Dynamic trees for online analysis of massive data.

Christoforos Anagnostopoulos
Statistical Laboratory, U. of Cambridge
Wilberforce Road, Cambridge CB3 0WB
canagnos@statslab.cam.ac.uk

Robert B. Gramacy
Booth Business School, U. of Chicago
Chicago, Illinois, 60637
rbgramacy@ChicagoBooth.edu

Abstract

Data collection at a massive scale is becoming ubiquitous in a wide variety of settings, from vast offline databases to online real-time information. Learning algorithms deployed in such contexts must rely on single-pass inference, where the data history is never revisited. Although rapidly developing, online non-parametric Bayesian inference remains challenged by massive, transient data, as the complexity of such models will typically increase with the sample size. In this work, we take steps to overcome these challenges by porting data discarding into a fully Bayesian framework via the use of informative priors and active learning heuristics. We showcase our methods by augmenting a modern non-parametric modelling framework, dynamic trees, and illustrate its performance on a number of practical examples. The end product is a powerful fully online regression and classification tool, whose performance compares favorably to the state-of-the-art.

1 Introduction

In certain simple cases, online estimation without information loss is possible via exact recursive update formulae, e.g., via conjugate Bayesian updating. In parametric dynamic modelling, approximate samples from the filtering distribution for a variable of interest may be obtained online via sequential Monte Carlo (SMC) techniques, under quite general conditions. In [13], SMC is use in a non-parametric context, where a 'particle cloud' of trees are employed to track parsimonious regression and classification surfaces as data arrive sequentially. However, the resulting algorithm is not, strictly speaking, *online*, since local tree moves may revisit the partial data history depending on how the process evolves. This complication arises as an essential by-product of non-parametric modelling, wherein the complexity of the estimator is allowed to increase with the dataset size. Consequently, maintaining constant operational cost per timestep will necessarily entail discarding datapoints over time. Our concern in this paper is managing the information loss entailed in data discarding. First, we propose datapoint *retirement* (Section 3), whereby discarded datapoints are partially 'remembered' through conjugate informative priors, updated sequentially. This technique is well-suited to trees, which combine non-parametric flexibility with simple parametric models with conjugate priors by partitioning. Nevertheless, forming new partitions in the tree still requires access to actual datapoints, and consequently data discarding comes at a cost of both information and degrees of freedom. We show that this can be managed, to a surprising extent, by the right retirement scheme even when discarding data randomly. We further show that borrowing active learning heuristics to prioritise points for retirement, i.e., *active retiring*, leads to better performance still.

2 Dynamic Trees

Dynamic trees (DTs) [13] are a process–analog of Bayesian treed models [4]. The model specification, and tree prior, are defined in a way that is amenable to fast sequential inference by SMC, and that yields a predictive surface which organically increases in complexity as more data arrive. The

computations involved in sequential updates are local, and completely determined by the set of trees $\{\mathcal{T}_t\}$ which are reachable from $\mathcal{T}_{t-1}$ when a new predictor $\mathbf{x}_t$ arrives, and the form of the model fit in each leaf. The tree may *stay*, *prune* or *grow* local to the leaf node in $\mathcal{T}_{t-1}$ where $\mathbf{x}_t$ lands. The new observation, y_t, completes a stochastic rule for determining how the update $\mathcal{T}_{t-1} \to \mathcal{T}_t$ chooses amongst those three options, via $p(y^t|\mathcal{T}_t, \mathbf{x}^t)$. To keep computations efficient, the leaf model, parameterised by a leaf-specific parameter θ, must be simple enough for the marginal likelihood to be analytic, and will be equipped with (leaf-specific again) priors over θ. We therefore restrict our attention to *constant* and *linear* models for regression, and *multinomial* models for classification [13]. As with all particle simulation methods, some Monte Carlo error will accumulate. Nevertheless, DT out-of-sample performance compares favorably to other nonparametric methods, like Gaussian processes (GPs), but at a fraction of the computational cost [13].

