Supplementary Material for: Predictive Entropy Search for Multi-objective Bayesian Optimization

Daniel Hernández-Lobato
Universidad Autónoma de Madrid
Francisco Tomás y Valiente 11
28049, Madrid, Spain
daniel.hernandez@uam.com

José Miguel Hernández-Lobato
Harvard University
33 Oxford street
Cambridge, MA 02138, USA
jmhl@seas.harvard.edu

Amar Shah
Cambridge University
Trumpington Street, Cambridge
CB2 1PZ, United Kingdom.
as793@cam.ac.uk

Ryan P. Adams
Harvard University and Twitter
33 Oxford street
Cambridge, MA 02138, USA
rpa@seas.harvard.edu

1 Detailed Description of Expectation Propagation

In this section we describe in detail the specific steps of the EP algorithm that is required for the evaluation of the proposed acquisition function, PESMO. More precisely, we show how to compute the EP approximation to the conditional predictive distribution of each objective f_k. From the main manuscript we know that that this distribution is obtained by multiplying the GP posteriors by the product of all the approximate factors. We also show how to implement the EP updates to refine each approximate factor. In our implementation we assume independence among the K objective functions.

We assume the reader is familiar with the steps of the expectation propagation algorithm, as described in [8].

Recall from the main manuscript that all EP approximate factors $\tilde{\psi}$ are Gaussian and given by:

$$\tilde{\psi}(\mathbf{x}_i, \mathbf{x}_j^\star) = \prod_{k=1}^{K} \tilde{\phi}_k(\mathbf{x}_i, \mathbf{x}_j^\star)\,, \tag{1}$$

where

$$\begin{aligned}\tilde{\phi}_k(\mathbf{x}_i, \mathbf{x}_j^\star) = \exp\big\{&-0.5\left(f_k(\mathbf{x}_i)^2 \tilde{v}_{i,j,k} + f_k(\mathbf{x}_j^\star)^2 \tilde{v}_{i,j,k}^\star + f_k(\mathbf{x}_j^\star) f_k(\mathbf{x}_i) \tilde{c}_{i,j,k}\right) \\ &+\tilde{m}_{i,j,k} f_k(\mathbf{x}_i) + \tilde{m}_{i,j,k}^\star f_k(\mathbf{x}_j^\star)\big\}\ ,\end{aligned} \tag{2}$$

for some input location $\mathbf{x}_i$ and for some $\mathbf{x}_j^\star$ extracted from the current sampled Pareto set $\mathcal{X}^\star$. Note that in (2), $\tilde{v}_{i,k}$, $\tilde{v}_{j,k}^\star$, $\tilde{c}_{i,j,k}$, $\tilde{m}_{i,k}$ and $\tilde{m}_{j,k}^\star$ are parameters fixed by EP.

1.1 Reconstruction of the Conditional Predictive Distribution

In this section we show how to obtain a conditional predictive distribution for each objective function f_k, given a sampled Pareto set $\mathcal{X}^\star = \{\mathbf{x}_1^\star, \ldots, \mathbf{x}_M^\star\}$ of size M, and a set of N input locations $\hat{\mathcal{X}} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, with corresponding observations of the k-th objective $\mathbf{y}_k$. We also assume that we are given the EP approximate factors $\tilde{\psi}$.

Define $\mathbf{f}_k = (f_k(\mathbf{x}_1^\star), \ldots, f_k(\mathbf{x}_M^\star), f_k(\mathbf{x}_1), \ldots, f_k(\mathbf{x}_N))^\mathrm{T}$. We are interested in computing

$$p(\mathbf{f}_k|\mathcal{X}^\star, \hat{\mathcal{X}}, \mathbf{y}_k) \approx q(\mathbf{f}_k) = Z^{-1} p(\mathbf{f}_k|\hat{\mathcal{X}}, \mathbf{y}_k) \prod_{\mathbf{x}\in\mathcal{X}^\star\cup\hat{\mathcal{X}}} \prod_{\substack{\mathbf{x}^\star\neq\mathbf{x} \\ \mathbf{x}^\star\in\mathcal{X}^\star}} \tilde{\phi}_k(\mathbf{x}, \mathbf{x}^\star)\,, \tag{3}$$

for some normalization constant Z. In (3) we have only considered those approximate factors that depend on the current objective function f_k and ignored the rest. We note that $p(\mathbf{f}_k|\hat{\mathcal{X}}, \mathbf{y}_k)$ is simply the posterior distribution of the Gaussian process, which is a multi-variate Gaussian over $N + M$ variables with natural parameters $\tilde{\mathbf{\Sigma}}^k$ and $\tilde{\boldsymbol{\mu}}^k$. Furthermore, all EP approximate factors $\tilde{\psi}$ are Gaussian. Because the Gaussian distribution is closed under the product operation, $q(\mathbf{f}_k)$ is a multi-variate Gaussian distribution over $N + M$ variables with natural parameters $\mathbf{S}^k$ and $\mathbf{m}^k$ obtained as:

