
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

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 which liberates a user quickly design and test a very broad class of models. The engine builds on a stochastic procedure interface. This stochastic procedure interface specifies and encapsulates primitive random variables analogously to 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). This levitates the design and testing of new models as well as the incorporation of complex prior knowledge which currently is a difficult and time consuming task. 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. We consider Venture the most compelling of the probabilistic programming languages because it is 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 to 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. Making GPs available in probabilistic programming is crucial to allow a language to solve a wide range of problems. GPs have been part of a recent system for inductive learning of symbolic expressions called the Automated Statistician Duvenaud et al. (2013); Lloyd et al. (2014). Learning such expressions is a hard problem that requires careful design of approximation techniques since standard inference method do not apply. In the following, we will present GPs as a novel feature for probabilistic programming languages that solves such problems. 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.

We evaluate the contribution on hard problems posed by the GP community using real world and synthetic data by assessing quality in terms of posterior distributions of symbolic outcome and in terms of the residuals produced by the model. 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. Its implementation is based on 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.

108 **2.2 Gaussian Processes**
109

110 In the following, we will introduce GP related theory and notations. We will exclusively work on
111 two variable regression problems. Let the data be real-valued scalars $\{x_i, y_i\}_{i=1}^n$ (complete data will
112 be denoted by column vectors \mathbf{x}, \mathbf{y}). GPs present a non-parametric way to express prior knowledge
113 on the space of possible functions f that we assume to have generated the data. f is assumed latent
114 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
115 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
116 function that summarizes the covariance of all functions that map to y_i at x_i . We can absorb the
117 mean function into the covariance function so without loss of generality we can set the mean to
118 zero. The marginal likelihood can be expressed as:

119
120
$$p(\mathbf{y}|\mathbf{x}) = \int p(\mathbf{y}|\mathbf{f}, \mathbf{x}) p(\mathbf{f}|\mathbf{x}) d\mathbf{f} \quad (3)$$

121

123 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
124 predictive posterior with

126
$$\mathbf{y}^* \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \quad (4)$$

127

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

130
$$\boldsymbol{\mu} = \mathbf{K}(\mathbf{x}, \mathbf{x}^*) \mathbf{K}(\mathbf{x}^*, \mathbf{x}^*)^{-1} \mathbf{y} \quad (5)$$

131

132 and covariance

134
$$\boldsymbol{\Sigma} = \mathbf{K}(\mathbf{x}, \mathbf{x}) + \mathbf{K}(\mathbf{x}, \mathbf{x}^*) \mathbf{K}(\mathbf{x}^*, \mathbf{x}^*)^{-1} \mathbf{K}(\mathbf{x}^*, \mathbf{x}). \quad (6)$$

135

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

139
$$\log P(\mathbf{y} | \mathbf{X}) = -\frac{1}{2}\mathbf{y}^\top (\mathbf{K} + \sigma^2 \mathbf{I})^{-1} \mathbf{y} - \frac{1}{2} \log |\mathbf{K} + \sigma^2 I| - \frac{n}{2} \log 2\pi \quad (7)$$

140

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

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

150
$$k(x, x') = \sigma^2 \exp\left(-\frac{(x - x')^2}{2\ell^2}\right) \quad (8)$$

151

153 where σ and ℓ are hyper-parameters. σ is a scaling factor and ℓ is the typical length-scale. Smaller
154 variations can be achieved by adapting these hyper-parameters.