3 Datapoint retirement

At time t, the DT algorithm of [13] requires access to the entire data history to compute $p(\mathcal{T}_t \mid \mathcal{T}_{t-1}, \mathbf{x}^t)$ for each particle. To enable online operation, the covariate pool, $\mathbf{x}^t = (\mathbf{x}_1, \ldots, \mathbf{x}_t)$ has to be of size constant with t. This can only be achieved via data discarding. However, the analytic/parametric nature of DT leaves allows us to retain part of the discarded information in the form of informative priors in the leaves, yielding in effect a *soft* implementation of data discarding, which we refer to as *datapoint retirement*. Intuitively, the DT with retirement manages two types of information: a non-parametric memory comprising an active data pool of constant size w, which forms the leaf likelihoods; and a parametric memory consisting of possibly informative leaf priors.

Updating the relevant leaf priors before discarding a datapoint proceeds via standard conjugate Bayesian updating. Let us focus on a single leaf η, letting $(\mathbf{x}_n, y_n)_{n\in\mathcal{R}}$ denote the data retired from η so far. The leaf prior $\pi(\theta)$ is then given by the posterior of θ given the retired data, $\pi(\theta) =_{\text{df}} P\left(\theta \mid (\mathbf{x}_n, y_n)_{n\in\mathcal{R}}\right) \propto L\left(\theta; (\mathbf{x}_n, y_n)_{n\in\mathcal{R}}\right) \pi_0(\theta)$, where $\pi_0(\theta)$ is a non-informative prior employed initially. To retire one more datapoint, $(\mathbf{x}_m, y_m)$, we recursively update:

$$\pi^{(\text{new})}(\theta) =_{\text{df}} P\left(\theta \mid (\mathbf{x}_n, y_n)_{n\in\mathcal{R}\cup\{m\}}\right) \propto L(\theta; \mathbf{x}_m, y_m) P\left(\theta \mid (\mathbf{x}_n, y_n)_{n\in\mathcal{R}}\right) \tag{1}$$

The calculation in (1) is tractable whenever conjugate priors are employed. Take for instance the linear regression model, $\mathbf{y} \sim N(\mathbf{X}\beta, \sigma^2\mathbf{I})$, where $\mathbf{y} = (y_n)_{n\in\mathcal{R}}$ is the retired response data, and $\mathbf{X}$ the retired *augmented* design matrix, i.e., consisting of augmented covariate vectors $[1, \mathbf{x}_n']'$, so that β_1 represents an intercept. With $\pi_0(\beta, \sigma^2) \propto \frac{1}{\sigma^2}$, we obtain $\pi_t(\beta, \sigma^2) =_{\text{df}} P(\beta, \sigma^2 \mid \mathbf{y}, \mathbf{X}) = \text{NIG}(\nu/2, s\nu/2, \beta, \mathbf{A})$, where NIG stands for Normal-Inverse-Gamma, and,

$$\mathbf{A} = (\mathbf{X}'\mathbf{X})^{-1},\ \mathbf{R} = \mathbf{X}'\mathbf{y},\ r = \mathbf{y}'\mathbf{y},\ \nu = n - p,\ \beta = \mathbf{A}\mathbf{R},\ s^2 = \frac{1}{\nu}\left(r - \beta'\mathbf{A}\beta\right) \tag{2}$$

assuming $\mathbf{X}'\mathbf{X}$ is invertible. Although the retired data $(y_n, \mathbf{x}_n)_{n\in\mathcal{R}}$ are unavailable, we can afford to keep in memory the values of the above statistics, as their dimension does not grow with the size of $\mathcal{R}$. If we now wish to retire an additional datapoint $(y_m, \mathbf{x}_m)$, we perform the following updates:

$$\left(\mathbf{A}^{(\text{new})}\right)^{-1} = \mathbf{A}^{-1} + X_m' X_m,\ \mathbf{R}^{(\text{new})} = \mathbf{R} + \mathbf{X}_m'\mathbf{y}_m,\ s^{(\text{new})} = s + y_m' y_m,\ \nu^{(\text{new})} = \nu + 1. \tag{3}$$

