Online Learning with Bayesian Classification Trees

Samuel Rota Bulò
FBK-irst
Trento, Italy
rotabulo@fbk.eu

Peter Kontschieder
Microsoft Research
Cambridge, UK
Mapillary
Graz, Austria
pkontschieder@gmail.com

Abstract

Randomized classification trees are among the most popular machine learning tools and found successful applications in many areas. Although this classifier was originally designed as offline learning algorithm, there has been an increased interest in the last years to provide an online variant. In this paper, we propose an online learning algorithm for classification trees that adheres to Bayesian principles. In contrast to state-of-the-art approaches that produce large forests with complex trees, we aim at constructing small ensembles consisting of shallow trees with high generalization capabilities. Experiments on benchmark machine learning and body part recognition datasets show superior performance over state-of-the-art approaches.

1. Introduction

Random classification forests are often the preferred machine learning tool due to their efficiency, scalability, and robustness. They found successful application in many areas such as computer vision [21], bio-informatics [26], medical data analysis [9], data-mining [24], *etc*.

Random forests are usually trained in batch (or offline) modality, *i.e*. all training data is expected to be given in advance and once a forest is trained it cannot be updated with new, labelled samples. However, there are application domains where the training data is not available in advance, but is collected over time, and inference might be required at any moment. Online learning approaches can serve such purposes and have a number of advantages over batch methods: they are more flexible as they can work in the presence of online data generation processes, can adapt to distributions varying over time, do not need to store the training set during the learning phase, to name just a few.

Joining the online learning paradigm with classification trees is particularly difficult due to the recursive nature of this classifier. Moreover, the presence of (typically) deterministic split decisions within the tree complicates using new data samples to correct decisions that were taken at an earlier level. Consequently, the number of works addressing online learning within decision trees in the literature is rather small and we review the most related ones next. The Hoeffding tree algorithm [11] maintains a set of candidate splits in the leaves and tracks their quality as new data arrives. The Hoeffding bound is used to control the amount of data that should be collected before a probably-optimal split selection can be ensured. In [20] a similar idea is pursued, but the leaf splitting condition changes and an online bagging [19] strategy is adopted. A modification over [20] was presented in [23], which uses reservoir sampling to keep track of a fixed-length, unbiased set of training samples for updating the trees. The work in [2] also extends the Hoeffding tree algorithm by introducing an adaptive-size version, allowing to mix differently-sized trees. Their approach imposes restrictions on the number of split nodes such that shallower trees can adapt more quickly to changes in the distribution of the incoming data stream while deeper trees adapt more slowly and therefore maintain a longer-term memory. The work in [10] also maintains sets of candidate splits in the leaf nodes. There, the authors provide a decision forest construction algorithm, which dynamically partitions the data into structure and estimation samples, the former being used to influence the tree structure and the latter being used to estimate the leaf predictions. The work of [13] presents an alternative approach by governing the tree growing phase by a Mondrian process. There, label distributions are kept at each node and controlled by a hierarchy of normalized, stable processes.

A remaining limitation of state-of-the-art online algorithms for random forests is a tendency to producing oversized trees, as the split selection process is unaware of the tree complexity and split functions are very simple. Moreover, trees tend to overfit, thus requiring large forests to achieve good performance, which however implies slowdown of the inference process and increased memory requirements. In particular, the algorithms that keep candidate splits in the leaves during training suffer from a prohibitive memory cost as trees get deep [10] or require multiple passes over the training data [10, 20].

Contributions. In this paper, we introduce a novel online learning algorithm for Bayesian classification trees that tries to overcome the aforementioned limitations, by taking a different perspective. Our goal is to learn tree classifiers with a shallow structure and high generalization capability. We trade shallow tree structures for more complex split decision functions that jointly take into account multiple feature dimensions and their correlations. Our online learning procedure is a Bayesian approach that iteratively replaces a posterior distribution over trees, obtained after observing a new training sample, with a simpler, parametric distribution. This surrogate posterior is determined within the parametric family in a way to fit the updated posterior distribution with the minimum loss of information. Our algorithm is characterized by update rules for the tree hyper-parameters that are free from cumbersome learning rate selection and allow us to naturally absorb the information carried by each new sample. Due to the Bayesian learning principle, which takes the uncertainty about the tree's parameters into account, we can obtain tree classifiers that do not overfit. In summary, the proposed method matches our initial intention of obtaining ensembles containing few, shallow and well-generalizing trees. The provided experimental evaluation confirms our model choice and its benefits are shown with quantitative experiments on standard benchmark machine learning and more complex body part recognition datasets.

Relationship to Bayesian tree models. Bayesian models for decision trees have appeared in the literature for offline learning – namely the Bayesian hierarchical mixture of experts (BHMEs) [3], Bayesian CART models [5, 6, 8, 25], Bayesian BART models [7] – and for the online learning modality with dynamic tree models [22, 1]. Our model differentiates from these approaches since we are proposing a parametric, Bayesian model for decision trees that is trained in an online fashion, *i.e*. we update over time a posterior distribution over the space of decision trees, whereas other similar approaches like BHMEs (despite sharing structural similarities) work in an offline fashion and consider sigmoid-gated hierarchical mixture of experts in the underlying model hypothesis space.

2. Classification Trees

In this section we recap classification trees to provide all the necessary notation and definitions that are subsequently used for introducing our Bayesian online learning approach.

Consider a classification problem with input space $\mathcal{X}$ and a finite set of categorical labels $\mathcal{Y}$. A *classification tree* is a classifier consisting of decision nodes and prediction nodes, arranged into a tree-structure. Decision nodes correspond to the tree's internal nodes $\mathcal{N}$ and are responsible for routing data samples to an appropriate prediction node (*i.e*. leaf) in $\mathcal{L}$. Each decision node $n \in \mathcal{N}$ takes a routing decision for a data sample $\boldsymbol{x} \in \mathcal{X}$ via a *routing function* $b_n : \mathcal{X} \to \{0, 1\}$. If $b_n(\boldsymbol{x}) = 1$, then $\boldsymbol{x}$ is routed to the left sub-tree, otherwise it goes to the right one. Akin to conventional decision trees, we consider binary decision functions of the following type:

$$b_n(\boldsymbol{x}) = \mathbb{1}_{\boldsymbol{\theta}_n^\top \xi_n(\boldsymbol{x}) \geq 0} \,, \tag{1}$$

where $\mathbb{1}_P$ is an indicator function for the truth value of P. The decision function b_n depends on a node-specific feature map $\xi_n : \mathcal{X} \to \mathbb{R}^{\mathsf{d}_n}$, and a d_n-dimensional parameter vector $\boldsymbol{\theta}_n \in \mathbb{R}^{\mathsf{d}_n}$ (see also [15]). Starting from the root node and after visiting a number of decision nodes, $\boldsymbol{x}$ ends up in a prediction node, where the actual class assignment takes place. Indeed, each prediction node $\ell \in \mathcal{L}$ holds a probability distribution $\boldsymbol{\pi}_\ell = (\pi_{\ell y})_{y \in \mathcal{Y}}$ over labels in $\mathcal{Y}$ that will be used to deliver the final prediction for the data sample reaching it.

