A Alternative strategies for specifying weights

In section 2.2 one specific choice for specifying the weights $\boldsymbol{\alpha}$ in feature engineering was introduced where weights are obtained by optimizing (5). The corresponding strategy might be abbreviated as 'optimize then transform', because the non-linear transformation happens after the weights have been computed. Here we present three alternative strategies of increasing computational complexity.

Strategy 2 (transform then optimize): Like in the original strategy the weights $\boldsymbol{\alpha}$ are specified conditional on the $F_{r_l}(\boldsymbol{x})$ terms defined at earlier steps but now optimization happens after applying the transformation $g(\cdot)$. In other words the weights are obtained as maximum likelihood estimates using model (1) with $F_{r_l}, r_l = 1, ..., w_j$ as covariates and $g^{-1}(\mathsf{h}(\cdot))$ as a link function, thus fitting the model $g^{-1}(\mathsf{h}(\mu)) = \boldsymbol{\alpha}_j^T \mathbf{F}^d(\boldsymbol{x}) + \alpha_{j,0}$. This strategy yields a particularly simple optimization problem if h is the canonical link function and $g(\cdot)$ a concave function in which case the estimates are uniquely defined. However, if we want to use gradient based optimizers then we have to make a restriction on $g(\cdot)$ to be continuous and differentiable in the regions of interest. Otherwise gradient free continuous optimization techniques have to be applied.

Strategy 3 (transform then optimize across all layers): Similarly as in Strategy 2 parameters are obtained as maximum likelihood estimates using model (1) but we include now parameters from all layers as covariates. We are again fitting the model $g^{-1}(\mathsf{h}(\mu)) = \boldsymbol{\alpha}_j^T \mathbf{F}^d(\boldsymbol{x}) + \alpha_{j,0}$, but now the optimization is performed with respect to parameters across all layers. There has to be made the same restrictions on $g(\cdot)$ to be continuous and differentiable in the regions of interest as in Strategy 2 if one wants to use gradient based optimizers. One drawback of this strategy is that now there is no guarantee to find a unique global optimum of the likelihood of the feature, even if $g(\cdot)$ is concave. If gradient free optimizers are used the problem becomes computationally extremely demanding given the difficulty of the optimization problem. Furthermore different local optima define different features having structurally the same configuration and hence the topology of the feature space is getting more complex.

Strategy 4 (fully Bayesian): All parameters across all layers are drawn from a prior distributions (in the implementation we used $N(0, 1)$). There are no restrictions on the nonlinear transformations, only the link function needs to be differentiable. The problem with this strategy is that to get the posterior mode a rather high-dimensional integral has to be solved. The probability of getting close to the mode is extremely low and the convergence requires typically a huge number of iterations. This might be improved by drawing around the modes obtained by the previously suggested strategies, but to develop this idea is a topic of further research. Just like the 3rd strategy, all different values of the vector of parameters will define different features. With this strategy the joint space of configurations and parameters is (at least in principle) systematically explored, which is extremely demanding computationally.

In Tables 7-9 the predictive performance of these strategies is compared for the NEO asteroids classification problem (Example 1), the breast cancer data (Example 2), and the spam data (Example 3). Comparing Table 7 with Table 1, Table 8 with Table 2 and Table 9 with Table 3, we see that there is no substantial difference in predictive performance between the strategies used for specifying weights.

Table 7: Comparison of performance (ACC, FPR, FNR) of alternative feature engineer strategies (indicated with _2, _3, _4 in the table) for Example 1. For methods with random outcome the median measures (with minimum and maximum in parentheses) are displayed. The algorithms are sorted according to median power.

Algorithm	ACC	FNR	FPR
DBRM_G_3	0.9998 (0.9959,1.0000)	0.0002 (0.0001,0.0056)	0.0002 (0.0000,0.0042)
DBRM_R_3	0.9998 (0.9953,1.0000)	0.0002 (0.0001,0.0068)	0.0002 (0.0000,0.0070)
DBRM_R_4	0.9998 (0.9945,1.0000)	0.0002 (0.0001,0.0080)	0.0002 (0.0000,0.0069)
DBRM_G_2	0.9998 (0.9933,1.0000)	0.0002 (0.0001,0.0089)	0.0002 (0.0000,0.0048)
DBRM_G_4	0.9998 (0.9932,0.9999)	0.0002 (0.0001,0.0097)	0.0002 (0.0000,0.0042)
DBRM_R_2	0.9998 (0.9925,1.0000)	0.0002 (0.0001,0.0105)	0.0002 (0.0000,0.0032)

Table 8: Comparison of performance (ACC, FPR, FNR) of alternative feature engineer strategies for Example 2. See caption of Table 7 for details.

