# Ensembles on Random Patches

Gilles Louppe and Pierre Geurts

Université de Liège

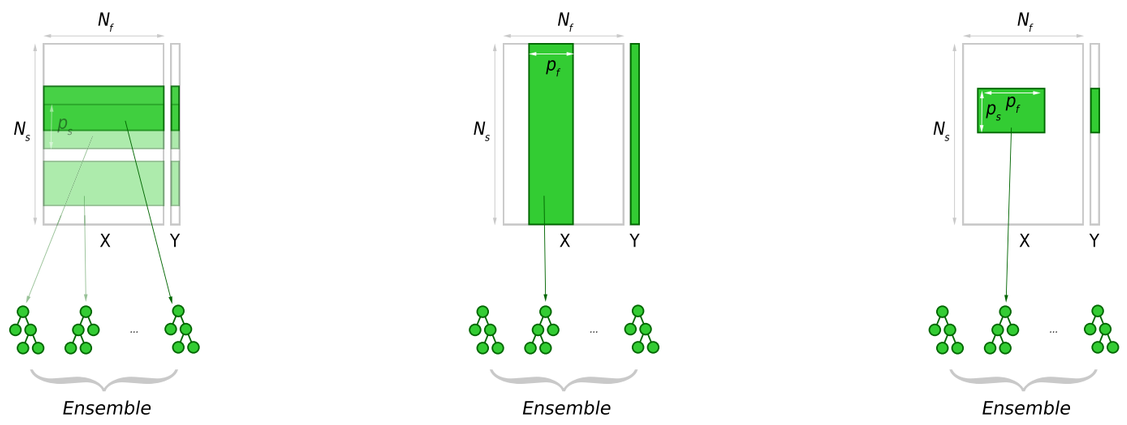
## Abstract

In this paper, we consider supervised learning under the assumption that the available memory is small compared to the dataset size. This general framework is relevant in the context of big data, distributed databases and embedded systems. **We investigate a very simple, yet effective, ensemble framework that builds each individual model of the ensemble from a random patch of data obtained by drawing random subsets of both instances and features from the whole dataset.** We carry out an extensive and systematic evaluation of this method on 29 datasets, using decision tree-based estimators. With respect to popular ensemble methods, these experiments show that **the proposed method provides on par performance in terms of accuracy while simultaneously lowering the memory needs, and attains significantly better performance when memory is severely constrained.**

## Framework

*Ensembles on Random Patches ?*

Pasting (P) [Breiman, 1999] + Random Subspaces (RS) [Ho, 1998] = Random Patches (RP) [This work]




1. Draw a subsample $r$ of $p_s N_s$ random examples, with $p_f N_f$ random features.
2. Build a base estimator on $r$.
3. Repeat 1-2 for a number $T$ of estimators.
4. Aggregate the predictions by voting.

**Goal :** *Reduce computing times (as P) while improving accuracy (as RS) ?*

## Tree-based methods

*Trees ?*

**Random Forest (RF)** [Breiman, 2001]

- Ensemble of randomized trees built on bootstrap samples (approx., $p_s = 0.632$).
- At each internal node, the chosen split is the best among *optimized* splits (cut-points) over $K$ features drawn at random.

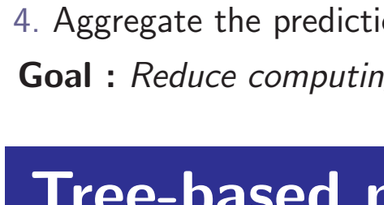
**Extra-Trees (ET)** [Geurts, 2006]

- Ensemble of randomized trees built on the entire set ($p_s = 1.0$).
- At each internal node, the chosen split is the best among $K$ *random* splits (cut-points) over $K$ features drawn at random.

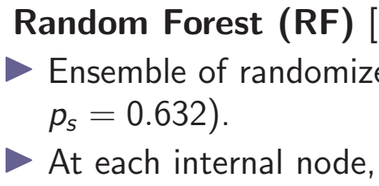
For both methods, $K$ features are re-drawn locally at each node. By contrast, in Random Patches, $p_f N_f$ features are drawn **once**, globally.

## Accuracy

*How such ensembles compare with others ?*




*Results on small datasets.*




*Results on larger datasets.*

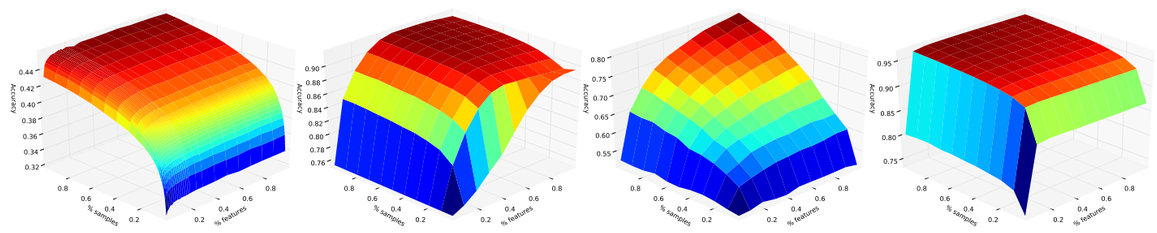
- Full comparison of 8 methods on 16+13 datasets, using either standard decision trees (-DT) or randomized decision trees (-ET) as base estimators.
- As expected, RP shows to be as good as P and RS. It improves wrt P but not wrt RS.
- Global feature sampling does not impair accuracy. RP and RS are as good as ET and better than RF.
- Tuned example sampling, as P does, is often ineffective. (Though it reduces computing times.)