A classification tree, denoted by t, is identified by its structure S and its parameters, *i.e*. $t = (\boldsymbol{\theta}, \boldsymbol{\pi}, S)$, where $\boldsymbol{\theta} = \{\boldsymbol{\theta}_n\}_{n \in \mathcal{N}}$ holds the parameters of all decision nodes and $\boldsymbol{\pi} = \{\boldsymbol{\pi}_\ell\}_{\ell \in \mathcal{L}}$ contains the class distributions of all prediction nodes. The structure of the tree comprises the set of nodes $\mathcal{N}$, leaves $\mathcal{L}$ and their relations. Moreover, it includes the node-specific feature maps ξ_n.

Given a tree $t = (\boldsymbol{\theta}, \boldsymbol{\pi}, S)$, the predictive distribution $p\left(y|t; \boldsymbol{x}\right)$ for data sample $\boldsymbol{x}$ is defined as

$$p\left(y|t; \boldsymbol{x}\right) = \sum_{\ell \in \mathcal{L}} r(\ell|t; \boldsymbol{x}) \pi_{\ell y} \,, \tag{2}$$

where $\pi_{\ell y}$ denotes the probability of a sample ending in leaf ℓ to take on class y, and $r(\ell|\boldsymbol{x})$ is regarded as the *routing function*, which is 1 for the leaf ℓ where sample $\boldsymbol{x}$ is actually routed to, and 0 elsewhere.

To give an explicit form to the routing function, we introduce the following binary relations that depend on the tree structure: $\ell \swarrow n$ is true if ℓ belongs to the left subtree of node n, and $n \searrow \ell$ is true if ℓ belongs to the right subtree of n. By exploiting these relations we can factorize the routing function as

$$r(\ell|t; \boldsymbol{x}) = \prod_{n \in \mathcal{N}} b_n(\boldsymbol{x})^{\mathbb{1}_{\ell \swarrow n}} (1 - b_n(\boldsymbol{x}))^{\mathbb{1}_{n \searrow \ell}} \,. \tag{3}$$

Although the product in (3) runs over all nodes, only the decision nodes along the path from the root node to the leaf ℓ are affected. [1]

3. Bayesian Online Classification Trees

In this section we present our novel online learning algorithm for classification trees, adhering to Bayesian principles. Our approach falls within the theoretical framework

[1] $\mathbb{1}_{\ell \swarrow n}$ and $\mathbb{1}_{n \searrow \ell}$ are both zero for all n not being ancestors of ℓ. Hence, the corresponding factors in (3) yield 1 when assuming $0^0 = 1$.

of *Bayesian Online Learning* (BOL) [18], also known as *assumed density filtering* in the control literature [14], and is related to *expectation propagation* [17].

3.1. Overview

Let $\mathcal{D}_i = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_i, y_i)\} \subset \mathcal{X} \times \mathcal{Y}$ denote a collection of i labelled examples, and assume to have a prior distribution $p(t)$ defined over the set of decision trees. In standard Bayesian inference it is possible to compute the posterior distribution of t given the collection of data points $\mathcal{D}_i$ by recursively employing the Bayes rule in the following way:

$$p\left(t; \mathcal{D}_i\right) \propto p\left(y_i | t; \boldsymbol{x}_i\right) p\left(t; \mathcal{D}_{i-1}\right) \,, \tag{4}$$

where $p\left(t; \mathcal{D}_0\right) = p(t)$. This formula captures the change in the posterior distribution due to an added training sample $(\boldsymbol{x}_i, y_i)$, when the posterior distribution after $i-1$ samples is treated as the new prior for the incoming data point.

Although the rule in (4) seems to be structurally suitable for an online scenario, it cannot instantly be used for online learning, because in general it requires knowledge about the *entire* training set. However, if we could store the posterior from the previous $(i-1)$ samples, and compute the normalizing constant, we would obtain an online learning approach by repeatedly applying (4), without the need of revisiting past samples. The BOL framework implements this idea by recursively building a *surrogate* distribution for the *true* posterior $p\left(t; \mathcal{D}_i\right)$. The surrogate distribution is confined to a pre-defined, parametric family of distributions $\mathcal{Q}$, which can be compactly stored. In the rest of the paper we will denote by $q(t; h_i)$, or more compactly $q^i(t)$, the surrogate distribution of the true posterior of t given i samples, h_i being the parametrization of the distribution.

The recursive construction of this surrogate distribution alternates a Bayes update step, integrating information conveyed by new training data akin to (4), with a projection step, which re-maps the obtained distribution in the parametric family $\mathcal{Q}$.

Update step. Let $q^i(t)$ be the surrogate posterior distribution of t from i samples, which is taken as prior for the update step of a new data sample $(\boldsymbol{x}_{i+1}, y_{i+1})$. By application of the Bayes rule we obtain the following updated posterior distribution:

$$\hat{q}^{i+1}(t) \propto p\left(y_{i+1} | t; \boldsymbol{x}_{i+1}\right) q^i(t) \,. \tag{5}$$

The family $\mathcal{Q}$, where q^i belongs to, is typically selected in a way to make the computation of (5) tractable. This property however is not necessarily preserved by the distribution $\hat{q}^{i+1}$ in case we use it for a subsequent update. For this reason, a projection step is required.

Projection step. The projection step finds the best approximation of $\hat{q}^{i+1}$ within the parametric family $\mathcal{Q}$, in a way to minimize the loss of information. In this regard, the sought distribution $q^{i+1} \in \mathcal{Q}$ is the one that minimizes the following Kullback-Leibler (KL) divergence:

$$q^{i+1} \in \arg\min_{q \in \mathcal{Q}} D_{\mathrm{KL}}\left(\hat{q}^{i+1} \| q\right) \,. \tag{6}$$

After performing the projection, q^{i+1} can be regarded as a surrogate of the true posterior distribution $p(t; \mathcal{D}_{i+1})$ of t given $i+1$ samples, which can be used as a new prior distribution for subsequent updates (see, Fig. 1).

