Mondrian Forests:
Efficient Online Random Forests

Balaji Lakshminarayanan

Joint work with Daniel M. Roy and Yee Whye Teh

Outline

Background and Motivation

Mondrian Forests
- Randomization mechanism
- Online training
- Experiments

Conclusion

Introduction

- **Input**: attributes $X = \{x_n\}_{n=1}^N$, labels $Y = \{y_n\}_{n=1}^N$ (i.i.d)
- $x_n \in \mathcal{X}$ and $y_n \in \{1, \ldots, K\}$ (classification)
- **Goal**: Predict y_* for test data x_*

Introduction

- **Input**: attributes $X = \{x_n\}_{n=1}^N$, labels $Y = \{y_n\}_{n=1}^N$ (i.i.d)
- $x_n \in \mathcal{X}$ and $y_n \in \{1, \ldots, K\}$ (classification)
- **Goal**: Predict y_* for test data x_*
- **Recipe for prediction**: Use a random forest
  - Ensemble of randomized decision trees
  - State-of-the-art for lots of real world prediction tasks

Introduction

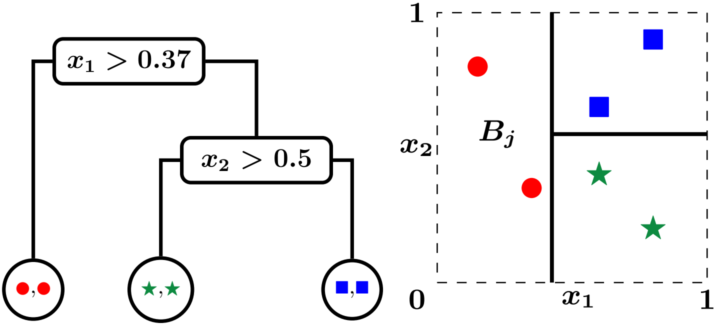
- **Input**: attributes $X = \{x_n\}_{n=1}^{N}$, labels $Y = \{y_n\}_{n=1}^{N}$ (i.i.d)
- $x_n \in \mathcal{X}$ and $y_n \in \{1, \ldots, K\}$ (classification)
- **Goal**: Predict y_* for test data x_*
- **Recipe for prediction**: Use a random forest
  - Ensemble of randomized decision trees
  - State-of-the-art for lots of real world prediction tasks
  - 'An empirical comparison of supervised learning algorithms' [Caruana and Niculescu-Mizil, 2006]
  - 'Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?' [Fernández-Delgado et al., 2014]

Introduction

- **Input**: attributes $X = \{x_n\}_{n=1}^N$, labels $Y = \{y_n\}_{n=1}^N$ (i.i.d)
- $x_n \in \mathcal{X}$ and $y_n \in \{1, \dots, K\}$ (classification)
- **Goal**: Predict y_* for test data x_*
- **Recipe for prediction**: Use a random forest
  - Ensemble of randomized decision trees
  - State-of-the-art for lots of real world prediction tasks
  - 'An empirical comparison of supervised learning algorithms' [Caruana and Niculescu-Mizil, 2006]
  - 'Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?' [Fernández-Delgado et al., 2014]
- What is a decision tree?

Example: Classification tree

- Hierarchical axis-aligned binary partitioning of input space
- Rule for predicting label within each block



$\mathcal{T}$: list of nodes, feature-id + location of splits for internal nodes
$\boldsymbol{\theta}$: Multinomial parameters at leaf nodes

Prediction using decision tree

- Example:
  - Multi-class classification: $\theta = [\theta_r, \theta_b, \theta_g]$
  - Prediction = smoothed empirical histogram in node j
  - Label counts in left node $[n_r = 2, n_b = 0, n_g = 0]$
  - $\theta \sim \mathcal{D}irichlet(\alpha/3, \alpha/3, \alpha/3)$
  - Prediction = Posterior mean of $\theta = \left[\frac{2+\alpha/3}{2+\alpha}, \frac{\alpha/3}{2+\alpha}, \frac{\alpha/3}{2+\alpha}\right]$

