaller dimensional (random) space (the Nyström method); select training examples with most information; (stochastically) divide the data into subsets (infinite Mixture of Gaussian Process Experts); Extensions to non-Gaussian and dependent noise models are quite feasible.