Chapter 1

Dropout in Gaussian processes

1.1 Dropout in Gaussian processes

Dropout is a method for regularizing neural networks (**??**). Training with dropout entails randomly and independently "dropping" (setting to zero) some proportion p of features or inputs, in order to improve the robustness of the resulting network by reducing co-dependence between neurons. To maintain similar overall activation levels, weights are multiplied by $1/p$ at test time. Alternatively, feature activations are multiplied by $1/p$ during training. At test time, the set of models defined by all possible ways of dropping-out neurons is averaged over, usually in an approximate way.

? and **?** analyzed dropout in terms of the effective prior induced by this procedure in several models, such as linear and logistic regression. In this section, we examine the priors on functions that result from performing dropout in the one-hidden-layer neural network implicitly defined by a GP (equation (**??**)).

1.1.1 Dropout on feature activations

First, we examine the prior that results from randomly dropping features from $\mathbf{h}(\mathbf{x})$ with probability p. If these features have a weight distribution with finite moments $\mathbb{E}[\alpha_i] = \mu, \mathbb{V}[\alpha_i] = \sigma^2$, then the distribution of weights after each one has been dropped out with probability p is:

$$r_i \overset{\text{iid}}{\sim} \text{Ber}(p) \qquad \mathbb{E}[r_i\alpha_i] = p\mu, \mathbb{V}[r_i\alpha_i] = p^2\sigma^2 \; . \tag{1.1}$$

However, after we increase the remaining activations to maintain the same expected activation by multiplying them by $1/p$, the resulting moments are again:

$$\mathbb{E}\left[\frac{1}{p}r_i\alpha_i\right] = \frac{p}{p}\mu = \mu, \quad \mathbb{V}\left[\frac{1}{p}r_i\alpha_i\right] = \frac{p^2}{p^2}\sigma^2 = \sigma^2 \,. \tag{1.2}$$

Thus, dropping out features of an infinitely-wide MLP does not change the model at all, since no individual feature can have more than infinitesimal contribution to the network output.

1.1.2 Dropping out inputs

In a GP with kernel $k(\mathbf{x}, \mathbf{x}') = \prod_{d=1}^{D} k_d(\mathbf{x}_d, \mathbf{x}'_d)$, exact averaging over all possible ways of dropping out inputs with probability $1/2$ results in a mixture of GPs, each depending on only a subset of the inputs:

$$p\left(f(\mathbf{x})\right) = \frac{1}{2^D} \sum_{\mathbf{r}\in\{0,1\}^D} \mathrm{GP}\left(0, \prod_{d=1}^{D} k_d(\mathbf{x}_d, \mathbf{x}'_d)^{r_d}\right) \tag{1.3}$$

We present two results ways to gain intuition about this model.

First, if the kernel on each dimension has the form $k_d(\mathbf{x}_d, \mathbf{x}'_d) = g\left(\frac{\mathbf{x}_d - \mathbf{x}'_d}{w_d}\right)$, as does the SE kernel (**??**), then any input dimension can be dropped out by setting its lengthscale w_d to ∞. Thus, dropout on the inputs of a GP corresponds to a spike-and-slab prior on lengthscales, with each dimension independently having $w_d = \infty$ with probability $1/2$.

Another way to understand the resulting prior is to note that the dropout mixture (1.3) has the same covariance as

$$f(\mathbf{x}) \sim \mathrm{GP}\left(0, \frac{1}{2^{-2D}} \sum_{\mathbf{r}\in\{0,1\}^D} \prod_{d=1}^{D} k_d(\mathbf{x}_d, \mathbf{x}'_d)^{r_d}\right) \tag{1.4}$$

A GP whose covariance is a sum of kernels corresponds to a sum of functions, each distributed according to a GP. Therefore, (1.4) describes a sum of 2^D functions, each depending on a different subset of the inputs. This model class was studied by **?**, who showed that exact inference in these models can be performed in $\mathcal{O}(N^2D^2 + N^3)$.

In this chapter, we introduce a Gaussian process model of functions which are *additive*. An additive function is one which decomposes into a sum of low-dimensional functions, each depending on only a subset of the input variables. Additive GPs generalize both Generalized Additive Models, and the standard GP models which use squared-

exponential kernels. Hyperparameter learning in this model can be seen as Bayesian Hierarchical Kernel Learning (HKL). We introduce an expressive but tractable parameterization of the kernel function, which allows efficient evaluation of all input interaction terms, whose number is exponential in the input dimension. The additional structure discoverable by this model results in state-of-the-art predictive power in regression tasks, as well as increased interpretability.