Algorithm	ACC	FNR	FPR
DBRM_G_3	0.9695 (0.9507,0.9789)	0.0536 (0.0479,0.0862)	0.0148 (0.0000,0.0361)
DBRM_R_2	0.9695 (0.9554,0.9789)	0.0536 (0.0422,0.0756)	0.0148 (0.0000,0.0396)
DBRM_R_4	0.9695 (0.9577,0.9789)	0.0536 (0.0479,0.0756)	0.0148 (0.0000,0.0361)
DBRM_R_3	0.9671 (0.9577,0.9789)	0.0536 (0.0422,0.0756)	0.0148 (0.0037,0.0361)
DBRM_G_4	0.9671 (0.9577,0.9789)	0.0536 (0.0305,0.0756)	0.0184 (0.0000,0.0361)
DBRM_G_2	0.9671 (0.9531,0.9789)	0.0536 (0.0422,0.0862)	0.0184 (0.0000,0.0361)

Table 9: Comparison of performance (ACC, FPR, FNR) of alternative feature engineer strategies for Example 3. See caption of Table 7 for details.

Algorithm	ACC	FNR	FPR
DBRM_G_2	0.9243 (0.9100,0.9357)	0.0927 (0.0780,0.1103)	0.0545 (0.0445,0.0686)
DBRM_G_3	0.9237 (0.9100,0.9321)	0.0924 (0.0766,0.1122)	0.0548 (0.0474,0.0714)
DBRM_G_4	0.9237 (0.9113,0.9315)	0.0931 (0.0821,0.1077)	0.0562 (0.0470,0.0714)
DBRM_R_3	0.9240 (0.9132,0.9334)	0.0951 (0.0752,0.1155)	0.0552 (0.0465,0.0672)
DBRM_R_2	0.9240 (0.9132,0.9321)	0.0917 (0.0801,0.1142)	0.0550 (0.0465,0.0676)
DBRM_R_4	0.9237 (0.9109,0.9341)	0.0931 (0.0787,0.1096)	0.0562 (0.0455,0.0686)

B Interpretability of DBRM results

The key feature of DBRM which allows to obtain interpretable models is that there is a whole set $\mathcal{G}$ of non-linear transformations and hence feature engineering becomes highly flexible. To illustrate the importance of the choice of $\mathcal{G}$ we have reanalyzed Example 5 on Kepler's third law with DBRM_G_1_PAR_64 using only the sigmoid function as non-linear transformation and considering different restrictions on the search space:

1. $\mathcal{G} = \{\text{sigmoid}(x)\}$, $D_{max} = 5$;
2. $\mathcal{G} = \{\text{sigmoid}(x)\}$, $D_{max} = 300$, and $P_c = 0$;
3. $\mathcal{G} = \{\text{sigmoid}(x)\}$, $D_{max} = 300$, and $P_c = 0$ and $p(\gamma_j) \propto 1$.

Clearly for these settings it is not possible to obtain the correct model in closed form, but according to the universal approximation theorem (Hornik 1991) Kepler's 3rd law can still be well approximated. In the first setting the true model is infeasible since the cubic root function is not a part of $\mathcal{G}$ but at least multiplication of features via the crossover operator is still possible. In the second setting crossovers are not allowed but on the other hand there is no longer any real restriction on the depth of features. Finally in the third setting all features get a uniform prior in the feature space. As a consequence from this lack of regularization we expect that highly complex features are generated.

Table 10 illustrates the effects of making these changes in the DBRM setting on the interpretability of models by reporting the ten most frequently detected features over $N = 100$ simulations. To simplify the reporting we denote *TypeFlag*, *RadiusJpt*, *PeriodDays*, *PlanetaryMassJpt*, *Eccentricity*, *HostStarMassSlrMass*, *HostStarRadiusSlrRad*, *HostStarMetallicity*, *HostStarTempK*, *PlanetaryDensJpt* as x_1-x_{10}, correspondingly, and use the symbol σ for the sigmoid function.

