

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

# Gaussian Processes with Probabilistic Programming

## Anonymous Author(s)

Affiliation

Address

email

## Abstract

We introduce Venture GPs, a way to formulate Gaussian Processes with probabilistic programming. Gaussian Processes are flexible non-parametric models that can be applied to a broad class of problems. We represent Gaussian Processes in Venture, a Turing complete probabilistic programming language. Venture provides a compositional language with a generalized inference engine that builds on a stochastic procedure interface. This stochastic procedure interface specifies and encapsulates primitive random variables analogously conditional probability tables in Bayesian Networks. The programming language is extended with a set of stochastic processes that allow a user to formulate Gaussian Process models and to perform numerically stable inference over them while hiding the linear algebra needed for this inside the language. We show how we can extend the non-parametric model to incorporate hierarchical causal priors on model structure and hyper-parameters with only a few lines of code. We also show state-of-the-art applications of Gaussian Processes in this framework, namely structure discovery of high-level properties of Gaussian Processes, Bayesian Optimization and hyper-parameter inference. We evaluate the performance of the programs with synthetic and real world data.

## 1 Introduction

Probabilistic programming could be revolutionary for machine intelligence due to universal inference engines and the rapid prototyping for novel models (Ghahramani, 2015). Probabilistic programming languages aim to provide a formal language to specify probabilistic models in the style of computer programming and can represent any computable probability distribution as a program. In this work, we will introduce new features of Venture, a recently developed probabilistic programming language and the first probabilistic programming language suitable for general purpose use (Mansinghka et al., 2014). Venture comes with scalable performance on hard problems and with a general purpose inference engine. The inference engine is based on Markov Chain Monte Carlo (MCMC) methods (for an introduction, see Andrieu et al. (2003)). MCMC lends itself models with complex structures such as probabilistic programs or hierarchical Bayesian non-parametric models since they can provide a vehicle to express otherwise intractable integrals necessary for a fully Bayesian representation. MCMC is scalable, often distributable and also compositional. That is, one can arbitrarily chain MCMC kernels to infer over several hierarchically connected or nested models as they will emerge in probabilistic programming.

One very powerful model yet unseen in probabilistic programming languages are Gaussian Processes (GPs). GPs are gaining increasing attention for representing unknown functions by posterior probability distributions in various fields such as machine learning, signal processing, computer vision and bio-medical data analysis. In the following, we will present Gaussian Processes as a novel feature for probabilistic programming languages. Our contribution is threefold:

- we introduce a new stochastic process for GPs in a probabilistic programming language;

- we show how one can solve hard problems of state-of-the-art machine learning with only a few lines of Venture code; and
- we introduce an additional stochastic process that samples from a probabilistic context free grammar for GP covariance structure generation.

The paper is structured as follows, we will first provide some background on probabilistic programming in Venture and GPs. We will then elaborate on our new stochastic processes. Finally, we will show how we can apply those on problems of hyper-parameter inference, structure discovery for Gaussian Processes and Bayesian Optimization including experiments with real world and synthetic data.

## 2 Background

### 2.1 Venture

Venture is a compositional language for custom inference strategies that comes with a Scheme- and Java-Script-like front-end syntax. It's implementation is based on three concepts:

1. stochastic procedure interfaces that specify and encapsulate random variables, analogously to conditional probability tables in a Bayesian network;
2. probabilistic execution traces that represent execution histories and capture conditional dependencies; and
3. scaffolds that partition execution histories and factor global inference problems into sub-problems.

These building blocks provide a powerful way to represent probability distributions; some of which cannot be expressed with density functions. For the purpose of this work the most important Venture directives that operate on these building blocks to understand are ASSUME, OBSERVE, SAMPLE and INFER. ASSUME induces a hypothesis space for (probabilistic) models including random variables by binding the result of an expression to a symbol. SAMPLE simulates a model expression and returns a value. OBSERVE adds constraints to model expressions. INFER instructions incorporate observations and cause Venture to find a hypothesis that is probable given the data.