1.2 Introduction

Most statistical regression models in use today are of the form: $g(y) = f(x_1) + f(x_2) + \cdots + f(x_D)$. Popular examples include logistic regression, linear regression, and Generalized Linear Models**?**. This family of functions, known as Generalized Additive Models (GAM)**?**, are typically easy to fit and interpret. Some extensions of this family, such as smoothing-splines ANOVA **?**, add terms depending on more than one variable. However, such models generally become intractable and difficult to fit as the number of terms increases.

At the other end of the spectrum are kernel-based models, which typically allow the response to depend on all input variables simultaneously. These have the form: $y = f(x_1, x_2, \ldots, x_D)$. A popular example would be a Gaussian process model using a squared-exponential (or Gaussian) kernel. We denote this model as SE-GP. This model is much more flexible than the GAM, but its flexibility makes it difficult to generalize to new combinations of input variables.

In this paper, we introduce a Gaussian process model that generalizes both GAMs and the SE-GP. This is achieved through a kernel which allow additive interactions of all orders, ranging from first order interactions (as in a GAM) all the way to Dth-order interactions (as in a SE-GP). Although this kernel amounts to a sum over an exponential number of terms, we show how to compute this kernel efficiently, and introduce a parameterization which limits the number of hyperparameters to $O(D)$. A Gaussian process with this kernel function (an additive GP) constitutes a powerful model that allows one to automatically determine which orders of interaction are important. We show that this model can significantly improve modeling efficacy, and has major advantages for model interpretability. This model is also extremely simple to implement, and we provide example code.

We note that a similar breakthrough has recently been made, called Hierarchical Kernel Learning (HKL)**?**. HKL explores a similar class of models, and sidesteps the

possibly exponential number of interaction terms by cleverly selecting only a tractable subset. However, this method suffers considerably from the fact that cross-validation must be used to set hyperparameters. In addition, the machinery necessary to train these models is immense. Finally, on real datasets, HKL is outperformed by the standard SE-GP **?**.

We can see that a GP with a first-order additive kernel is an example of a GAM: Each function drawn from this model is a sum of orthogonal one-dimensional functions. Compared to functions drawn from the higher-order GP, draws from the first-order GP have more long-range structure.

We can expect many natural functions to depend only on sums of low-order interactions. For example, the price of a house or car will presumably be well approximated by a sum of prices of individual features, such as a sun-roof. Other parts of the price may depend jointly on a small set of features, such as the size and building materials of a house. Capturing these regularities will mean that a model can confidently extrapolate to unseen combinations of features.

1.3 Additive Kernels

We now give a precise definition of additive kernels. We first assign each dimension $i \in \{1 \dots D\}$ a one-dimensional *base kernel* $k_i(x_i, x_i')$. We then define the first order, second order and nth order additive kernel as:

$$k_{add_1}(\mathbf{x}, \mathbf{x}') = \sigma_1^2 \sum_{i=1}^{D} k_i(x_i, x_i') \tag{1.5}$$

$$k_{add_2}(\mathbf{x}, \mathbf{x}') = \sigma_2^2 \sum_{i=1}^{D} \sum_{j=i+1}^{D} k_i(x_i, x_i') k_j(x_j, x_j') \tag{1.6}$$

$$k_{add_n}(\mathbf{x}, \mathbf{x}') = \sigma_n^2 \sum_{1 \le i_1 < i_2 < \ldots < i_n \le D} \left[\prod_{d=1}^{n} k_{i_d}(x_{i_d}, x_{i_d}') \right] \tag{1.7}$$

where D is the dimension of our input space, and σ_n^2 is the variance assigned to all nth order interactions. The nth covariance function is a sum of $\binom{D}{n}$ terms. In particular, the Dth order additive covariance function has $\binom{D}{D} = 1$ term, a product of each dimension's covariance function:

$$k_{add_D}(\mathbf{x}, \mathbf{x}') = \sigma_D^2 \prod_{d=1}^{D} k_d(x_d, x_d') \tag{1.8}$$

In the case where each base kernel is a one-dimensional squared-exponential kernel, the Dth-order term corresponds to the multivariate squared-exponential kernel:

$$k_{add_D}(\mathbf{x}, \mathbf{x}') = \sigma_D^2 \prod_{d=1}^{D} k_d(x_d, x'_d) = \sigma_D^2 \prod_{d=1}^{D} \exp\left(-\frac{(x_d - x'_d)^2}{2l_d^2}\right) = \sigma_D^2 \exp\left(-\sum_{d=1}^{D} \frac{(x_d - x'_d)^2}{2l_d^2}\right) \tag{1.9}$$

also commonly known as the Gaussian kernel. The full additive kernel is a sum of the additive kernels of all orders.

