
Gaussian Processes with Probabilistic Programming

000
001
002
003
004
005
006
007
008
009
010
011
012
013
014
015
016
017
018
019
020
021
022
023
024
025
026
027
028
029
030
031
032
033
034
035
036
037
038
039
040
041
042
043
044
045
046
047
048
049
050
051
052
053
Anonymous Author(s)

Affiliation
Address
email

Abstract

Abstract

1 Introduction

Probabilistic programming could be revolutionary for machine intelligence due to universal inference engines and the rapid prototyping for novel models (Ghahramani, 2015).

In the following, we will present how to do Bayesian non-parametrics in form of Gaussian Processes. MCMC lends itself to Bayesian interpretations of Gaussian Processes since they can provide a vehicle to express otherwise intractable integrals necessary for a fully Bayesian representation.

2 Gaussian Processes

In the following, we will introduce GP related theory and notations. We will exclusively work on two variable regression problems. Let the data be real-valued scalars $\{x_i, y_i\}_{i=1}^n$ (complete data will be denoted by column vectors \mathbf{x}, \mathbf{y}). GPs present a non-parametric way to express prior knowledge on the space of possible functions f that we assume to have generated the data. f is assumed latent and the GP prior is given by a multivariate Gaussian with mean and covariance $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}'))$, where $m(\mathbf{x})$ is a function of the mean of all functions that map to y_i at x_i and $k(\mathbf{x}, \mathbf{x}')$ is a kernel or covariance function that summarizes the covariance of all functions that map to y_i at x_i . We can absorb the mean function into the covariance function so without loss of generality we can set the mean to zero. The marginal likelihood can be expressed as:

$$p(\mathbf{y}|\mathbf{x}) = \int p(\mathbf{y}|\mathbf{f}, \mathbf{x}) p(\mathbf{f}|\mathbf{x}) d\mathbf{f} \quad (1)$$

where the prior is Gaussian $\mathbf{f}|\mathbf{x} \sim \mathcal{N}(0, k(\mathbf{x}, \mathbf{x}'))$. For a zero mean Gaussian Process this results in a Gaussian posterior $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with mean:

$$\boldsymbol{\mu} = \mathbf{K}(\mathbf{x}, \mathbf{x}^*) \mathbf{K}(\mathbf{x}^*, \mathbf{x}^*)^{-1} \mathbf{y} \quad (2)$$

and covariance

$$\boldsymbol{\Sigma} = \mathbf{K}(\mathbf{x}, \mathbf{x}) + \mathbf{K}(\mathbf{x}, \mathbf{x}^*) \mathbf{K}(\mathbf{x}^*, \mathbf{x}^*)^{-1} \mathbf{K}(\mathbf{x}^*, \mathbf{x}). \quad (3)$$

where \mathbf{K} is a covariance function. The covariance function covers general high-level properties of the observed data such as linearity, periodicity and smoothness. The most widely used type of covariance function is the squared exponential covariance function:

$$k(x, x') = \sigma^2 \exp\left(-\frac{(x - x')^2}{2\ell^2}\right) \quad (4)$$

where σ and ℓ are hyper-parameters. σ is a scaling factor and ℓ is the typical length-scale. Smaller variations can be achieved by exchanging these hyper-parameters. Below, we see how we can express simple GP smoothing with a few

054 lines of Venture code while allowing users to custom design covariance functions.
 055

056 Listing 1: GP Smoothing

```

057 [ASSUME 1 1] ∈ {hyper-parameters}
058 [ASSUME sf 2] ∈ {hyper-parameters}
059
060  $k(x, x') := \sigma^2 \exp\left(-\frac{(x-x')^2}{2\ell^2}\right)$ 
061
062 [ASSUME f VentureFunction(k, σ, ℓ) ]
063 [ASSUME SE make-se (apply-function f 1 sf) ]
064 [ASSUME (make-gp 0 SE) ]
065
066 [SAMPLE GP (array 1 2 3)] % Prior
067
068 [OBSERVE GP D]
069
070 [SAMPLE GP (array 1 2 3)]
071
072 [INFER (MH {hyper-parameters} one 100) ]
073
074 [SAMPLE GP (array 1 2 3)] % Posterior

```

075 The first two lines depict the hyper-parameters. We tag both of them to belong to the set {hyper-
 076 parameters}. Every member of this set belongs to the same inference scope. This scope controls the
 077 application of the inference procedure used. In this paper, we use MH throughout. Each scope is
 078 further subdivided into blocks that allow to do block-proposals. In the following we omit the block
 079 notation for readability, since we always choose the block of a certain scope at random.

