10-708: Probabilistic Graphical Models 10-708, Spring 2016

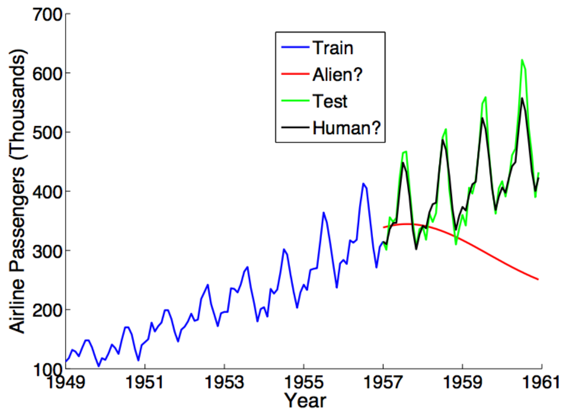
20: Gaussian Processes

Lecturer: Andrew Gordon Wilson *Scribes: Sai Ganesh Bandiatmakuri*

1 Discussion about ML

Here we discuss an introduction to Machine learning and practicality of a few historical techniques. In one example, we observe that sampling using a Hamiltonian Monte Carlo algorithm, although can be effective in practice, it is still associated with some hard problems, like handtuning some of the hyper-parameters (e.g the step size) and many recent efforts in Machine learning attempt to automate such manual steps. (For example, the step sampling algorithm). Machine learning is also primarily adaptive function learning - learning to adapt based on data, rather than have a fixed set of rules and in order to successfully do this, a model should be able to automatically discover patterns and extrapolate to new situations. In this context, we present an example of a trend graph of the number of airline passengers by year and qualitatively show the effectiveness of a spectral mixture/kernel model.

Figure 1: Extrapolating airline passenger count



In the above picture, the red curve is generated by an RBF kernel, which makes strong smoothness approximations. Te black curve is generated by a different kind of kernel based on spectral mixtures. It is very close to the ground truth in green.

1.1 Support and Inductive biases

Whenever we want to discuss the effectiveness of a model, we need to consider its support and its inductive Bias. The support is what solutions we think are a-priori possible. It is the space of solutions that can possibly be represented by the model. For example, if we restrict the model to a polynomial of degree 3, the support involves only polynomials of degree ≤ 3 and it cannot model cases where the true distribution is more complex. The inductive bias of a model is what solutions (represented in the support) are a-priori likely. It is a measure of how we distribute the support. Another example discussed was the inductive bias of conv-nets. These layers are less flexible than fully connected layers, but their inductive bias is often better suited to computer vision tasks.

In principle, we want a model to be able to represent all likely functions, although how it distributes the support may be biased. Today's deep learning approaches and the Gaussian process are considered universal approximators. We can get a better idea of complexity of models by plotting the likelihoods of certain datasets given a model.