Table 10: 10 most frequent features detected under Settings 1, 2 and 3

Setting 1		Setting 2		Setting 3	
Fq	Feature	Fq	Feature	Fq	Feature
99	x_3	100	x_3	100	x_3
98	x_3*x_3	72	σ(-10.33+0.24x_4-8.83x_8)	54	x_2
93	x_3*x_{10}	64	x_{10}	21	σ(-16.91-4.94x_2)
4	x_3*x_3*x_{10}	62	x_2	19	x_9
1	x_9*x_3	16	σ(0.21+0.01x_3+0.20x_7)	16	x_5
1	x_9*x_3*x_3	9	x_4	14	x_{10}
1	x_{10}*x_{10}*x_3	7	σ(-13.11-7.76x_8-3.33x_2+0.40x_{10})	10	σ(6.88×10^9-3.92x_2+ 3.44×$10^9$$\sigma$(-13.57-0.17$x_4$- 2.84$x_2-7.66x_8$+0.54$x_{10}$) -13.76×$10^9$$\sigma$($\sigma$(-13.57- 0.17$x_4-2.84x_2-7.66x_8$+ 0.54$x_{10}$)))
1	x_7*x_3*x_3	5	σ(-3.36+2.83x_3+0.21x_3-3.36x_9)	9	x_4
1	x_6*x_3*x_3	3	σ(σ(-10.33+0.24x_4)-8.83x_8)	8	σ(-13.57-0.17x_4- 2.84x_2-7.66x_8+0.54x_{10})
1	x_3*x_3*x_3	3	σ(0.15+0.05x_4-0.01x_3+0.15x_7)	7	σ(0.21+0.21x_3)
0	Others	4	Others	$>$ 300	Others

The results shown in Table 10 are not too surprising. Restricting the set of non-linear transformations results in increasingly more complex features. Using Setting 1 there is not a single occurrence of a sigmoid function while in Setting 2 the feature σ(-10.33+0.24x_4-8.83x_8) is selected in almost 3 out of 4 runs. Removing the complexity penalty in Setting 3 yields highly complex features which are however no longer that much replicable over simulation runs.

The general conclusion is that more flexible sets of non-linear transformations $\mathcal{G}$ provide the possibility to obtain interpretable models which have similar predictive performance than complex models based on a less flexible set of transformations. Problems with the latter approach include potential overfitting, substantially more need of memory and computational effort (if one for instance is interested in predictions). In contrast DBRM will often construct architectures that reach state of the art performance in terms of prediction and still remain relatively simple, hence representing sophisticated phenomena in a fairly parsimonious way.

C Further applications

C.1 Example 6: Simulated data with complex combinatorial structures

In this simulation study we generated $N = 100$ datasets with $n = 1000$ observations and $p = 50$ binary covariates. The covariates were assumed to be independent and were simulated for each simulation run as $X_j \sim \text{Bernoulli}(0.5)$ for $j \in= 1, \ldots, 50$. In the first simulation study the responses were simulated according to a Gaussian distribution with error variance $\sigma^2 = 1$ and individual expectations specified as follows:

$$\begin{aligned} E(Y) &= 1 + 1.5X_7 + 1.5X_8 + 6.6X_{18} * X_{21} + 3.5X_2 * X_9 + 9X_{12} * X_{20} * X_{37} \\ &+ 7X_1 * X_3 * X_{27} + 7X_4 * X_{10} * X_{17} * X_{30} + 7X_{11} * X_{13} * X_{19} * X_{50} \end{aligned}$$

We compare the results of GMJMCMC, RGMJMCMC for DBRM with the Bayesian logic regression model in Hubin et al. (2018). The latter model differs from the current one in that the model prior is different. For a given logical tree (which is the only allowed feature form) we use $a^{c(L_j)} = \frac{1}{N(s_j)}\ , \quad s_j \leq C_{max}$, where $N(s_j) = \binom{m}{s_j} 2^{2s_j - 2}$. Q and priors for the model parameters are the same as defined in DBRM model. All algorithms were run on 32 threads until the same number of models were visited after the last change of the model space. In particular, in each of the threads the algorithms were run until 20000 unique models were obtained after the last population of models had been generated at iteration 15000. Specification of the Bayesian Logic Regression model corresponds exactly to the one used in simulation Scenario 6 in Hubin et al. (2018). In this example a detected feature is only counted as a true positive if it exactly coincides with a feature of the data generating model. The results are summarized in Table 11. Detection in this example corresponds to the features having marginal inclusion probabilities above $\eta^* = 0.5$ after the search is completed.