080 The ASSUME directives describe the assumptions we make for the GP model, we assume the hyper-
 081 parameters 1 and sf (corresponding to ℓ, σ) to be 1 and 2. The squared exponential covariance
 082 function can be defined outside the Venture code with foreign conventional programming languages,
 083 e.g. Python. In that way, the user can define custom covariance functions without being restricted to
 084 the most common ones. We then integrate the foreign function into Venture as VentureFunction. In
 085 the next line this function is associated with the hyper-parameters. Finally, we assume a Gaussian
 086 Process SP with a zero mean and the previously assumed squared exponential covariance function.

087 In the case where hyper-parameters are unknown they can be found deterministically by optimizing
 088 the marginal likelihood using a gradient based optimizer. Non-deterministic, Bayesian representa-
 089 tions of this case are also known (Neal, 1997). Extending the program described in listing 1 for a
 090 Bayesian treatment of hyper-parameters is simple using the build in stochastic procedure that simu-
 091 lates drawing samples from a gamma distribution:

092

093 Listing 2: Bayesian GP Smoothing

```

094 [ASSUME 1 (gamma 1 3)]
095 [ASSUME sf (gamma 1 2)]
096
097  $k(x, x') := \sigma^2 \exp\left(-\frac{(x-x')^2}{2\ell^2}\right)$ 
098
099 [ASSUME f VentureFunction(k, σ, ℓ) ]
100 [ASSUME SE make-se (apply-function f 1 sf) ]
101 [ASSUME (make-gp 0 SE) ]

```

102

103 Larger variations are achieved by changing the type of the covariance function structure. A different
 104 type could be a linear covariance function:

105

106

107

$$k(x, x') = \sigma^2(x - \ell)(x' - \ell). \quad (5)$$

108 Note that covariance function structures are compositional. We can add covariance functions if we
 109 want to model globally valid structures

$$111 \quad k_3(x, x') = k_1(x, x') + k_2(x, x') \quad (6)$$

112 and we can multiply covariance functions if the data is best explained by local structure

$$114 \quad k_4(x, x') = k_1(x, x') \times k_2(x, x'); \quad (7)$$

115 both, k_3 and k_4 are valid covariance function structures. This leads to an infinite space of possible
 116 structures that could potentially explain the observed data best (e.g. Fig. ??). In the following, we
 117 will refer to covariance functions that are not composite as base covariance functions. Note that this
 118 form of composition can be easily expressed in Venture, for example if one wishes to add a linear
 119 and a periodic kernel:

121 Listing 3: LIN \times PER

```
122 [ASSUME 1 (gamma 1 3)]
123 [ASSUME sf (gamma 1 2)]
124 [ASSUME a (gamma 2 2)]
125
126  $k_{LIN}(x, x') = \sigma_1^2(x - \ell)(x' - \ell)$ 
127
128  $k_{PER}(x, x') := \sigma_2^2 \exp\left(-\frac{2 \sin^2(\pi(x-x')/p)}{\ell^2}\right)$ 
129
130 [ASSUME f_LIN VentureFunction(k_LIN, sigma_1) ]
131 [ASSUME f_PER VentureFunction(k_PER, sigma_2, ell, p) ]
132 [ASSUME LIN (make-LIN (apply-function f_LIN a)) ]
133 [ASSUME PER (make-PER (apply-function f_PER 1 sf)) ]
134 [ASSUME (make-gp 0 (function-times LIN PER)) ]
```

135 Knowledge about the composite nature of covariance functions is not new, however, until recently,
 136 the choice and the composition of covariance functions were done ad-hoc. The Automated Statisti-
 137 cian Project came up with an approximate search over the possible space of kernel structures (Du-
 138 venaud et al., 2013; Lloyd et al., 2014).