Figure 2: Simple models can only generate certain datasets

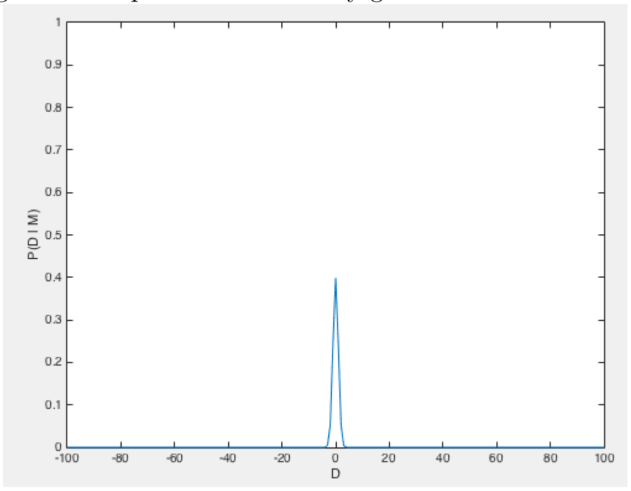


Figure 3: More flexible models can generate more datasets

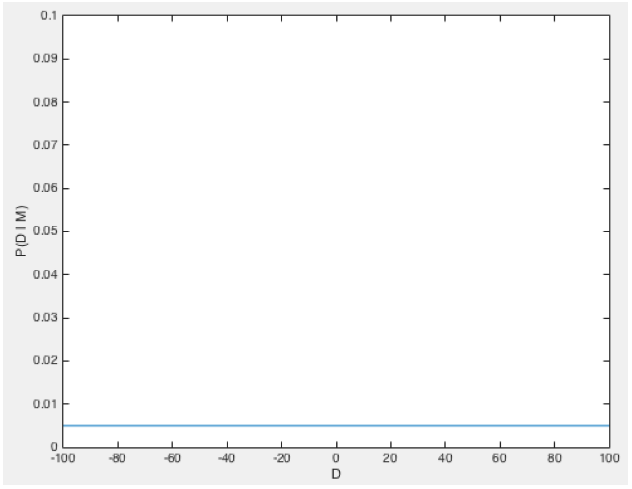
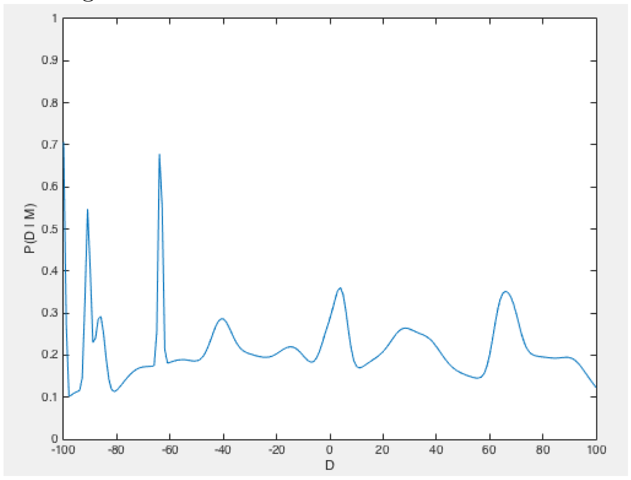


Figure 4: Flexible model with inductive bias



1.2 Basic regression problem

The basic regression problem is to predict a function value at a data point x_* which may be far away from given training data points. The typical approach is to assume some parametric form of the function, by parameters w, formulate an error function $E(w)$ and minimize wrt w. There are different choices for the function type $f(x, w)$ - $w^T x$ (linear in w and x), $w^T \phi(x)$ (linear basis function model), etc. An example of $E(w)$ is the L_2 error.

$$E(w) = \sum_{i=1}^{N} |f(x_i, w) - y(x_i)|^2$$

1.2.1 A probabilistic approach by modeling noise

In the above expression, although we may arrive at a set of parameters $\mathbf{w}$, it may not provide an intuitive understanding of the error measure. A probabilistic approach is to also model the noise.

$$y(x) = f(x, w) + \epsilon(x)$$

If we model the noise as i.i.d additive gaussian, with mean 0 and variance σ^2,

$$p(y(x)|x, w, \sigma^2) = \mathcal{N}(y(x); f(x, w), \sigma^2)$$

$$p(Y|X, w, \sigma^2) = \prod_{i=1}^{N} \mathcal{N}(y(x_i); f(x_i, w), \sigma^2)$$

$$logp(Y|X, w, \sigma^2) \propto -\frac{1}{2\sigma^2} \sum_{i=1}^{N} |f(x_i, w) - y(x_i)|^2$$

The above approach makes the same predictions as an L_2 error function, but it can now be optimized for both w and σ^2 and thus provides a more intuitive understanding of the error function and the noise. For example, if there are too many outliers, we can change the representation of $\epsilon(x)$ to a laplacian distribution and the squared error to sum of absolute errors. This method thus provides an intuitive framework to represent uncertainty and model development.

However, both the above approaches are still prone to overfitting (when using highly complex models). One way to address this is to introduce regularization. This process has its own issues - it is a difficult design decision to pick the regularization penalty term and the weight and performing cross validation.

1.3 Bayesian model averaging

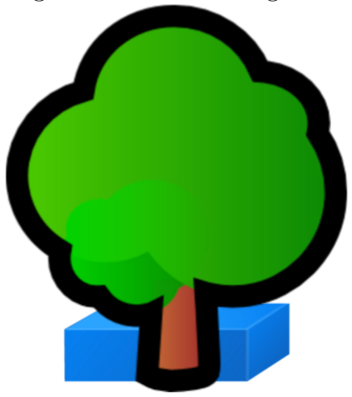
An alternate way to avoid overfitting is by Bayesian model averaging. Here, we're more interested in the distribution of functions than the parameters w themselves. We average over infinitely many models $p(y|x_*, w)$ by the posterior probabilities of those models. This gives us automatically calibrated complexity and does not have the problem of overfitting.

$$p(y|x_*, \mathbf{y}, w) = \int p(y|x_*, w)p(w|\mathbf{y}, X)dw$$

1.4 Examples of Occam's razor

One example discussed in the class was the example of a tree blocking a block as shown below. The simplest explanation is there exists a regular sized block with no deformities behind the tree. There can be more elaborate and far fetched hypotheses for the occluded portion of the box (e.g it can be covered, distorted, etc), but Occam's razor favors the first explanation.

