Supplementary material:
Bayesian Optimization with Tree-structured Dependencies

Rodolphe Jenatton [1] **Cedric Archambeau** [1] **Javier Gonzalez** [2] **Matthias Seeger** [1]

Abstract

In this supplementary material, we provide additional experimental results along with some details about the inference in our model and its non-linear extension.

1. Experiments

In this section, we provide complementary experimental results.

1.1. Optimization of synthetic tree-structured functions

In addition to the results presented in the core of the paper, we include simulations with the non-linear extension of `tree`, denoted by `tree-nonlinear`, where the shared parameters $\mathbf{z}_p = [\mathbf{r}_v]_{v \in V_p}$ are modeled in a non-linear fashion (see details in Section 3 of the supplementary material).

Moreover, we consider as well settings where the shared variables have a quadratic dependency in the leaf objectives, on both balanced and unbalanced binary trees; see Figure 1 and Figure 2.

The trees on Figure 1 are referred to as `small balanced`, while those on Figure 2 are referred to as `small unbalanced`. We consider also higher-dimensional versions of those, with a depth of 4, resulting in respectively 8 and 9 leaves whose constant shifts are respectively $\{a \times 0.1\}_{a=1}^{8}$ and $\{a \times 0.1\}_{a=1}^{9}$ (they are referred to as `large balanced` and `large unbalanced`).

As defined in the core paper, all the non-shared continuous variables x_j's are defined in $[-1, 1]$, while the shared ones are in $[0, 1]$. This implies that the best function value will always be 0.1.

We can see that the observations made in the core of the paper are still valid for the unbalanced trees. In particular, with linearly-dependent shared variables, `tree-nonlinear` converges more slowly than the linear version `tree` (middle panels of Figures 3 and 4), which may be the price to pay for having higher-dimensional latent variables $\mathbf{c}$. Moreover, in presence of quadratically-dependent shared variables (right panels of Figures 3 and 4), we observe that `tree` fails to model adequately the non-linearities, while `tree-nonlinear`, as expected, can. In absence of shared variables (left panels of Figures 3 and 4), `tree` and `tree-nonlinear` are simply equivalent, which explains why their curves are superimposed. Finally, as dimension increases (i.e., going from the `small` to `large` settings), the performance of `independent` worsens, while that of `tree` and `tree-nonlinear` are barely affected.

1.2. Multi-layer perceptron tuning

We report in Figure 5 the results of all methods including the non-linear extension of `tree`. The setting is identical to that described in the core of the paper. We can see that the linear version `tree` performs better than `tree-nonlinear`. This conclusion, perhaps surprising, is in good agreement with the recent observations from Zhang et al. (2016) where simple linear models outperformed more sophisticated, non-linear competing methods within the context of the optimization of data analytic pipelines.

2. Details about the Tree-structured semi-parametric Gaussian process regression model

We start by providing details about the posterior inference.

2.1. Posterior Inference

The joint distribution $P(\mathbf{y}, \mathbf{g}, \mathbf{c})$ of our model is given by

$$P(\mathbf{c}) \prod_p \mathcal{N}(\mathbf{g}_p; \mathbf{b}_p, \mathbf{K}_p) \mathcal{N}(\mathbf{y}_p; \mathbf{g}_p + \mathbf{Z}_p^\top \mathbf{c}, \sigma^2 \mathbf{I}_{n_p}), \quad (1)$$

where $\mathbf{K}_p = [\mathcal{K}_p(\mathbf{x}_i, \mathbf{x}_j)]_{i,j \in I_p}$ are kernel matrices, with the prior $P(\mathbf{c}) = \mathcal{N}(\mathbf{c}; \mathbf{0}, \mathbf{\Sigma}_c)$. Our goal is to obtain the posterior process $P(g_p(\cdot)|\mathbf{c}, \mathbf{y}_p)$ and the posterior distribution

[1] Amazon, Berlin, Germany. [2] Amazon, Cambridge, United Kingdom. Correspondence to: Rodolphe Jenatton <jenatton@amazon.de>, Cedric Archambeau <cedrica@amazon.de>, Javier Gonzalez <gojav@amazon.co.uk>, Matthias Seeger <matthias@amazon.de>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