### Conclusions (I)

- In terms of accuracy, ensembles built on random patches are usually as good as the other methods.
- Random Patches and Random Subspaces are on par, while Pasting performs less well. Sampling features is critical to improve accuracy.
- N.B. : Randomizing cut-points (à la Extra-Trees) is most of the time beneficial.

## Sensitivity

*Why tuning both $p_s$ and $p_f$ ?*



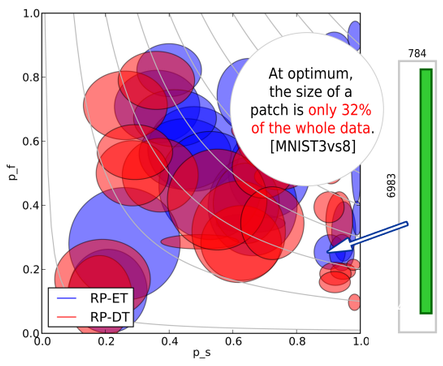

Accuracy increases either with $p_s$, or with $p_f$, or with both or plateaus.

### Conclusions (II)

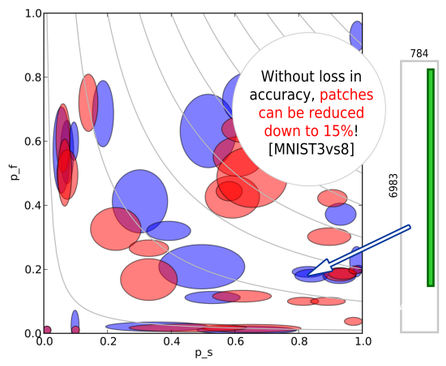
- Neither Pasting nor Random Subspaces can work well for all datasets.
- Both $p_s$ and $p_f$ need to be chosen on a per-dataset basis.

## Size of the patches

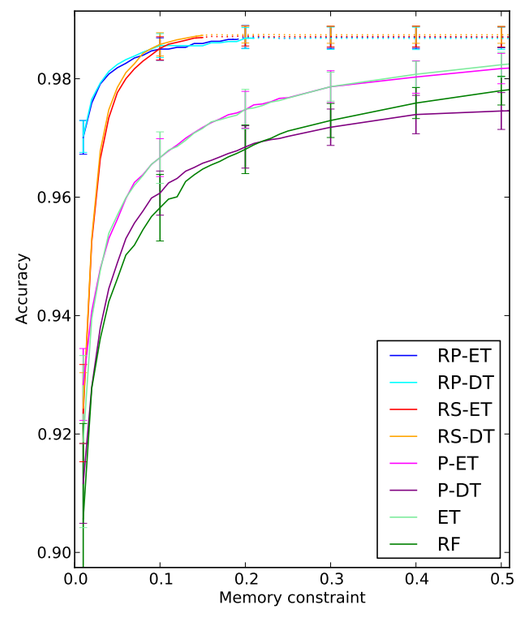
*Optimal size ? Smaller size ?*




*Optimally tuned patch sizes.*




*Minimal size without significant impact on accuracy.*




Accuracy using only 1% of data per tree [MNIST3vs8] :

| Method | Accuracy |
|---|---|
| Random Patches | 0.970 |
| Pasting | 0.928 |
| Random Subspaces | 0.924 |
| Extra-Trees | 0.918 |
| Random Forest | 0.905 |

- At the cost of accuracy, the size of the patches can be reduced even further.
- Though, RP minimizes that loss because it can find the right trade-off between $p_s$ and $p_f$.

### Conclusions (III)

- Training each estimator on the whole data is (often) useless. The size of the random patches can be reduced without (significant) loss in accuracy.
- As a result, both memory consumption and training time can be reduced, at low cost.
- With very small patches, accuracy degrades. Yet, RP exploits data better than the other methods.
- Building estimators on different subsamples is better than building them all on a same sample.

## Big data

*So what ?*

- Assume that your dataset $D$ is much larger than your memory of size $M$. *How to build a model out of it ?*
- Solution : Build a Random Patches ensemble on $D$ !

1. Draw random patches of size $p_s N_s \times p_f N_f < M$ and build an ensemble out of them.
2. Adjust both $p_s$ and $p_f$ to maximize accuracy.

*Questions?* `http://www.montefiore.ulg.ac.be/~glouppe`