$$\begin{aligned}
S^k_{i,i} &= \tilde{\Sigma}^k_{i,i} + \sum_{j=M+1}^{N+M} \tilde{v}^\star_{j,i,k} + \sum_{j\neq i} \tilde{v}_{i,j,k} + \sum_{\substack{j=1 \\ j\neq i}}^{M} \tilde{v}^\star_{j,i,k} \quad \text{for} \quad i = 1, \ldots, M\,, \\
S^k_{i,i} &= \tilde{\Sigma}^k_{i,i} + \sum_{j=1}^{M} \tilde{v}_{i,j,k} \quad \text{for} \quad i = M+1, \ldots, N+M\,, \\
S^k_{i,j} &= \tilde{\Sigma}^k_{i,j} + \tilde{c}_{i,j,k} \quad \text{for} \quad i = M+1, \ldots, N+M\,, \quad \text{and} \quad j = 1, \ldots, M\,, \\
S^k_{i,j} &= \tilde{\Sigma}^k_{j,i} + \tilde{c}_{j,i,k} + \tilde{c}_{i,j,k} \quad \text{for} \quad i = 1, \ldots, M\,, \quad \text{and} \quad j = 1, \ldots, M\,, \quad \text{and} \quad i \neq j\,, \\
S^k_{i,j} &= \tilde{\Sigma}^k_{i,j} \quad \text{for} \quad i = M+1, \ldots, N+M\,, \quad \text{and} \quad j = M+1, \ldots, M+M\,, \quad \text{and} \quad i \neq j\,, \\
S^k_{j,i} &= S^k_{i,j} \quad \text{for} \quad i \neq j\,, \\
m^k_i &= \tilde{\mu}^k_i + \sum_{j=M+1}^{N+M} \tilde{m}^\star_{j,i,k} + \sum_{j\neq i} \tilde{m}_{i,j,k} + \sum_{\substack{j=1 \\ j\neq i}}^{M} \tilde{m}^\star_{j,i,k} \quad \text{for} \quad i = 1, \ldots, M\,, \\
m^k_i &= \tilde{\mu}^k_i + \sum_{j=1}^{M} \tilde{m}_{i,j,k} \quad \text{for} \quad i = M+1, \ldots, N+M\,.
\end{aligned} \tag{4}$$

From these natural parameters we can obtain, respectively, the covariance matrix $\mathbf{\Sigma}^k$ and the mean vector $\boldsymbol{\mu}^k$ by computing $(\mathbf{S}^k)^{-1}$ and $(\mathbf{S}^k)^{-1}\mathbf{m}^k$. This has a total cost that is $\mathcal{O}((N + M)^3)$ since we have to invert a matrix of size $(N + M) \times (N + M)$. Importantly, this operations has to be performed only once at each iteration of the optimization process, and the result can be reused when evaluating the acquisition at different input locations.

1.2 The Conditional Predictive Distribution at a New Point

Consider now the computation of the conditional distribution for f_k at a new candidate location $\mathbf{x}_{N+1}$. Assume that we have already obtained $q(\mathbf{f}_k)$ from the previous section and that we have already obtained the parameters of the required approximate factors by using EP. We are interested in evaluating the conditional predictive variance for $f_k(\mathbf{x}_{N+1})$. For this, we need to evaluate:

$$p(f_k(\mathbf{x}_{N+1})|\hat{\mathcal{X}}, \mathcal{X}^\star, \mathbf{x}_{N+1}) \approx \int Z^{-1} q(\mathbf{f}_k, f_k(\mathbf{x}_{N+1})) \prod_{\mathbf{x}^\star\in\mathcal{X}^\star} \tilde{\phi}_k(\mathbf{x}_{N+1}, \mathbf{x}^\star) d\mathbf{f}_k\,, \tag{5}$$

where Z is simply a normalization constant and $q(\mathbf{f}_k, f_k(\mathbf{x}_{N+1}))$ is a multivariate Gaussian distribution which results by extending $q(\mathbf{f}_k)$ with one extra dimension for $f_k(\mathbf{x}_{N+1})$. Recall that $\mathbf{f}_k = (f_k(\mathbf{x}_1^\star), \ldots, f_k(\mathbf{x}_M^\star), f_k(\mathbf{x}_1), \ldots, f_k(\mathbf{x}_N))^\mathrm{T}$. Again, in (5) we have only considered those approximate factors that depend on f_k. The covariances between $\mathbf{f}_k$ and $f_k(\mathbf{x}_{N+1})$ are obtained from the GP posterior for f_k given the observed data. The mean and the variance of $f_k(\mathbf{x}_{N+1})$ to be used in $q(\mathbf{f}_k, f_k(\mathbf{x}_{N+1}))$ can also be obtained in a similar way. Because all the factors in the r.h.s. of (5) are Gaussian, the result of the integral is a univariate Gaussian distribution.