157 **3 Venture GPs**
158

160 Given a stochastic process that implements the GP algebra above we can implement a GP sam-
161 pler (4) to perform GP inference in a few lines of code. We can express simple GP smoothing
162 with fixed hyper-parameters and perform MH on it while allowing users to custom design

162 covariance functions. Throughout the paper, we will use the Scheme-like front-end syntax.
 163

164 Listing 1: GP Smoothing

```

165 [ASSUME 1 1] ∈ {hyper-parameters}
166 [ASSUME sf 2] ∈ {hyper-parameters}
167
168  $k(x, x') := \sigma^2 \exp\left(-\frac{(x-x')^2}{2\ell^2}\right)$ 
169
170 [ASSUME f VentureFunction(k, σ, ℓ) ]
171 [ASSUME SE make-se (apply-function f 1 sf) ]
172 [ASSUME (make-gp 0 SE) ]
173
174 [SAMPLE GP (array 1 2 3)] % Prior
175 [OBSERVE GP D]
176 [SAMPLE GP (array 1 2 3)]
177 [INFER (MH {hyper-parameters} one 100) ]
178 [SAMPLE GP (array 1 2 3)] % Posterior

```

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

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

191 In the case where hyper-parameters are unknown they can be found deterministically by optimizing
 192 the marginal likelihood using a gradient based optimizer. Non-deterministic, Bayesian representa-
 193 tions of this case are also known (Neal, 1997) where we draw hyper-parameters from Γ distributions:

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

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

194 Extending the program described in listing 1 to draw the hyper-parameters for a Bayesian treatment
 195 of hyper-parameters is simple using the build in stochastic procedure that simulates drawing samples
 196 from a gamma distribution:
 197

201 Listing 2: Bayesian GP Smoothing

```

202 [ASSUME 1 (gamma 1 3)]
203 [ASSUME sf (gamma 1 2)]
204
205  $k(x, x') := \sigma^2 \exp\left(-\frac{(x-x')^2}{2\ell^2}\right)$ 
206
207 [ASSUME f VentureFunction(k, σ, ℓ) ]
208 [ASSUME SE make-se (apply-function f 1 sf) ]
209 [ASSUME (make-gp 0 SE) ]

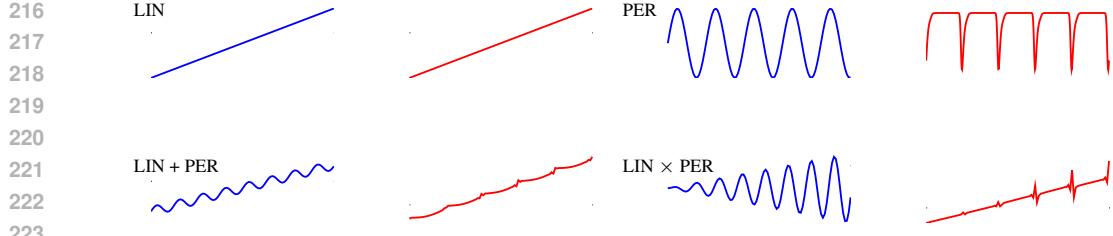
```

210 Larger variations are achieved by changing the type of the covariance function structure. A different
 211 type could be a linear covariance function:
 212

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

213 Note that covariance function structures are compositional. We can add covariance functions if we
 214 want to model globally valid structures
 215

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



216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269

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

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

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

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

Listing 3: LIN \times PER

```
[ASSUME 1 (gamma 1 3)]
[ASSUME sf (gamma 1 2)]
[ASSUME a (gamma 2 2)]

kLIN(x, x') = σ12(x - ℓ)(x' - ℓ)
kPER(x, x') := σ22 exp(- $\frac{2 \sin^2(\pi(x-x')/p)}{\ell^2}$ )

[ASSUME fLIN VentureFunction(kLIN, σ1) ]
[ASSUME fPER VentureFunction(kPER, σ2, ℓ, p) ]
[ASSUME LIN (make-LIN (apply-function fLIN a)) ]
[ASSUME PER (make-PER (apply-function fPER 1 sf)) ]
[ASSUME (make-gp 0 (function-times LIN PER)) ]
```