140 2.1 A Bayesian interpretation

141 In the following, we will explore a Bayesian representation of GP. The probability of the hyper-
 142 parameters of a GP with assumptions as above and given covariance function structure \mathbf{K} can be
 143 described as:

$$145 \quad P(\boldsymbol{\theta} | \mathbf{D}, \mathbf{K}) = \frac{P(\mathbf{D} | \boldsymbol{\theta}, \mathbf{K})P(\boldsymbol{\theta} | \mathbf{K})}{P(\mathbf{D} | \mathbf{K})}. \quad (8)$$

147 We are interested in the case where covariance structure is not given. Our probabilistic programming
 148 based MCMC framework approximates the following intractable integrals of the expectation for the
 149 prediction:

$$151 \quad \mathbb{E}[y^* | x^*, D, \mathbf{K}_\Omega^s] = \int \int f(x^*, \boldsymbol{\theta}, \mathbf{K}) P(\boldsymbol{\theta} | \mathbf{D}, \mathbf{K}) P(\mathbf{K} | \boldsymbol{\Omega}, s, n) d\boldsymbol{\theta} d\mathbf{K}. \quad (9)$$

153 This is done by sampling from the posterior probability distribution of the hyper-parameters and the
 154 possible kernel:

$$155 \quad y^* \approx \frac{1}{T} \sum_{t=1}^T f(x^* | \boldsymbol{\theta}^{(t)}, \mathbf{K}^{(t)}). \quad (10)$$

158 3 Stochastic Processes

161 In order to provide the sampling of the kernel, we introduce a stochastic process to the SP that
 simulates the grammar for algebraic expressions of kernel algebra. Here, we start with a set of

possible kernels and draw a random subset. For this subset of size n , we sample a set of possible operators that operate on the base kernels.

The marginal probability of a kernel structure which allows us to sample is characterized by the probability of a uniformly chosen subset of the set of n possible covariance functions times the probability of sampling a global or a local structure which is given by a binomial distribution:

$$P(\mathbf{K} \mid \Omega, s, n) = P(\Omega \mid s, n) \times P(s \mid n) \times P(n), \quad (11)$$

with

$$P(\Omega \mid s, n) = \binom{n}{r} p_{+ \times}^k (1 - p_{+ \times})^{n-k} \quad (12)$$

and

$$P(s \mid n) = \frac{n!}{|s|!} \quad (13)$$

where $P(n)$ is a prior on the number of base kernels used. It is possible to also assign a prior for the probability to sample global or local priors, however, we have assigned complete uncertainty to this with the binomial $p = 0.5$.

4 Experiments

Neal suggested the treatment of outliers as a use-case for a Bayesian treatment of Gaussian processes (1997). He evaluates his MCMC setting using the following synthetic data problem. Let f be the underlying function that generates the data:

$$f(x) = 0.3 + 0.4x + 0.5 \sin(2.7x) + \frac{1.1}{(1+x^2)} + \eta \quad \text{with } \eta \sim \mathcal{N}(0, \sigma) \quad (14)$$

We synthetically generate outliers by setting $\sigma = 0.1$ in 95% of the case and to $\sigma = 1$ in the remaining cases. Venture GPs can capture the true underlying function within only 100 MH steps (see Fig. 2). Note that Neal devices an additional noise model and performs large numbe of Hybrid-Monte Carlo and Gibbs steps.

4.1 Structure Learning

Many equivalent covariance structures can be sampled due to covariance function algebra and equivalent representations with different parameterization (Lloyd et al., 2014). Certain covariance functions can differ in terms of the hyper-parameterization but can be absorbed into a single covariance function with a different parameterization. To inspect the posterior of these equivalent structures we convert each kernel expression into a sum of products and subsequently simplify expressions using the following grammar:

Listing 4: Grammar to simplify expressions

SE \times SE	\rightarrow	SE
{SE, PER, C, WN} \times WN	\rightarrow	WN
LIN + LIN	\rightarrow	LIN
{SE, PER, C, WN, LIN} \times C	\rightarrow	{SE, PER, C, WN, LIN}

For reproducing results from the Automated Statistician Project in a Bayesian fashion we first define a prior on the hypothesis space. Note that, as in the implementation of the Automated Statistician, we upper-bound the complexity of the space of covariance functions we want to explore. We also put vague priors on hyper-parameters.