Figure 5: Tree covering a box



Another example discussed was predicting the next two numbers from a given sequence of integers {-1, 3, 7, 11, ? ?}. The first hypothesis H_1, the simplest is assuming its an arithmetic progression and predicting 15 and 19. The second more complicated hypothesis tries to fit a polynomial $H_2 : y(x) = \frac{x^3}{11} + \frac{9x^2}{11} + \frac{23}{11}$.

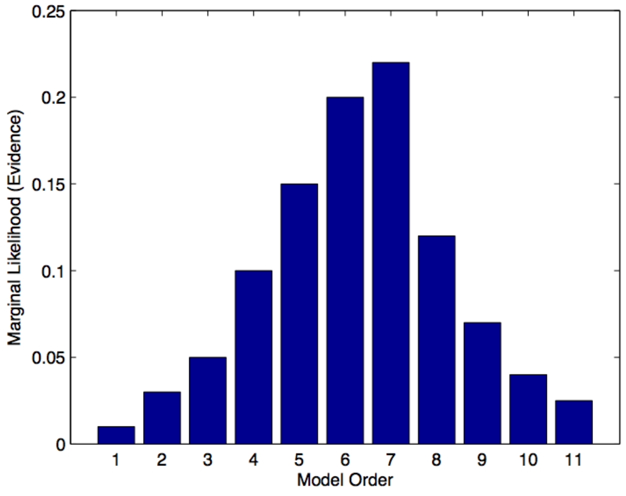
We can quantitatively see that Occam's razor favors the first hypothesis over the other. We first write the posterior in terms of the likelihood and the priors using Bayes rule. $\frac{p(H_1|D)}{p(H_2|D)} = \frac{p(D|H_1)}{p(D|H_2)}\frac{p(H_1)}{p(H_2)}$ For x $\epsilon[-50, 50]$, we can arrive at $\frac{p(H_1|D)}{p(H_2|D)} \approx \frac{10^{-4}}{2.3*10^{-12}}$ (i.e $\sim 40M$ higher even if we assume $P(H_1) = P(H_2)$).

1.5 Occam's asymptote

Since we have techniques like Bayesian model averaging, regularization, etc, it then seems intuitive to pick a "sufficiently complex" model to represent the data and not worry about overfitting. However, we'll see that choice of prior is important in making such a decision.

If we try to fit a function using different models, where each model k is a polynomial of order k, we note that for the naive case, the graph of the marginal likelihood forms an occam's hill - i.e too simple models don't explain the data neither do too complex models.

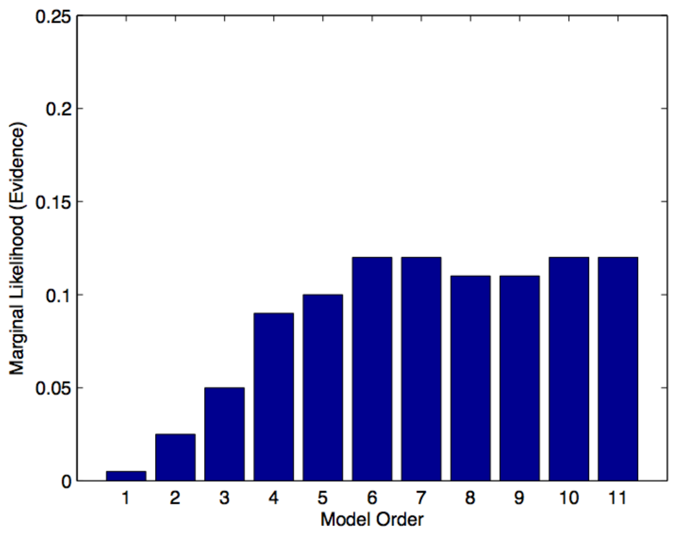
Figure 6: Occam's hill



The above paradox arises when we use an isotropic prior (i.e equal weights on all parameters). If the coefficients are **a**, we have $p(a) = \mathcal{N}(0, \sigma^2 I)$.