INFER is most commonly done by deploying the Metropolis-Hastings algorithm (MH) (Metropolis et al., 1953). Many algorithms used in the MCMC world can be interpreted as special cases of MH (Andrieu et al., 2003). We can outline the MH algorithm as follows. For  $T$  steps we sample  $x^*$  from a proposal distribution  $q$ :

$$x^* \sim q(x^* | x^{(t)}) \quad (1)$$

which we accept ( $x^{t+1} \leftarrow x^*$ ) with ratio:

$$\alpha = \min \left\{ 1, \frac{p(x^*)q(x^t | x^*)}{p(x^{(t)})q(x^* | x^t)} \right\} \quad (2)$$

Venture implements an MH transition operator for probabilistic execution traces.

### 2.2 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  $f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), k(x_i, x'_i))$ , where  $m(\mathbf{x})$  is a function of the mean of all functions that map to  $y_i$  at  $x_i$  and  $k(x_i, x'_i)$  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:

$$p(\mathbf{y}|\mathbf{x}) = \int p(\mathbf{y}|\mathbf{f}, \mathbf{x}) p(\mathbf{f}|\mathbf{x}) d\mathbf{f} \quad (3)$$

108 where the prior is Gaussian  $\mathbf{f}|\mathbf{x} \sim \mathcal{N}(0, k(\mathbf{x}, \mathbf{x}'))$ . We can sample a vector of unseen data from the  
 109 predictive posterior with

$$110 \quad \mathbf{y}^* \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \quad (4)$$

111 for a zero mean prior GP with a posterior mean of:

$$112 \quad \boldsymbol{\mu} = \mathbf{K}(\mathbf{x}, \mathbf{x}^*) \mathbf{K}(\mathbf{x}^*, \mathbf{x}^*)^{-1} \mathbf{y} \quad (5)$$

114 and covariance

$$115 \quad \boldsymbol{\Sigma} = \mathbf{K}(\mathbf{x}, \mathbf{x}) + \mathbf{K}(\mathbf{x}, \mathbf{x}^*) \mathbf{K}(\mathbf{x}^*, \mathbf{x}^*)^{-1} \mathbf{K}(\mathbf{x}^*, \mathbf{x}). \quad (6)$$

116  $\mathbf{K}$  is a covariance function. The log-likelihood is defined as:

$$117 \quad \log P(\mathbf{y} | \mathbf{X}) = -\frac{1}{2} \mathbf{y}^\top (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{y} - \frac{1}{2} \log |\mathbf{K} + \sigma^2 \mathbf{I}| - \frac{n}{2} \log 2\pi \quad (7)$$

118 with  $n$  being the number of data-points and sigma the independent observation noise. Both log-  
 119 likelihood and predictive posterior can be computed efficiently in a Venture SP with an algorithm  
 120 that resorts to Cholesky factorization(Rasmussen and Williams, 2006, chap. 2) resulting in a com-  
 121 putational complexity of  $\mathcal{O}(n^3)$  in the number of data-points.  
 122

123 The covariance function covers general high-level properties of the observed data such as linear-  
 124 ity, periodicity and smoothness. The most widely used type of covariance function is the squared  
 125 exponential covariance function:  
 126

$$127 \quad k(x, x') = \sigma^2 \exp\left(-\frac{(x - x')^2}{2\ell^2}\right) \quad (8)$$

128 where  $\sigma$  and  $\ell$  are hyper-parameters.  $\sigma$  is a scaling factor and  $\ell$  is the typical length-scale. Smaller  
 129 variations can be achieved by adapting these hyper-parameters.  
 130

### 132 3 Venture GPs

134 Given a stochastic process that implements the GP algebra above we can implement a GP sam-  
 135 pler (4) to perform GP inference in a few lines of code. We can express simple GP smooth-  
 136 ing with fixed hyper-parameters and perform MH on it code while allowing users to custom de-  
 137 sign covariance functions. Throughout the paper, we will use the Scheme-like front-end syntax.  
 138