Both GMJMCMC and RGMJMCMC performed exceptionally well for fitting this DBRM with slight advantages of the former. The original GMJMCMC(LR) algorithm for fitting Bayesian Logic Regression in this case performed almost as well as GMJMCMC and RGMJMCMC, except for a significant drop in power in one of the four-way interactions. This is however not too surprising because the crossover operator of DBRM models perfectly fits the data generating model whereas the logic regression model focuses on general logic expressions and provides in that sense a larger chance to generate features which are closely related to the data generating four-way interaction (Hubin et al. 2018).

Table 11: Results for Example 6. Power for individual trees, overall power (average power over trees), expected number of false positives (FP), and false discovery rate (FDR) are compared between GMJMCMC, RGMJMCMC and Bayesian Logic regression.

	DBRM_G	DBRM_R	Bayesian Logic regression
X_7	1.0000	1.0000	0.9900
X_8	1.0000	1.0000	1.0000
$X_2 * X_9$	1.0000	0.9600	1.0000
$X_{18} * X_{21}$	1.0000	1.0000	0.9600
$X_1 * X_3 * X_{27}$	1.0000	1.0000	1.0000
$X_{12} * X_{20} * X_{37}$	1.0000	1.0000	0.9900
$X_4 * X_{10} * X_{17} * X_{30}$	0.9900	0.9200	0.9100
$X_{11} * X_{13} * X_{19} * X_{50}$	0.9800	0.8900	0.3800
Overall Power	0.9963	0.9712	0.9038
FP	0.5100	1.1400	1.0900
FDR	0.0601	0.1279	0.1310

C.2 Example 7: Epigenetic data with latent Gaussian variables

This example illustrates how the extended DBRM model (9) can be used for feature engineering while simultaneously modeling correlation structures with latent Gaussian variables. To this end we consider genomic and epigenomic data from *Arabidopsis thaliana*. Arabidopsis is an extremely well studied model organism for which plenty of genomic and epigenomic data sets are publicly available (see for example Becker et al. 2011). In terms of epigenetic data we consider methylation markers. DNA locations with a nucleotide of type cytosine nucleobase (C) can be either methylated or not. Our focus will be on modeling the amount of methylated reads through different covariates including (local) genomic structures, gene classes and expression levels. The studied data was obtained from the NCBI GEO archive (Barrett et al. 2013), where we consider a sample of $n = 500$ base-pairs chosen from a random genetic region of a single plant. Only cytosine nucleobases can be methylated, hence these 500 observations correspond to 500 sequential cytosine nucleobases from the selected genetic region.

At each location i there are R_i reads of which Y_i are methylated. Although a binomial distribution would be most appropriate here, we have, due to numerical considerations, assumed a Poisson distribution for Y_i with mean $\mu_i \in \mathbb{R}^+$. In the extended DBRM model (9) we use the logarithm as the canonic link function. For the feature engineering part of the model we consider $p = 14$ input variables which are defined as follows. The first factor with three levels is coded with two dummy variables X_1 and X_2 and describes whether a location belongs to a CGH, CHH or CHG genetic region, where H is either A, C or T. The second factor is concerned with the distance of the location to the previous cytosine nucleobase (C). The dummy variables $X_3 - X_8$ code whether the distance is 2, 3, 4, 5, from 6 to 20 or greater than 20, respectively, taking a distance of 1 as reference. The third factor describes whether a location belongs to a gene, and if yes whether this gene belongs to a particular group of biological interest. These groups are denoted by M_α, M_γ, M_δ and M_0 and are coded by 3 additional dummy variables $X_9 - X_{11}$. Two further covariates are derived from the expression level for a nucleobase being either greater than 3000 FPKM or greater than 10000 FPKM, defining binary covariates X_{12} and X_{13}. The last covariate X_{14} is the offset defined by the total number of reads per location $R_t, \in \mathbb{N}$. The offset mentioned above is modeled as an additional component of the model and hence can be a matter of model choice.