Inference. At any time, we can use the current surrogate posterior distribution, say $q^i(t)$, to compute the *posterior predictive distribution* for a new data sample $\boldsymbol{x}$. The posterior predictive distribution, denoted by $p(y; \boldsymbol{x}, h_i)$, provides the expected class distribution that we obtain for $\boldsymbol{x}$ with a decision tree t sampled from the surrogate posterior distribution q^i:

$$p(y; h_i, \boldsymbol{x}) = \mathbb{E}_{q^i}[p(y|t; \boldsymbol{x})] \tag{7}$$

where $\mathbb{E}_{q^i}[\cdot]$ denotes expectation with respect to q^i. In the rest of the paper we will refer to (7) as the posterior predictive distribution for convenience, but the reader should keep in mind that it is actually an approximation of the *true* posterior predictive distribution $p\left(y; \boldsymbol{x}, \mathcal{D}_i\right)$ that one obtains in standard (offline) Bayesian inference from the observed set of labelled samples $\mathcal{D}_i$. Indeed, we replace the true posterior $p\left(t; \mathcal{D}_i\right)$ with the surrogate posterior $q^i(t)$, which has been sequentially estimated from $\mathcal{D}_i$ as previously detailed.

3.2. Surrogate Posterior for Classification Trees

The key component of the online learning algorithm described above is the surrogate posterior. On one hand, we

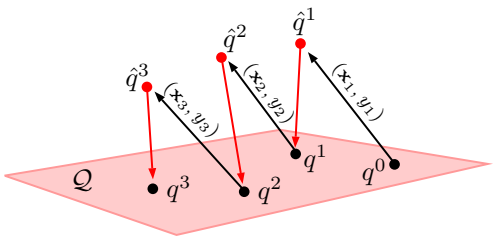

Figure 1. Example of the Bayesian online learning process: we start with a prior distribution $q^0(t)$ over the set of decision trees, and apply the update rule in (5) to incorporate the information from the first sample $(\boldsymbol{x}_1, y_1)$. The obtained posterior $\hat{q}^1(t)$ is then reprojected in the parametric family of distributions $\mathcal{Q}$ using (6). The resulting distribution $q^1(t)$ is a surrogate distribution of the true posterior $p(t; \mathcal{D}_1)$, which can be used as a new prior for the next update step. We keep iterating this process as new training samples arrive. At any moment, the most recent surrogate posterior q^i can be used for inference on unlabelled data samples.

would like to keep it simple such that the projection step in (6) and the computation of the posterior predictive distribution in (7) remain tractable. On the other hand, we would like to have it complex and multi-modal to best possibly capture the true posterior distribution. The solution we propose is a compromise between these two contradicting requirements.

Unimodal with fixed structure. Assume in first place to have a pre-defined tree structure $\hat{S}$. The surrogate posterior over trees $t = (\boldsymbol{\theta}, \boldsymbol{\pi}, S)$ can be defined as a factorization of independent distributions over the tree's parameters, which takes the following parametric form:

$$q(t; h) = \delta_{\hat{S}}(S) \prod_{n \in \mathcal{N}} q_n(\boldsymbol{\theta}_n; \Sigma_n, \boldsymbol{\mu}_n) \prod_{\ell \in \mathcal{L}} q_\ell(\boldsymbol{\pi}_\ell; \boldsymbol{\alpha}_\ell) \,. \quad (8)$$

The first term in (8) is a Dirac measure that supports only trees with structure $\hat{S}$. Each decision function's parameter $\boldsymbol{\theta}_n$ follows a multivariate Gaussian with mean $\boldsymbol{\mu}_n$ and covariance Σ_n, *i.e.* $q_n \sim \texttt{Gauss}(\boldsymbol{\mu}_n, \Sigma_n)$, while each prediction node's class distribution $\boldsymbol{\pi}_\ell$ follows a Dirichlet distribution with parameter $\boldsymbol{\alpha}_\ell$, *i.e.* $q_\ell \sim \texttt{Dir}(\boldsymbol{\alpha}_\ell)$ (see Fig. 2). The argument $h = (\hat{S}, \Sigma, \boldsymbol{\mu}, \boldsymbol{\alpha})$ of q holds all the parameters of the distribution (*a.k.a.* hyperparameters of the tree).

The distribution in (8) is unimodal for most parametrizations and, under this modelling choice, the projection step in (6) turns into the following, independent minimizations over the different parameters of q^i (see, Subsection A.1 of the supplementary material):

$$(\Sigma_n^{i+1}, \boldsymbol{\mu}_n^{i+1}) \in \arg\min_{\Sigma, \boldsymbol{\mu}} \mathbb{E}_{\hat{q}^{i+1}}[-\log\, q_n(\boldsymbol{\theta}_n; \Sigma, \boldsymbol{\mu})] \,, \quad (9)$$

$$\boldsymbol{\alpha}_\ell^{i+1} \in \arg\min_{\boldsymbol{\alpha}} \mathbb{E}_{\hat{q}^{i+1}}[-\log q_\ell(\boldsymbol{\pi}_\ell; \boldsymbol{\alpha})] \,, \quad (10)$$

where the expectations are with respect to the updated posterior $\hat{q}^{i+1}$. Moreover, the structure of the tree is preserved by the update in (6), *i.e.* $\hat{S}^{i+1} = \hat{S}^i$. In Sec. 4, we describe how to solve (9) and (10).

Multi-modal. A simple way to better capture the true posterior distribution consists in modeling the surrogate posterior as a uniform mixture of distributions

$$q(t; h^1, \ldots, h^{\mathsf{m}}) = \frac{1}{\mathsf{m}} \sum_{j=1}^{\mathsf{m}} q(t; h^j) \,, \quad (11)$$

where $q(t; h^j)$ is defined as in (8). In such a way, each of the mixture components might consider a different tree structure (including possibly different feature maps ξ_n per node). Accordingly, the tree structures are still assumed to be given, but multiple structures can now be integrated into a single, multi-modal distribution. The projection step in (6) for the multi-modal posterior can be approximated by independent projections of each single surrogate posterior forming the mixture in (11) (see, Subsection A.2 in the supplementary material). Finally, the posterior predictive distribution under the multi-modal posterior has the simple closed-form

$$p(y; h_i^1, \ldots, h_i^{\mathsf{m}}, \boldsymbol{x}) = \frac{1}{\mathsf{m}} \sum_{j=1}^{\mathsf{m}} p(y; h_i^j, \boldsymbol{x}) \,, \quad (12)$$

where h_i^j is the parameter of the surrogate posterior distribution of t obtained from i samples for the jth mixture component in (11). Please note that in (12), we average the posterior predictive distribution over m online-trained Bayesian trees, as known from conventional forests [4, 9]. Next, we will focus on algorithmic details (*e.g.* parameter updates, implementation notes, *etc.*) of individual trees.