Define $\tilde{\mathbf{f}}_k = (f_k(\mathbf{x}_1^\star), \ldots, f_k(\mathbf{x}_M^\star), f_k(\mathbf{x}_{N+1}))^\mathrm{T}$. Because $\prod_{\mathbf{x}^\star \in \mathcal{X}^\star} \tilde{\phi}_k(\mathbf{x}_{N+1}, \mathbf{x}^\star)$ does not depend on $f_k(\mathbf{x}_1), \ldots, f_k(\mathbf{x}_N)$, we can marginalize these variables in the r.h.s. of (5) to get something proportional to:

$$\int q(\tilde{\mathbf{f}}_k) \prod_{\mathbf{x}^\star \in \mathcal{X}^\star} \tilde{\phi}_k(\mathbf{x}_{N+1}, \mathbf{x}^\star) \prod_{i=1}^M df_k(\mathbf{x}_i^\star) \propto \int \mathcal{N}(\tilde{\mathbf{f}}_k | (\mathbf{S}^x)^{-1}\mathbf{m}^x, (\mathbf{S}^x)^{-1}) \prod_{i=1}^M df_k(\mathbf{x}_i^\star) = \mathcal{N}(f_k(\mathbf{x}_{N+1}) | m_x, \sigma_x^2)\,, \qquad (6)$$

where $\mathbf{m}^x$ and $\mathbf{S}^x$ are the natural parameters of the approximate conditional predictive distribution for $\tilde{\mathbf{f}}_k$, which is Gaussian. Similarly, m_x and σ_x^2 are the mean and variance of the Gaussian approximation to $p(f_k(\mathbf{x}_{N+1} | \hat{\mathcal{X}}, \mathcal{X}^\star, \mathbf{x}_{N+1})$.

We are interested in the evaluation of σ_x^2, which is required for entropy computation. It is clear that σ_x^2 is given by the last diagonal entry of $(\mathbf{S}^x)^{-1}$. In consequence, we now show how to compute $\mathbf{S}^x$ and $(\mathbf{S}^x)^{-1}_{M+1,M+1}$. We do not give the details for computing m_x, because only the variance is required for the entropy computation.

Each entry in $\mathbf{S}^x$ is given by:

$$\begin{aligned}
S^x_{i,j} &= S^k_{i,j} \quad \text{for} \quad 1 \leq i \leq M \quad \text{and} \quad 1 \leq j \leq M\,, \quad \text{and} \quad i \neq j\,,\\
S^x_{i,j} &= \mathrm{cov}(f_k(\mathbf{x}_{N+1}), f(\mathbf{x}_j^\star)) + \tilde{c}_{N+1,j,k} \quad \text{for} \quad 1 \leq j \leq M \quad \text{and} \quad i = M+1\,,\\
S^x_{j,i} &= S^x_{i,j} \quad \text{for} \quad j \neq i\,, \quad \text{and} \quad 1 \leq i, j \leq M\,,\\
S^x_{i,i} &= S^k_{i,i} + \tilde{v}^\star_{N+1,j,k}\,, \text{for} \quad 1 \leq i \leq M\,,\\
S^x_{M+1,M+1} &= \mathrm{var}(f_k(\mathbf{x}_{N+1})) + \sum_{j=1}^M \tilde{v}_{N+1,j,k}\,,
\end{aligned} \qquad (7)$$

where $\tilde{v}_{N+1,i,k}$, $\tilde{v}^\star_{N+1,i,k}$, and $\tilde{c}_{N+1,i,k}$ are the parameters of each of the M factors $\tilde{\phi}_k(\mathbf{x}_{N+1}, \mathbf{x}_j^\star)$, for $j = 1, \ldots, M$. Furthermore, $\mathrm{var}(f_k(\mathbf{x}_{N+1}))$ and $\mathrm{cov}(f_k(\mathbf{x}_{N+1})$ are the posterior variance of $f_k(\mathbf{x}_{N+1})$ and the posterior covariance between $f_k(\mathbf{x}_{N+1})$ and $f_k(\mathbf{x}_j^\star)$.

We note that $\mathbf{S}^x$ has a block structure in which only the last row and column depend on $\mathbf{x}_{N+1}$. This allows to compute $\sigma_x^2 = (\mathbf{S}^x)^{-1}_{M+1,M+1}$ with cost $\mathcal{O}(M^3)$ using the formulas for block matrix inversion. All these computations are carried out using the open-BLAS library for linear algebra operations which is particularly optimized for each processor.

Given σ_x^2 we only have to add the variance of the additive Gaussian noise ϵ^k_{N+1} to obtain the final variance of the Gaussian approximation to the conditional predictive distribution of $y^k_{N+1} = f_k(\mathbf{x}_{N+1}) + \epsilon^k_{N+1}$.

