Heteroscedastic Gaussian Processes with Applications in Video Reconstruction

Landon Carter
MIT Department of Computer Science
6.882 Bayesian Modeling and Inference Final Project
lcarter@mit.edu

Abstract

We replicate the work of Kersting et al. in implementing a modified Gaussian Process for regression using input-dependent noise rates. We also explore the motivation and framework behind using Gaussian Processes for temporally-compressed video reconstruction, and show some preliminary results in this vein.

1 Introduction

Gaussian Processes (GP's), are useful in Bayesian-based regression modeling, allowing for a separation of model function and goodness of fit. In normal GPs, you are given an input $\mathbf{x_i} \in \mathbf{R}^D$, and output associated with each point, $y_i \in \mathbf{R}$. We define mean function $m(x) = \mathcal{E}[f(x)]$ and covariance function $k(x, x') = \mathcal{E}[(f(x) - m(x))(f(x') - m(x'))]$. We then write the GP as

$$f(x) \sim \mathcal{GP}(m(x), k(x, x')).$$

In particular, we choose the covariance as the squared exponential,

$$k(x_p, x_q) = \exp\left(-\frac{1}{2}|x_p - x_q|^2\right).$$

To do regression, we apply the standard Bayesian approach, yielding a joint distribution on training outputs, f, and test outputs (used to reconstruct the hidden function), f_*:

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{f}_* \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} K(X, X) & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right)$$

One of the nice features about GP's is that we can encode our entire prior belief in the covariance function. In the case of the squared exponential, we simply encode a belief that the function is smooth over a given length scale (specified by a hyperparameter, θ, which modifies the standard form presented above).

2 Model

2.1 Mathematical Setup

The modification to Gaussian Processes that we make is to allow the noise model to be input-dependent. In a noisy GP, we recover the functional dependency $t_i = f(\mathbf{x_i}) + \epsilon$, where ϵ is a Gaussian noise term. In the modification originally proposed by Goldberg et al., ϵ is replaced by $\epsilon_i \sim \mathcal{N}(0, \sigma_i)$, and $\sigma_i = r(\mathbf{x_i})$, a function of $\mathbf{x_i}$. Based on this, we can determine that the predictive distribution $P(\mathbf{t}^* | x_1^*, \ldots, x_q^*)$ is a multivariate Gaussian:

$$P(\mathbf{t}^*|x_1^*,\dots,x_q^*) = \mathcal{N}(\mu^*,\Sigma^*)$$
$$\mu^* = E[\mathbf{t}^*] = K^*(K+R)^{-1}\mathbf{t}$$
$$\Sigma^* = \text{var}[\mathbf{t}^*] = K^{**} + R^* - K^*(K+R)^{-1}K^{*\intercal}$$

Here, $K \in \mathbf{R}^{n\times n}$, $K_{ij} = k(x_i, x_j)$, $K^* \in \mathbf{R}^{q\times n}$, $K^*_{ij} = k(x^*_i, x_j)$. $K^{**} \in \mathbf{R}^{q\times q}$, $K^{**}_{ij} = k(x^*_i, x^*_j)$, $\mathbf{t} = (t_1, t_2, \dots, t_n)^\intercal$, $R = \text{diag}(\mathbf{r}$ with $\mathbf{r} = (r(x_1), r(x_2), \dots, r^n_x))^\intercal$, and $R^* = \text{diag}(\mathbf{r}^*)$ with $\mathbf{r}^* = (r(x^*_1, r(x^*_2), \dots, r(x^*_q))^\intercal$.

In the implementation originally suggested by Goldberg et al., $r(x)$ has a Gaussian prior placed on it and is modeled using an independent GP (more specifically, the log of the noise is nodeled using an independent GP). Goldberg et al. set $z(x) = \log(r(x))$, letting $\mathbf{z} = z_1, z_2, \dots z_n$ coincide with the input points to the main GP. Since the noise rates are now independent latent variables, the predictive distribution becomes

$$P(\mathbf{t}^*|X^*, D) = \iint P(\mathbf{t}^*|X^*, \mathbf{z}, z^*, D)P(\mathbf{z}, z^*|X^*, D)d\mathbf{z}dz^*$$

However, $P(\mathbf{z}, z^8|X^*, D)$ is difficult to handle analytically, and thus Goldberg et al. apply MCMC sampling. However, we can approximate using the most likely noise levels, $(\tilde{\mathbf{z}}, \tilde{z}^*)$: $P(\mathbf{t}^*|X^*, D) \approx P(\mathbf{t}^*|X^*, \tilde{\mathbf{z}}, \tilde{z}^*, D)$.