1.3.1 Parameterization

The only design choice necessary in specifying an additive kernel is the selection of a one-dimensional base kernel for each input dimension. Any parameters (such as length-scales) of the base kernels can be learned as usual by maximizing the marginal likelihood of the training data.

In addition to the hyperparameters of each dimension-wise kernel, additive kernels are equipped with a set of D hyperparameters $\sigma_1^2 \dots \sigma_D^2$ controlling how much variance we assign to each order of interaction. These "order variance" hyperparameters have a useful interpretation: The dth order variance hyperparameter controls how much of the target function's variance comes from interactions of the dth order. Table 1.1 shows examples of normalized order variance hyperparameters learned on real datasets.

Table 1.1 Relative variance contribution of each order in the additive model, on different datasets. Here, the maximum order of interaction is set to 10, or smaller if the input dimension less than 10. Values are normalized to sum to 100.

Order of interaction	1st	2nd	3rd	4th	5th	6th	7th	8th	9th	10th
pima	0.1	0.1	0.1	0.3	1.5	**96.4**	1.4	0.0		
liver	0.0	0.2	**99.7**	0.1	0.0	0.0				
heart	**77.6**	0.0	0.0	0.0	0.1	0.1	0.1	0.1	0.1	22.0
concrete	**70.6**	13.3	13.8	2.3	0.0	0.0	0.0	0.0		
pumadyn-8nh	0.0	0.1	0.1	0.1	0.1	0.1	0.1	**99.5**		
servo	**58.7**	27.4	0.0	13.9						
housing	0.1	0.6	**80.6**	1.4	1.8	0.8	0.7	0.8	0.6	12.7

On different datasets, the dominant order of interaction estimated by the additive model varies widely. An additive GP with all of its variance coming from the 1st order

is equivalent to a GAM; an additive GP with all its variance coming from the Dth order is equivalent to a SE-GP.

Because the hyperparameters can specify which degrees of interaction are important, the additive GP is an extremely general model. If the function we are modeling is decomposable into a sum of low-dimensional functions, our model can discover this fact and exploit it (see Figure **??**) . If this is not the case, the hyperparameters can specify a suitably flexible model.

1.3.2 Interpretability

As noted by **?**, one of the chief advantages of additive models such as GAM is their interpretability. Plate also notes that by allowing high-order interactions as well as low-order interactions, one can trade off interpretability with predictive accuracy. In the case where the hyperparameters indicate that most of the variance in a function can be explained by low-order interactions, it is useful and easy to plot the corresponding low-order functions, as in Figure 1.2.

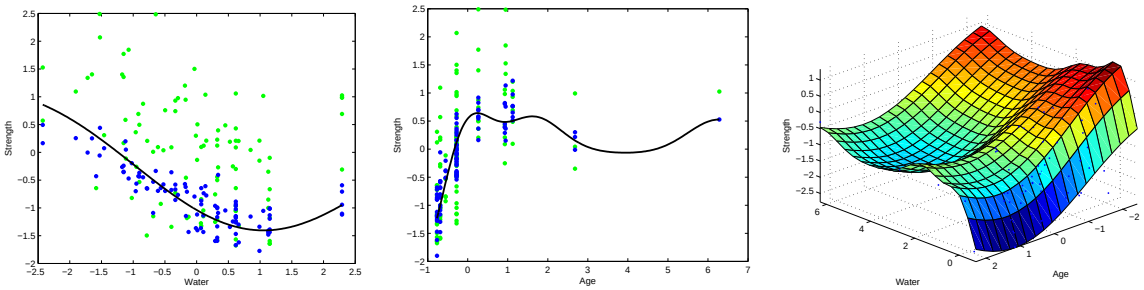


Fig. 1.1 Low-order functions on the concrete dataset. Left, Centre: By considering only first-order terms of the additive kernel, we recover a form of Generalized Additive Model, and can plot the corresponding 1-dimensional functions. Green points indicate the original data, blue points are data after the mean contribution from the other dimensions' first-order terms has been subtracted. The black line is the posterior mean of a GP with only one term in its kernel. Right: The posterior mean of a GP with only one second-order term in its kernel.