1.3 Update of an Approximate Factor

EP updates until convergence each of the approximate factors $\tilde{\psi}$. Given an exact factor $\psi(\mathbf{x}_i, \mathbf{x}_j^\star)$, in this section we show how to update the corresponding EP approximate factor $\tilde{\psi}(\mathbf{x}_i, \mathbf{x}_j^\star)$. For this, we assume that we have already obtained the parameters $\boldsymbol{\mu}^k$ and $\boldsymbol{\Sigma}^k$ of each of the K conditional predictive distributions, $q(\mathbf{f}_k)$, as described in Section 1.1. The form of the exact factor is:

$$\psi(\mathbf{x}_i, \mathbf{x}_j^\star) = 1 - \prod_{k=1}^K \Theta\left(f_k(\mathbf{x}_i) - f_k(\mathbf{x}_j^\star)\right)\,. \qquad (8)$$

Note that this factor only depends on $f_k(\mathbf{x}_i)$ and $f_k(\mathbf{x}_j^\star)$ for $k = 1, \ldots, K$. This means that we are only interested in the distribution of these variables under $q(\mathbf{f}_k)$, for $k = 1, \ldots, K$, and can ignore (marginalize in q) all other variables. Thus, in practice we will work with $q(f_k(\mathbf{x}_i), f_k(\mathbf{x}_j^\star))$, for $k = 1, \ldots, K$. These are bi-variate Gaussian distributions. Let the means, variances and covariance parameters of one of these distributions be respectively: $m_{i,j,k}$, $m^\star_{i,j,k}$, $v_{i,j,k}$, $v^\star_{i,j,k}$ and $c_{i,j,k}$.

1.3.1 Computation of the Cavity Distribution

The first step of the update is to compute and old distribution q^{old}, known as the cavity distribution, which is obtained by removing the approximate factor $\tilde{\psi}(\mathbf{x}_i, \mathbf{x}_j^\star)$ from the product of the K approximations $q(f_k(\mathbf{x}_i), f_k(\mathbf{x}_j^\star))$, for $k = 1, \ldots, K$. Recall that $\tilde{\psi}(\mathbf{x}_i, \mathbf{x}_j^\star) = \prod_{k=1}^K \tilde{\phi}_k(\mathbf{x}_i, \mathbf{x}_j^\star)$. This can be done by division. Namely, $q^{\text{old}}(f_k(\mathbf{x}_i), f_k(\mathbf{x}_j^\star)) \propto q(f_k(\mathbf{x}_i), f_k(\mathbf{x}_j^\star))/\tilde{\phi}_k(\mathbf{x}_i, \mathbf{x}_j^\star)$, for $k = 1, \ldots, K$. Because all the factors are Gaussian, the result is another bi-variancete Gaussian distribution. Let the corresponding *old* parameters be: $m_{i,j,k}^{\text{old}}$, $m_{i,j,k}^{\text{old}\star}$, $v_{i,j,k}^{\text{old}}$, $v_{i,j,k}^{\text{old}\star}$ and $c_{i,j,k}^{\text{old}}$. These parameters are obtained by subtracting from the natural parameters of $q(f_k(\mathbf{x}_i), f_k(\mathbf{x}_j^\star))$, the natural parameters of $\tilde{\phi}_k$. The resulting natural parameters are then transformed into standard mean and covariance parameters to get the parameters of $q^{\text{old}}(f_k(\mathbf{x}_i), f_k(\mathbf{x}_j^\star))$. This step is performed as indicated in the last paragraph of Section 1.1, and it involves computing the inverse of a 2×2 matrix, which is something very easy and inexpensive to do in practice.

1.3.2 Computation of the Moments of the Tilted Distribution

Given each $q^{\text{old}}(f_k(\mathbf{x}_i), f_k(\mathbf{x}_j^\star))$, for $k = 1, \ldots, K$, the next step of the EP algorithm is to compute the moments of a tilted distribution defined as:

$$\hat{p}(\{f_k(\mathbf{x}_i), f_k(\mathbf{x}_j^\star)\}_{k=1}^K) = \hat{Z}^{-1}\psi(\mathbf{x}_i, \mathbf{x}_j^\star) \prod_{k=1}^K q^{\text{old}}(f_k(\mathbf{x}_i), f_k(\mathbf{x}_j^\star))\,, \tag{9}$$

where $\hat{Z}$ is just a normalization constant that guarantees that $\hat{p}$ integrates up to one.

Importantly, the normalization constant $\hat{Z}$ can be computed in closed form and is given by:

$$\hat{Z} = 1 - \prod_{k=1}^K \Phi\left(\frac{m_{i,j,k}^{\text{old}} - m_{i,j,k}^{\text{old}\star}}{\sqrt{v_{i,j,k}^{\text{old}} + v_{i,j,k}^{\text{old}\star} - 2c_{i,j,k}^{\text{old}}}}\right)\,, \tag{10}$$

where $\Phi(\cdot)$ is the c.p.f. of a standard Gaussian distribution. The moments (mean vector and covariance matrix) of $\hat{p}$ can be readily obtained from the derivatives of $\log \hat{Z}$ with respect to the parameters of $q^{\text{old}}(f_k(\mathbf{x}_i), f_k(\mathbf{x}_j^\star))$, as indicated in the Appendix of [5].