2.2 Algorithm

Finally, we may now write the algorithm proposed in Kersting et al.:

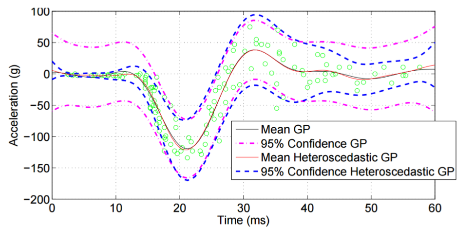
1. Given the input data $D = (x_i, t_i)_{i=1}^n$, estimate a standard, homoscedastic GP G_1 maximizing the likelihood for predicting t from x.
2. Given G_1, estimate the empirical noise levels for the training data, i.e., $z'_i = \log(\text{var}[t_i, G_1(x_i, D)])$, forming a new data set $D' = (x_1, z'_1), (x_2, z'_2), \dots, (x_n, z'_n)$.
3. On D', estimate a second GP G_2. This is the GP of the log noise.
4. Now estimate the combined GP G_3 on D using G_2 to predict the logarithmic noise levels r_i.
5. If not converged, set $G_1 = G_3$ and go to step 2.

This is an EM-like algorithm, where noise levels and data are computed in an alternating fashion until convergence - first, noise levels are calculated given the current prediction (G_1, step 2), then the GP is computed for the noise levels. Afterwards, compute the maximum likelihood parameter given the current prediction G_1, and the current prediction for noise, G_2, repeating until this maximum likelihood converges.

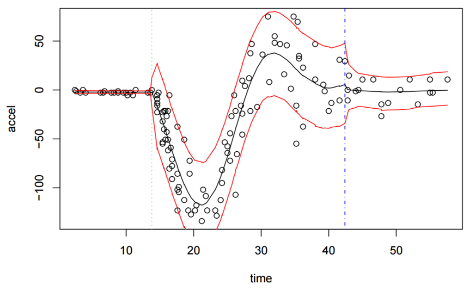
For the variance in step 2, viewing t_i and $G_1(x_i, D)$ as independent observations of a noise-free unknown target, the arithmetic mean $(t_i - G_1(x_i, D))^2/2$ is a natural estimate for the noise level.

2.3 Comparison to treed GP

The primary benefit of the modifications we've made to standard GPs is to allow for heteroscedastic noise. One other modification on standard GPs that also allows for heteroscedastic noise is treed GPs, as discussed by Grammacy et al. Treed GPs employ the idea that GPs may have multiple distinct regions, each with independent noise levels. And, indeed, using the standard motorcycle crash test set, we can see that Treed GPs and Heteroscedastic GPs are both able to produce much tighter noise bounds in the region before initial impact:



(a) Heteroscedastic GPs - Kersting et al. 2007



(b) Treed GPs - Grammacy et al. 2007

Figure 1: Comparison between Heteroscedastic GPs and Treed GPs on the motorcycle crash test set, showing that they both achieve much better noise levels in the pre-impact region than standard GPs and show similar performance elsewhere.

In general, Heteroscedastic GPs offer a few benefits over treed GPs:

- Continuous noise functions.
- No need to decide where to branch the tree, nor how many branches to create.
- Easier to implement. Though being easier to implement does not necessarily make a model superior, in this case it alludes to the simpler and more straightforward nature of the Heteroscedastic algorithm.

3 Video Reconstruction

One of the original goals of this project was to apply GPs to video reconstruction. First, we'll explore why this might be useful, then go into the details of the propoesd method and show some preliminary results.

3.1 Motivation

Given the situation where one has a temporally-compressed video file and one wishes to reconstruct the missing frames, we propose that GPs may be useful. In particular, the "dumb averaging" algorithm cannot account for movement in the video frame:

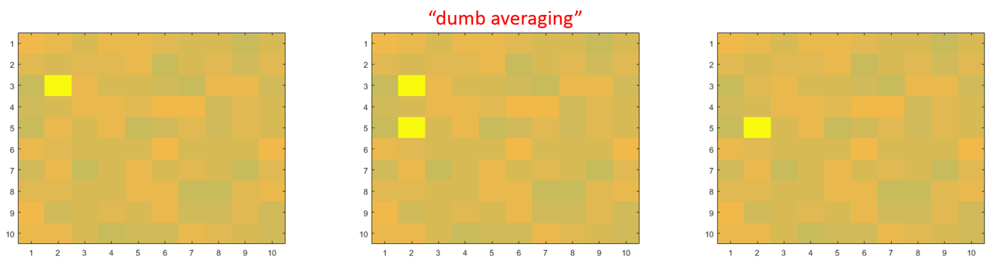


Figure 2: "dumb averaging" algorithm applied to a temporally-compressed video file. Given the frame on the right and left, the "dumb averaging" algorithm simply tries to average the two together to produce the missing center frame. Unfortunately, rather than capturing the movement of the yellow square, it instead just produces half of a square in one location and half of a square in another location.