1.3.3 Efficient Evaluation of Additive Kernels

An additive kernel over D inputs with interactions up to order n has $O(2^n)$ terms. Naïvely summing over these terms quickly becomes intractable. In this section, we show how one can evaluate the sum over all terms in $O(D^2)$.

The nth order additive kernel corresponds to the nth *elementary symmetric polynomial*? ?, which we denote e_n. For example: if $\mathbf{x}$ has 4 input dimensions ($D = 4$), and if we let $z_i = k_i(x_i, x_i')$, then

$$
\begin{aligned}
k_{add_1}(\mathbf{x}, \mathbf{x}') &= e_1(z_1, z_2, z_3, z_4) = z_1 + z_2 + z_3 + z_4 \\
k_{add_2}(\mathbf{x}, \mathbf{x}') &= e_2(z_1, z_2, z_3, z_4) = z_1 z_2 + z_1 z_3 + z_1 z_4 + z_2 z_3 + z_2 z_4 + z_3 z_4 \\
k_{add_3}(\mathbf{x}, \mathbf{x}') &= e_3(z_1, z_2, z_3, z_4) = z_1 z_2 z_3 + z_1 z_2 z_4 + z_1 z_3 z_4 + z_2 z_3 z_4 \\
k_{add_4}(\mathbf{x}, \mathbf{x}') &= e_4(z_1, z_2, z_3, z_4) = z_1 z_2 z_3 z_4
\end{aligned}
$$

The Newton-Girard formulae give an efficient recursive form for computing these polynomials. If we define s_k to be the kth power sum: $s_k(z_1, z_2, \ldots, z_D) = \sum_{i=1}^{D} z_i^k$, then

$$
k_{add_n}(\mathbf{x}, \mathbf{x}') = e_n(z_1, \ldots, z_D) = \frac{1}{n} \sum_{k=1}^{n} (-1)^{(k-1)} e_{n-k}(z_1, \ldots, z_D) s_k(z_1, \ldots, z_D) \tag{1.10}
$$

Where $e_0 \triangleq 1$. The Newton-Girard formulae have time complexity $O(D^2)$, while computing a sum over an exponential number of terms.

Conveniently, we can use the same trick to efficiently compute all of the necessary derivatives of the additive kernel with respect to the base kernels. We merely need to remove the kernel of interest from each term of the polynomials:

$$
\frac{\partial k_{add_n}}{\partial z_j} = e_{n-1}(z_1, \ldots, z_{j-1}, z_{j+1}, \ldots z_D) \tag{1.11}
$$

This trick allows us to optimize the base kernel hyperparameters with respect to the marginal likelihood.

1.3.4 Computation

The computational cost of evaluating the Gram matrix of a product kernel (such as the SE kernel) is $O(N^2 D)$, while the cost of evaluating the Gram matrix of the additive kernel is $O(N^2 DR)$, where R is the maximum degree of interaction allowed (up to D). In higher dimensions, this can be a significant cost, even relative to the fixed $O(N^3)$ cost of inverting the Gram matrix. However, as our experiments show, typically only the first few orders of interaction are important for modeling a given function; hence if one is computationally limited, one can simply limit the maximum degree of interaction

without losing much accuracy.

Additive Gaussian processes are particularly appealing in practice because their use requires only the specification of the base kernel. All other aspects of GP inference remain the same. All of the experiments in this paper were performed using the standard GPML toolbox[1]; code to perform all experiments is available at the author's website.[2]

1.4 Related Work

Plate**?** constructs a form of additive GP, but using only the first-order and Dth order terms. This model is motivated by the desire to trade off the interpretability of first-order models, with the flexibility of full-order models. Our experiments show that often, the intermediate degrees of interaction contribute most of the variance.

A related functional ANOVA GP model**?** decomposes the *mean* function into a weighted sum of GPs. However, the effect of a particular degree of interaction cannot be quantified by that approach. Also, computationally, the Gibbs sampling approach used in **?** is disadvantageous.

Christoudias et al.**?** previously showed how mixtures of kernels can be learnt by gradient descent in the Gaussian process framework. They call this *Bayesian localized multiple kernel learning*. However, their approach learns a mixture over a small, fixed set of kernels, while our method learns a mixture over all possible products of those kernels.