1.3.3 Computation of the Individual Approximate Factors

Given the moments of the tilted distribution $\hat{p}(\{f_k(\mathbf{x}_i), f_k(\mathbf{x}_j^\star)\}_{k=1}^K)$, it is straight-forward to obtain the parameters of the approximate factors $\tilde{\phi}_k$, for $k = 1, \ldots, K$, whose product approximates $\psi(\mathbf{x}_i, \mathbf{x}_j^\star)$. The idea is that the product of $\tilde{\psi}(\mathbf{x}_i, \mathbf{x}_j^\star) = \prod_{i=1}^K \tilde{\phi}_k(\mathbf{x}_i, \mathbf{x}_j^\star)$ and $\prod_{k=1}^K q^{\text{old}}(f_k(\mathbf{x}_i), f_k(\mathbf{x}_j^\star))$ should lead to a Gaussian distribution with the same moments as the tilted distribution $\hat{p}$.

The detailed steps to find each $\tilde{\phi}_k$ are: (i) Define a Gaussian distribution with the same moments as $\hat{p}$, denoted $\prod_{k=1}^K q^{\text{new}}(f_k(\mathbf{x}_i), f_k(\mathbf{x}_j^\star))$. Note that this distribution factorizes across each objective k. Let the parameters of this distribution be $m_{i,j,k}^{\text{new}}$, $m_{i,j,k}^{\text{new}\star}$, $v_{i,j,k}^{\text{new}}$, $v_{i,j,k}^{\text{new}\star}$ and $c_{i,j,k}^{\text{new}}$, for $k = 1, \ldots, K$. (ii) Transform these parameters to natural parameters, and subtract to them the natural parameters of $\prod_{k=1}^K q^{\text{old}}(f_k(\mathbf{x}_i), f_k(\mathbf{x}_j^\star))$. (iii) The resulting natural parameters are the natural parameters of each updated $\tilde{\phi}_k$. Note that this operation involves going from standard parameters to natural parameters. Again, this can be done as indicated in the last paragraph of Section 1.1. For this, the inverse of the corresponding 2×2 covariance matrix of each Gaussian factor of $\prod_{k=1}^K q^{\text{new}}(f_k(\mathbf{x}_i), f_k(\mathbf{x}_j^\star))$ is required. Because these are 2×2 matrices, this operation is inexpensive and very fast to compute.

1.4 Parallel EP Updates and Damping

In our EP implementation we updated in parallel each of the approximate factors $\tilde{\psi}$, as indicated in [4]. That is, we computed the corresponding cavity distribution for each factor ψ and updated the corresponding approximate factor $\tilde{\psi}$ afterwards. Next, the EP approximation was reconstructed as indicated in Section 1.1.

We also employed damped EP updates in our implementation [9]. That is, the parameters of each updated factor are set to be a linear combination of the old parameters and the new parameters. The use of damped updates prevents very large changes in the parameter values. It is hence very useful to improve the convergence properties of the algorithm. Finally, damping does not change the convergence points of EP.

2 Finding a Small and Accurate Ensemble of Decision Trees

In this section we evaluate each of the methods from the main manuscript in the task of finding an ensemble of decision trees of small size that has low prediction error. We measure the ensemble size in terms of the sum of the total number of nodes in each of trees of the ensemble. Note that the objectives considered are conflicting because it is expected that an ensemble of small size has higher prediction error than an ensemble of larger size. The dataset considered is the *German Credit* dataset, which is extracted from the UCI repository [6]. This is a binary classification dataset with 1,000 instances and 9 attributes. The prediction error is measured using a 10-fold-cross validation procedure that is repeated 5 times to reduce the variance of the estimates.

Critically, to get ensembles of decision trees with good prediction properties one must encourage diversity in the ensemble [3]. In particular, if all the decision trees are equal, there is no gain from aggregating them in an ensemble. However, too much diversity can also lead to ensembles of poor prediction performance. For example, if the predictions made are completely random, one cannot obtain improved results by aggregating the individual classifiers. In consequence, we consider here several mechanisms to encourage diversity in the ensemble, and let the amount of diversity be specified in terms of adjustable parameters.

To build the ensemble we employed decision trees in which the data is split at each node, and the best split is chosen by considering each time a random set of attributes —we use the *Decision-Tree* implementation provided in the python package *scikit-learn* for this, and the number of random attributes is an adjustable parameter. This is the approach followed in Random Forest [1] to generate the ensemble classifiers. Each tree is trained on a random subset of the training data of a particular size, which is another adjustable parameter. This approach is known in the literature as subbagging [2], and has been shown to lead to classification ensembles with good prediction properties. We consider also an extra method to introduce diversity known as class-switching [7]. In class-switching, the labels of a random fraction of the training data are changed to a different class. The final ensemble prediction is computed by majority voting.

In summary, the adjustable parameters are: the number of decision trees built (between 1 and $1,000$), the number of random features considered at each split in the building process of each tree (between 1 and 9), the minimum number of samples required to split a node (between 2 and 200), the fraction of randomly selected training data used to build each tree, and the fraction of training instances whose labels are changed (after doing the sub-sampling process).

