Ensembles on Random Patches

Gilles Louppe and Pierre Geurts

Dept. of EE & CS, & GIGA-R
Université de Liège, Belgium

September 25, 2012

Big data

Big data has become ubiquitous

- Life sciences, computer vision, web applications, finance, ...

Big data

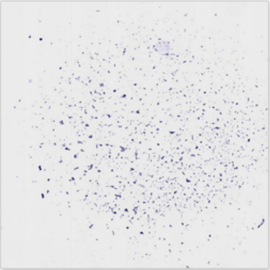
Big data has become ubiquitous

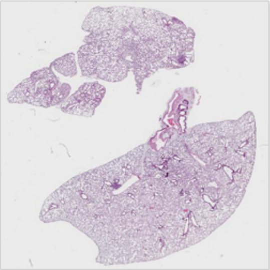
- Life sciences, computer vision, web applications, finance, ...

Big data is challenging !

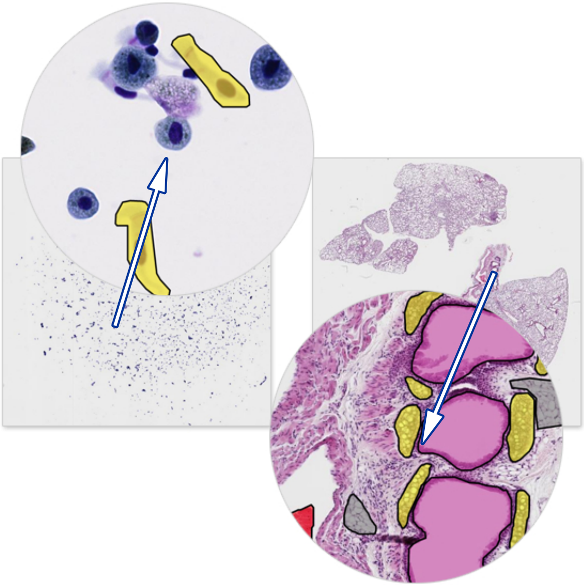
- Large number of examples (millions to billions), large number of features (thousands to millions)
- So large that classical machine learning algorithms are no longer fit.

Big data, an example





Big data, an example



Outline

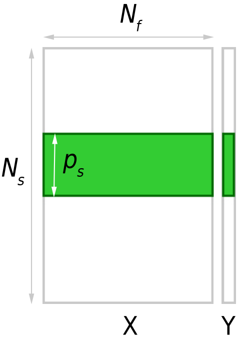
Ensembles on Random Patches?

1. Framework
   - Pasting
   - Random Subspaces
   - Random Patches
   - Tree-based methods

Pasting (P) [Breiman, 1999]

Goal : Reduce computing times.

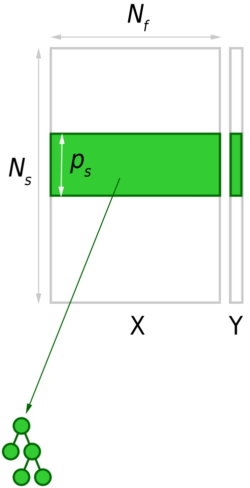
Pasting (P) [Breiman, 1999]



Goal : Reduce computing times.

1. Draw a subsample r of $p_s N_s$ ($p_s \in (0, 1]$) random examples, with all N_f features.

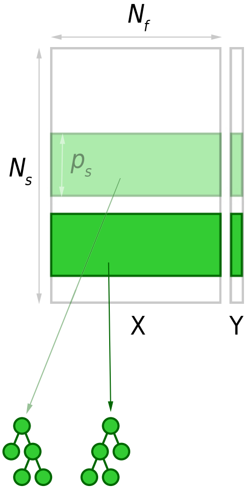
Pasting (P) [Breiman, 1999]



Goal : Reduce computing times.

1. Draw a subsample r of $p_s N_s$ ($p_s \in (0, 1]$) random examples, with all N_f features.
2. Build a base estimator on r.