139 Listing 1: GP Smoothing

```
140 [ASSUME 1 1] ∈ {hyper-parameters}
141 [ASSUME sf 2] ∈ {hyper-parameters}
142
143 k(x, x') := σ² exp(-\frac{(x - x')²}{2ℓ²})
144
145 [ASSUME f VentureFunction(k, σ, ℓ) ]
146 [ASSUME SE make-se (apply-function f 1 sf) ]
147 [ASSUME (make-gp 0 SE) ]
148
149 [SAMPLE GP (array 1 2 3)] % Prior
150
151 [OBSERVE GP D]
152
153 [SAMPLE GP (array 1 2 3)]
154
155 [INFER (MH {hyper-parameters} one 100) ]
156
157 [SAMPLE GP (array 1 2 3)] % Posterior
```

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

162 The ASSUME directives describe the assumptions we make for the GP model, we assume the hyper-  
 163 parameters  $\ell$  and  $\sigma$  (corresponding to  $\ell, \sigma$ ) to be 1 and 2. The squared exponential covariance  
 164 function can be defined outside the Venture code with foreign conventional programming languages,  
 165 e.g. Python. In that way, the user can define custom covariance functions without being restricted to  
 166 the most common ones. We then integrate the foreign function into Venture as VentureFunction. In  
 167 the next line this function is associated with the hyper-parameters. Finally, we assume a Gaussian  
 168 Process SP with a zero mean and the previously assumed squared exponential covariance function.

169 In the case where hyper-parameters are unknown they can be found deterministically by optimizing  
 170 the marginal likelihood using a gradient based optimizer. Non-deterministic, Bayesian representa-  
 171 tions of this case are also known (Neal, 1997) where we draw hyper-parameters from  $\Gamma$  distributions:  
 172

$$\ell^{(t)} \sim \Gamma(\alpha_1, \beta_1) \quad (9)$$

$$\sigma^{(t)} \sim \Gamma(\alpha_2, \beta_2) \quad (10)$$

182 Extending the program described in listing 1 to draw the hyper-parameters for a Bayesian treatment  
 183 of hyper-parameters is simple using the build in stochastic procedure that simulates drawing samples  
 184 from a gamma distribution:

186 Listing 2: Bayesian GP Smoothing

```
187 [ASSUME 1 (gamma 1 3)]
188 [ASSUME sf (gamma 1 2)]
189
190  $k(x, x') := \sigma^2 \exp\left(-\frac{(x-x')^2}{2\ell^2}\right)$ 
191
192 [ASSUME f VentureFunction(k, sigma, ell) ]
193 [ASSUME SE make-se (apply-function f 1 sf) ]
194 [ASSUME (make-gp 0 SE ) ]
```

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

$$k(x, x') = \sigma^2(x - \ell)(x' - \ell). \quad (11)$$

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

$$k_3(x, x') = k_1(x, x') + k_2(x, x') \quad (12)$$

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

$$k_4(x, x') = k_1(x, x') \times k_2(x, x'); \quad (13)$$

212 both,  $k_3$  and  $k_4$  are valid covariance function structures. This leads to an infinite space of possible  
 213 structures that could potentially explain the observed data best (e.g. Fig. 1). In the following, we  
 214 will refer to covariance functions that are not composite as base covariance functions. Note that this  
 215 form of composition can be easily expressed in Venture, for example if one wishes to add a linear  
 and a periodic kernel:

216

217

Listing 3: LIN × PER

```

218 [ASSUME 1 (gamma 1 3)]
219 [ASSUME sf (gamma 1 2)]
220 [ASSUME a (gamma 2 2)]
221
222  $k_{LIN}(x, x') = \sigma_1^2(x - \ell)(x' - \ell)$ 
223  $k_{PER}(x, x') := \sigma_2^2 \exp\left(-\frac{2 \sin^2(\pi(x-x')/p)}{\ell^2}\right)$ 
224
225 [ASSUME fLIN VentureFunction( $k_{LIN}, \sigma_1$ ) ]
226 [ASSUME fPER VentureFunction( $k_{PER}, \sigma_2, \ell, p$ ) ]
227 [ASSUME LIN (make-LIN (apply-function fLIN a)) ]
228 [ASSUME PER (make-PER (apply-function fPER 1 sf)) ]
229 [ASSUME (make-gp 0 (function-times LIN PER)) ]
230

```

231 Knowledge about the composite nature of covariance functions is not new, however, until recently,  
 232 the choice and the composition of covariance functions were done ad-hoc. The Automated Statisti-  
 233 cian Project came up with an approximate search over the possible space of kernel structures (Du-  
 234 venaud et al., 2013; Lloyd et al., 2014). However, a fully Bayesian treatment of this was not done  
 235 before.