Prediction using decision tree

- Example:
  - Multi-class classification: $\theta = [\theta_r, \theta_b, \theta_g]$
  - Prediction = smoothed empirical histogram in node j
  - Label counts in left node $[n_r = 2, n_b = 0, n_g = 0]$
  - $\theta \sim \mathcal{D}irichlet(\alpha/3, \alpha/3, \alpha/3)$
  - Prediction = Posterior mean of $\theta = \left[\frac{2+\alpha/3}{2+\alpha}, \frac{\alpha/3}{2+\alpha}, \frac{\alpha/3}{2+\alpha}\right]$
- Likelihood for n^{th} data point $= p(y_n|\theta_j)$ assuming x_n lies in leaf node j of $\mathcal{T}$
- Prior over θ_j: independent or hierarchical
- Prediction for x_* falling in $j = \mathbb{E}_{\theta_j|\mathcal{T},X,Y}\left[p(y_*|\theta_j)\right]$, where

$$p(\theta_j \mid \mathcal{T}, X, Y) \propto \underbrace{p(\theta_j|...)}_{\text{prior}} \underbrace{\prod_{n \in N(j)} p(y_n|\theta_j)}_{\text{likelihood of data points in node j}}$$

- Smoothing is done independently for each tree

Random forest (RF)

- Generate randomized trees $\{\mathcal{T}_m\}_1^M$
- Prediction for x_*:

$$p(y_*|x_*) = \frac{1}{M}\sum_m p(y_*|x_*, \mathcal{T}_m)$$

- Model combination and not Bayesian model averaging

Random forest (RF)

- Generate randomized trees $\{\mathcal{T}_m\}_1^M$
- Prediction for x_*:

$$p(y_*|x_*) = \frac{1}{M}\sum_m p(y_*|x_*, \mathcal{T}_m)$$

- Model combination and not Bayesian model averaging

- Advantages of RF
  - Excellent predictive performance (test accuracy)
  - Fast to train (in batch setting) and test
  - Trees can be trained in parallel

Disadvantages of RF

- Not possible to train incrementally
  - Re-training batch version periodically is slow $\mathcal{O}(N^2 \log N)$
  - Existing online RF variants [Saffari et al., 2009, Denil et al., 2013] require
    - lots of memory / computation or
    - need lots of training data before they can deliver good test accuracy (data inefficient)

Disadvantages of RF

- Not possible to train incrementally
  - Re-training batch version periodically is slow $\mathcal{O}(N^2 \log N)$
  - Existing online RF variants [Saffari et al., 2009, Denil et al., 2013] require
    - lots of memory / computation or
    - need lots of training data before they can deliver good test accuracy (data inefficient)

Mondrian forests = Mondrian process + Random forests

- Can operate in either batch mode or online mode
- Online speed $\mathcal{O}(N \log N)$
- Data efficient (predictive performance of online mode equals that of batch mode!)

Outline

Background and Motivation

Mondrian Forests
- Randomization mechanism
- Online training
- Experiments

Conclusion

Popular batch RF variants

How to generate individual trees in RF?

- **Breiman-RF** [Breiman, 2001]: Bagging + Randomly subsample features and choose best location amongst subsampled features

Popular batch RF variants

How to generate individual trees in RF?

- **Breiman-RF** [Breiman, 2001]: Bagging + Randomly subsample features and choose best location amongst subsampled features
- **Extremely Randomized Trees** [Geurts et al., 2006] (ERT-*k*): Randomly sample *k* (feature-id, location) pairs and choose the best split amongst this subset
  - no bagging
  - ERT-1 does not use labels *Y* to guide splits!

Mondrian process [Roy and Teh, 2009]

- MP$(\lambda, \mathcal{X})$ specifies a distribution over hierarchical axis-aligned binary partitions of $\mathcal{X}$ (e.g. $\mathbb{R}^D$, $[0, 1]^D$)
- λ is complexity parameter of the Mondrian process

Mondrian process [Roy and Teh, 2009]

- MP$(\lambda, \mathcal{X})$ specifies a distribution over hierarchical axis-aligned binary partitions of $\mathcal{X}$ (e.g. $\mathbb{R}^D$, $[0, 1]^D$)
- λ is complexity parameter of the Mondrian process