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

3.1 A Bayesian interpretation

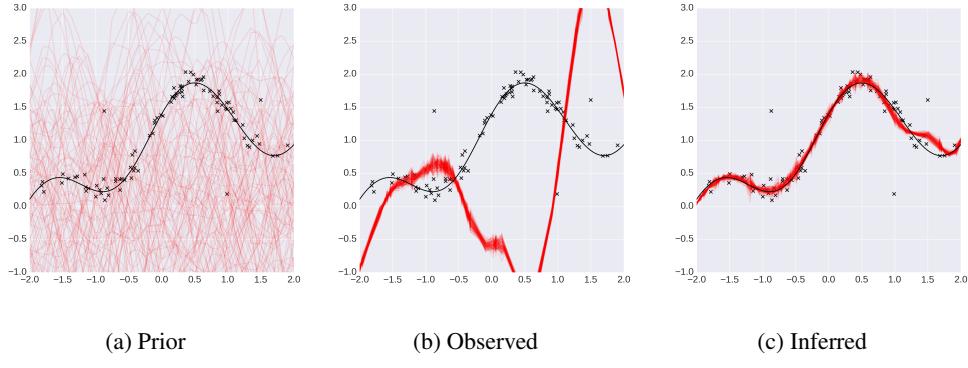
In the following, we will explore a Bayesian representation of GP. The probability of the hyperparameters of a GP with assumptions as above and given covariance function structure \mathbf{K} can be described as:

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

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:

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

We synthetically generate outliers by setting $\sigma = 0.1$ in 95% of the cases 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 number of Hybrid-Monte Carlo and Gibbs steps.



283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
281
282
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.

3.2 Structure Learning

The case where the covariance structure is not given is even more interesting. Our probabilistic programming based MCMC framework approximates the following intractable integrals of the expectation for the prediction:

$$293 \quad \mathbb{E}[y^* | x^*, D, \mathbf{K}_\Omega^s] = \int \int f(x^*, \boldsymbol{\theta}, \mathbf{K}) P(\boldsymbol{\theta} | \mathbf{D}, \mathbf{K}) P(\mathbf{K} | \boldsymbol{\Omega}, s, n) d\boldsymbol{\theta} d\mathbf{K}. \quad (16)$$

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

$$297 \quad 298 \quad 299 \quad y^* \approx \frac{1}{T} \sum_{t=1}^T f(x^* | \boldsymbol{\theta}^{(t)}, \mathbf{K}^{(t)}). \quad (17)$$

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 covariance function algebra:

$$303 \quad \mathbf{K}^{(t)} \sim P(\mathbf{K} | \boldsymbol{\Omega}, s, n) \quad (18)$$

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:

$$311 \quad 312 \quad P(\mathbf{K} | \boldsymbol{\Omega}, s, n) = P(\boldsymbol{\Omega} | s, n) \times P(s | n) \times P(n), \quad (19)$$

313
with

$$314 \quad 315 \quad P(\boldsymbol{\Omega} | s, n) = \binom{n}{r} p_{+\times}^k (1 - p_{+\times})^{n-k} \quad (20)$$

316
and

$$317 \quad 318 \quad P(s | n) = \frac{n!}{|s|!} \quad (21)$$

319
320
321
322
323
324
where $P(n)$ is a prior on the number of base kernels used which can sample from a discrete uniform
distribution. This will strongly prefer simple covariance structures with few base kernels since
individual base kernels are more likely to be sampled in this case due to (21). Alternatively, we
can approximate a uniform prior over structures by weighting $P(n)$ towards higher numbers. It is
possible to also assign a prior for the probability to sample global or local structures, however, we
have assigned complete uncertainty to this with the probability of a flip $p = 0.5$.

324 Many equivalent covariance structures can be sampled due to covariance function algebra
 325 and equivalent representations with different parameterization (Lloyd et al., 2014). Certain
 326 covariance functions can differ in terms of the hyper-parameterization but can be
 327 absorbed into a single covariance function with a different parameterization. To inspect
 328 the posterior of these equivalent structures we convert each kernel expression into
 329 a sum of products and subsequently simplify expressions using the following grammar:
 330

331 Listing 4: Grammar to simplify expressions

332 SE × SE	→ SE
333 {SE, PER, C, WN} × WN	→ WN
334 LIN + LIN	→ LIN
335 {SE, PER, C, WN, LIN} × C	→ {SE, PER, C, WN, LIN}