Finally, we note that this setting is suited to the decoupled version of PESMO since both objectives can be evaluated separately. In particular, the total number of nodes is estimated by building only once the ensemble without leaving any data aside for validation, as opposed to the cross-validation approach used to estimate the ensemble error, which requires to build several ensembles on subsets of the data, to then estimate the prediction error on the data left out for validation.

We run each method for 200 evaluations of the objectives and report results after 100 and 200 evaluations. That is, after 100 and 200 evaluations, we optimize the posterior means of the GPs and provide a recommendation in the form of a Pareto set. As in the experiments reported in the main manuscript with neural networks, we re-estimate three times the objectives associated to each Pareto point from the recommendation made by each method, and average results. The goal of this averaging process is to reduce the noise in the final evaluation of the objectives. These final evaluations are used to estimate the performance of each method using the hyper-volume.

We repeat these experiments 50 times and report the average results across repetitions.

Table 1: Avg. hyper-volume after 100 and 200 evaluations of the objectives.

# Eval.	PESMO	PESMO$_{\text{dec}}$	ParEGO	SMSego	EHI	SUR
100	8.742±.006	**8.755±.009**	8.662±.019	8.719±.012	8.731±.009	8.739±.007
200	**8.764±.007**	8.758±.007	8.705±.008	8.742±.006	8.727±.008	8.756±.006

Table 1 shows the average hyper-volume of the recommendations made by each method, after 100 and 200 evaluations of the objective functions. The table also shows the corresponding error bars. In this case the observed differences among the different methods are smaller than in the experiments with neural networks. Nevertheless, we observe that the decoupled version of PESMO obtains the best results after 100 evaluations. After this, PESMO in the coupled setting performs best, closely followed by SUR. After 200 evaluations, the best method is the coupled version of PESMO, closely followed by its decoupled version and by SUR. SMSego and EHI give worse results than these methods, in general. Finally, as in the experiments with neural networks reported in the main manuscript, ParEGO is the worst performing method. In summary, the best methods are PESMO in either setting (coupled or decoupled) and SUR. All other methods perform worse. Furthermore, the decoupled version of PESMO gives slightly better results at the beginning, *i.e.*, after 100 evaluations.

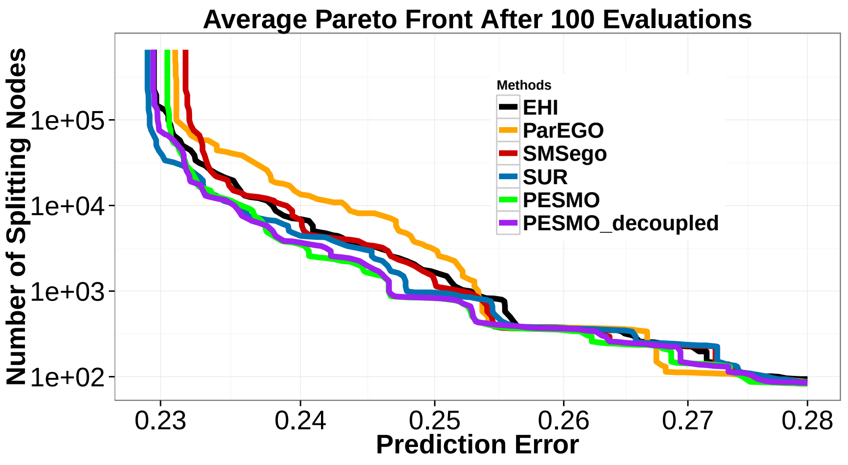
Figure 1 shows the average Pareto front (this is simply the values in functional space associated to the Pareto set) corresponding to the recommendations made by each method after 100 (top) and 200 evaluations of the objectives (bottom). We observe that PESMO finds ensembles with better properties than the ones found by EHI, SMSego and ParEGO. Namely, ensembles of smaller size for a similar or even better prediction error. The most accurate ensembles are found by SUR. Nevertheless, they have a very similar error to the one of the most accurate ensembles found by PESMO. Finally, we note that in some cases, PESMO is able to find ensembles of intermediate size with better prediction error than the ones found by SUR.

Figure 2 shows the average number of times that the decoupled version of PESMO evaluates each objective. We observe that in this case the objective that measures the number of nodes in the ensemble is evaluated more times. However, the difference between the number of evaluations of each objective is smaller than the difference observed in the case of the experiments with neural networks. Namely, 135 evaluations of one objective versus 65 evaluations of the other, in this case, compared to 175 evaluations versus 25 evaluations, in the case of the experiments with neural networks. This may explain why in this case the differences between the coupled and the decoupled version of PESMO are not as big as in the experiments reported in the main manuscript.