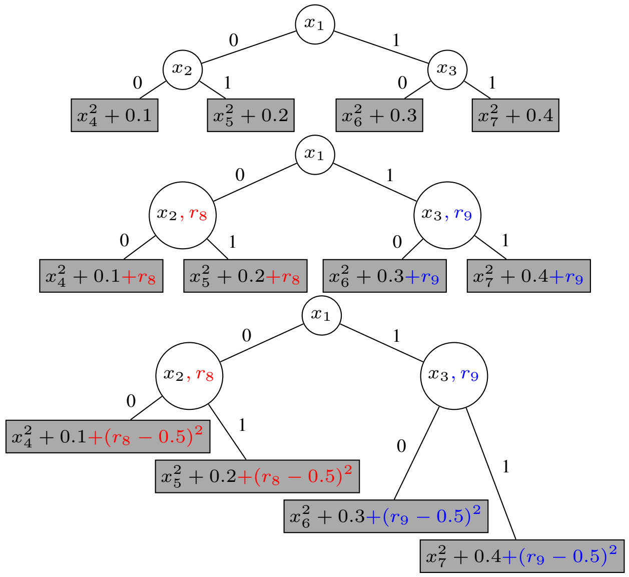

Figure 1. Three examples of functions with tree-structured conditional relationships. Each inner node is a binary variable, and a path in those trees represents successive binary decisions. The leaves contain univariate quadratic functions that are shifted by different constant terms. *Top*: Setting without shared variables. *Middle*: r_8 and r_9 are shared variables that are common to the functions at the leaves of their respective subtrees. In this example, the shared variables have a linear dependency in the leaf objectives. *Bottom*: Similar to the middle tree, except that r_8 and r_9 have a quadratic dependency in the leaf objectives.

$P(\mathbf{c}|\mathbf{y})$. This can be done by invoking standard multivariate Gaussian identities (Petersen & Pedersen, 2012) or by directly reading of the posterior parameters after rewriting the joint distribution into the following form:

$$P(\mathbf{y})P(\mathbf{c}|\mathbf{y})\prod\nolimits_p P(\mathbf{g}_p|\mathbf{c},\mathbf{y}_p).$$

First, we can decompose the joint $P(\mathbf{y}_p,\mathbf{g}_p|\mathbf{c})$:

$$P(\mathbf{g}_p|\mathbf{c},\mathbf{y}_p)\mathcal{N}(\mathbf{y}_p;\mathbf{Z}_p^\top\mathbf{c}+\mathbf{b}_p,\mathbf{M}_p), \qquad (2)$$

where $\mathbf{M}_p=\mathbf{K}_p+\sigma^2\mathbf{I}_{n_p}$. In the sequel, we compute expressions such as $\mathbf{M}_p^{-1}$ and $\log|\mathbf{M}_p|$ by using the Cholesky decomposition $\mathbf{M}_p=\mathbf{L}_p\mathbf{L}_p^\top$. The second factor in (2) is the distribution $P(\mathbf{y}_p|\mathbf{c})$, obtained by integrating out $\mathbf{g}_p$:

$$\int_{\mathbf{g}_p}\mathcal{N}(\mathbf{y}_p;\mathbf{g}_p+\mathbf{Z}_p^\top\mathbf{c},\mathbf{M}_p)\mathcal{N}(\mathbf{g}_p;\mathbf{b}_p,\mathbf{K}_p)d\mathbf{g}_p.$$

It will play a role shortly. The first factor in (2) is the posterior over the latent functions values:

$$\begin{aligned}P(\mathbf{g}_p|\mathbf{c},\mathbf{y}_p)&\propto\mathcal{N}(\mathbf{g}_p;\mathbf{b}_p,\mathbf{K}_p)\mathcal{N}(\mathbf{y}_p;\mathbf{g}_p+\mathbf{Z}_p^\top\mathbf{c},\sigma^2\mathbf{I}_{n_p})\\&\propto\mathcal{N}(\mathbf{g}_p;\mathbf{b}_p,\mathbf{K}_p)\mathcal{N}(\mathbf{y}_p-\mathbf{Z}_p^\top\mathbf{c};\mathbf{g}_p,\sigma^2\mathbf{I}_{n_p}).\end{aligned}$$