The DT with retirement algorithm will, at time $t+1$, add the $(t+1)$th datapoint to the active pool, and update the model by SMC exactly as explained in Section 2. Once t exceeds w, the algorithm will also select some datapoint, $(\mathbf{x}_r, y_r)$, to retire from the active pool: the associated leaf prior for $\eta(\mathbf{x}_r)^{(i)}$ for each particle i is updated to incorporate the removed datapoint $(\mathbf{x}_r, y_r)$. At this level, data retirement performs a shifting of information from the likelihood part of the posterior to the prior that preserves, exactly, the posterior predictive distribution and the value of the marginal likelihood calculation both locally at each node $\eta(x)^{(i)}$, as well as globally for each tree $\mathcal{T}^{(i)}$. Moreover, this predictive distribution is preserved for any future updates involving data points that either belong to a different node, or belong to $\eta(\mathbf{x})^{(i)}$ which stochastically chose a *stay* move. The retiring mechanism suggests a method for splitting and combining that information in 'grow' and 'prune' moves. Following a 'prune' move, retired datapoints from the pruned leaves are pooled together to form the new leaf prior additivelty, as implied by conjugate updating. This does not require access to the actual retired datapoints. Following a 'grow' move, we let both novel leaves inherit the parent prior, but split its strength $\nu^{(P)}$ between the two at proportions equal to the active data proportions in the children. This preserves the total strength of retired information.

4 Active discarding

Ideally, we would rather retire datapoints that will be of "less" use to the model going forward. We formulate the choice of which active data points to retire as an *active retiring* (AR) problem, borrowing techniques from the *active learning* (AL) literature. Regression and classification must be treated separately, as they require different AR techniques. We focus on the former due to space constraints, but note that in either case AR is easier than AL for DTs due to thrifty analytic calculations.

Consider the regression active learning statistic due to Cohn [5, ALC] popularised in the modern nonparametric regression literature [11] using GPs, and subsequently ported to DTs [13]. ALC chooses $\mathbf{x}^\star$ to maximise the expected reduction in predictive variance averaged over the input space. A drawback is that it isensitive to the choice of (and density of) a search and reference grid over which the variance statistics are evaluated. Our first simplification when porting AL to AR is to recognise that no grids are needed. The program is to evaluate the ALC statistic at each active data location, and choose the smallest one for discarding. We focus on the linear case, as the constant case may be derived as a special case. We have: $\Delta\sigma^2_{\mathbf{x}}(\mathbf{z}|\mathcal{T}) = \Delta\sigma^2_{\mathbf{x}}(\mathbf{z}|\eta) \equiv \sigma^2(\mathbf{z}|\eta) - \sigma^2_{\mathbf{x}}(\mathbf{z}|\eta)$, where the latter two terms represent the respective predictive variance at $\mathbf{z}$ with or without $\mathbf{x}$. A quick calculation shows that their difference equals $\frac{s^2_\eta - \mathcal{R}_\eta}{|\eta|-m-3} \times \frac{\left(\frac{1}{|\eta|} + \mathbf{z}'\mathcal{G}_\eta^{-1}\mathbf{x}\right)^2}{1+\frac{1}{|\eta|}+\mathbf{x}'\mathcal{G}_\eta^{-1}\mathbf{x}}$, when *both* $\mathbf{x}$ and $\mathbf{z}$ are in $\eta \in \mathcal{L}_\mathcal{T}$, and zero otherwise. The s^2_η is the leaf sum of squares, $|\eta|$ is the number of data points in the leaf, and $\mathcal{G}_\eta$ is the Grahm matrix for η for the active *and* retired data [13]. Integrating over $\mathbf{z}$, only $(1/|\eta| + \mathbf{z}'\mathcal{G}_\eta^{-1}\mathbf{x})^2$ features z in the expression above, and can be integrated over the rectangular region η with an $O(m^2)$ implementation:

$$\int_{a_1}^{b_1}\cdots\int_{a_m}^{b_m}\left(c+\sum_{i=1}^m \tilde{z}_i x_i\right)^2 dz_1\cdots dz_m = A_\eta c^2 + c\sum_i\left(\prod_{k\neq i}(b_k-a_k)\right)x_i(b_i^2-a_i^2)$$
$$+\sum_i\left(\prod_{k\neq i}(b_k-a_k)\right)\frac{x_i^2}{3}(b_i^3-a_i^3)+\sum_i\sum_{j<i}\left(\prod_{k\neq i,j}(b_k-a_k)\right)\frac{x_i x_j}{2}(b_i^2-a_i^2)(b_j^2-a_j^2),$$