1.4.1 Hierarchical Kernel Learning

Bach**?** uses a regularized optimization framework to learn a weighted sum over an exponential number of kernels which can be computed in polynomial time. The subsets of kernels considered by this method are restricted to be a *hull* of kernels.[3] Given each dimension's kernel, and a pre-defined weighting over all terms, HKL performs model selection by searching over hulls of interaction terms. In **?**, Bach also fixes the relative weighting between orders of interaction with a single term α, computing the sum over

[1] Available at `http://www.gaussianprocess.org/gpml/code/`

[2] Example code available at: `http://mlg.eng.cam.ac.uk/duvenaud/`

[3] In the setting we are considering in this paper, a hull can be defined as a subset of all terms such that if term $\prod_{j \in J} k_j(\mathbf{x}, \mathbf{x}')$ is included in the subset, then so are all terms $\prod_{j \in J/i} k_j(\mathbf{x}, \mathbf{x}')$, for all $i \in J$. For details, see **?**.

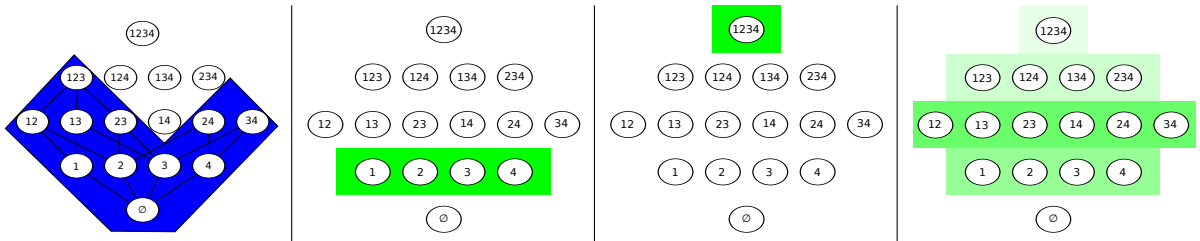


Fig. 1.2 A comparison of different additive model classes. Nodes represent different interaction terms, ranging from first-order to fourth-order interactions. Far left: HKL can select a hull of interaction terms, but must use a pre-determined weighting over those terms. Far right: the additive GP model can weight each order of interaction seperately. Neither the HKL nor the additive model dominate one another in terms of flexibility, however the GP-GAM and the SE-GP are special cases of additive GPs.

all orders by:

$$k_a(\mathbf{x}, \mathbf{x}') = v_D^2 \prod_{d=1}^{D} (1 + \alpha k_d(x_d, x'_d)) \tag{1.12}$$

which has computational complexity $O(D)$. However, this formulation forces the weight of all nth order terms to be weighted by α^n.

Figure 1.3 contrasts the HKL hull-selection method with the Additive GP hyperparameter-learning method. Neither method dominates the other in flexibility. The main difficulty with the approach of **?** is that hyperparameters are hard to set other than by cross-validation. In contrast, our method optimizes the hyperparameters of each dimension's base kernel, as well as the relative weighting of each order of interaction.

1.4.2 ANOVA Procedures

Vapnik **?** introduces the support vector ANOVA decomposition, which has the same form as our additive kernel. However, they recommend approximating the sum over all D orders with only one term "of appropriate order", presumably because of the difficulty of setting the hyperparameters of an SVM. Stitson et al.**?** performed experiments which favourably compared the support vector ANOVA decomposition to polynomial and spline kernels. They too allowed only one order to be active, and set hyperparameters by cross-validation.

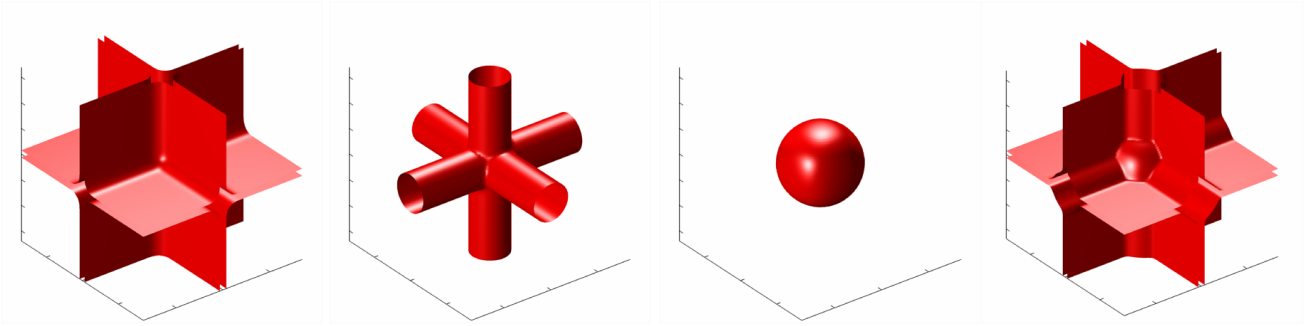
A closely related procedure from the statistics literature is smoothing-splines ANOVA (SS-ANOVA)**?**. An SS-ANOVA model is estimated as a weighted sum of splines along