236

### 237 3.1 A Bayesian interpretation

238

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

$$P(\boldsymbol{\theta} | \mathbf{D}, \mathbf{K}) = \frac{P(\mathbf{D} | \boldsymbol{\theta}, \mathbf{K})P(\boldsymbol{\theta} | \mathbf{K})}{P(\mathbf{D} | \mathbf{K})}. \quad (14)$$

242

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

$$f(x) = 0.3 + 0.4x + 0.5 \sin(2.7x) + \frac{1.1}{(1+x^2)} + \eta \quad \text{with } \eta \sim \mathcal{N}(0, \sigma) \quad (15)$$

246

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

251

### 252 3.2 Structure Learning

253

254 The case where the covariance structure is not given is even more interesting. Our probabilistic  
 255 programming based MCMC framework approximates the following intractable integrals of the ex-  
 256 pectation for the prediction:

$$\mathbb{E}[y^* | x^*, D, \mathbf{K}_\Omega^s] = \iint f(x^*, \boldsymbol{\theta}, \mathbf{K}) P(\boldsymbol{\theta} | \mathbf{D}, \mathbf{K}) P(\mathbf{K} | \boldsymbol{\Omega}, s, n) d\boldsymbol{\theta} d\mathbf{K}. \quad (16)$$

257

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

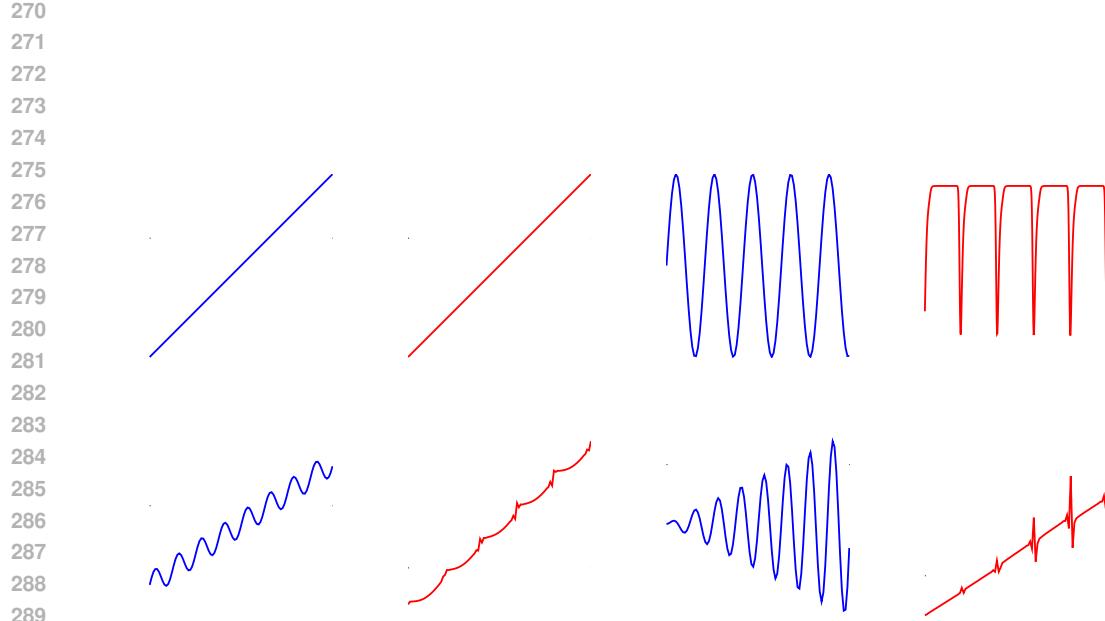
$$y^* \approx \frac{1}{T} \sum_{t=1}^T f(x^* | \boldsymbol{\theta}^{(t)}, \mathbf{K}^{(t)}). \quad (17)$$

260

261 In order to provide the sampling of the kernel, we introduce a stochastic process to the SP that  
 262 simulates the grammar for algebraic expressions of covariance function algebra:

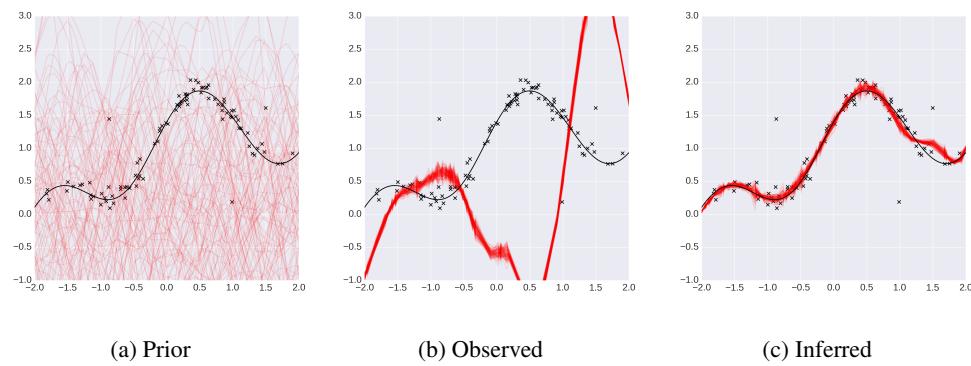
263

$$\mathbf{K}^{(t)} \sim P(\mathbf{K} | \boldsymbol{\Omega}, s, n) \quad (18)$$



290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315

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



316  
317  
318  
319  
320  
321  
322  
323

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.

324 Here, we start with a set of possible kernels and draw a random subset. For this subset of size  $n$ , we  
 325 sample a set of possible operators that operate on the base kernels.  
 326

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

$$P(\mathbf{K} \mid \Omega, s, n) = P(\Omega \mid s, n) \times P(s \mid n) \times P(n), \quad (19)$$

330 with  
 331

$$P(\Omega \mid s, n) = \binom{n}{r} p_{+\times}^k (1 - p_{+\times})^{n-k} \quad (20)$$

332 and  
 333

$$P(s \mid n) = \frac{n!}{|s|!} \quad (21)$$

334 where  $P(n)$  is a prior on the number of base kernels used which can sample from a discrete uniform  
 335 distribution. This will strongly prefer simple covariance structures with few base kernels since  
 336 individual base kernels are more likely to be sampled in this case due to (21). Alternatively, we  
 337 can approximate a uniform prior over structures by weighting  $P(n)$  towards higher numbers. It is  
 338 possible to also assign a prior for the probability to sample global or local structures, however, we  
 339 have assigned complete uncertainty to this with the probability of a flip  $p = 0.5$ .  
 340

341 Many equivalent covariance structures can be sampled due to covariance function algebra  
 342 and equivalent representations with different parameterization (Lloyd et al., 2014). Certain  
 343 covariance functions can differ in terms of the hyper-parameterization but can be  
 344 absorbed into a single covariance function with a different parameterization. To inspect  
 345 the posterior of these equivalent structures we convert each kernel expression into  
 346 a sum of products and subsequently simplify expressions using the following grammar:  
 347

348 Listing 4: Grammar to simplify expressions  
 349

SE × SE	→ SE
{SE, PER, C, WN} × WN	→ WN
LIN + LIN	→ LIN
{SE, PER, C, WN, LIN} × C	→ {SE, PER, C, WN, LIN}

350 For reproducing results from the Automated Statistician Project in a Bayesian fashion we first define  
 351 a prior on the hypothesis space. Note that, as in the implementation of the Automated Statistician,  
 352 we upper-bound the complexity of the space of covariance functions we want to explore. We also  
 353 put vague priors on hyper-parameters.  
 354