However, most of the information is usually captured in the lower order polynomials with some noise terms in the higher order coefficients, so using an anisotropic prior, reducing the weights for higher order terms seems more intuitive. $p(a_i) = \mathcal{N}(0, \gamma^{-i})$ where γ is learned with the data. In this case, we arrive at the Occam's asymptote.

Figure 7: Occam's asymptote



2 Gaussian process

2.1 Linear model as a gaussian process

Consider a simple linear model $f(x) = a_0 + a_1 x$, where $a_0, a_1 \epsilon \mathcal{N}(0, 1)$. We have $\mathbb{E}[f(x)] = 0$ and covariance between two points $f(x_b)$ and $f(x_c)$ given by

$$\mathbb{E}[f(x_b)f(x_c)] = \mathbb{E}[a_0^2 + a_0 a_1(x_b + x_c) + a_1^2(x_b x_c)]$$
$$\mathbb{E}[f(x_b)f(x_c)] = \mathbb{E}[a_0^2] + 0 + \mathbb{E}[a_1^2]x_b x_c$$
$$\mathbb{E}[f(x_b)f(x_c)] = 1 + 0 + x_b x_c$$

Therefore any collection of values has a joint Gaussian distribution $[f(x_1), f(x_2), ...f(x_N)] \sim \mathcal{N}(0, \mathcal{K})$, where $\mathcal{K}_{ij} = 1 + x_i x_j$. By definition, f(x) is a gaussian process.

2.2 Definition

A Gaussian process (GP) is a collection of random variables, any finite number of which have a joint Gaussian distribution. We write $f(x) \sim GP(m, k)$ to mean $[f(x_1), ..., f(x_N)] \sim \mathcal{N}(\mu, \mathcal{K})$, where $\mu_i = m(x_i)$ and $\mathcal{K}_{ij} = k(x_i, x_j)$ for any collection of input values $x_1, ..., x_N$. In other words, f is a GP with mean function $m(x)$ and covariance kernel $k(x_i, x_j)$.

2.3 Linear basis function models

By modeling the regression function as a linear regression in the basis function space, we can show that it is still a Gaussian process.

$$\begin{aligned}
f(x,w) &= w^T\phi(x)\\
p(w) &= \mathcal{N}(0,\Sigma_w)\\
\mathbb{E}[f(x,w)] &= m(x) = \mathbb{E}[w]^T\phi(x) = 0\\
cov(f(x_i),f(x_j)) &= \mathbb{E}[f(x_i)f(x_j)] - \mathbb{E}[f(x_i)]\mathbb{E}[f(x_j)]\\
cov(f(x_i),f(x_j)) &= \phi(x_i)^T\mathbb{E}[ww^T]\phi(x_i) - 0 = \phi(x_i)^T\Sigma_w\phi(x_j)
\end{aligned}$$

Thus $f(x) \sim GP(m,\mathcal{K})$, where $m(x) = 0$ and $k(x_i,x_j) = \phi(x_i)^T\Sigma_w\phi(x_j)$. Thus this entire model is encapsulated as a distribution over functions with kernel $k(x,x')$

2.3.1 Inference for new test samples

The results from this section are derived in detail in Ch2 from [2]. For inference in this model, we have new test samples X_* and need to predict their outputs f_*. We assume the training data is given where y is the output of f with additive Gaussian noise ϵ, i.e $y = f + \epsilon$, where $\epsilon \sim \mathcal{N}(0,\sigma^2 I)$. As a result, $y \sim \mathcal{N}(0, K(X,X) + \sigma^2 I)$.

We write the joint distribution of y and f_* and by the definition of gaussian process, this is a Gaussian distribution:

$$\begin{bmatrix} y \\ f_* \end{bmatrix} \sim \mathcal{N}(0, \begin{bmatrix} K(X,X)+\sigma^2 I & K(X,X_*) \\ K(X_*,X) & K(X_*,X_*) \end{bmatrix})$$

therefore the conditional distribution $p(f_*|y)$ can be written as $(f_*|X,X_*,y) \sim \mathcal{N}(m,V)$, where $m = K(X_*,X)(K(X,X)+\sigma^2)^{-1}f$, and $V = K(X_*,X_*) - K(X_*,X)(K(X,X)+\sigma^2)^{-1}K(X,X_*)$