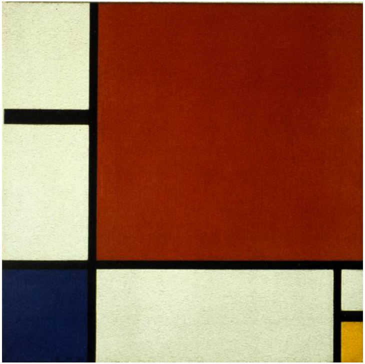
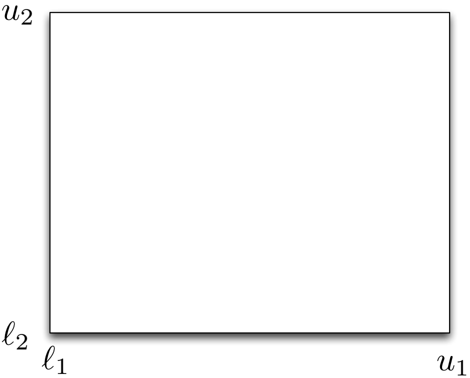


Figure: Mondrian Composition II in Red, Blue and Yellow (Source: Wikipedia)

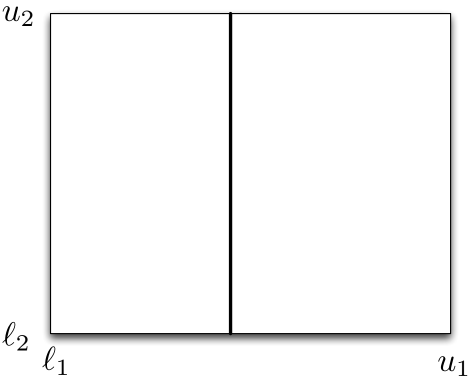
Generative process: MP$(\lambda, \{[\ell_1, u_1], [\ell_2, u_2]\})$

1. Draw Δ from exponential with rate $u_1 - \ell_1 + u_2 - \ell_2$
2. **IF** $\Delta > \lambda$ stop,



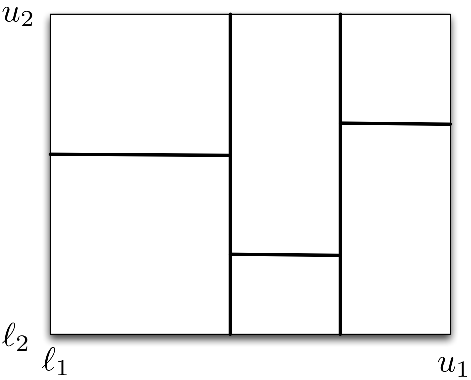
Generative process: MP$(\lambda, \{[\ell_1, u_1], [\ell_2, u_2]\})$

1. Draw Δ from exponential with rate $u_1 - \ell_1 + u_2 - \ell_2$
2. **IF** $\Delta > \lambda$ stop, **ELSE**, sample a split
   - split dimension: choose dimension d with prob $\propto u_d - \ell_d$
   - split location: choose uniformly from $[\ell_d, u_d]$



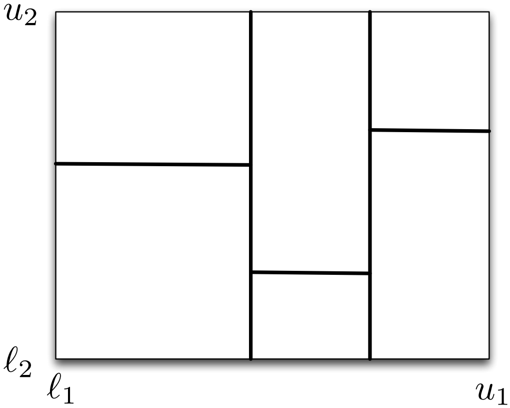
Generative process: MP$(\lambda, \{[\ell_1, u_1], [\ell_2, u_2]\})$

1. Draw Δ from exponential with rate $u_1 - \ell_1 + u_2 - \ell_2$
2. **IF** $\Delta > \lambda$ stop, **ELSE**, sample cut
   - Choose dimension d with probability $\propto u_d - \ell_d$
   - Choose cut location uniformly from $[\ell_d, u_d]$
   - Recurse on left and right subtrees with parameter $\lambda - \Delta$