The second expression has precisely the form of the posterior for a standard GP regression model: the first factor is the GP prior $P(\mathbf{g}_p)$, while the second is the Gaussian likelihood with observations $\mathbf{y}_p-\mathbf{Z}_p^\top\mathbf{c}$ and noise covariance $\sigma^2\mathbf{I}_{n_p}$. Following Rasmussen & Williams (2006), we obtain the posterior GP distribution as:

$$g_p(\cdot)|\mathbf{c},\mathbf{y}_p\sim\mathcal{GP}\big(m_p(\cdot),S_p(\cdot,\cdot)\big),$$

where $m_p(\mathbf{x})=\mathbf{k}_p(\mathbf{x})^\top\mathbf{M}_p^{-1}(\mathbf{y}_p-\mathbf{Z}_p^\top\mathbf{c}-\mathbf{b}_p)+b_p$ and $S_p(\mathbf{x},\mathbf{x}')=\mathcal{K}_p(\mathbf{x},\mathbf{x}')-\mathbf{k}_p(\mathbf{x})^\top\mathbf{M}_p^{-1}\mathbf{k}_p(\mathbf{x}')$. Next, to obtain the posterior $P(\mathbf{c}|\mathbf{y})$, we note that

$$P(\mathbf{c}|\mathbf{y})\propto\mathcal{N}(\mathbf{c};\mathbf{0},\boldsymbol{\Sigma}_c)\prod\nolimits_p\mathcal{N}(\mathbf{y}_p;\mathbf{Z}_p^\top\mathbf{c}+\mathbf{b}_p,\mathbf{M}_p).$$

After some algebra, we can also re-arrange this expression into a quadratic form $P(\mathbf{c}|\mathbf{y})\propto e^{-\frac{1}{2}\mathbf{c}^\top\boldsymbol{\Lambda}_c\mathbf{c}+\mathbf{f}_c^\top\mathbf{c}}$, where $\mathbf{f}_c=\sum_p\mathbf{Z}_p\mathbf{M}_p^{-1}(\mathbf{y}_p-\mathbf{b}_p)$ and $\boldsymbol{\Lambda}_c=\boldsymbol{\Sigma}_c^{-1}+\sum_p\mathbf{Z}_p\mathbf{M}_p^{-1}\mathbf{Z}_p^\top$. Hence, we see that the posterior $P(\mathbf{c}|\mathbf{y})$ is the Gaussian distribution $\mathcal{N}(\boldsymbol{\Lambda}_c^{-1}\mathbf{f}_c,\boldsymbol{\Lambda}_c^{-1})$. Expressions depending on this distribution are computed using the Cholesky decomposition of $\boldsymbol{\Lambda}_c$.

We next give details about the computation of the marginal likelihood.

2.2. Marginal Likelihood

Recall the full joint from (1). Using (2) and dividing out common terms leads to

$$P(\mathbf{y})P(\mathbf{c}|\mathbf{y})=\mathcal{N}(\mathbf{c};\mathbf{0},\boldsymbol{\Sigma}_c)\prod\nolimits_p\mathcal{N}(\mathbf{y}_p;\mathbf{Z}_p^\top\mathbf{c}+\mathbf{b}_p,\mathbf{M}_p).$$

Hence, we obtain the following expression for the log-marginal likelihood $\log P(\mathbf{y})$:

$$\sum\nolimits_p\log\mathcal{N}(\mathbf{y}_p;\mathbf{Z}_p^\top\mathbf{c}+\mathbf{b}_p,\mathbf{M}_p)+\log P(\mathbf{c})-\log P(\mathbf{c}|\mathbf{y}).$$

3. Details about the non-linear extension of `tree`

For space limitation reasons, we have omitted in the core paper the details describing how we use in practice the random Fourier features (Rahimi et al., 2007).