Due to the failure of "dumb averaging" to capture motion, we wish to employ GPs to enable object tracking.

3.2 Proposed Method

In order to account for movement within the frame, we need a measure of the position of each "object" in the frame. In the most simple case, we take pixels as objects, and allow for pixels to move around the frame. We model the movement of each pixel in x and y as independent GPs, with just a 1-dimensional input of timestep. In order to determine the position of each pixel or object in each frame, a number of methods can be employed, such as image recognition, object tracking, cross correlation, or others. In the end, we choose cross correlation for ease of use and speed. The final step, then, is how to reconstruct missing video frames from this set of trained GPs - we propose to have a voting structure - each pixel of the output image will be the maximum of the likelihoods given to it by sampling each GP at the given (x, y). Basically, this allows for each of the pixels to “move" - have a velocity in each of the output variables, which is the gradient of the associated GP. This also means that if a pixel has a high “velocity", its position can be filled in with background pixels that have a low but high-variance “velocity".

One may ask - if we have position, why not just interpolate? Or fit to a quadratic if we expect accelerating motion (eg, ballistics motion). There are a few benefits to using a GP approach over interpolation:

- Automatic adaption to different motion curves. This is similar to the treed GP argument - if our data has different regimes, finding a suitable fit for each regime, as well as lining up the different regimes, may be difficult - that’s where GPs come in.
- Automatic background maintenance via the voting phase of the reconstruction algorithm, which, to be automatic, requires a noise estimate for the position at each timestep.

3.3 Toy Results

Applying this method to a toy dataset video, we can easily extract the position of objects, and track them successfully throughout the video. Implementing the Heteroscedastic algorithm allows for us to have more dynamic and tighter noise bounds where possible.

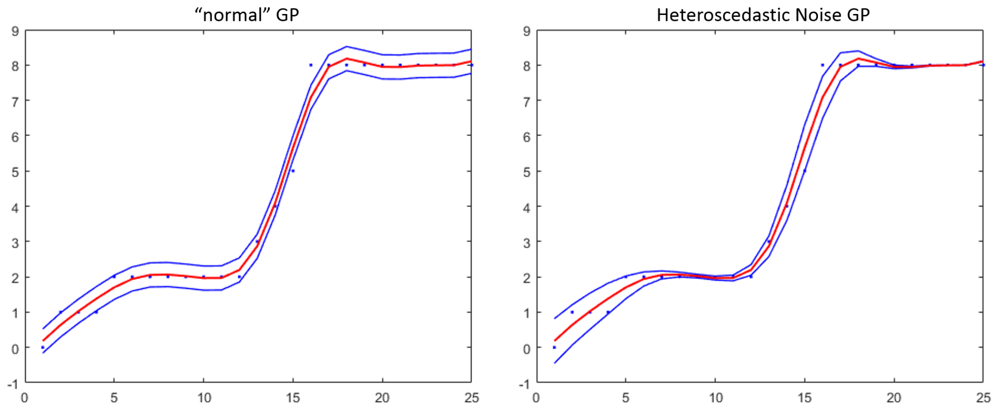


Figure 3: Comparison of tracking of a yellow pixel across a noisy background, showing the superior noise thresholds of the Heteroscedastic implementation.

4 Conclusions

We have implemented a Heteroscedastic Gaussian Process, replicating the basic results of Kersting et al., and have proposed a new use case for this in temporally-compressed video reconstruction. We have implemented the proposed algorithm, and report good preliminary results at tracking objects, though the voting structure and background “backfill” features were never quite tuned correctly. Future work would include refining this, most notably, using a Gaussian estimate to position to update the GPs, initially feeding in the standard deviation as part of the dataset as G_2 in the Heteroscedastic GP algorithm.

5 References

- Goldberg, P., Williams, C., & Bishop, C. (1998). Regression with input-dependent noise: A gaussian process treatment. *NIPS 1998*.
- Kersting, K., Plagemann, C., Pfaff, P., & Burgard, W. (2007). Most Likely Heteroscedastic Gaussian Process Regression. *ICML 2007*.
- Gramacy, R., & Lee, H. (2007). Bayesian Treed Gaussian Process Models with an Application to Computer Modeling. *ArXiv*.