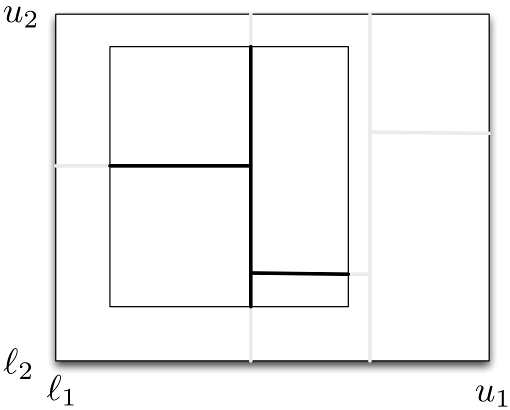
Self-consistency of Mondrian process

- Simulate $\mathcal{T} \sim \mathrm{MP}(\lambda, [\ell_1, u_1], [\ell_2, u_2])$



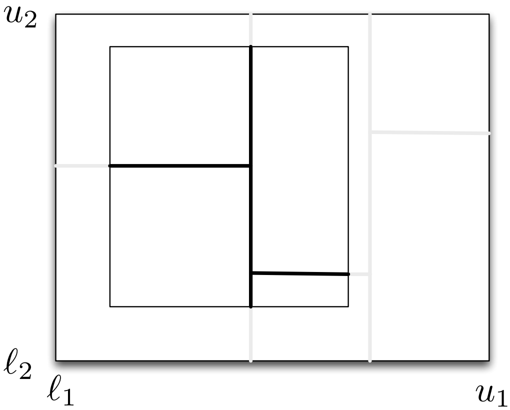
Self-consistency of Mondrian process

- Simulate $\mathcal{T} \sim \mathrm{MP}(\lambda, [\ell_1, u_1], [\ell_2, u_2])$
- Restrict $\mathcal{T}$ to a smaller rectangle $[\ell'_1, u'_1] \times [\ell'_2, u'_2]$



Self-consistency of Mondrian process

- Simulate $\mathcal{T} \sim \mathrm{MP}(\lambda, [\ell_1, u_1], [\ell_2, u_2])$
- Restrict $\mathcal{T}$ to a smaller rectangle $[\ell'_1, u'_1] \times [\ell'_2, u'_2]$



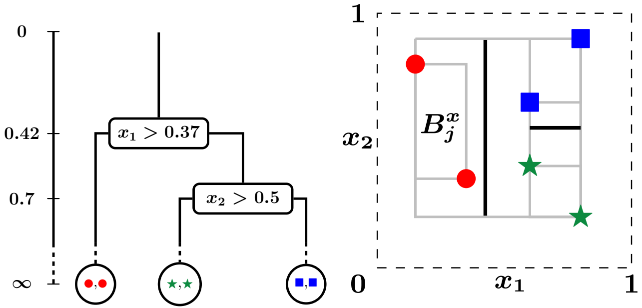
- Restriction has distribution $\mathrm{MP}(\lambda, [\ell'_1, u'_1], [\ell'_2, u'_2])$!

Mondrian trees

- Use X to define lower and upper limits within each node and use MP to sample splits

Mondrian trees

- Use X to define lower and upper limits within each node and use MP to sample splits
- Difference between Mondrian tree and usual decision tree
  - split in node j is committed only within extent of training data in node j
  - node j is associated with 'time of split' $t_j > 0$ (split time increases with depth and will be useful in online training)
  - splits are chosen independent of the labels Y



Outline

Mondrian trees: online learning

- As dataset grows, we extend the Mondrian tree $\mathcal{T}$ by simulating from a conditional Mondrian process MTx

Mondrian trees: online learning

- As dataset grows, we extend the Mondrian tree $\mathcal{T}$ by simulating from a conditional Mondrian process MTx

$$\left.\begin{array}{c} \mathcal{T} \sim \mathrm{MT}(\lambda, \mathcal{D}_{1:n}) \\ \mathcal{T}' \mid \mathcal{T}, \mathcal{D}_{1:n+1} \sim \mathrm{MTx}(\lambda, \mathcal{T}, \mathcal{D}_{n+1}) \end{array}\right. \implies \mathcal{T}' \sim \mathrm{MT}(\lambda, \mathcal{D}_{1:n+1})$$

- Distribution of batch and online trees are the same!
- Order of the data points does not matter