For a given vertex $v\in V$ with feature representation $\mathbf{r}_v\in\mathbb{R}^{D_v}$, we associate the new R_v-dimensional representation

$$\phi_v(\mathbf{r}_v)\triangleq\sqrt{2/R_v}\cdot\cos\big(\mathbf{W}_v\mathbf{r}_v+\mathbf{q}_v\big)$$

where $\cos(\cdot)$ operates element-wise, $\mathbf{W}_v\in\mathbb{R}^{R_v\times D_v}$ has entries identically and independently distributed from $\mathcal{N}(0,\frac{1}{\gamma})$ while $\mathbf{q}_v\in\mathbb{R}^{R_v}$ has its components generated uniformly in $[0,2\pi]$. Those randomly generated features can be shown to approximate a RBF kernel with bandwidth γ^2 (Rahimi et al., 2007).

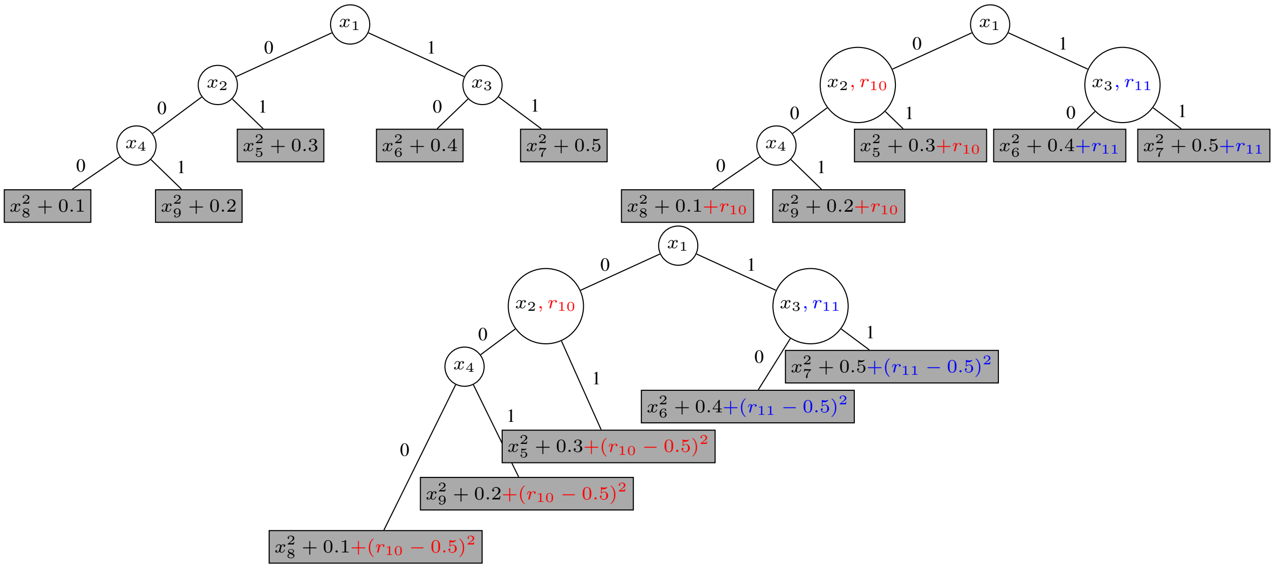

Figure 2. Three examples of functions with tree-structured conditional relationships. Each inner node is a binary variable, and a path in those trees represents successive binary decisions. The leaves contain univariate quadratic functions that are shifted by different constant terms. Compared with Figure 2, the trees are not balanced anymore. *Top left*: Setting without shared variables. *Top right*: r_{10} and r_{11} are shared variables that are common to the functions at the leaves of their respective subtrees. In this example, the shared variables have a linear dependency in the leaf objectives. *Bottom*: Similar to the middle tree, except that r_{10} and r_{11} have a quadratic dependency in the leaf objectives.

