Contents

Variational Gaussian Processes (Dai)

Very good slides!

Q = dimensionality of the input. N is the number of data points.

Computational complexity

When we talk about the computational complexity of GPs we are talking about the $O(N^3 + N^2 + N)$ complexity of calculating the log pdf $p(y|x)$.

Calculating the kernel matrix can take up a surprising portion of the budget. For example, for the RBF kernel K N^2 exponentials have to be calculated.

Speeding up the Cholesky decomposition with parallel computing/GPUs is an emerging field. But there is the cost evaluating K and the $O(N^2)$ memory footprint, which you'll typically hit before wall time problems.

Currently on typical laptops you can get up to 10,000 datapoints.

Precomputing

Precomputing happens wherever it can: that's how very fast (linear time) prediction happens at production time, with a given learned model.

But for learning (parameter estimations) this is not easily possible: the kernel is a function of the current point in parameter space.

Big data (?)

In typical industry applications of ML with a large amount of data subsampling would be the first approach.

But lots of data does not imply complex relations. Tend to get a lot of data on common cases and very few data or rare cases (heavy tails). Sampling representative data from such an imbalanced distribution can be hard.

Covariance matrix of redundant data

K becomes low rank because of spurious information (K is a Gram matrix and the dimension of its column space becomes degenerate). The degree of low-rankness can be quantified by the number of very small eigenvalues. (Right plot on slide 14).

Low-rank approximation: Nyström approximation

Since K is low-rank, approximate it by constructing another low-rank matrix K' using $M \ll N$ sampled inputs. In this way the eigenvalue spectrum of K is interpolated (?) by the spectrum K'.

Note that K' is still $N \times N$, so where is the computational benefit? It follows from the Woodbury formula, which rewrites the inversion by exploiting the low-rank decomposition of K', achieving $O(NM^2)$.

Note: You can replace the Tr(K - K') by any metric that goes to zero when $K \to K'$.

Note: doing the low-rank approximation using PCA would also work, but PCA is also $O(N^3)$.

Problem: subsampling data is problematic because the data is often heavy-tailed.

GPs with pseudo data (FITC)

Formulate joint GP with pseudo data (u, Z).

FITC approximation to reduce computational cost: assume Λ is diagonal. This also allows the marginalization over u in $p(y|X) = \int dup(y, u|X)$. Then apply Woodbury formula to take advantage of the "FITC decomposition" of the covariance matrix, reducing cost to $O(NM^2)$.

Model approximation vs. approximate inference

FITC changes the model definition. This is different from variational approaches, where an approximation to the model is optimized (VI).

Variational sparse GPs

Sparse refers to inducing points, i.e. using a sparse amount of data. It follows initially exactly the same concept of augmenting the GP with pseudo data.

Special variational posterior:

$$q(f, u|X, Z, \mu, \Sigma) = p(f|u, X, Z)q(u|\mu, \Sigma)$$

The optimal values of μ and Σ can be derived analytically, probably through identification using the KL-divergence.

Increasing the number of inducing points can only improve the fit, since these are variational parameters. They can be chosen to accomodate the computational budget.

Stochastic variational sparse GPs

Use minibatches and stochastic gradient descent (SGDs).

Pros and cons on slide 44.

Unsupervised and composite GPs (Ek)

Best way to think about GPs!

The Gaussian identities

Taking $p(x, y) = N(\mu, \Sigma)$, derive $p(x)$ and $p(x|y)$. Very instructive to do at least once.

The exchangeability and marginal conditions are trivially true for Gaussians, but are very insightful to understand intuitively.

Gaussians are projections of GPs.

Unsupervised learning

> Unsupervised learning doesn't exist. But we need even stronger priors to learn anything useful

Assigning x values to y values using an inductive bias.

Our inductive bias is given by GPs: assign x values and fit GPs through the (x, y) pairs. See which set of x's are liked best by the GP.

But we need to place a preference both about the distribution of the x values alone ($p(x)$) *and* the (x, y) pairs ($p(f)$).

- $p(f)$: Defines inductive bias in the function
- $p(x)$: Defines inductive bias in the latent space

Then $p(y) = \int p(y|f)p(f|x)p(x)df dx$.

This integral, however, is about as intractable as it gets. Thus resort to VI.

> Nature laughs at the difficulties of integration (Laplace)

Variational inference

Unsupervised learning is about finding $p(y)$, which usually seens as a normalization constant in other problems!

An unusual way of writing Bayes' theorem:

$$p(y) = p(y|x)p(x)/p(x|y)$$

Thus we try to find the posterior $p(x|y)$ *in order to get the marginal probability of the data* $p(y)$. We get to the posterior with VI. The variational parameters *are hallucinated data points* (u, z) which try to encapsulate the given data y.

In other words, we're trying to induce the marginal distributions at the y, $p(y)$ as defined above, by moving around the pseudo-observations u, z. Just like a GP predicts marginal distributions at unobserved points given observed data.