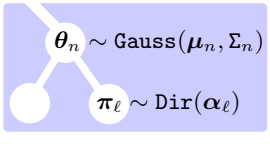

Figure 2. The parameters of the tree are given by $\boldsymbol{\theta}_n$ for each decision node n and $\boldsymbol{\pi}_\ell$ for each prediction node ℓ. Each decision node's parameter $\boldsymbol{\theta}_n$ is a multivariate Gaussian with mean $\boldsymbol{\mu}_n$ and covariance Σ_n. Each prediction node's parameter $\boldsymbol{\pi}_\ell$ is a Dirichlet-variate with concentration vector $\boldsymbol{\alpha}_\ell$.

4. Algorithmic Details

In this section we provide further details about the form of the posterior predictive distribution in (7) as well as the update formulae for the surrogate posterior's parameters, providing the solutions for (9) and (10). Due to a lack of space we omit the full derivations and a detailed discussion about the computational complexity of our model, which are however available in the supplementary material.

Posterior predictive distribution. The posterior predictive distribution $p\,(y; h_i, \boldsymbol{x})$ as given in (7) can be computed in closed-form as follows:

$$p\,(y; h_i, \boldsymbol{x}) = \sum_{\ell \in \mathcal{L}} \frac{\alpha_{\ell y}^i}{|\boldsymbol{\alpha}_\ell^i|_1} \rho(\ell; h_i, \boldsymbol{x}) \,. \quad (13)$$

This distribution is the counterpart of (2), which we obtain by marginalizing out the tree using the surrogate posterior distribution $q(t; h_i)$. The first term in the summation is the expectation of $\pi_{\ell y}$ under the Dirichlet distribution with parameter $\boldsymbol{\alpha}_\ell^i$, where $|\cdot|_1$ is the ℓ_1 norm. The second term is a stochastic routing function, which takes the form:

$$\rho(\ell; h_i, \boldsymbol{x}) = \prod_{n \in \mathcal{N}} \beta_n^i(\boldsymbol{x})^{\mathbb{1}_{\ell \swarrow n}} (1 - \beta_n^i(\boldsymbol{x}))^{\mathbb{1}_{n \searrow \ell}} \,, \quad (14)$$

where $\beta_n^i(\boldsymbol{x})$ represents the following probability of sample $\boldsymbol{x}$ to be routed to the left child at node n in a random tree t distributed as $q(t; h_i)$:

$$\beta_n^i(\boldsymbol{x}) = \Phi(\boldsymbol{\mu}_n^{i\top} \tilde{\xi}_n^i(\boldsymbol{x}))\,.$$

Function Φ is the cumulative distribution function of the standard normal distribution, and $\tilde{\xi}_n^i$ is a Σ_n^i-normalized version of the node-specific feature map ξ_n, given by

$$\tilde{\xi}_n^i(\boldsymbol{x}) = \xi_n(\boldsymbol{x}) \left[\xi_n(\boldsymbol{x})^\top \Sigma_n^i \xi_n(\boldsymbol{x})\right]^{-1/2}\,. \tag{15}$$

Inuitively, β_n^i represent a softening of the hard decision rule b_n in (1), induced by the uncertainty about the decision node's parametrization.

Remark 1 *Function* (14) *is still valid if we replace ℓ with any other node m in the tree, and in that case $\rho(m; h_i, \boldsymbol{x})$ provides the probability of reaching node m. We will use this later in this section. Moreover, we will also use a variant of the posterior predictive distribution, denoted by $p(y|m; h_i, \boldsymbol{x})$, which conditions on a node m of the tree. This provides the posterior predictive distribution if we took m as the starting node for the prediction.*

Update rule for $\boldsymbol{\mu}$ and Σ. The update rules for $\boldsymbol{\mu}$ and Σ can be obtained by solving the cross-entropy minimization problem in (9). Since the Gaussian distribution is in the exponential family, we can determine a solution to the aforementioned minimization problem by moment-matching. The resulting update rules are given by

$$\boldsymbol{\mu}_n^{i+1} = \boldsymbol{\mu}_n^i + \kappa_n \Sigma_n^i \tilde{\xi}_n^i\,, \tag{16}$$

$$\Sigma_n^{i+1} = \Sigma_n^i - \left(\kappa_n^2 + \kappa_n \boldsymbol{\mu}_n^{i\top} \tilde{\xi}_n^i\right) (\Sigma_n^i \tilde{\xi}_n^i) \left(\Sigma_n^i \tilde{\xi}_n^i\right)^\top\,, \tag{17}$$

where we wrote $\tilde{\xi}_n^i$ as a shortcut for $\tilde{\xi}_n^i(\boldsymbol{x}_{i+1})$ and

$$\kappa_n = \phi\left(\boldsymbol{\mu}_n^{i\top} \tilde{\xi}_n^i\right) (u_{n_\mathsf{L}} - u_{n_\mathsf{R}})\, a_n\,, \tag{18}$$

where $\phi(\cdot)$ is the probability density function of the standard normal distribution, n_L and n_R denote the left and right child of node n, respectively, $a_n = \frac{\rho(n; h_i, \boldsymbol{x}_{i+1})}{p(y_{i+1}; h_i, \boldsymbol{x}_{i+1})}$, and $u_n = p(y_{i+1}|n; h_i, \boldsymbol{x}_{i+1})$.

Remark 2 (Scale invariance) *The posterior predictive distribution will not change if we scale each mean $\boldsymbol{\mu}_n^i$ by c_n and each covariance Σ_n^i by c_n^2 for any $c_n > 0$ (*e.g. *$c_n = 1/\|\boldsymbol{\mu}_n^i\|_2$). Moreover, the update rules in* (16) *and* (17) *are consistent under the same transformation,* i.e. *we obtain $c_n \boldsymbol{\mu}_n^{i+1}$ and $c_n^2 \Sigma_n^{i+1}$ if we transform $\boldsymbol{\mu}_n^i$ and Σ_n^i in the same way. Hence, we can safely scale $\boldsymbol{\mu}_n^{i+1}$ and Σ_n^{i+1} in this sense after each update round, without affecting the outcome of the algorithm. This helps to avoid numerical instabilities.*

Update rule for $\boldsymbol{\alpha}$. The update rule for $\boldsymbol{\alpha}$ can be determined by solving (10). Since $\boldsymbol{\pi}_\ell$ follows a Dirichlet distribution with parameter $\boldsymbol{\alpha}_\ell$, we obtain the following moment-matching equations:

$$\mathbb{E}_{q_\ell^{i+1}}[\log(\boldsymbol{\pi}_\ell)] = \mathbb{E}_{\hat{q}^{i+1}}[\log(\boldsymbol{\pi}_\ell)]\,. \tag{19}$$

With some manipulations of the two expectation terms, we end up with the following system of equalities for $z \in \mathcal{Y}$:

$$\begin{aligned}&\psi(\alpha_{\ell z}^{i+1}) - \psi(|\boldsymbol{\alpha}_\ell^{i+1}|) \\ &\qquad = \psi(\alpha_{\ell z}^i) - \psi(|\boldsymbol{\alpha}_\ell^i|) + \frac{a_\ell(\mathbb{1}_{z=y_{i+1}} - u_\ell)}{|\boldsymbol{\alpha}_\ell^i|_1}\end{aligned} \tag{20}$$

where a_ℓ and u_ℓ are defined as in (18), and $\psi(\cdot)$ is the digamma function. We solve the system using Newton-Raphson iterations, where few iterations (typically 5–10) are necessary to achieve a good accuracy. This provides us with a solution to (10) as required. We refer to [16] for the derivation of the fixed-point iterations.