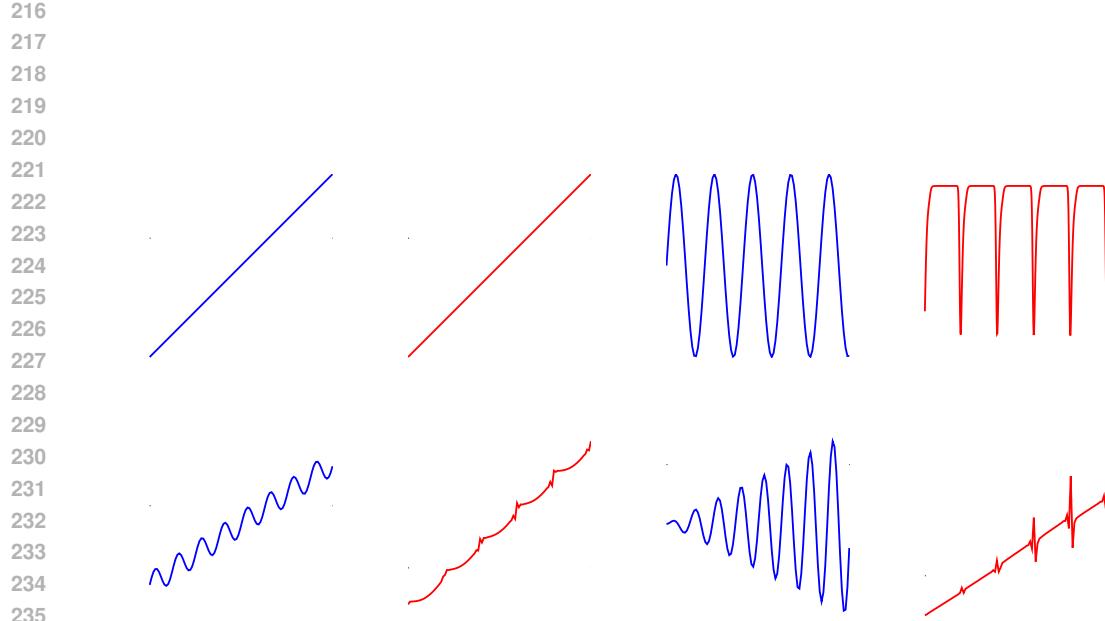


Figure 1: Composition of covariance functions (blue, left) and samples from the distribution of curves they can produce (red, right).

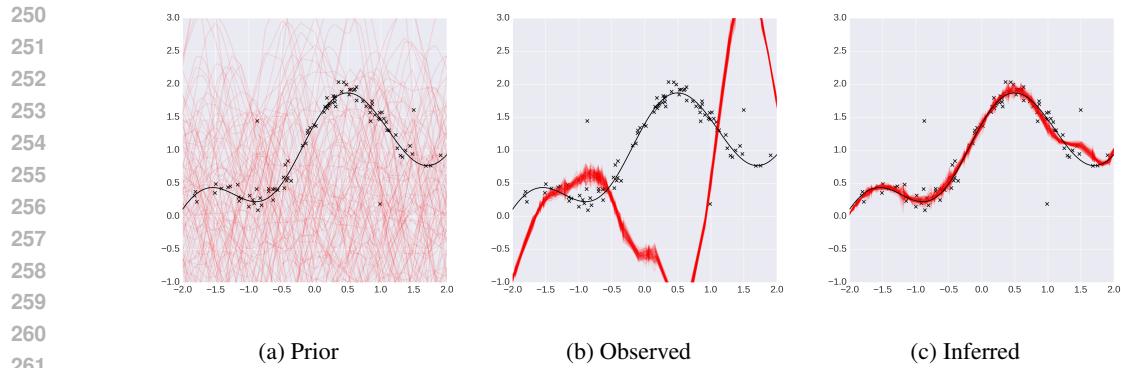


Figure 2: Running a Venture GP on Neal's example for MCMC showing the prior, after having observed the data and after performing inference on the hyper-parameters. Note how the GP is choosing outliers to smooth instead of essential data before inference takes place.

Listing 5: Venture Code for Bayesian GP Structure Learning

```

[ASSUME S (array K1,K2,...,Kn)] // (defined as above)
[ASSUME pn (uniform_structure n)]
[ASSUME S (array K1,K2,...,Kn)]
[ASSUME K* (grammar S pn)]
[ASSUME GP (make-gp 0 K*)]

[OBSERVE GP D]

[INFER (REPEAT 2000 (DO
    (MH 10 pn one 1)
    (MH 10 K* one 1)
    (MH 10 {hyper-parameters} one 10)) ]

```

We defined the space of covariance structures in a way allowing us to reproduce results for covariance function structure learning as in the Automated Statistician. We ran a leave one out cross-validation to gain confidence on the posterior. This resulted in 545 independent runs of the Markov chain that produced a coherent posterior. Our Bayesian interpretation of GP structure and GPs produced a posterior of structures that is in line with previous results on this data set (Duvenaud et al., 2013; see Fig. 3).

We found the final sample of multiple runs to be most informative. This kind of Markov Chain seems to produce samples that are highly auto-correlated.