336 For reproducing results from the Automated Statistician Project in a Bayesian fashion we first define
 337 a prior on the hypothesis space. Note that, as in the implementation of the Automated Statistician,
 338 we upper-bound the complexity of the space of covariance functions we want to explore. We also
 339 put vague priors on hyper-parameters.
 340

341 Listing 5: Venture Code for Bayesian GP Structure Learning

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

347 [OBSERVE GP D]

348 [INFER (REPEAT 2000 (DO
349
350   (MH 10 pn one 1)
351   (MH 10 K* one 1)
352   (MH 10 {hyper-parameters} one 10)) ]
```

353 We defined the space of covariance structures in a way allowing us to reproduce results for covariance
 354 function structure learning as in the Automated Statistician. This lead to coherent results, for
 355 example for the airline data set. We will elaborate the result using a sample from the posterior (Fig.
 356 3). The sample is identical with the highest scoring result reported in previous work using a search-
 357 and-score method (Duvenaud et al., 2013) and the predictive capability is comparable. However, the
 358 components factor in a different way due to different parameterization of the individual base kernels.
 359

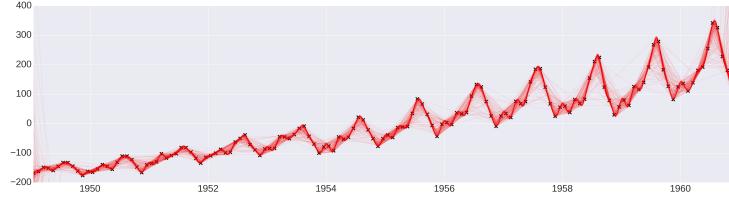
360 We further investigated the quality of our stochastic processes by running a leave one out cross-
 361 validation to gain confidence on the posterior. This resulted in 545 independent runs of the Markov
 362 chain that produced a coherent posterior: our Bayesian interpretation of GP structure and GPs pro-
 363 duced a posterior of structures that is in line with previous results on this data set (Duvenaud et al.,
 364 2013; see Fig. 4).

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

369 4 Bayesian Optimization