Prior distribution. We select a prior distribution from the same family $\mathcal{Q}$ as the surrogate posterior given in (8) and it is identified by the timestamp $i = 0$, *i.e.* $p(t) = q^0(t)$. We instantiate a prior distribution by providing a tree structure $\hat{S}^0$ with some pre-defined depth and with randomized feature map functions ξ_n in each node having the form, $\xi_n(\boldsymbol{x}) = [\mathtt{P}_n \boldsymbol{x}; 1]$, where $\mathtt{P}_n$ is a projection matrix and 1 accounts for a bias term. The prior terms for the decision nodes' parameters are improper, flat priors with mean $\boldsymbol{\mu}^0 = \mathbf{0}$ and $\Sigma^0 \to \infty \mathtt{I}$, which induces the following update once the first sample is observed: $\boldsymbol{\mu}_n^1 = \xi_n(\boldsymbol{x}_1)/\|\xi_n(\boldsymbol{x}_1)\|_2$ and $\Sigma_n^1 = \mathtt{I}/\kappa_n^2 - \boldsymbol{\mu}_n^1 \boldsymbol{\mu}_n^{1\top}$, where κ_n is computed as per (18) with $\tilde{\xi}_n^0 = \mathbf{0}$. The prior parameters for the prediction nodes are uniformly sampled in the range $(0, \epsilon]$, *i.e.* $0 < \alpha_{\ell y}^0 \leq \epsilon$, where $0 < \epsilon \ll 1$ is a small, non-negative constant, with the exception of having one peaked preference per class uniformly distributed across the leaves.

Implementation notes The update of the surrogate tree posterior distribution after having seen a new training sample $(\boldsymbol{x}_{i+1}, y_{i+1})$ can be carried out by traversing the tree twice. The first traversal is top-down and computes $\rho_n = \rho(n|h_i, \boldsymbol{x}_{i+1})$ for each node n, which is required in the definition of κ_n, in (18) and in (20). This is done by initializing $\rho_\top = 1$, where $\top \in \mathcal{N}$ is the root node, and by computing for each decision node $n \in \mathcal{N}$ visited in breadth-first order $\rho_{n_\mathsf{L}} = \beta_n^i(\boldsymbol{x}_{i+1})\rho_n$ and $\rho_{n_\mathsf{R}} = \rho_n - \rho_{n_\mathsf{L}}$, where n_L and n_R are the left and right child of node n. It is then possible to run over the leaves $\ell \in \mathcal{L}$ to compute $u_\ell = \alpha_{\ell y_{i+1}}^i / |\boldsymbol{\alpha}_\ell^i|_1$ and finally obtain the posterior predictive probability $p(y_{i+1}; h_i, \boldsymbol{x}_{i+1}) = \sum_{\ell \in \mathcal{L}} \rho_\ell u_\ell$. We have then all the required quantities to compute $\boldsymbol{\alpha}_\ell^{i+1}$ for each

Algorithm 1 Online learning of Bayesian classification tree

Require: $(\boldsymbol{x}_{i+1}, y_{i+1})$: next training sample
Require: h_i: latest surrogate posterior parameter (if $i > 0$)
Require: $\hat{S}^0$: a tree structure (if $i = 0$)
if $i = 0$ **then**
  initialize $\boldsymbol{\alpha}^0$, $\boldsymbol{\mu}^0$ and Σ^0 (see, **Prior distribution**)
▷ Forward pass over tree computes $\rho_n = \rho(n|h_i, \boldsymbol{x}_{i+1})$
$\rho_\top \leftarrow 1$
for all $n \in \mathcal{N}$ in top-down, breadth-first order **do**
  $\rho_{n_\mathsf{L}} \leftarrow \beta_n^i(\boldsymbol{x}_{i+1})\rho_n$ ▷ n_L: left child of n
  $\rho_{n_\mathsf{R}} \leftarrow \rho_n - \rho_{n_\mathsf{L}}$ ▷ n_R: right child of n
$u_\ell \leftarrow \frac{\alpha^i_{\ell y_{i+1}}}{|\boldsymbol{\alpha}^i_\ell|_1}, \forall \ell \in \mathcal{L}$
$p\left(y_{i+1}; h_i, \boldsymbol{x}_{i+1}\right) \leftarrow \sum_{\ell \in \mathcal{L}} \rho_\ell u_\ell$
compute $\boldsymbol{\alpha}_\ell^{i+1}$ by solving (20), $\forall \ell \in \mathcal{L}$
▷ Backward pass over tree: $u_n = p\left(y_{i+1}|n; h_i, \boldsymbol{x}_{i+1}\right)$
for all n in bottom-up, breadth-first order **do**
  $u_n \leftarrow u_{n_\mathsf{R}} + (u_{n_\mathsf{L}} - u_{n_\mathsf{R}})\beta_n^i(\boldsymbol{x}_{i+1})$
  compute κ_n as per (18)
  if $i = 0$ **then** ▷ Prior initialization
    $\boldsymbol{\mu}_n^1 \leftarrow \frac{\xi_n(\boldsymbol{x}_{i+1})}{\|\xi_n(\boldsymbol{x}_{i+1})\|_2}$
    $\Sigma_n^1 \leftarrow \mathtt{I}/\kappa_n^2 - \boldsymbol{\mu}_n^1 \boldsymbol{\mu}_n^{1\top}$
  else
    compute $\boldsymbol{\mu}_n^{i+1}$ as per (16)
    compute Σ_n^{i+1} as per (17)
    rescale $\boldsymbol{\mu}_n^{i+1}$ and Σ_n^{i+1} as per Remark 2
return h_{i+1}: new surrogate posterior parameters

prediction node ℓ by solving the system (20). The second traversal is bottom-up and computes the updates for the decision nodes' hyperparameters. We run again over the nodes $n \in \mathcal{N}$, but in bottom-up, breadth-first order. Once a node n is visited, we compute $u_n = u_{n_\mathsf{R}} + (u_{n_\mathsf{L}} - u_{n_\mathsf{R}})\beta_n^i(\boldsymbol{x}_{i+1})$ and κ_n as per (18). Finally, we calculate Σ_n^{i+1} and $\boldsymbol{\mu}_n^{i+1}$ using (16) and (17), since all the required quantities are available. A summary is provided in Alg. 1.