4.2 Log-Likelihood

4.3 Residuals

5 Bayesian Optimization

Bayesian Optimization poses the problem of finding the global maximum of an unknown function as a hierarchical decision problem (Ghahramani, 2015). Evaluating the actual function can be very expensive. For example, finding the best configuration for the learning algorithm of a large convolutional neural network implies expensive function evaluations to compare a potentially infinite number of configurations. Another common example is the example of data acquisition. For problems with large amounts of data available it may be interested to chose certain informative data-points to evaluate a model on. In continuous domains, many Bayesian Optimization methods deploy GPs (e.g. Snoek et al., 2012).

The hierarchical nature of Bayesian Optimization makes it an ideal application for GPs in Venture. The following Bayesian Optimization scheme is closely related to Thompson Sampling Thompson (1933).

We consider a true unknown reward function $r(x)$ that we estimate with a GP prior $\mathcal{GP}(0, K(\mathbf{x}, \mathbf{x}))$.

References

- Duvenaud, D., Lloyd, J. R., Grosse, R., Tenenbaum, J., and Ghahramani, Z. (2013). Structure discovery in nonparametric regression through compositional kernel search. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1166–1174.

Ghahramani, Z. (2015). Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459.

Lloyd, J. R., Duvenaud, D., Grosse, R., Tenenbaum, J., and Ghahramani, Z. (2014). Automatic construction and natural-language description of nonparametric regression models. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Neal, R. M. (1997). Monte carlo implementation of gaussian process models for bayesian regression and classification. *arXiv preprint physics/9701026*.

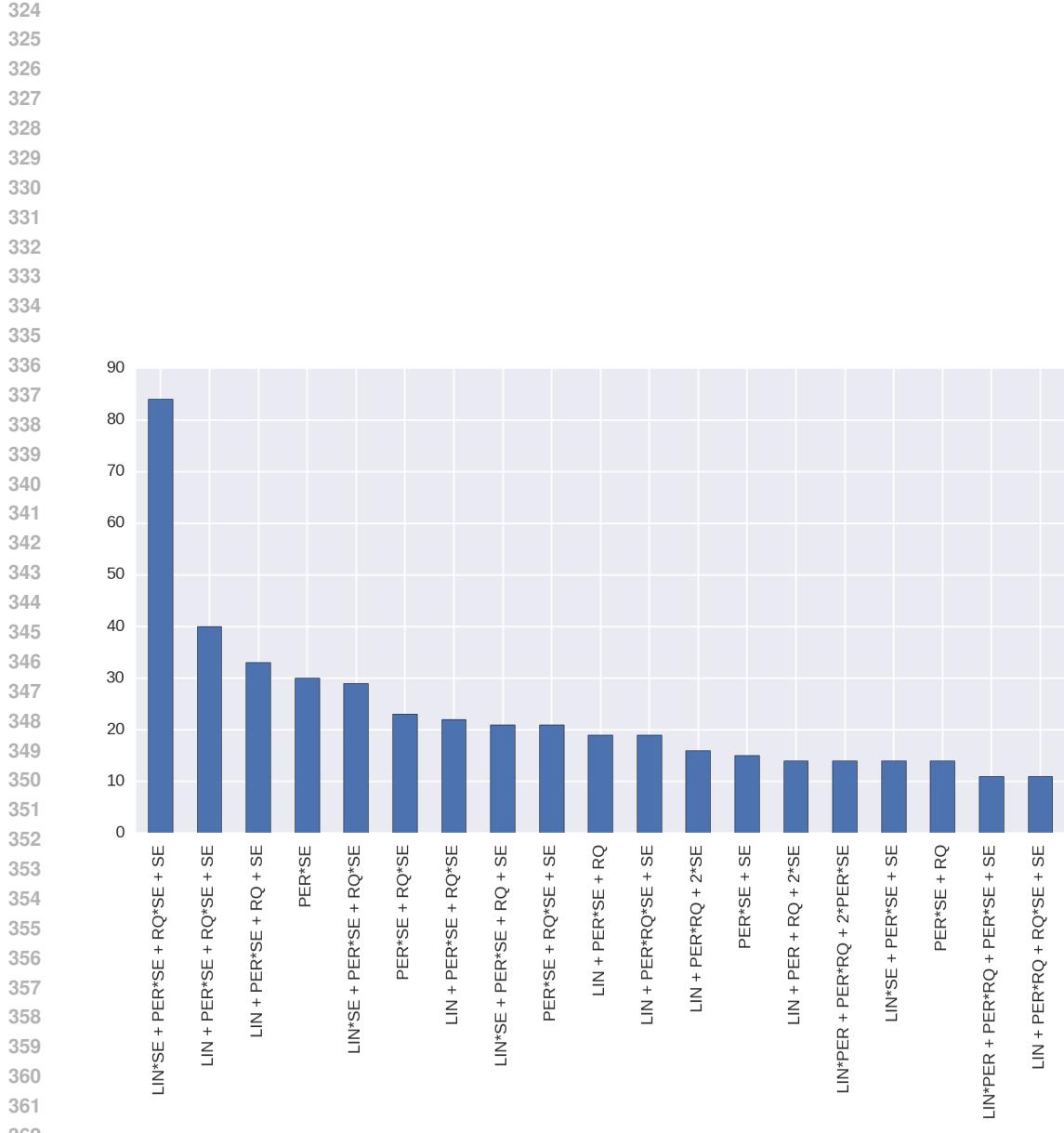
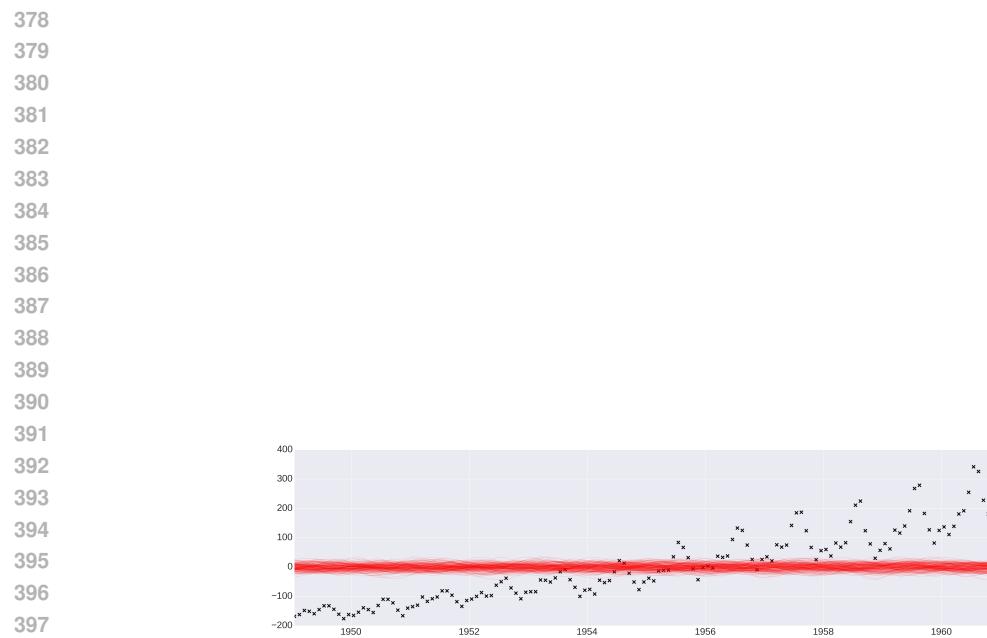