3 Accuracy of the Acquisition in the Decoupled Setting

One question to be experimentally addressed is whether the proposed approximations for the individual acquisition functions $\alpha_k(\cdot)$, for $k = 1, \ldots, K$, with K the total number of objectives are sufficiently accurate in the decoupled case of PESMO. For this, we extend the experiment carried out in the main manuscript, and compare in a one-dimensional problem with two objectives the acquisition functions $\alpha_1(\cdot)$ and $\alpha_2(\cdot)$ computed by PESMO, with a more accurate estimate obtained via expensive Monte Carlo sampling and a non-parametric estimator of the entropy [10]. This estimate measures the expected decrease in the entropy of the predictive distribution of one of the objectives, at a given location of the input space, after conditioning to the Pareto set. Importantly, in the decoupled case, the observations corresponding to each objective need not be located at the same input locations.

Figure 3 (top) shows at a given step of the optimization process, the observed data and the posterior mean and the standard deviation of each of the two objectives. The figure on the middle shows the corresponding acquisition function corresponding to the first objective, $\alpha_1(\cdot)$, computed



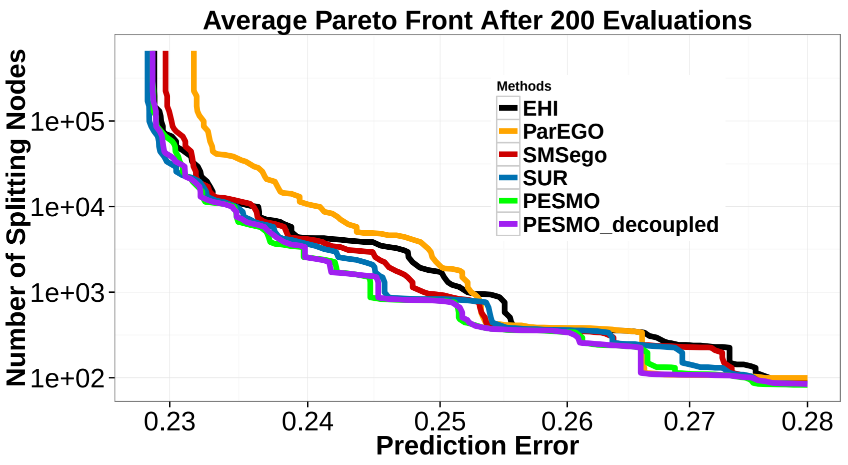


Figure 1: Avg. Pareto fronts obtained by each method after 100 (top) and 200 (bottom) evaluations of the objectives. Best seen in color.

by PESMO and by the Monte Carlo method (Exact). The figure on the bottom shows the same results for the acquisition function corresponding to the second objective, $\alpha_2(\cdot)$. We observe that both functions look very similar, including the location of the global maximizer. This indicates that the approximation obtained by expectation propagation is potentially good also in the decoupled setting.

References

[1] Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[2] Peter Bühlmann and Bin Yu. Analyzing Bagging. *Annals of Statistics*, 30:927–961, 2001.

[3] Thomas G Dietterich. Ensemble methods in machine learning. In *Multiple classifier systems*, pages 1–15. Springer, 2000.

[4] Marcel Van Gerven, Botond Cseke, Robert Oostenveld, and Tom Heskes. Bayesian source localization with the multivariate Laplace prior. In *Advances in Neural Information Processing Systems 22*, pages 1901–1909, 2009.

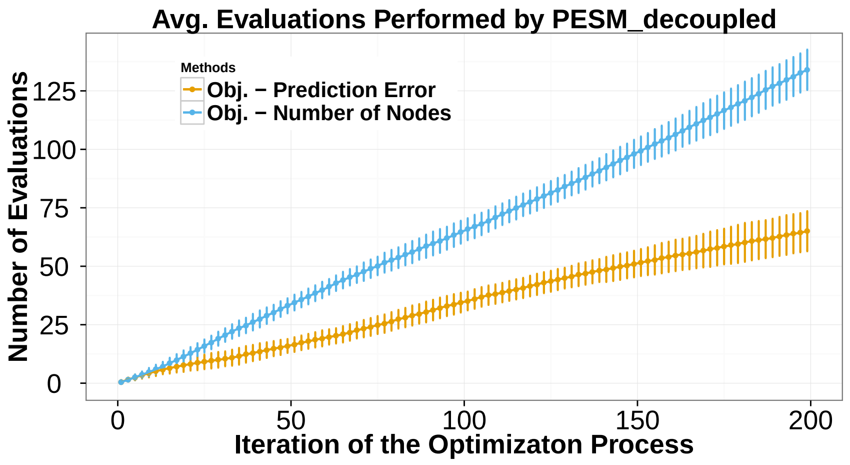


Figure 2: Number of evaluations of each objective done by $\text{PESMO}_{\text{decoupled}}$, as a function of the iteration number N, in the problem of finding a good ensemble of decision trees. Best seen in color.

[5] Daniel Hernández-Lobato. *Prediction Based on Averages over Automatically Induced Learners: Ensemble Methods and Bayesian Techniques.* PhD thesis, Computer Science Department, Universidad Autónoma de Madrid, 2009. Online available at: http://arantxa.ii.uam.es/d̃hernan/ docs/Thesis_color_links.pdf.