Fast, single path inference During inference, exact computation of the posterior predictive distribution requires traversing the tree entirely. The complexity is thus $O(|\mathcal{N}|\mathsf{d}^2)$, where d is the maximum decision node feature dimensionality. However, since the routing function of each tree gets peaked on a single path after a reasonable number of samples ($i \gg 0$) have been observed, we can obtain a good approximation of the posterior predictive distribution by taking at each decision node n the direction where the sample $\boldsymbol{x}$ has the highest probability to be routed to, *i.e.* left if $\beta_n^i(\boldsymbol{x}) > 0.5$, and right otherwise. This decision can be taken efficiently by evaluating the sign of $\boldsymbol{\mu}_n^{i\top}\xi_n(\boldsymbol{x})$, because $\beta_n^i(\boldsymbol{x}) > 0.5 \iff \boldsymbol{\mu}_n^{i\top}\xi_n(\boldsymbol{x}) > 0$. With this trick, we reduce the per-tree complexity during inference to $O(\mathsf{d}\log_2|\mathcal{N}|)$, which is the same as for offline, oblique decision trees [12, 15]. Please note, that $\log_2|\mathcal{N}|$ is much smaller for our compact trees than for typically deeper oblique decision trees.

5. Experiments

We assess several variants of our algorithm on different datasets, including standard machine learning (ML) classification benchmarks (Sec. 5.1) and pixel-wise semantic labelling of Kinect [21] data (Sec. 5.2). For all experiments, we provide baseline results of state-of-the-art online random forest approaches, using their publicly available reference implementations. We validate our learner against Mondrian Forests (MF) [13], Online Random Forests (ORF) [20], and Consistent Online Forests (COF) [10] (the latter code includes a re-implementation of ORF). As additional baselines, we provide results for offline random forests (RF) [4] and offline oblique random forests (obRF) [15]. Please note however, that offline forest results are not directly comparable to online results, as their training expects the *entire* dataset to be given in advance. We fix $\epsilon = 0.01$ (see last section), which may serve as guideline for other datasets (we did not experience large sensitivity when varying it). For each dataset we train at most 8 trees for our method (no reasonable improvement was found with more) and we allow only a *single epoch over the data* for our trees to properly simulate an online scenario unless explicitly stated otherwise. Instead, all forest competitors (offline and online) comprised 100 trees with up to 15 epochs over the data (specifically recommended for [20, 10]).

5.1. Classification performance on ML datasets

We tested on `G50c`, `dna`, `satimages` and `USPS` since they were also (partially) selected in [10, 13], and cover different granularity of difficulty with respect to dataset characteristics (#feature dimensions, #classes, #train/#test samples). A summary is provided on top of Tab. 1, followed by blocks for offline ([4, 15], grayed block) and online forest results, respectively. All reported scores are average classification errors with standard deviations in [%] (from 10 repetitions or cross-validation folds for standard partitioning of datasets), *i.e.*, lower is better. Offline forest results should mainly demonstrate the effects due to different complexities of decision node functions: For instance, [4] uses randomly selected, single feature channels (*i.e.* axis-aligned splits) while [15] applies more complex, oriented hyperplanes, thus incorporating a larger feature space. Online forest results for ORF, MF and COF could be approximately reproduced with default parameter settings in their code (or suggested in their papers), which we also used for training/testing on datasets not evaluated in their papers.

We dub our method as Bayesian Online Forest[2] (BOF) in

[2]While knowing that we introduced an ensemble of Bayesian trees rather than a Bayesian forest, we use the term forest as we average akin to conventional forests

		G50c	dna		satimages	USPS
	#features	50	180	60	36	256
	#classes	2	3		6	10
	#train/#test	50/500	1400/1186		3104/2000	7291/2007
Offline forests	RF [4]	18.91 ±1.33	6.05 ±0.48	5.19 ±0.3	9.7 ±0.25	6.50 ±0.13
	obRF$^{100}_{25}$ [15]	9.01 ±0.77	17.38 ±0.43	6.75 ±0.25	9.36 ±0.16	5.87 ±0.21
	obRF$^{8}_{25}$ [15]	16.16 ±1.92	22.81 ±1.08	10.35 ±0.56	10.62 ±0.34	7.47 ±0.34
	obRF$^{100}_{\text{BOF}}$ [15]	8.92 ±0.79	29.2 ±0.63	9.02 ±0.41	9.86 ±0.22	7.85 ±0.24
	obRF$^{8}_{\text{BOF}}$ [15]	15.0 ±2.14	29.48 ±1.19	12.49 ±0.81	11.3 ±0.41	10.38 ±0.51
Online forests	ORF$_{\times 15}$ [20]	19.44 ±1.71	8.2 ±0.9	5.77 ±0.56	11.8±0.40	6.60 ±0.20
	MF$_{\times 15}$ [13]	13.8 ±1.79	32.75 ±0.54	9.01 ±0.41	10.46 ±0.15	6.79 ±0.23
	COF$_{\times 15}$ [10]	19.71 ±1.36	7.36 ±0.32	6.5 ±0.15	10.4 ±0.13	7.06 ±0.22
	ORF$_{\times 1}$ [20]	33.02 ±3.52	13.2 ±0.47	8.85 ±0.64	13.5 ±0.37	9.6 ±0.19
	MF$_{\times 1}$ [13]	14.22 ±1.53	32.93 ±0.5	9.10 ±0.56	10.45 ±0.26	6.96 ±0.19
	COF$_{\times 1}$ [10]	41.64 ±3.79	16.37 ±0.86	11.28 ±1.66	15.29 ±0.30	11.53 ±0.22
	BOF	7.74 ±1.29	5.78 ±0.31	4.87 ±0.19	11.14 ±0.30	6.17 ±0.28
	BOF-P	3.64 ±0.25	6.87 ±0.74	5.84 ±0.46	13.25 ±0.20	6.03 ±0.21
	BOF-B	7.86 ±1.38	5.73 ±0.31	4.86 ±0.28	11.42 ±0.38	6.47 ±0.18
	BOF-PB	3.72 ±0.27	7.32 ±0.57	6.61 ±0.26	13.91 ±0.30	6.25 ±0.26

Table 1. Mean classification errors with standard deviations in [%] over 10 runs. Grayed block: Offline forest variants. Middle block: Online forest competitors. Bottom block: Proposed Bayesian Online forest variants. See Sec. 5.1 for description.

case no projection is performed, *i.e.* $\mathtt{P}_n = \mathtt{I}$, BOF-P when we perform a randomly selected projection, BOF-B when using bagging (*i.e.* a tree discards a sample with probability τ), and BOF-BP when bagging and random projection are applied. All methods were trained on the same splits into training/testing. As a rule of thumb, we define a dataset-specific set of possible tree depths from where the actual tree depth is randomly selected. Specifically, we sample the tree depth from $\{\lceil \log_2(|\mathcal{Y}|) \rceil, \ldots, \lceil \log_2(|\mathcal{Y}|) \rceil + 2\}$ such that there are at least as many leaves as number of classes. *E.g.*, the `satimages` dataset has 6 classes which means that we randomly select a tree depth between 3 and 5. This max tree depth is also applied for some oblique forest configurations, indicated by the super- and subscripts. For instance, obRF$^{8}_{\text{BOF}}$ means that 8 trees with the same max depth as our Bayesian trees were grown, while obRF$^{100}_{25}$ means that 100 trees with max depth 25 were trained.