2.4 The Gaussian process graphical model

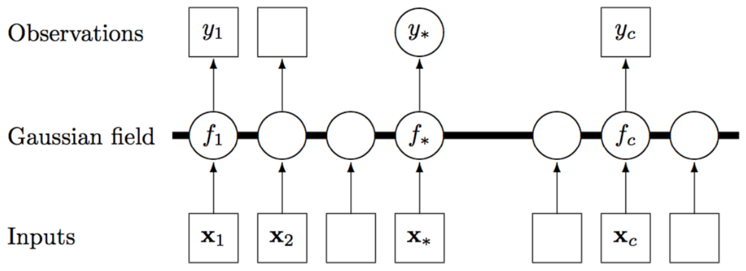


Figure 8: The gaussian process graphical model, Squares are observed variables, circles are latent. and the think horizontal bar is a set of fully connected nodes. Note that each y_i is conditionally independent given f_i. Because of the marginalization property of GP, addition of further inputs x and unobserved targets y does not change the distribution of any other variables. Originally from Rasmussen and Williams (2006) [2]

2.5 Example: RBF kernel

This is one of the most popular kernels, with also nice theoretical properties.

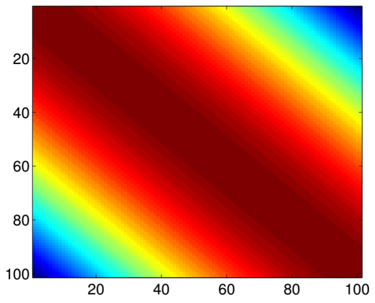
It can be viewed as an infinite basis model, where each feature can be written as $\phi_i(x) = \sigma^2 exp(\frac{-(x-c_i)^2}{2l^2})$. If we use J features, we can write the kernel entry as:

$$k(x_p, x_q) = \frac{\sigma^2}{J} \sum_{i=1}^{J} \phi_i(x_p)\phi_i(x_q)$$

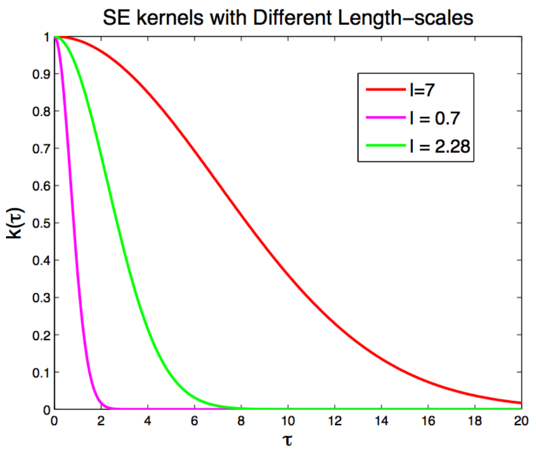
By taking limits $J \to \infty$, we can write the kernel function as $K_{RBF}(x_i, x_j) = a^2 exp(\frac{||x_i - x_j||^2}{2l^2})$, here the hyperparameters a and l control the amplitude and wiggliness of the function respectively.

Here, the intuition in the RBF kernel is nearby samples are more correlated than samples that are farther away (controlled by l).

Figure 9: Visualization of the covariance matrix



l controls how the correlations are decay wrt L_2 distance. A larger value of l indicates slow decay and thus farther points will have non zero covariance or correlations. This can be represented in the figure below, the x axis is the distance between samples τ.



For larger l, we can see long range correlations. If l is very large, we see high correlations between samples. If l is very small, we see a lot of wiggliness where the distribution tries to fit individual data points. If l is

0, this is equivalent to white noise.

2.6 Learning and model selection

We can integrate away the entire gaussian process $f(x)$ to obtain the marginal likelihood as a function of the model hyperparameters alone.

$$p(Y|\theta, X) = \int p(y|f, \theta X)p(f|\theta, X)df$$
$$log(P(Y|\theta, X)) = -\frac{1}{2}y^T(\mathcal{K} + \sigma^2 I)^{-1}y - \frac{1}{2}log|\mathcal{K} + \sigma^2 I| - \frac{N}{2}log(2\pi)$$

The first term above evaluates the model fit and the second term penalizes the model complexity.

2.7 Learning by full Bayesian treatment

The following derivation is from Section 5.2 in [2]. For each function f, we may have some low level parameters w (which may be weights in a neural network for example) and we have hyper parameters θ on top of them (e.g parameters like weight decay for neural networks, a and l for Gaussian processes, etc). On top of them, we may have a discrete set of possible model structures $\mathcal{H}_i$.