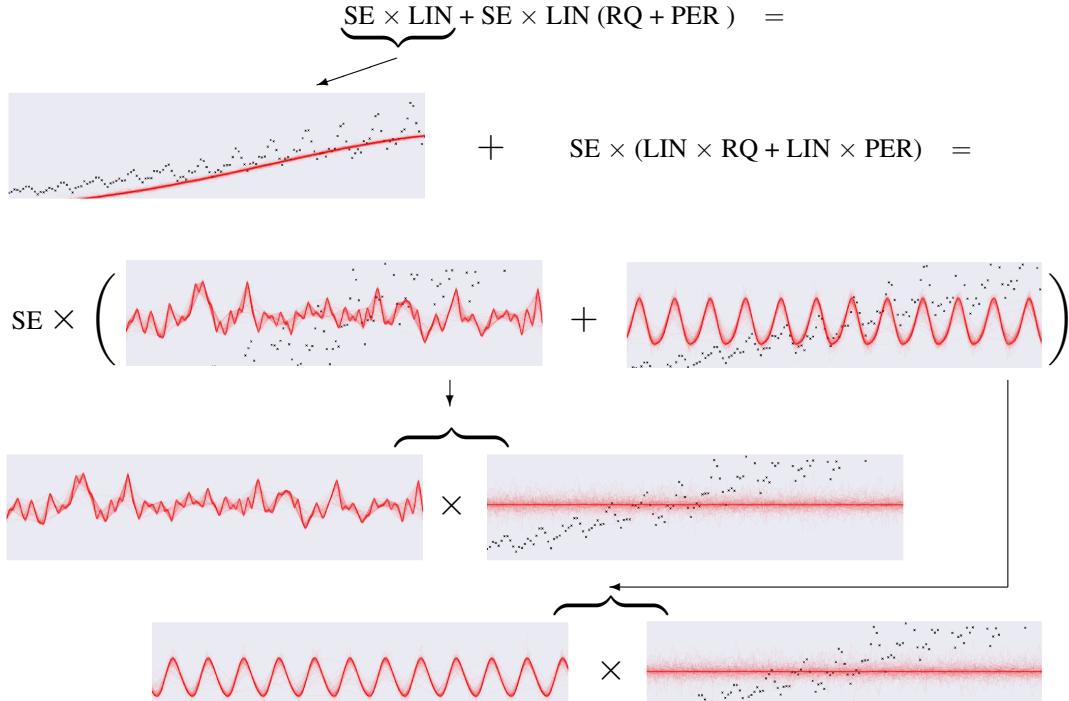
370 Bayesian Optimization poses the problem of finding the global maximum of an unknown function
 371 as a hierarchical decision problem (Ghahramani, 2015). Evaluating the actual function can be
 372 very expensive. For example, finding the best configuration for the learning algorithm of a large
 373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

378
379
380
381
382
383
384
385
386
387
388
389
390



(a) The predictive posterior using the full grammar structure.

391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417



(b) Compositional Structure

418
419
420
421
422
423
424
425
426
427
428
429
430
431

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

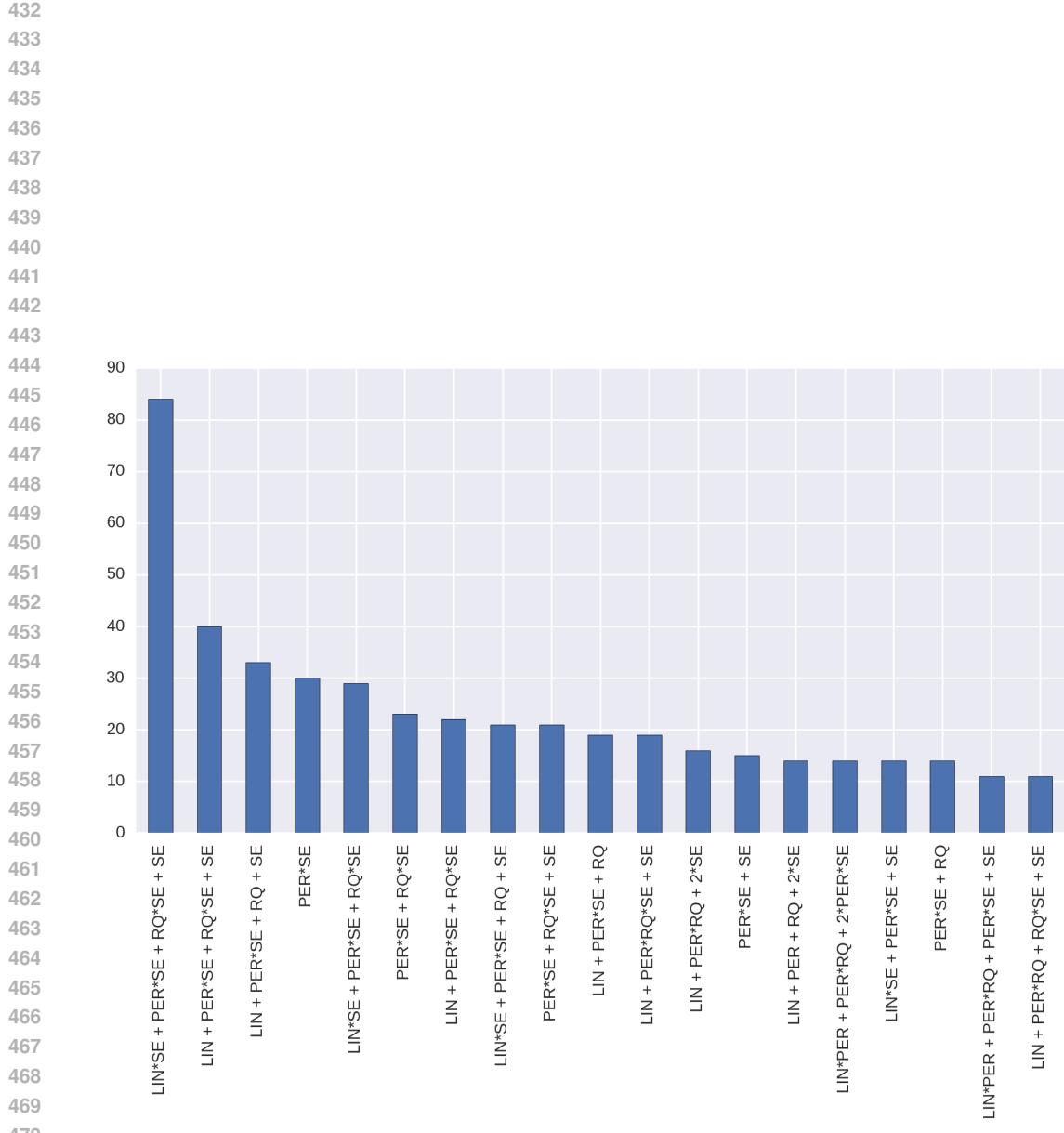


Figure 4: Posterior on structure of the CO₂ data. We have cut the tail of the distribution for space reasons since the number of possible structures is large. We see the final sample of the each of the 545 chains with 2000 nested steps each. Note that Duvenaud et al. (2013) report LIN × SE + PER × SE + RQ × SE.

486 The hierarchical nature of Bayesian Optimization makes it an ideal application for GPs in Venture.
 487 The following Bayesian Optimization scheme is closely related to Thompson Sampling Thompson
 488 (1933). Thompson Sampling is a general framework to solve exploration-exploitation problems that
 489 applies to our notion of Bayesian Optimization. We sample a probe from the posterior

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

490 and then optimize the expected reward by choosing an action a
 491

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

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

498 5 Conclusion

500 501 References

- 502 Andrieu, C., De Freitas, N., Doucet, A., and Jordan, M. I. (2003). An introduction to mcmc for