We obtain scores that are similar or better than all online methods we compare to, considering their 15-epoch results ORF$_{\times 15}$, MF$_{\times 15}$, COF$_{\times 15}$. For the `dna` dataset we evaluate with two different feature space sizes like [13]. MF seems to struggle with higher-dimensional inputs (see error values of ≈33% vs. ≈9%), whereas pre-selection of informative dimensions yields ≈1% in accuracy gain for our approach. On the `satimages` dataset we perform similar (or slightly worse) than our competitors. Finally, we also list single-epoch results for ORF$_{\times 1}$, MF$_{\times 1}$ and COF$_{\times 1}$, which, except for MF, show drastic performance reductions, inhibiting online learning without additional samples.

To illustrate the ensemble effect, we obtain the following classification errors (in [%]) when using (1,4,8) BOF trees. `G50c`: (8.1, 7.9, 7.7). `dna(180)`: (6.1, 5.9, 5.8). `dna(60)`: (5.5, 5.0, 4.9). `satimages`: (13.0, 11.3, 11.1). `USPS`: (8.3, 6.5, 6.2). Since our trees are Bayesian, they exhibit less variance than standard trees. We thus require smaller ensembles to achieve similar/better accuracy. For different variants of our method we experience only minor performance drop when applying bagging ($\tau = 0.3$), which however linearly reduces training times. We obtain both, improvement of classification error and reduction in training time for `G50c` and `USPS` when applying randomly chosen projections to lower-dimensional feature spaces (by applying projection matrices $\mathtt{P}_n$). As target feature projection dimensionality we choose values approximately around $\mathtt{d}/2$, *i.e.* $\mathtt{P}_n \in \mathbb{R}^{\mathtt{d}/2 \times \mathtt{d}}$. We provide a table with detailed information on Matlab timings in the supplementary material (Sec. C.1), corresponding to the experiments in Tab. 1.

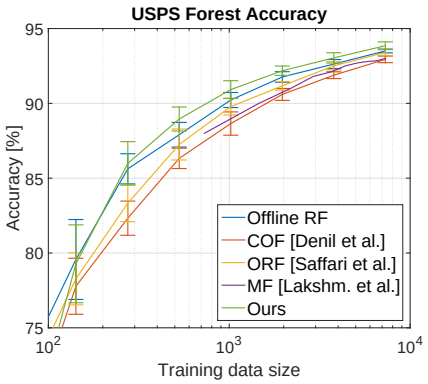

Figure 3. Sequential data arrival experiments for USPS dataset (see last paragraph in Sec. 5.1), showing test data classification accuracies as function of seen training samples.

Sequential data arrival performance on `USPS` In Fig. 3 we show the results when training and testing our tree ensemble from sequentially arriving data of the `USPS` dataset, akin to the experiment in [10]. The curves show the performance on the test dataset as a function of the number of training samples presented to the algorithms. As can be seen, our algorithm initially performs comparably with the other methods but begins to even surpass re-trained offline forests [4] after seeing more than 500 training samples. The numbers for our method are averaged over 10 runs and again each sample was presented only once while [10, 20] allowed 15 epochs in their training protocol.

5.2. Kinect dataset

In this experiment we perform the task of pixel-wise semantic labelling (body part recognition, see Fig. 5), using synthetically generated depth input images. We used the publicly available[3] dataset of [10], which provides predefined splits into training and test images, as well as the specific order and training sample center locations presented to the learners. The training set contains 2000 images (*i.e.* poses with 19 body part classes + 1 background

[3] `http://mdenil.com/projects/#random-forests`

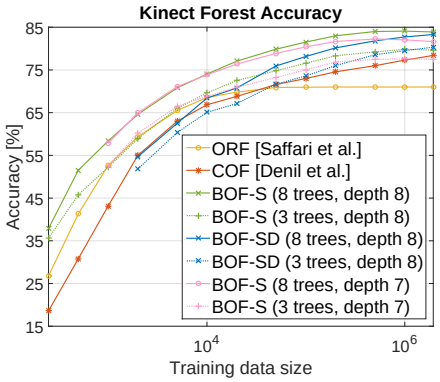

Figure 4. Sequential data arrival experiments for Kinect dataset (see Sec. 5.2), showing test data classification accuracies as function of seen training samples.

class label) from where ≈50 samples per body part and image were collected, resulting in roughly 2 million training samples. Testing was conducted on all foreground pixels of the 500 images in the test set. Since the exact order and locations of training samples are given, we can provide a direct comparison to the baseline scores reported for [20] and [10]. Both trained forests of 25 trees, [20] was limited to depth 8 for memory reasons (to keep memory consumption <10GB) while [10] reports no restriction on their maximum depth. Moreover, [10] reports that their trees were allowed to evaluate 2000 candidate offsets with 10 candidate split node tests at a memory consumption of 1.6GB, when limiting their approach to 1000 active leaf nodes.

We trained again ensembles comprising 8 balanced Bayesian trees, using maximum depths of 7 or 8 (yielding 127/255 split nodes and 128/256 leaf nodes per tree, respectively). Instead of granting access to 2000 candidate offsets, we used $\mathsf{d} = 100$ randomly chosen offsets (+1 bias dimension) per split node (dubbed BOF-S), which we sampled from a log-polar space with maximum distance of ≈35 pixels, akin to [10]. In such a way, the total number of parameters per tree for our approach is $\left(|\mathcal{N}| \cdot ((\mathsf{d}+1)^2 + (\mathsf{d}+1) + \mathsf{d}) + 2|\mathcal{N}| \cdot |\mathcal{Y}|\right)$ (requiring 101^2 per Σ_n, 101 per $\boldsymbol{\mu}_n$, 100 for subspace selection via P_n and 20 per $\boldsymbol{\alpha}_\ell$), resulting in ≈5/10MB per tree (depth 7/8). When using only axis-aligned, diagonal covariance matrices Σ_n (denoted as BOF-SD), the model memory requirement reduces to ≈160/320kB per tree. Once only inference has to be performed, memory consumption can be reduced to ≈110/220kB (for both, BOF-S and BOF-SD) when using the fast, single-path routing described in Sec. 4..