In practice, to address the problem of the selection of γ, we consider a strategy consisting of approximating a sum of RBF kernels for different values of γ. Our experiments showed that taking $\gamma \in \{0.1, 1, 10\}$ provided good results. Moreover, we fix the dimension $R_v = 3R$ for all vertices, where R corresponds to the dimension of $\phi_v^\gamma(\mathbf{r}_v) \triangleq \sqrt{2/R_v} \cdot \cos(\mathbf{W}_v^{(\gamma)} \mathbf{r}_v + \mathbf{q}_v)$ for each $\gamma \in \{0.1, 1, 10\}$ with $\mathbf{W}_v^{(\gamma)}$ having its entries drawn from $\mathcal{N}(0, \frac{1}{\gamma})$. The final representation $\phi_v(\mathbf{r}_v) = [\phi_v^\gamma(\mathbf{r}_v)]_{\gamma \in \{0.1,1,10\}}$ is obtained via a concatenation of the ϕ_v^γ's. In practice, we take $R = 25$.

References

Petersen, K. B. and Pedersen, M. S. The matrix cookbook. Technical report, Technical University of Denmark, nov 2012. Version 20121115.

Rahimi, Ali, Recht, Benjamin, et al. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, volume 3, pp. 5, 2007.

Rasmussen, Carl and Williams, Chris. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

Zhang, Yuyu, Bahadori, Mohammad Taha, Su, Hang, and Sun, Jimeng. Flash: Fast bayesian optimization for data analytic pipelines. In Krishnapuram, Balaji, Shah, Mohak, Smola, Alexander J., Aggarwal, Charu, Shen, Dou, and Rastogi, Rajeev (eds.), *KDD*, pp. 2065–2074. ACM, 2016. ISBN 978-1-4503-4232-2.