Mondrian trees: online learning

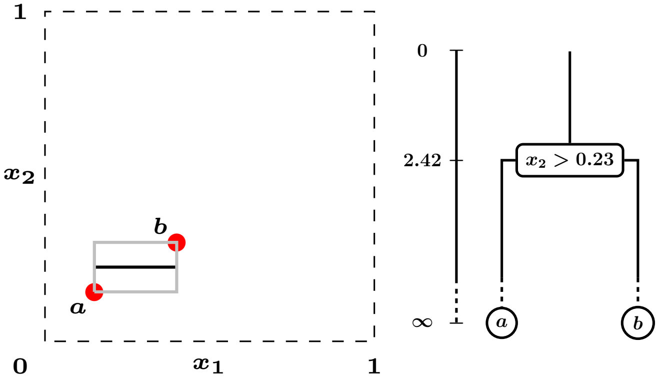
- As dataset grows, we extend the Mondrian tree $\mathcal{T}$ by simulating from a conditional Mondrian process MTx

$$\left.\begin{array}{c}\mathcal{T} \sim \mathrm{MT}(\lambda, \mathcal{D}_{1:n}) \\ \mathcal{T}' \mid \mathcal{T}, \mathcal{D}_{1:n+1} \sim \mathrm{MTx}(\lambda, \mathcal{T}, \mathcal{D}_{n+1})\end{array}\right. \implies \mathcal{T}' \sim \mathrm{MT}(\lambda, \mathcal{D}_{1:n+1})$$

- Distribution of batch and online trees are the same!
- Order of the data points does not matter
- MTx can perform one or more of the following 3 operations
  - insert new split above an existing split
  - extend existing split to new range
  - split leaf further
- Computational complexity MTx is linear in depth of tree

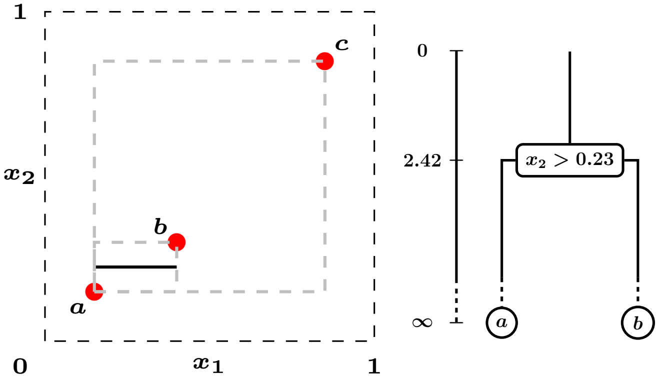
Online training cartoon

Start with data points *a* and *b*



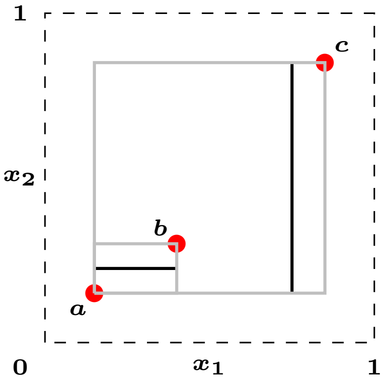
Online training cartoon

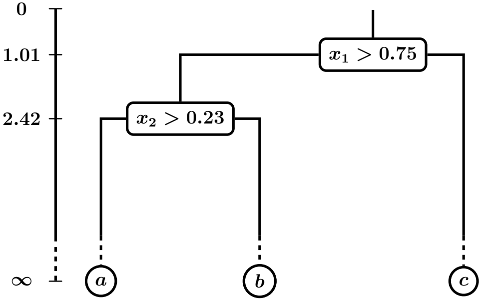
Adding new data point c: update visible range



Online training cartoon

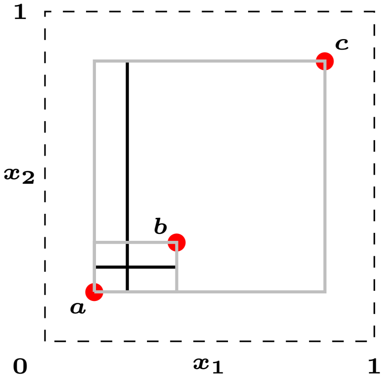
Adding new data point c: introduce new split (above an existing split). New split in R_{abc} should be consistent with R_{ab}.