At the bottom level, the posterior over the parameters is given using the Bayes rule

$$p(w, |\mathbf{y}, X, \theta, \mathcal{H}_i) = \frac{p(\mathbf{y}|X, w, \mathcal{H}_i)p(w|\theta, \mathcal{H}_i)}{p(\mathbf{y}|X, \theta, \mathcal{H}_i)}$$

Where $p(\mathbf{y}|X, w, \mathcal{H}_i)$ is considered the likelihood and $p(w|\theta, \mathcal{H}_i)$ is considered the prior. The normalizing constant in the denominator is obtained by marginalizing out w and called the marginal likelihood.

$$p(\mathbf{y}|X, \theta, \mathcal{H}_i) = \int p(\mathbf{y}|X, w, \mathcal{H}_i)p(w|\theta, \mathcal{H}_i)dw$$

At the next level, we express the posterior distribution of the hyperparameters, where the marginal likelihood defined above plays the role of the likelihood.

$$p(\theta|\mathbf{y}, X, \mathcal{H}_i) = \frac{p(\mathbf{y}|X, \theta, \mathcal{H}_i)p(\theta|\mathcal{H}_i)}{p(\mathbf{y}|X, \mathcal{H}_i)}$$

Here, $p(\theta|\mathcal{H}_i)$ is the *hyperprior* (prior over the hyper-parameters) and the normalizing constant is

$$p(\mathbf{y}|X, \mathcal{H}_i) = \int p(\mathbf{y}|X, \theta, \mathcal{H}_i)p(\theta|\mathcal{H}_i)d\theta \tag{1}$$

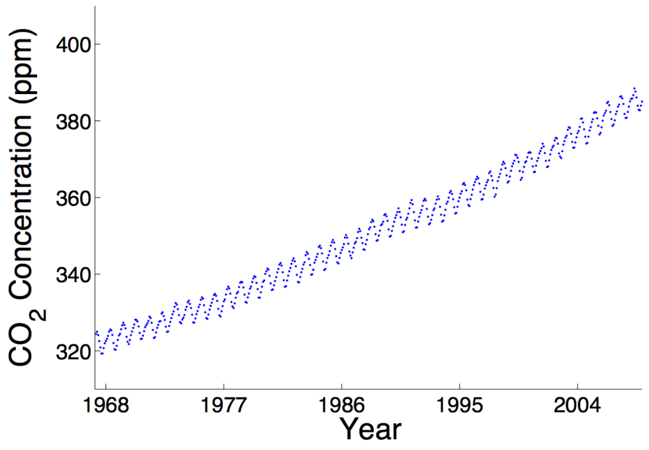
At the top level, we compute the posterior of the model.

$$p(\mathcal{H}_i|\mathbf{y}, X) = \frac{p(\mathbf{y}|X, \mathcal{H}_i)p(\mathcal{H}_i)}{p(\mathbf{y}|X)}$$
$$p(\mathbf{y}|X) = \sum_i p(\mathbf{y}|X, \mathcal{H}_i)p(\mathcal{H}_i)$$

However computing the integral in Equation (1) may be intractable and one may have to refer to analytic approximations (e.g variational inference) or MCMC methods.

2.8 Example: Combining kernels to predict CO_2 trends

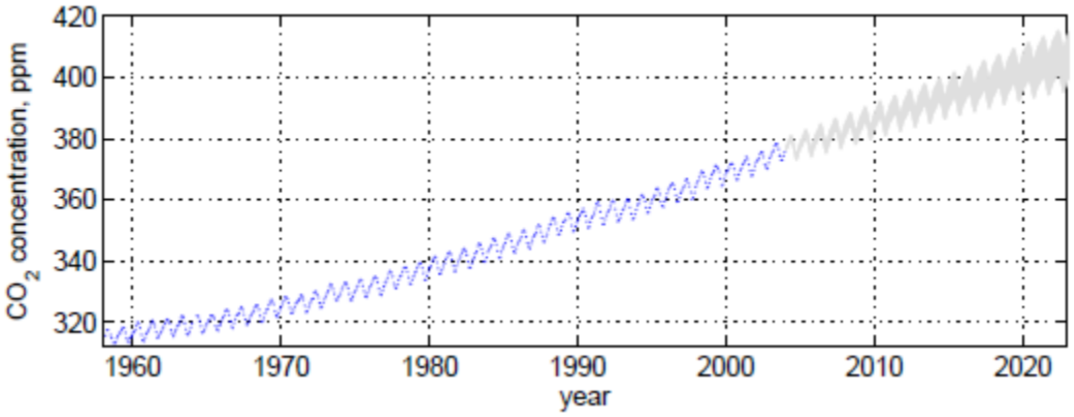
The data consists of monthly average atmospheric CO_2 concentrations derived from air samples collected at the Mauna Loa Observatory, Hawaii, between 1958 and 2003. The data is shown below. Our goal is to model the CO2 concentration as a function of time x.