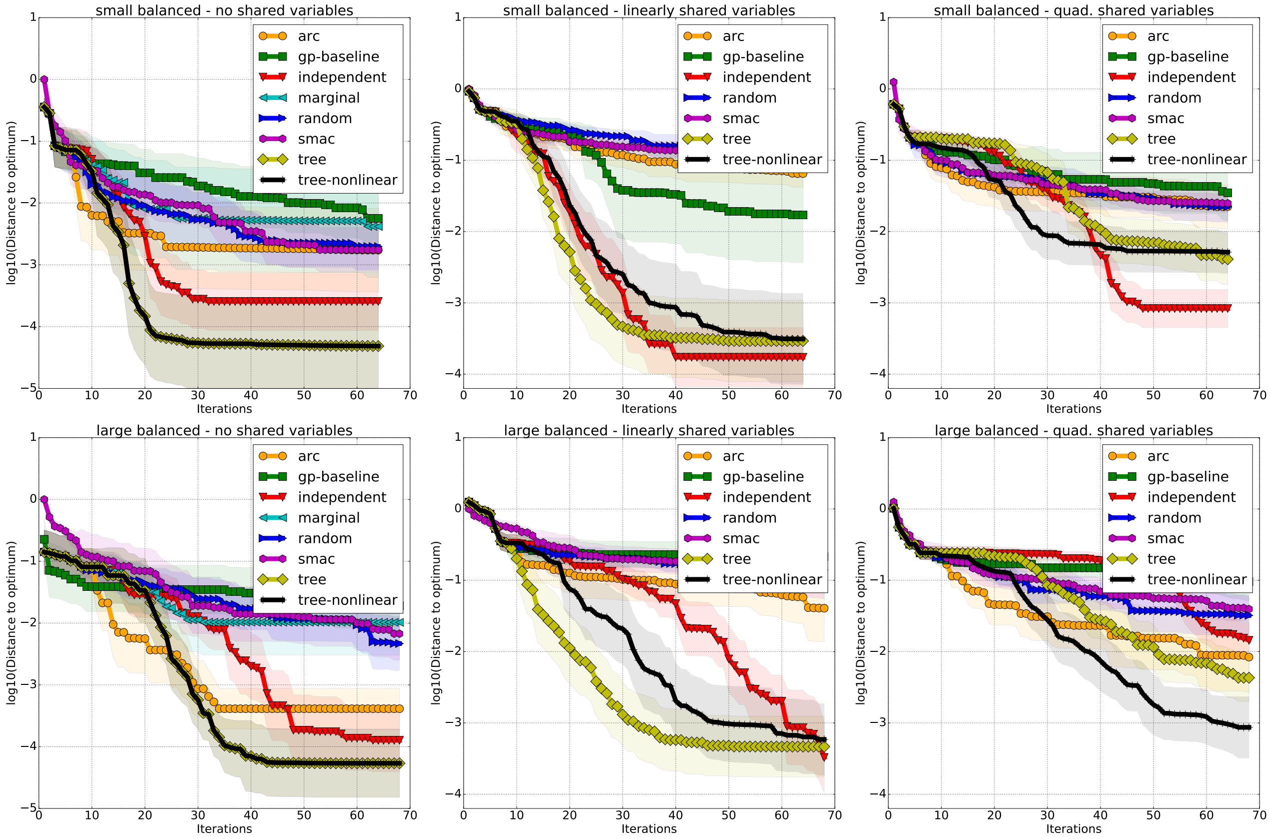

Figure 3. Optimization performance over synthetic tree-structured functions, as measured by the $\log_{10}$ distance to the (known) minimum versus the number of iterations. *Top*: Results for the small balanced binary trees displayed in Figure 1, without (left), with linearly-dependent (middle) and quadratically-dependent (right) shared variables. *Bottom*: Results for larger balanced binary trees with depth 4 and a total of 8 leaves, without (left), with linearly-dependent (middle) and quadratically-dependent (right) shared variables. Best seen in color.