each dimension, plus a sum of splines over all pairs of dimensions, all triplets, etc, with each individual interaction term having a separate weighting parameter. Because the number of terms to consider grows exponentially in the order, in practice, only terms of first and second order are usually considered. Learning in SS-ANOVA is usually done via penalized-maximum likelihood with a fixed sparsity hyperparameter.

In contrast to these procedures, our method can easily include all D orders of interaction, each weighted by a separate hyperparameter. As well, we can learn kernel hyperparameters individually per input dimension, allowing automatic relevance determination to operate.

1.4.3 Non-local Interactions

By far the most popular kernels for regression and classification tasks on continuous data are the squared exponential (Gaussian) kernel, and the Matérn kernels. These kernels depend only on the scaled Euclidean distance between two points, both having the form: $k(\mathbf{x}, \mathbf{x}') = f(\sum_{d=1}^{D} (x_d - x'_d)^2 / l_d^2)$. Bengio et al.**?** argue that models based on squared-exponential kernels are particularily susceptible to the *curse of dimensionality*. They emphasize that the locality of the kernels means that these models cannot capture non-local structure. They argue that many functions that we care about have such structure. Methods based solely on local kernels will require training examples at all combinations of relevant inputs.



1st order interactions $k_1 + k_2 + k_3$ | 2nd order interactions $k_1k_2 + k_2k_3 + k_1k_3$ | 3rd order interactions $k_1k_2k_3$ (Squared-exp kernel) | All interactions (Additive kernel)

Fig. 1.3 Isocontours of additive kernels in 3 dimensions. The third-order kernel only considers nearby points relevant, while the lower-order kernels allow the output to depend on distant points, as long as they share one or more input value.

Additive kernels have a much more complex structure, and allow extrapolation based

on distant parts of the input space, without spreading the mass of the kernel over the whole space. For example, additive kernels of the second order allow strong non-local interactions between any points which are similar in any two input dimensions. Figure 1.4 provides a geometric comparison between squared-exponential kernels and additive kernels in 3 dimensions.

1.5 Experiments

1.5.1 Experimental Setup

On a diverse collection of datasets, we compared five different models. In the results tables below, GP Additive refers to a GP using the additive kernel with squared-exp base kernels. For speed, we limited the maximum order of interaction in the additive kernels to 10. GP-GAM denotes an additive GP model with only first-order interactions. GP Squared-Exp is a GP model with a squared-exponential ARD kernel. HKL[4] was run using the all-subsets kernel, which corresponds to the same set of kernels as considered by the additive GP with a squared-exp base kernel.

For all GP models, we fit hyperparameters by the standard method of maximizing training-set marginal likelihood, using L-BFGS **?** for 500 iterations, allowing five random restarts. In addition to learning kernel hyperparameters, we fit a constant mean function to the data. In the classification experiments, GP inference was done using Expectation Propagation **?**.

1.5.2 Bach Synthetic Dataset

In addition to standard UCI repository datasets, we generated a synthetic dataset following the same recipe as **?**: From a covariance matrix drawn from a Wishart distribution with 1024 degrees of freedom, we select 8 variables. We then construct the non-linear function $f(X) = \sum_{i=1}^{4} \sum_{j=1+1}^{4} X_i X_j + \epsilon$, which sums all 2-way products of the first 4 variables, and adds Gaussian noise ϵ. This dataset is one which can be predicted well by a kernel which is a sum of two-way interactions over the first 4 variables, ignoring the extra 4 noisy copies.

This dataset was designed by **?** to demonstrate the advantages of HKL over GP-ARD.