 503 machine learning. *Machine learning*, 50(1-2):5–43.
- 504 Duvenaud, D., Lloyd, J. R., Grosse, R., Tenenbaum, J., and Ghahramani, Z. (2013). Structure
 505 discovery in nonparametric regression through compositional kernel search. In *Proceedings of*
 506 *the 30th International Conference on Machine Learning (ICML-13)*, pages 1166–1174.
- 507 Ghahramani, Z. (2015). Probabilistic machine learning and artificial intelligence. *Nature*,
 508 521(7553):452–459.
- 509 Lloyd, J. R., Duvenaud, D., Grosse, R., Tenenbaum, J., and Ghahramani, Z. (2014). Automatic
 510 construction and natural-language description of nonparametric regression models. In *Twenty-*
 511 *Eighth AAAI Conference on Artificial Intelligence*.
- 512 Mansinghka, V. K., Selsam, D., and Perov, Y. (2014). Venture: a higher-order probabilistic pro-
 513 gramming platform with programmable inference. *arXiv preprint arXiv:1404.0099*.
- 514 Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation
 515 of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–
 516 1092.
- 517 Neal, R. M. (1997). Monte carlo implementation of gaussian process models for bayesian regression
 518 and classification. *arXiv preprint physics/9701026*.
- 519 Rasmussen, C. E. and Williams, C. K. I. (2006). *Gaussian Processes for Machine Learning (Adap-
 520 tive Computation and Machine Learning)*. The MIT Press.
- 521 Snoek, J., Larochelle, H., and Adams, R. P. (2012). Practical bayesian optimization of machine
 522 learning algorithms. In *Advances in Neural Information Processing Systems*, pages 2951–2959.
- 523 Thompson, W. R. (1933). On the likelihood that one unknown probability exceeds another in view
 524 of the evidence of two samples. *Biometrika*, pages 285–294.

525
 526
 527
 528
 529
 530
 531
 532
 533
 534
 535
 536
 537
 538
 539