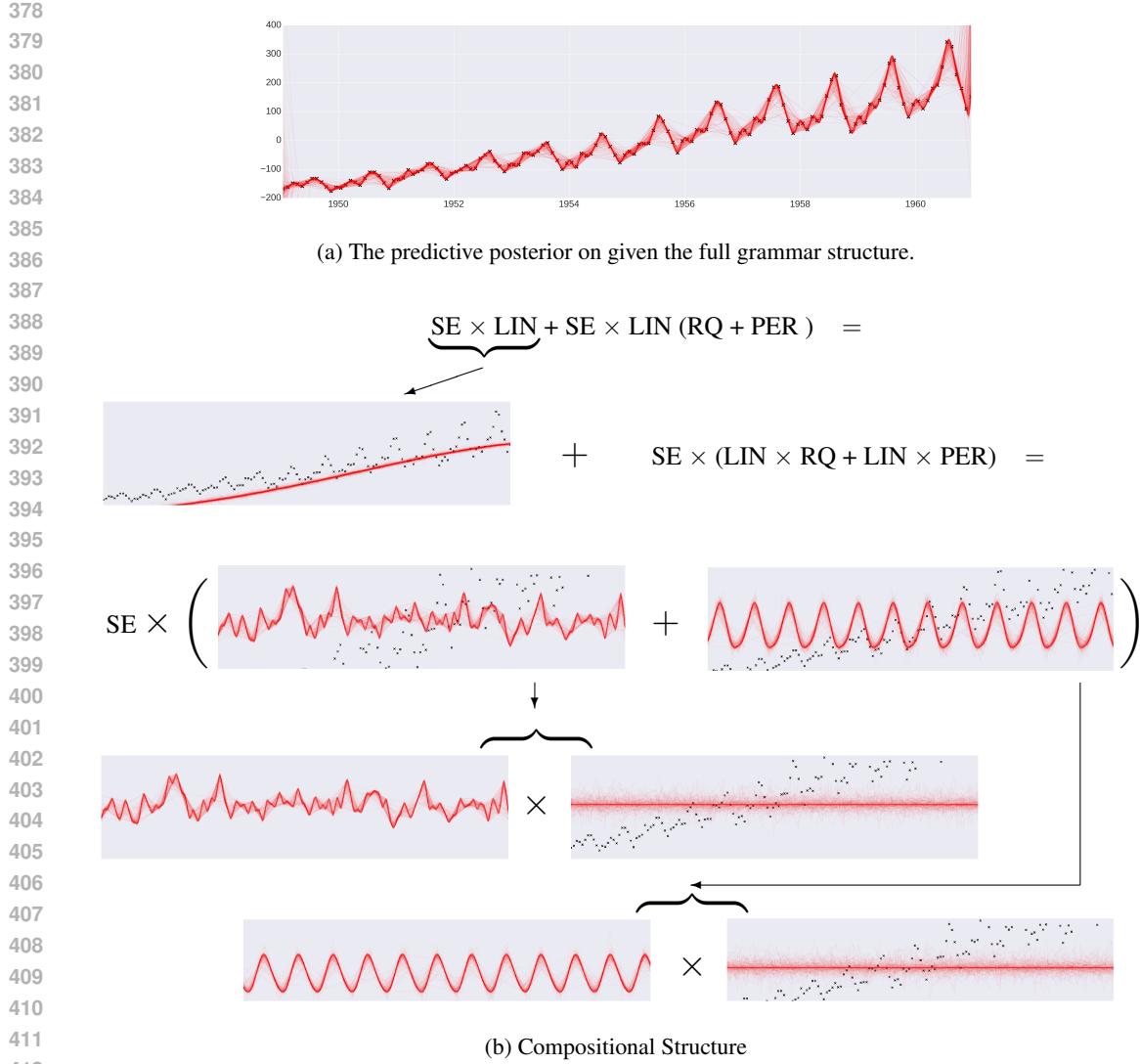
355 Listing 5: Venture Code for Bayesian GP Structure Learning  
 356

```
[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))]
```

357 We defined the space of covariance structures in a way allowing us to reproduce results for covariance  
 358 function structure learning as in the Automated Statistician. This lead to coherent results, for  
 359 example for the airline data set. We will elaborate the result using a sample from the posterior (Fig.  
 360



413 Figure 3: a) We see the predictive posterior as an result 1000 nested MH steps on the airline data set.  
 414 b) depicts a decomposition of this posterior for the structures sampled by Venture. RQ is the rational  
 415 quadratic covariance function. Note that although the overall result is in line with what Duvenaud  
 416 et al. (2013) report a slightly different composition is implied by the sampled parameters for the  
 417 structure. The first line shows the global trend and denotes the rest of the structure that is shown  
 418 above. In the second line, the see the periodic component on the right hand side. The left hand  
 419 side denotes short term deviations both multiplied by a smoothing kernel. The third and fourth lines  
 420 denote how we reach the second line: both periodic and rational quadratic covariance functions are  
 421 multiplied by a line with slope zero.