where the m-rectangle η is being described by $\{(a_i,b_i)\}^m$, $\tilde{\mathbf{z}} = \mathbf{z}'\mathcal{G}_\eta^{-1}$, and $c = 1/|\eta|$. The rectangular leaf regions generated by the trees is key. In the case of other partition models (like Voronoi tessellation models), this analytical integration would not be possible. Finally, the divide and conquer nature of trees—whose posterior distribution is approximated by thrifty, local, particle updates—allows AR statistics to be updated cheaply ($O(m^2N)$ for N particles) as long each leaf node stores its own AR statistics. We leave the details to the full version of the paper.

In repeated applications of ALC for AR, the active points that remain tend to cluster near the boundaries of high posterior partitioning boundaries. The number of such locations depends crucially on the number of active data points allowed, w. Consider the simple example with a parabolic response that must be learned sequentially via x-data sampled uniformly in $(-3, 2)$, with $w = 25$. The initial 25, before any retiring, are shown in the first panel of Figure 1. Each updating round then proceeds with one retirement followed by one new pair, and subsequent SMC update. The active points indeed shift towards likely partition boundaries. By $t = 150$ (third panel) the ability to learn about the mean with $w = 25$ is saturated, but the variance (errorbars) still improves for $t = 300$.

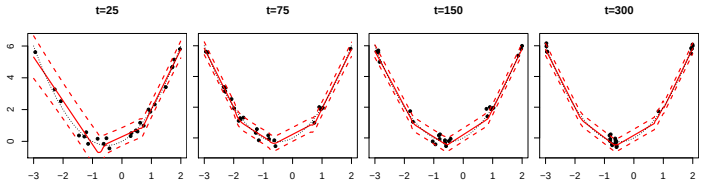
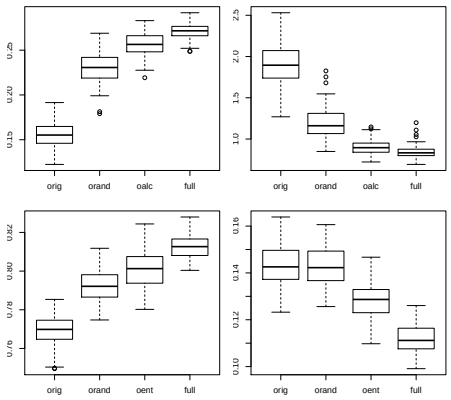


Figure 1: Snapshots of active data (25 points) and predictive surfaces.

4.1 Empirical results

Here we explore the benefit of AR over simpler heuristics, like random discarding and subsetted data estimators, by making predictive comparisons on benchmark datasets. To focus the discussion on our key objective, we compare only to full-data versions of DTs, and assess the impact of data discarding on performance. We emphasise however that discarding enables DTs to operate on arbitrarily long datasets, where the original DTs, as well as their main GP-based competitors, simply cannot be used. We first consider data originally used in [6], and then to demonstrate the competitiveness of DTs relative to modern (batch) nonparametric models [13]. The response is $10\sin(\pi x_1 x_2) + 20(x_3 - 0.5)^2 + 10x_4 + 5x_5$ plus $\mathcal{N}(0,1)$ additive error. Inputs $\mathbf{x}$ are random in $[0,1]^5$. We considered four estimators: one based on 200 pairs (ORIG), one based 1800 more for 2000 total (FULL), and two online versions [which saw the same stream of pairs] using either random (ORAND) or ALC (OALC) retiring to keep the total active data set limited to $w = 200$. We repeated the experiment 100 times in a MC fashion, each with new random training sets, and random testing sets of size 1000. $N = 1000$ particles and a linear leaf model were used throughout. Similar results were obtained for the constant model. From Figure 2 we can see that random retiring is better than subsetting, but retiring by ALC is even better, and can be nearly as good as the full-data estimator. In fact, it may be surprising to note that OALC was the *best* predictor 16% and 28% of the time by average predictive density (APD) and residual mean squared error (RMSE), respectively. The average time used by each estimator was approximately 1, 33, 45, and 67 seconds, respectively. So random retiring on this modestly-sized problem is 2-times faster than using the full data. ALC costs about 18% extra, time-wise, but leads to about a 35% reduction in RMSE relative to the full estimator. Note that the time-demands of the full estimator grow roughly as $t \log t$, whereas the online versions stay constant. Now consider the Spambase data set of [1]. The data contains binary classifications of 4601 emails based on 57 attributes (predictors). We do a similar experiment to the Friedman/regression example, above, except with classification leaves and 5-fold CV to create training and testing sets, repeated twenty times. We considered four estimators: one based on 1/10 of the training fold (ORIG), one based on the full fold (FULL), and two online versions trained on the same stream(s) using either random (ORAND) or entropy (OENT) retiring to keep the total active data set limited to 1/10 of full. Figure 2 tells a similar story to the Friedman experiment: random discarding is better than subsetting, but discarding by entropy is even better, and can be nearly as good as the full-data estimator. Entropy retiring resulted the best predictor 15% and 5% of the time by predictive probability and misclassification rate (MCR), respectively. While ORAND predicts labels comparably to ORIG, it is much better at capturing the full predictive distribution.