[6] M. Lichman. UCI machine learning repository, 2013.

[7] Gonzalo Martínez-Muñoz and Alberto Suárez. Switching class labels to generate classification ensembles. *Pattern Recognition*, 38:1483–1494, 2005.

[8] T. Minka. Expectation propagation for approximate Bayesian inference. In *Annual Conference on Uncertainty in Artificial Intelligence*, pages 362–36, 2001.

[9] T. Minka and J. Lafferty. Expectation-propagation for the generative aspect model. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, pages 352–359, 2002.

[10] H. Singh, N. Misra, V. Hnizdo, A. Fedorowicz, and E. Demchuk. Nearest neighbor estimates of entropy. *American journal of mathematical and management sciences*, 23:301–321, 2003.

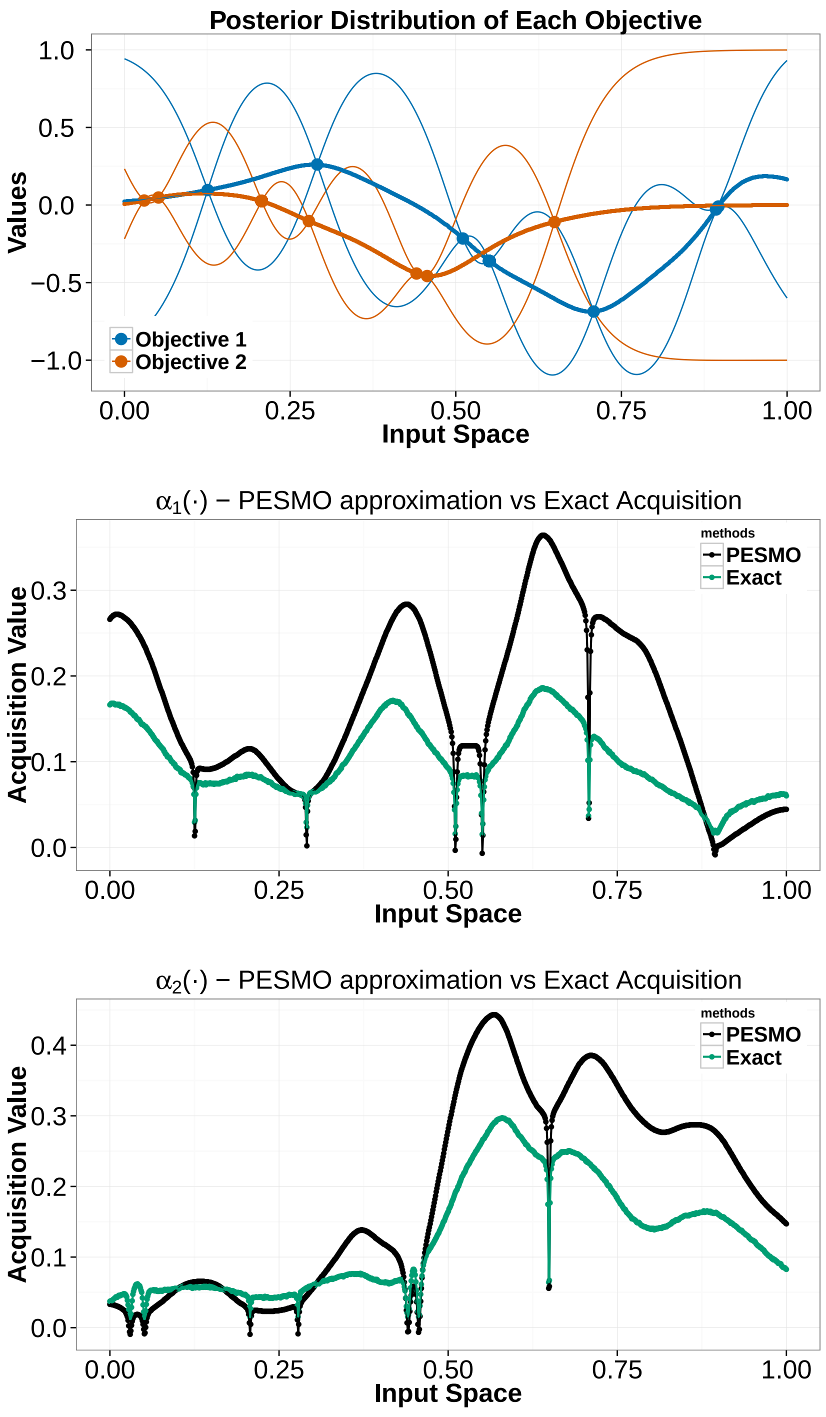


Figure 3: (top) Observations of each objective and posterior mean and standard deviations of each GP model. (middle) Estimates of the acquisition function corresponding to the first objective, $\alpha_1(\dot{})$, by PESMO, and by a Monte Carlo method combined with a non-parametric estimator of the entropy. (bottom) Same results for the acquisition function corresponding to the second objective $\alpha_2(\cdot)$. Best seen in color.