422  
 423  
 424  
 425 3). The sample is identical with the highest scoring result reported in previous work using a search-  
 426 and-score method (Duvenaud et al., 2013) and the predictive capability is comparable. However, the  
 427 components factor in a different way due to different parameterization of the individual base kernels.

428 We further investigated the quality of our stochastic processes by running a leave one out cross-  
 429 validation to gain confidence on the posterior. This resulted in 545 independent runs of the Markov  
 430 chain that produced a coherent posterior: our Bayesian interpretation of GP structure and GPs pro-  
 431 duced a posterior of structures that is in line with previous results on this data set ( Duvenaud et al.,  
 2013; see Fig. 4).

432 We found the final sample of multiple runs to be most informative. This kind of Markov Chain  
433 seems to produce samples that are highly auto-correlated.  
434

## 435 4 Bayesian Optimization 436

438 Bayesian Optimization poses the problem of finding the global maximum of an unknown function  
439 as a hierarchical decision problem (Ghahramani, 2015). Evaluating the actual function can be  
440 very expensive. For example, finding the best configuration for the learning algorithm of a large  
441 convolutional neural network implies expensive function evaluations to compare a potentially infinite  
442 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-  
443 points to evaluate a model on. In continuous domains, many Bayesian Optimization methods deploy  
444 GPs (e.g. Snoek et al., 2012).

445 The hierarchical nature of Bayesian Optimization makes it an ideal application for GPs in Venture.  
446 The following Bayesian Optimization scheme is closely related to Thompson Sampling Thompson  
447 (1933). Thompson Sampling is a general framework to solve exploration-exploitation problems that  
448 applies to our notion of Bayesian Optimization. We we sample a probe from the posterior  
449

$$450 \hat{\theta} \sim P(\hat{\theta} | \vec{x}) \quad (22)$$

451 and then optimize the expected reward by choosing an action  $a$   
452

$$453 a_i = \arg \max_a \mathbb{E}_{P(\dots|\theta)}[r(world_\theta(a))] \quad (23)$$

$$455 x_i \sim world_\theta(a_i) \quad (24)$$

## 457 References 458

- 459 Andrieu, C., De Freitas, N., Doucet, A., and Jordan, M. I. (2003). An introduction to mcmc for  
460 machine learning. *Machine learning*, 50(1-2):5–43.
- 461 Duvenaud, D., Lloyd, J. R., Grosse, R., Tenenbaum, J., and Ghahramani, Z. (2013). Structure  
462 discovery in nonparametric regression through compositional kernel search. In *Proceedings of  
463 the 30th International Conference on Machine Learning (ICML-13)*, pages 1166–1174.
- 464 Ghahramani, Z. (2015). Probabilistic machine learning and artificial intelligence. *Nature*,  
465 521(7553):452–459.
- 466 Lloyd, J. R., Duvenaud, D., Grosse, R., Tenenbaum, J., and Ghahramani, Z. (2014). Automatic  
467 construction and natural-language description of nonparametric regression models. In *Twenty-  
468 Eighth AAAI Conference on Artificial Intelligence*.
- 469 Mansinghka, V. K., Selsam, D., and Perov, Y. (2014). Venture: a higher-order probabilistic pro-  
470 gramming platform with programmable inference. *arXiv preprint arXiv:1404.0099*.
- 471 Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation  
472 of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–  
473 1092.
- 475 Neal, R. M. (1997). Monte carlo implementation of gaussian process models for bayesian regression  
476 and classification. *arXiv preprint physics/9701026*.
- 477 Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning (Adap-  
478 tive Computation and Machine Learning)*. The MIT Press.
- 479 Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine  
480 learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2951–2959.
- 481 Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view  
482 of the evidence of two samples. *Biometrika*, pages 285–294.