Furthermore we consider the following latent Gaussian variables to model spatial correlations, where marginal likelihoods are computed using the INLA package (Rue et al. 2018) and the parametrization is taken from there as well:

***AR*(1) process:** Autoregressive process of order 1 with parameter $\rho \in \mathbb{R}$, namely $\delta_i = \rho\delta_{i-1} + \epsilon_i \in \mathbb{R}$ with $\epsilon_i \sim N(0, \sigma_\epsilon^2)$, $i = 1, ..., n$. For this process the priors on the hyper-parameters are defined as follows: First reparametrize to $\psi_1 = \log \frac{1}{\sigma_{\epsilon,t}^2}(1-\rho^2)$, $\psi_2 = \log \frac{1+\rho}{1-\rho}$, then assume $\psi_1 \sim \text{logGamma}(1, 5 \times 10^{-5})$, $\psi_2 \sim N(0, 0.15^{-1})$.

***RW*(1) process:** Random walk of order 1 based on the Gaussian vector $\delta_1, ..., \delta_n$, which is constructed assuming independent increments: $\Delta\delta_i = \delta_i - \delta_{i-1} \sim N(0, \tau^{-1})$. Priors on the hyper-parameters are defined as follows: Reparametrize to $\psi = \log \tau$ and assume $\psi \sim \text{logGamma}(1, 5 \times 10^{-5})$.

***OU* process:** Ornstein-Uhlenbeck process (with mean zero), which is defined via the stochastic differential equation $d\delta(t) = -\phi\delta(t)dt + \sigma dW(t)$, where $\phi > 0$ and $\{W(t)\}$ is the Wiener process. This is the continuous time analogue to the discrete time $AR(1)$ model and the process is Markovian. Let $\delta_1, ..., \delta_n$ be the values of the process at increasing locations $t_1, ..., t_n$, then the conditional distribution $\delta_i|\delta_1, ..., \delta_{i-1}$ is Gaussian with mean $\delta_{i-1}e^{-\phi z_i}$ and precision $\tau(1 - e^{-2\phi z_i})^{-1}$, where $z_i = t_i - t_{i-1}$ and $\tau = 2\phi/\sigma^2$. Priors on the hyper-parameters are defined as follows: We first reparametrize to $\psi_1 = \log \tau$, $\psi_2 = \log \phi$ and then assume $\psi_1 \sim \text{logGamma}(1, 5 \times 10^{-5})$, $\psi_2 \sim N(0, 0.2^{-1})$.

***IG* process:** Independent Gaussian process $\{\delta_i\}$ with $\delta_i \sim N(0, \tau^{-1})$. Priors on the hyper-parameters are defined as follows: First reparametrize to $\psi = \log \tau$ and then assume $\psi \sim \text{logGamma}(1, 5 \times 10^{-5})$.

These different processes allow to model different spatial dependence structures of methylation rates along the genome. They can also account for variance which is not explained by the covariates. DBRM can be used to find the best combination of latent variables for modeling this dependence in combination with deep feature engineering.

Table 12: Results for Example 7: Features and latent Gaussian variables (LGV) with posterior probability above 0.25 found by GMJMCMC using 16 parallel threads.

	Variable	Posterior
Features	offset(log(total.bases))	1
	CG	0.999
	CHG	0.952
LGV	RW(1)	1

The Bayesian model is completed with Gaussian priors for the regression coefficients

$$\boldsymbol{\beta}|\boldsymbol{\gamma} \sim N_{p_{\boldsymbol{\gamma}}}(\mathbf{0}, I_{p_\gamma} e^{-\psi_{\beta_\gamma}}) \tag{29}$$

$$\psi_{\beta_\gamma} \sim \text{logGamma}(1, 5 \times 10^{-5}) \tag{30}$$

We then use prior (6) with $a = e^{-2\log n}$ for $\boldsymbol{\gamma}$. We use a similar prior for λ associated with selection of the latent Gaussian variables,

$$p(\boldsymbol{\lambda}) \propto \prod_{j=1}^{r} \exp(-2\log n\lambda_j)\ , \tag{31}$$

where each latent Gaussian variables has equal prior probability to be included.

From the results of Table 12 we learn that there are three features with large posterior probability: the offset for the total number of observations per location as well as two features indicating whether the location is CG or CHG. Among the latent Gaussian variables only the random walk process of order one was found to be of importance. None of the engineered features were found of importance for this example. Like in Example 1 and 2 we observe that although our feature space includes highly non-linear features the regularization due to our priors guarantees the choice of parsimonious models and non-linear features are only selected if really necessary.