The following were kernels or covariance functions handcrafted by inspecting the training data.

- Long rising trends: $k_1(x_p, x_q) = \theta_1^2 exp(-\frac{(x_p - x_q)^2}{2\theta_2^2})$
- Quasi periodic seasonal changes: $k_2(x_p, x_q) = k_{rbf}(x_p, x_q)k_{per}(x_p, x_q) = \theta_3^2 exp(-\frac{(x_p - x_q)}{2\theta_2^2} - 2\frac{sin^2(\pi(x_p - x_q))}{\theta_5^2})$
- Multi-scale medium term irregularities: $k_3(x_p, x_q) = \theta_6^2(1 + \frac{(x_p - x_q)^2}{2\theta_8\theta_7^2})^{-\theta_8}$
- Correlated and i.i.d noise: $k_4(x_p, x_q) = \theta_9^2 exp(-\frac{(x_p - x_q)^2}{2\theta_{10}^2}) + \theta_{11}^2\delta_{pq}$
- $k_{total}(x_p, x_q) = k_1(x_p, x_q) + k_2(x_p, x_q) + k_3(x_p, x_q) + k_4(x_p, x_q)$

Results from the predictions are shown below. We display the given training data, together with 95% predictive confidence region for a Gaussian process regression model, 20 years into the future. Rising trend and seasonal variations are clearly visible. Note also that the confidence interval gets wider the further the predictions are extrapolated. Original figure from [2].



2.9 Non gaussian likelihoods

If our model is non-gaussian, we can no longer integrate away the gaussian process to infer f_*. However, we can use a simple Monte Carlo sum:

$$\begin{aligned} p(f_*|\mathbf{y}, X, X_*) &= \int p(f_*|f, x_*)p(f|y)df \\ &\approx \frac{1}{J}\sum_{j=1}^{J} p(f_*|f^{(j)}, x_*), f^{(j)} \sim p(f|\mathbf{y}) \end{aligned}$$

We can sample from $p(f|\mathbf{y})$ using elliptical slice sampling [1].

For the hyperparameters, one possible approach is to perform Gibbs sampling:

$$\begin{aligned} p(\mathbf{f}|\mathbf{y}, \theta) &\propto p(\mathbf{y}|\mathbf{f})p(\mathbf{f}|\theta) \\ p(\theta|\mathbf{f}, \mathbf{y}) &\propto p(\mathbf{f}|\theta)p(\theta) \end{aligned}$$

Typically however since f and θ are highly correlated, the sampler will not mix very well and the approach usually does not work. According to the lecturer, a better alternative for hyperparameter inference is to compute an approximation of the distribution (using variational methods, for example) and use that instead as the source distribution to obtain samples.

3 Further extensions

Two major research areas devoted to extending Gaussian processes are related to automatic kernel selection and improving the scalability.

In the predictions related to CO_2 data explored above, there were a lot of manual decisions done by inspecting the data with handcrafted kernels. One possible direction is to automate this entire process and learn the kernel combinations from a data driven approach. It is also an active area in high dimensional problems. The RBF kernel uses the L_2 distance between data points, which need not be the best representative distance metric in higher dimensions.

Another focus area is scalability. When N (the number of variables) is approximately 1000 or small, Gaussian processes remain the gold standard for regression. Many results involve solving large systems of linear equations and performing Cholesky decompositions. The complexity for this is cubic in the size of the variables and thus not scalable when there are more than a few thousand variables. However, many of these matrices have predictable structure which can be exploited to make decompositions and inference more efficient.

References

[1] Iain Murray, Ryan Prescott Adams, and David JC MacKay. Elliptical slice sampling. *arXiv preprint arXiv:1001.0175*, 2009.

[2] Carl Edward Rasmussen. Gaussian processes for machine learning. 2006.