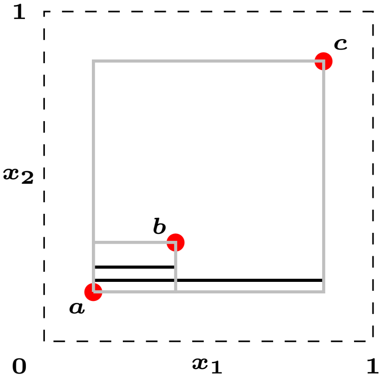




Online training cartoon

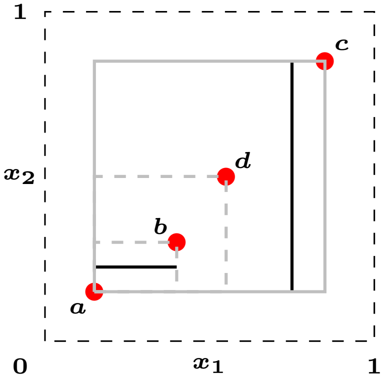
Examples of splits that are not self-consistent.

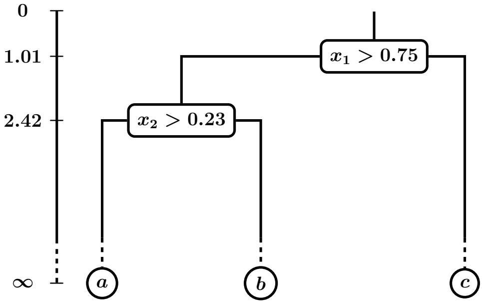




Online training cartoon

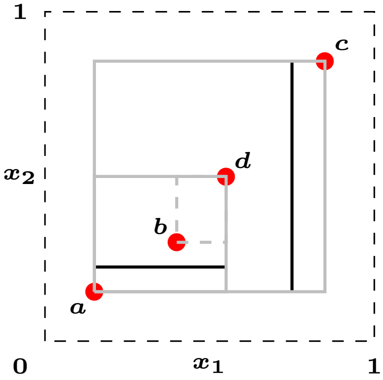
Adding new data point *d*: traverse to left child and update range

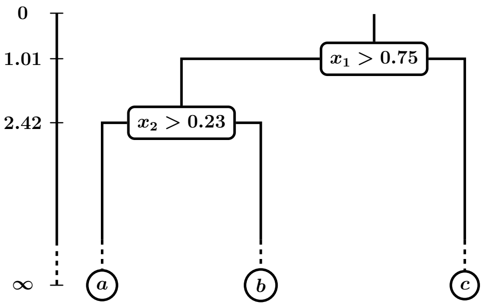




Online training cartoon

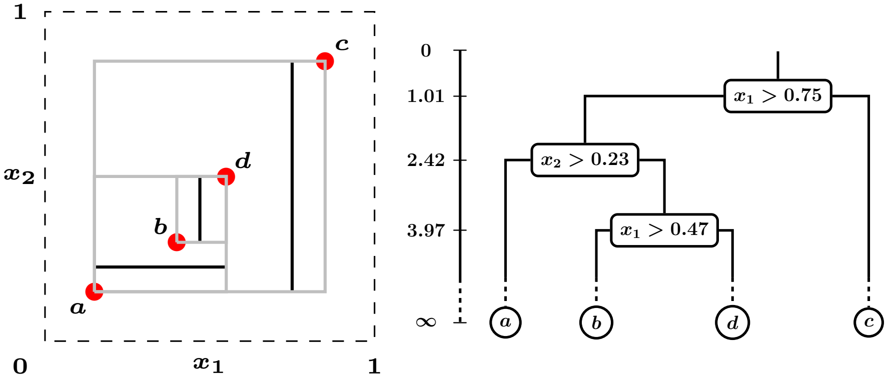
Adding new data point *d*: extend the existing split to new range





Online training cartoon

Adding new data point *d*: split leaf further



Key differences between Mondrian forests and existing online random forests

- Splits extended in a self-consistent fashion
- Splits not extended to unobserved regions
- New split can be introduced *anywhere* in the tree (as long as it's consistent with subtree below)

Prediction and Hierarchical smoothing

- Extend Mondrian to range of test data
  - Test data point can potentially branch off and form separate leaf node of its own!
  - Points far away from range of training data are more likely to brach off
  - We analytically average over every possible extension
- Hierarchical smoothing for posterior mean of $\theta|\mathcal{T}$
  - Independent prior would predict from prior if test data branches off into its own leaf node
  - Interpolated Kneser Ney approximation: fast
  - Can be interpreted as approximate posterior inference assuming Hierarchical Normalized Stable process prior
  - Smoothing done independently for each tree

Outline

Experimental setup

- Competitors
  - Periodically retrained RF, ERT
  - Online RF [Saffari et al., 2009]