ELBO derivation

Any concave function, in conjunction with Jensen's inequality, will yield a bound on $p(y)$ and a measure to optimize. That's why the derivation of the ELBO doesn't start with minimizing the KL. He argues that the question shouldn't be which divergence to use but which concave function is the most useful, since and concave function will yield a valid bound. And choosing the log function as this concave function is just useful because we use the exponential family a lot.

(But this ignores the theoretical underpinnings of KL divergence!)

The Trick

Settng a variational distribution to match a term in the product decomposition model posterior such that it cancels in the ELBO is an amazing trick.

Take-away

We can specify any prior over the latent space $p(x)$! For example a GP! This leads to composite GPs.

We can do mini-batch stuff.

Composite GPs ($\equiv$ deep GPs)

Appropriate when my knowledge is composite. Seems to be inherently difficult.

Collape an input space into small regions: classification. But tiny variations can make you jump between these small regions.

Round table with Ek

> PCA is factor analysis with a Gaussian prior (invariant to rotation). It can also be formulated as a GP with linear kernel. It's related to which variable in $f = Ux$ you integrate out or optimize.

How can I get away from Gaussian marginals?

This is attacking the fundamentals of GPs. You want the marginal at a data point to be asymmetric, positive, in an interval, etc.

1. Can I find a transformation? Can I rewrite the problem in a different parametrization such that it is a GP? Examples: log, monotonicity, differential equations (DE). For example you can model a drift term in a DE by a GP.
2. Is a GP really a suitable model?

Is KL divergence the right tool for VI?

KL bounds are incredibly loose in his experience. "KL divergence is bad."

But the biggest challenge in VI is the form of q. "But the forms of q are generally worse."

"These damn plots"

> Imagine you plot a conditional GP at 300 points. If you plot the uncertainty band rather than samples this GP then you throw away 300*299 values of the learned covariance matrix. The problem is that people tend to make the assumption that the GP samples can live anywhere between the uncertainty band, while this perspective misses out on the actual structure of the GP samples: e.g. how smooth they are. For this reason it is better to just show some samples. A good example is the Matern kernel: the uncertainty bands look quite smooth but the functions are not.

David MacKay plot (slide 96)

(Not entirely sure whether this explanation is 100% correct.)

This plot shows the data space D ordered from low to high complexity. The three curves represent $p(D|M_i)$, which are *normalized* pdfs over D. Since the D-axis is ordered in terms of complexity, the model complexities are ordered from low to high: M_1 is less complex than M_2, and so on.

When performing model selection for a given dataset D_0, we want to optimize $p(M_i|D_0)$ (we're not marginalizing over all possible data – this would be optimizing $p(M_i)$). Assuming an uniform prior $p(M_i)$, this is equivalent to optimizing the density $p(D_0|M_i)$, which in this case yields us M_2 as the best model. These densities are acquired by marginalizing over the model parameters.

The point is that a complex model will have to spread its mass over a large interval on the ordered D-axis such that it is unlikely to have a large density at the observed data D_0. A model that is too simply will not have large density at D_0. Thus, to get a high density at D_0, the model's complexity will have to be "just right"; the balance between simplicity (small support) and complexity (large support) is induced by the normalization requirement.

More on this MacKay (2005) and [Jaynes (2003); Ch. 20].

PCA

PCA is often presented as a mapping from observed data to latent space.

But that is the algorithm, not the model

The reason why the model leads to that algorithm, is because it is a linear mapping, and thus invertible.

Originally, in the old paper, PCA was conceived as a mapping from latent space (which has a Gaussian prior), to observed data.

Automatic Relevance Determination (ARD)

ARD is a misnomer

ARD is saying how linear a dimension is, like Pearson's correlation measures only linear correlation. But if it gives a zero, it is indeed irrelevant no mather if it was linear.

Good reference: https://users.aalto.fi/~ave/publications/Vehtari_B29.pdf

Multi-task GP modelling for point process data (Aglietti)

A point process is a model for (the number of) discrete events happening along time and in a space $\{x\}$. It is a generalization of the Poisson distribution whose rate (intensity function) $\lambda(x)$ depends on x. It also possible to include time in the intensity: $\lambda(x,t)$; but this is just another space dimension. In this talk we have

$$\lambda(x) = \exp(f(x))$$

where $f(x) \sim GP$. (Other transformation such as sigmoid or squaring are possible too.)

Multi-task model: a GP with different outputs (multi-output GPs). Introducing correlations between the different outputs can be done with various coregionalization models.

Jaynes, E. T. 2003. *Probability Theory: The Logic of Science*. Cambridge, UK; New York, NY: Cambridge University Press.

MacKay, David J C. 2005. *Information Theory, Inference, and Learning Algorithms*. https://doi.org/10.1198/jasa.2005.s54.