Figure 5. Color-coded, qualitative examples of Kinect semantic labelling experiment (ground truth vs. obtained result), see Sec. 5.2.

The plot in Fig. 4 shows the pixel labelling accuracy (percentage of correctly labelled foreground pixels of test set as in [10]) as a function of presented training data. We outperform both baselines by a significant margin over the entire sweep of training samples when using 8 BOF-S trees. For instance, after 500k training samples we improve by ≈6/8% over [10] and ≈11/13% over [20] (depth 7/8). Conversely, we approximately match the final performance of [10] at 2M samples with our depth 8 ensemble after seeing only 20k (*i.e.* $1/100$th) training samples. Also, we only need 3 of our trees in order to reach comparable final performance to [10], which we illustrate with dashed lines in the plot. With our faster 8 tree training variant BOF-SD (*ie.* diagonal Σ_n, shown in cyan), we approximately match the performance of the full covariance version at the maximum number of training samples. Finally, note that all results for our approaches were obtained by performing the fast, single-path inference described at the end of Sec. 4. More experimental insights in the supplementary material include i) details on timings in Sec. C.1, ii) plots and discussions of an increasing ensemble size for both, depth 7 and depth 8 BOF-S ensembles in Sec. C.2, iii) a guide on how to perform model selection based on the online development of the ensemble training loss in Sec. C.3.

6. Summary and Future Work

In this paper we have proposed a novel approach for online learning of classification trees, driven by ideas from Bayesian online learning theory. Our solution departs from state-of-the-art approaches by trying to build tree ensembles that consist of only few and compact trees with good generalization capability. We achieved this goal by adopting a Bayesian learning procedure that iteratively refines a posterior distribution within a pre-defined parametric family in a way to best incorporate information carried by new data samples. The experimental evaluation has shown that our approach is able to perform on par or better than state-of-the-art online forest algorithms on a variety of classification tasks, while using smaller models.

We plan to extend our approach to regression, by replacing the prediction model in the leaves and derive proper update formulæ for the related parameters. We also plan to investigate a semi-parametric, Bayesian setting in order to let the tree structure be driven by the data.

Acknowledgments. This research has received funding from the European Unions Horizon 2020 research and innovation programme under grant agreement No 687757.

References

[1] C. Anagnostopoulos and R. B. Gramacy. Dynamic trees for streaming and massive data contexts. ArXiv preprint arXiv:1201.5568, 2012.

[2] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavaldà. New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '09, pages 139–148, New York, NY, USA, 2009. ACM.

[3] C. M. Bishop and M. Svensén. Bayesian hierarchical mixtures of experts. In *Proc. of Conference on Uncertainty in Artificial Intelligence*, page 5764, 2003.

[4] L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[5] W. Buntine. Learning classification trees. *Stat. Comput.*, 2:63–73, 1992.

[6] H. Chipman, E. I. George, and R. E. McCulloch. Bayesian cart model search. *J. Am. Stat. Assoc.*, pages 935–948, 1998.

[7] H. Chipman, E. I. George, and R. E. McCulloch. BART: Bayesian additive regression trees. *The Annals of Applied Statistics*, 4(1):266–298, 2010.

[8] H. Chipman and R. E. McCulloch. Hierarchical priors for Bayesian cart shrinkage. *Stat. Comput.*, 10(1):17–24, 2000.

[9] A. Criminisi and J. Shotton. *Decision Forests in Computer Vision and Medical Image Analysis*. Springer, 2013.

[10] M. Denil, D. Matheson, and N. de Freitas. Consistency of online random forests. In *(ICML)*, 2013.

[11] P. Domingos and G. Hulten. Mining high-speed data streams. In *Int. Conf. on Knowl. Discov. and Data Mining*, pages 71–80, 2000.

[12] D. Heath, S. Kasif, and S. Salzberg. Induction of oblique decision trees. *Journal of Artificial Intelligence Research*, 2(2):1–32, 1993.

[13] B. Lakshminarayanan, D. Roy, and Y. W. Teh. Mondrian forests: Efficient online random forests. In *Advances in Neural Inform. Process. Syst.*, 2014.

[14] P. S. Maybeck. *Stochastic models, estimation and control*. Academic Press, 1982.

[15] B. H. Menze, B. M. Kelm, D. N. Splitthoff, U. Koethe, and F. A. Hamprecht. On oblique random forests. In *Machine Learning and Knowledge Discovery in Databases*, volume 6912. Springer, 2011.

[16] T. P. Minka. Estimating a dirichlet distribution. Unpublished, 2000.

[17] T. P. Minka. Expectation propagation for approximate bayesian inference. In *Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence*, UAI '01, pages 362–369, 2001.

[18] M. Opper. A bayesian approach to on-line learning. In D. Saad, editor, *On-line Learning in Neural Networks*, pages 363–378. Cambridge University Press, 1998.

[19] N. Oza and S. Russel. Online bagging and boosting. In *Proc. of Artificial Intell. and Statistics*, pages 105–112, 2001.

[20] A. Saffari, C. Leistner, J. Santner, M. Godec, and H. Bischof. On-line random forests. In *IEEE - ICCV Workshop on On-line Learning for Computer Vision*, 2009.

[21] J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, and A. Blake. Efficient human pose estimation from single depth images. *(PAMI)*, 2013.

[22] M. Taddy, R. B. Gramacy, and N. G. Polson. Dynamic trees for learning and design. *J. Am. Stat. Assoc.*, 106(493):109–123, 2011.

[23] J. Valentin, V. Vineet, M.-M. Cheng, D. Kim, J. Shotton, P. Kohli, M. Nießner, A. Criminisi, S. Izadi, and P. Torr. Semanticpaint: Interactive 3d labeling and learning at your finger tips. *ACM Transactions on Graphics*, 2015.

[24] A. Verikas, A. Gelzinis, and M. Bacauskiene. Mining data with random forests: A survey and results of new tests. *Pattern Recognition (PR)*, 44(2):330 – 349, 2011.

[25] Y. Wu, H. Tjelmeland, and M. West. Bayesian CART: Prior specification and posterior simulation. *J. Comput. Graph. Stat*, 16(1):44–66, 2007.

[26] P. Yang, Y. Hwa Yang, B. B. Zhou, and A. Y. Zomaya. A review of ensemble methods in bioinformatics. *Current Bioinformatics*, 5(4):296–308, 2010.