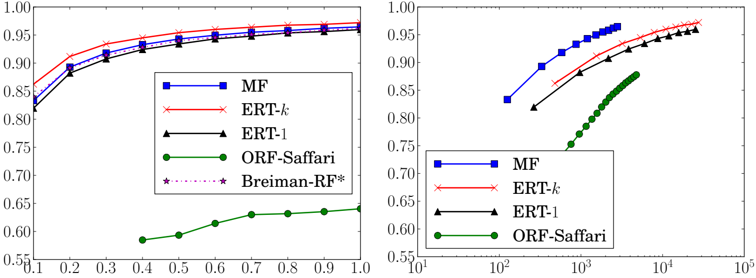
Experimental setup

- Competitors
  - Periodically retrained RF, ERT
  - Online RF [Saffari et al., 2009]
- Datasets:

Name	D	#Classes	#Train	#Test
Satellite images	36	6	3104	2000
Letter	16	26	15000	5000
USPS	256	10	7291	2007
DNA	180	3	1400	1186

- Training data split into 100 mini batches (unfair to MF)
- Number of trees $= 100$

Letter



Fraction of training data

Time (s)

Figure: Test accuracy

- Data efficiency: Online MF very close to batch RF (ERT, Breiman-RF) and significantly outperforms ORF-Saffari
- Speed: MF much faster than periodically re-trained batch RF and ORF-Saffari

USPS

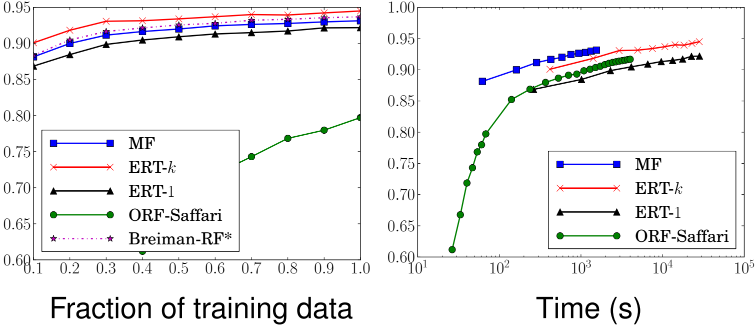
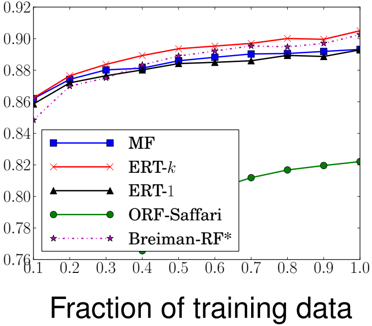


Figure: Test accuracy

Satellite Images



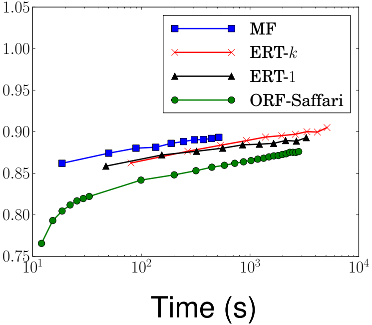


Figure: Test accuracy

So, what's the catch?

DNA

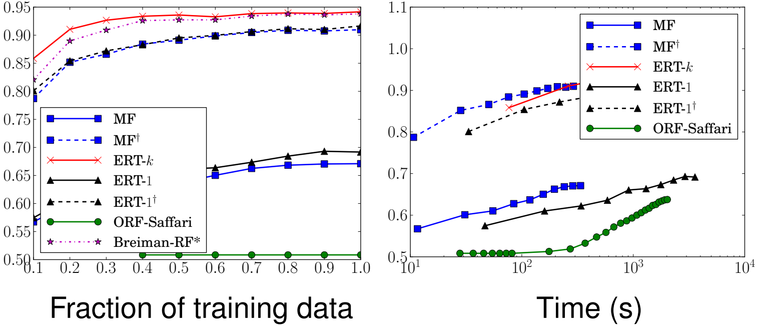


Figure: Test accuracy

- Irrelevant features: Choosing splits independent of labels (MF, ERT-1) harmful in presence of irrelevant features
- Removing irrelevant features (use only the 60 most relevant features[1]) improves test accuracy (MF†, ERT-1†)

[1] https://www.sgi.com/tech/mlc/db/DNA.names

Conclusion

- MF: Alternative to RF that supports incremental learning
- Computationally faster compared to existing online RF and periodically re-trained batch RF
- Data efficient compared to existing online RF
- Future work
  - Mondrian forests for regression
  - Mondrian forests for high dimensional data with lots of irrelevant features

Thank you!

code, paper: http://www.gatsby.ucl.ac.uk/~balaji

Questions?