[4]Code for HKL available at `http://www.di.ens.fr/~fbach/hkl/`

If the dataset is large enough, HKL can construct a hull around only those subsets of cross terms that are optimal for predicting the output. GP-ARD, in contrast, can only learn to ignore the noisy copy variables, but cannot learn to ignore the higher-term interactions between the predictive variables. However, a GP with an additive kernel can learn both to ignore irrelevant variables, and to ignore certain orders of interaction. In this example, the additive GP is able to recover the relevant structure.

1.5.3 Results

Tables 1.2, 1.3, 1.4 and 1.5 show mean performance across 10 train-test splits. Because HKL does not specify a noise model, it could not be included in the likelihood comparisons.

Table 1.2 Regression Mean Squared Error

Method	bach	concrete	pumadyn-8nh	servo	housing
Linear Regression	1.031	0.404	0.641	0.523	0.289
GP GAM	1.259	0.149	0.598	0.281	0.161
HKL	**0.199**	0.147	0.346	0.199	0.151
GP Squared-exp	**0.045**	0.157	**0.317**	**0.126**	**0.092**
GP Additive	**0.045**	**0.089**	**0.316**	**0.110**	**0.102**

Table 1.3 Regression Negative Log Likelihood

Method	bach	concrete	pumadyn-8nh	servo	housing
Linear Regression	2.430	1.403	1.881	1.678	1.052
GP GAM	1.708	0.467	1.195	0.800	0.457
GP Squared-exp	**−0.131**	0.398	**0.843**	0.429	**0.207**
GP Additive	**−0.131**	**0.114**	**0.841**	**0.309**	**0.194**

The model with best performance on each dataset is in bold, along with all other models that were not significantly different under a paired t-test. The additive model never performs significantly worse than any other model, and sometimes performs significantly better than all other models. The difference between all methods is larger in the case of regression experiments. The performance of HKL is consistent with the results in **?**, performing competitively but slightly worse than SE-GP.

Table 1.4 Classification percent error

Method	breast	pima	sonar	ionosphere	liver	heart
Logistic Regression	7.611	24.392	26.786	16.810	45.060	**16.082**
GP GAM	**5.189**	**22.419**	**15.786**	**8.524**	**29.842**	**16.839**
HKL	**5.377**	24.261	**21.000**	9.119	**27.270**	**18.975**
GP Squared-exp	**4.734**	**23.722**	**16.357**	**6.833**	**31.237**	**20.642**
GP Additive	**5.566**	**23.076**	**15.714**	**7.976**	**30.060**	**18.496**

Table 1.5 Classification negative log-likelihood

Method	breast	pima	sonar	ionosphere	liver	heart
Logistic Regression	0.247	0.560	4.609	0.878	0.864	0.575
GP GAM	**0.163**	**0.461**	**0.377**	**0.312**	**0.569**	**0.393**
GP Squared-exp	**0.146**	0.478	**0.425**	**0.236**	**0.601**	0.480
GP Additive	**0.150**	**0.466**	**0.409**	**0.295**	**0.588**	**0.415**

The additive GP performed best on datasets well-explained by low orders of interaction, and approximately as well as the SE-GP model on datasets which were well explained by high orders of interaction (see table 1.1). Because the additive GP is a superset of both the GP-GAM model and the SE-GP model, instances where the additive GP performs slightly worse are presumably due to over-fitting, or due to the hyperparameter optimization becoming stuck in a local maximum. Additive GP performance can be expected to benefit from integrating out the kernel hyperparameters.

1.5.4 Future Work

Since the non-local structure capturable by additive kernels is necessarily axis-aligned, we can naturally consider that combining the hyperparameter optimization with an initial rotation of the input space might allow us to recover non-axis aligned additivity in functions.

Note that we are free to choose a different covariance function along each dimension.

1.6 Conclusion

We present additive Gaussian processes: a simple family of models which generalizes two widely-used classes of models. Additive GPs also introduce a tractable new type of structure into the GP framework. Our experiments indicate that such additive structure is present in real datasets, allowing our model to perform better than standard GP models. In the case where no such structure exists, our model can recover arbitrarily flexible models, as well.

In addition to improving modeling efficacy, the additive GP also improves model interpretability: the order variance hyperparameters indicate which sorts of structure are present in our model.

Compared to HKL, which is the only other tractable procedure able to capture the same types of structure, our method benefits from being able to learn individual kernel hyperparameters, as well as the weightings of different orders of interaction. Our experiments show that additive GPs are a state-of-the-art regression model.