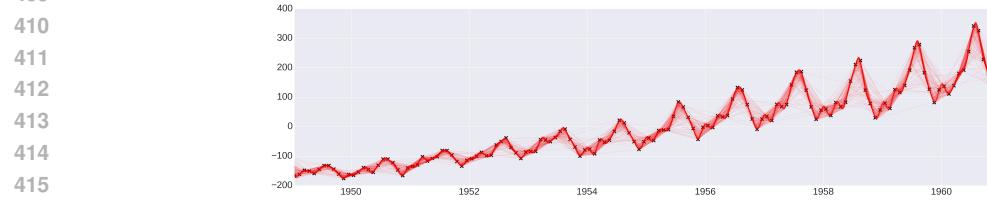
Figure 3: Posterior on structure of the CO₂ data. We have cut the tail of the distribution for space reasons since the number of possible structures is large. We see the final sample of the each of the 545 chains with 2000 nested steps each. Note that Duvenaud et al. (2013) report LIN × SE + PER × SE + RQ × SE.



(a) Prior, before having seen any data.



(b) After having seen any data but before inference



(c) After 1000 MH steps.

Figure 4: Running a Venuter GP with covariance structure PER x SE on the airline data

432 Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine
433 learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2951–2959.

434 Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view
435 of the evidence of two samples. *Biometrika*, pages 285–294.
436

437

438

439

440

441

442

443

444

445

446

447

448

449

450

451

452

453

454

455

456

457

458

459

460

461

462

463

464

465

466

467

468

469

470

471

472

473

474

475

476

477

478

479

480

481

482

483

484

485