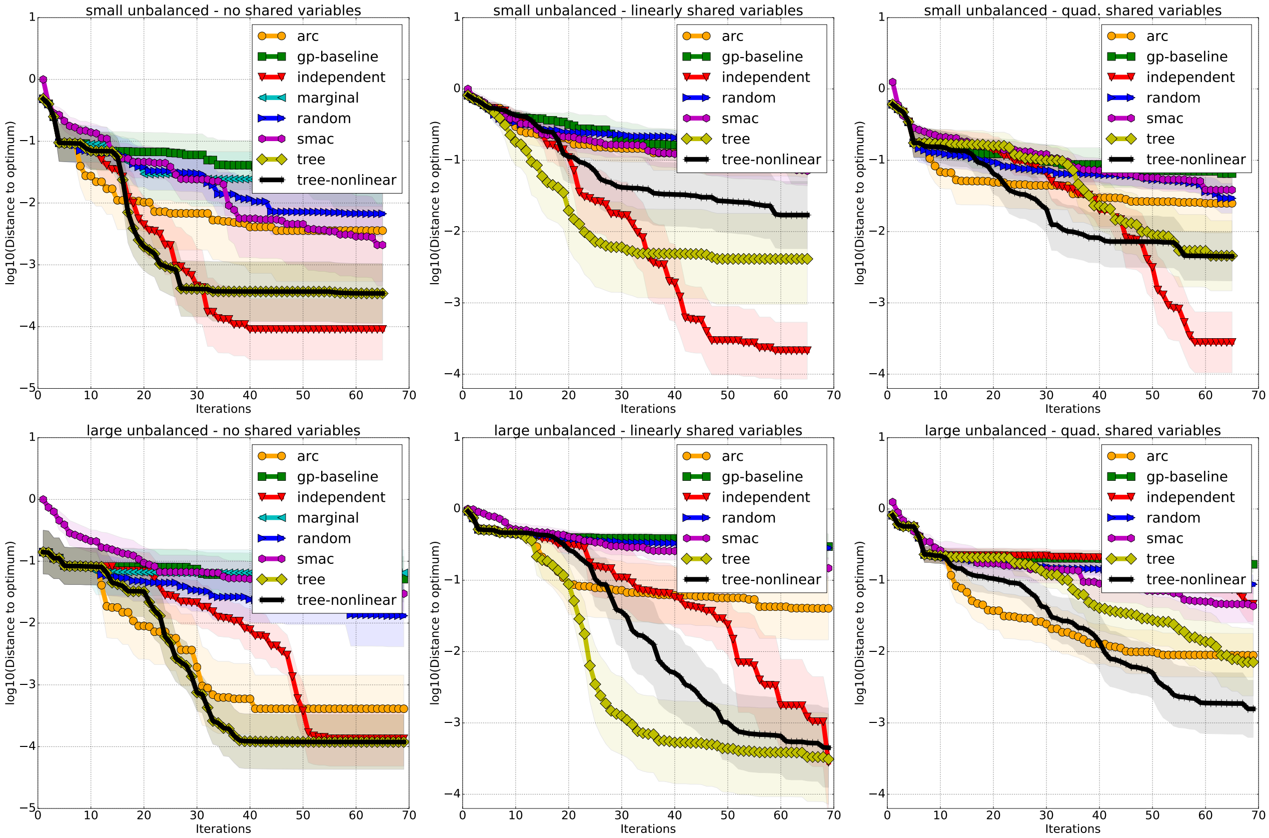

Figure 4. Optimization performance over synthetic tree-structured functions, as measured by the $\log_{10}$ distance to the (known) minimum versus the number of iterations. *Top*: Results for the small unbalanced binary trees displayed in Figure 2, without (left), with linearly-dependent (middle) and quadratically-dependent (right) shared variables. *Bottom*: Results for the large unbalanced binary trees with depth 4 and a total of 9 leaves, without (left), with linearly-dependent (middle) and quadratically-dependent (right) shared variables. Best seen in color.

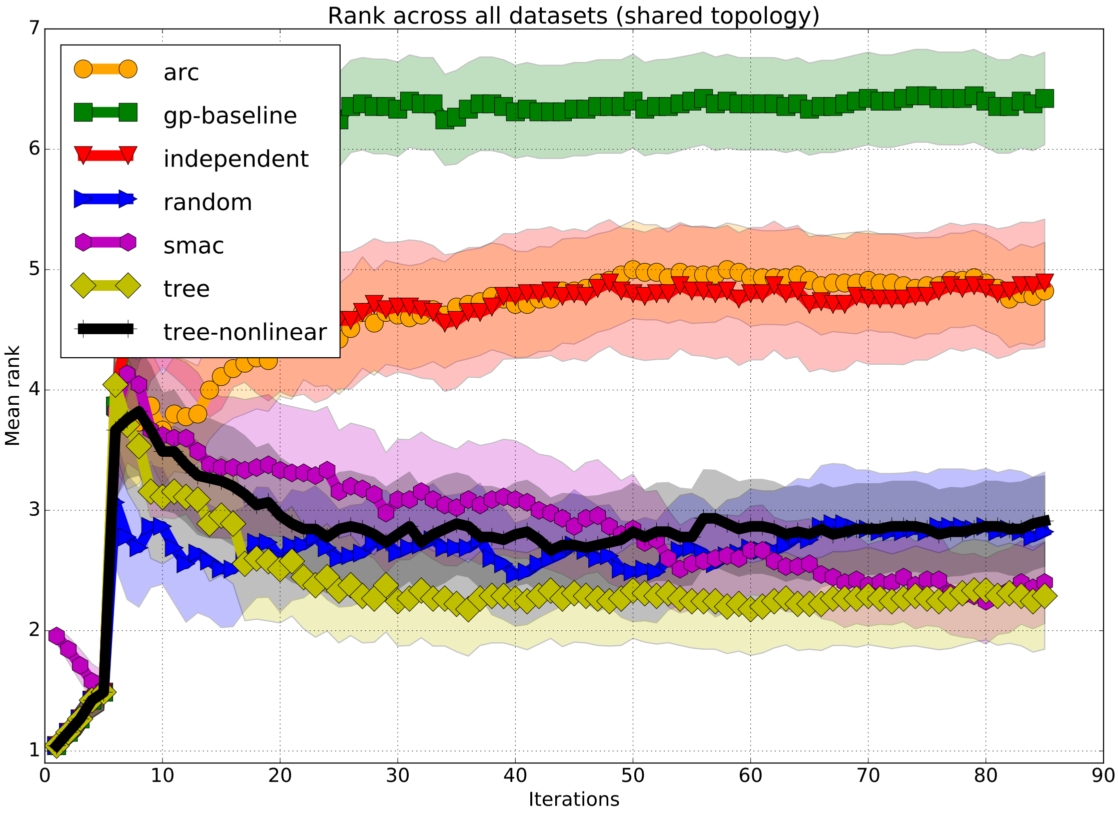

Figure 5. Tuning of a multi-layer perceptron for binary classification. Average rank aggregated over 45 datasets versus the number of iterations (lower is better; see details in the main article). Best seen in color.