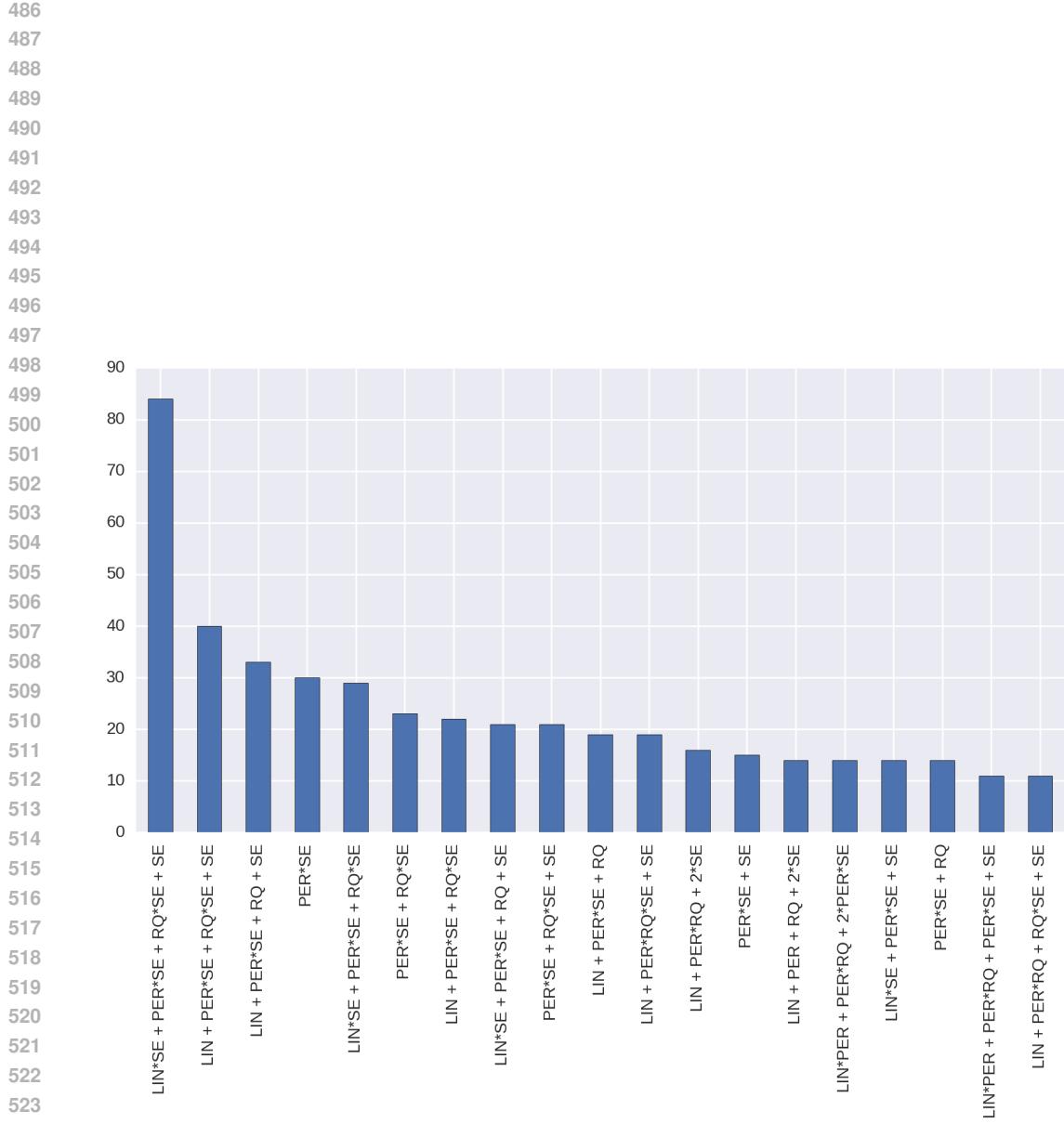


Figure 4: Posterior on structure of the CO<sub>2</sub> data. We have cut the tail of the distribution for space reasons since the number of possible structures is large. We see the final sample of the each of the 545 chains with 2000 nested steps each. Note that Duvenaud et al. (2013) report LIN × SE + PER × SE + RQ × SE.