	APD			RMSE		
	mean	5%	95%			
ORIG	.155	.133	.176	1.92	1.55	2.32
ORAND	.230	.206	.254	1.19	.957	1.46
OALC	.257	.239	.275	.896	.754	1.04
FULL	.271	.254	.282	.844	.737	.963

	APD			MCR		
	mean	5%	95%			
ORIG	.769	.752	.782	.143	.130	.161
ORAND	.792	.778	.805	.143	.129	.155
OENT	.801	.788	.817	.128	.114	.139
FULL	.813	.804	.823	.112	.102	.121

Figure 2: Friedman data (top) comparisons by APD (higher is better) and RMSE (lower is better); and spam data (bottom) by APD, and MCR (lower is better).

5 Conclusion

We rely on Bayesian machinery to facilitate fully online non-parametrics. The availability of tractable predictive distributions allows us to combine active retirement heuristics with conjugate updating to maintain a fixed budget of highly informative datapoints at minimum information loss.

References

[1] A. Asuncion and D.J. Newman. UCI machine learning repository, 2007.

[2] Carlos M. Carvalho, Michael Johannes, Hedibert F. Lopes, and Nicholas G. Polson. Particle learning and smoothing. *Statistical Science*, 25:88–106, 2010.

[3] H.A. Chipman, E.I. George, and R.E. McCulloch. Bayesian CART model search (with discussion). *Journal of the American Statistical Association*, 93:935–960, 1998.

[4] H.A. Chipman, E.I. George, and R.E. McCulloch. Bayesian treed models. *Machine Learning*, 48:303–324, 2002.

[5] D. A. Cohn. Neural network exploration using optimal experimental design. In *Advances in Neural Information Processing Systems*, volume 6(9), pages 679–686. Morgan Kaufmann Publishers, 1996.

[6] J. H. Friedman. Multivariate adaptive regression splines. *Annals of Statistics*, 19, No. 1:1–67, March 1991.

[7] Robert B. Gramacy and Matt A. Taddy. `dynaTree`*: Dynamic trees for learning and design*, 2011. R package version 2.0.

[8] A.J. Joshi, F. Porikli, and N. Papanikolopoulos. Multi-class active learning for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. to appear.

[9] D. J. C. MacKay. Information–based objective functions for active data selection. *Neural Computation*, 4(4):589–603, 1992.

[10] A. O'Hagan and J. Forster, editors. *Kendall's Advanced Theory of Statistics, Volume 2B, Bayesian Inference*. Arnold Publishers, 2004.

[11] S. Seo, M. Wallat, T. Graepel, and K. Obermayer. Gaussian process regression: Active data selection and test point rejection. In *Proceedings of the International Joint Conference on Neural Networks*, volume III, pages 241–246. IEEE, July 2000.

[12] R Development Core Team. R*: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2010. ISBN 3-900051-07-0.

[13] M.A. Taddy, R.B. Gramacy, and N.G. Polson. Dynamic trees for learning and design. *Journal of the American Statistical Association*, 106(493):109–123, 2011.