References I

Breiman, L. (2001).
Random forests.
Mach. Learn., 45(1):5–32.

Caruana, R. and Niculescu-Mizil, A. (2006).
An empirical comparison of supervised learning algorithms.
In *Proc. Int. Conf. Mach. Learn. (ICML)*.

Denil, M., Matheson, D., and de Freitas, N. (2013).
Consistency of online random forests.
In *Proc. Int. Conf. Mach. Learn. (ICML)*.

Fernández-Delgado, M., Cernadas, E., Barro, S., and Amorim, D. (2014).
Do we need hundreds of classifiers to solve real world classification problems?
Journal of Machine Learning Research, 15:3133–3181.

References II

Geurts, P., Ernst, D., and Wehenkel, L. (2006).
Extremely randomized trees.
Mach. Learn., 63(1):3–42.

Roy, D. M. and Teh, Y. W. (2009).
The Mondrian process.
In *Adv. Neural Inform. Proc. Syst. (NIPS)*.

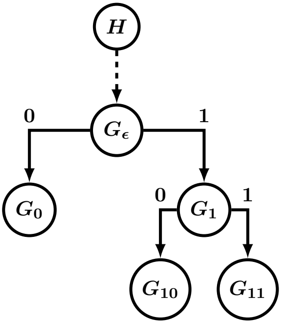
Saffari, A., Leistner, C., Santner, J., Godec, M., and Bischof, H. (2009).
On-line random forests.
In *Computer Vision Workshops (ICCV Workshops)*. IEEE.

Teh, Y. W. (2006).
A hierarchical Bayesian language model based on Pitman–Yor processes.
In *Proc. 21st Int. Conf. on Comp. Ling. and 44th Ann. Meeting Assoc. Comp. Ling.*, pages 985–992. Assoc. for Comp. Ling.

Extra slides

Hierarchical prior over θ

- G_j parametrizes $p(y|x)$ in B_j^x
- Normalized stable process (NSP): special case of PYP where concentration = 0
- $d_j \in (0, 1)$ is discount for node j
- $G_\epsilon | H \sim \text{NSP}(d_\epsilon, H)$,
  $G_{j0} | G_j \sim \text{NSP}(d_{j0}, G_j)$,
  $G_{j1} | G_j \sim \text{NSP}(d_{j1}, G_j)$



- $\mathbb{E}[G_\epsilon(s)] = H(s)$
- $\text{Var}[G_\epsilon(s)] = (1 - d_H) H(s) \big(1 - H(s)\big)$
- Closed under Marginalization: $G_0 | H \sim \text{NSP}(d_\epsilon d_0, H)$
- $d_j = e^{-\gamma \Delta_j}$ where $\Delta_j = t_j - t_{\text{parent}(j)}$ (time difference between split times)

Posterior inference for NSP

- Special case of approximate inference for PYP [Teh, 2006]
- Chinese restaurant process representation
- **Interpolated Kneser-Ney smoothing**
  - fast approximation
  - Restrict number of tables serving a dish to at most 1
  - popular smoothing technique in language modeling

Interpolated Kneser-Ney smoothing

- Prediction for x_* lying in node j is given by

$$\overline{G}_{jk} = p(y_* = k | x_* \in B_j^x, X, Y, \mathcal{T})$$
$$= \begin{cases} \dfrac{c_{j,k} - d_j \, \mathrm{tab}_{j,k}}{c_{j,\cdot}} + \dfrac{d_j \, \mathrm{tab}_{j,\cdot}}{c_{j,\cdot}} \, \overline{G}_{\mathrm{parent}(j),k} & c_{j,\cdot} > 0 \\ \overline{G}_{\mathrm{parent}(j),k} & c_{j,\cdot} = 0 \end{cases}$$

- $c_{j,k}$ = number of points in node j with label k
- $\mathrm{tab}_{j,k} = \min(c_{j,k}, 1)$ and $d_j = \exp\big(-\gamma(t_j - t_{\mathrm{parent}(j)})\big)$