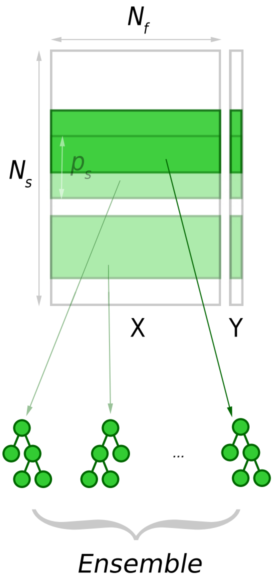
Pasting (P) [Breiman, 1999]



Goal : Reduce computing times.

1. Draw a subsample r of $p_s N_s$ ($p_s \in (0, 1]$) random examples, with all N_f features.
2. Build a base estimator on r.
3. Repeat 1-2 for a number T of estimators.

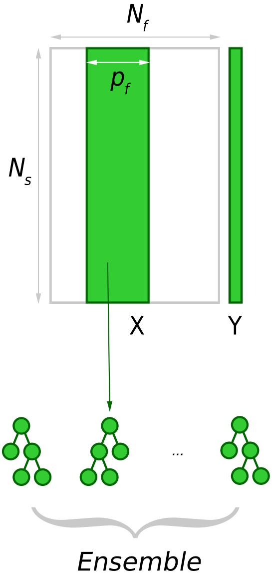
Pasting (P) [Breiman, 1999]



Goal : Reduce computing times.

1. Draw a subsample r of $p_s N_s$ ($p_s \in (0, 1]$) random examples, with all N_f features.
2. Build a base estimator on r.
3. Repeat 1-2 for a number T of estimators.
4. Aggregate the predictions by voting.

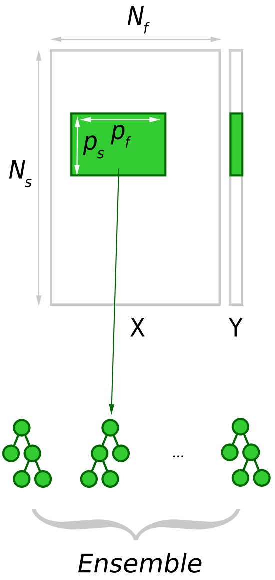
Random Subspaces (RS) [Ho, 1998]



Goal : Improve accuracy.

1. Draw a subsample r of all N_s examples, with $p_f N_f$ ($p_f \in (0, 1]$) 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.

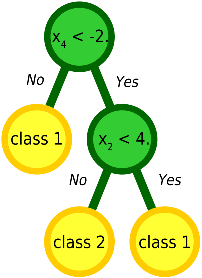
Random Patches (RP) [This work]



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.

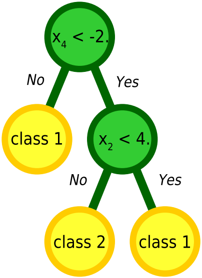
Goal : *Reduce computing times while improving accuracy ?*

Tree-based methods



A decision tree

Tree-based methods

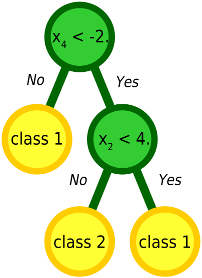


A decision tree

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.

Tree-based methods



A decision tree

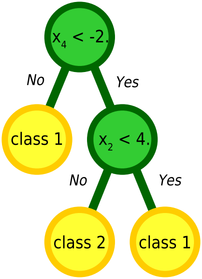
Random Forest (RF) [Breiman, 2001]

- 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.

Tree-based methods



A decision tree

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.

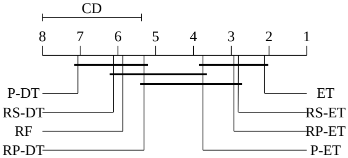
Extra-Trees (ET) [Geurts, 2006]

- 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.

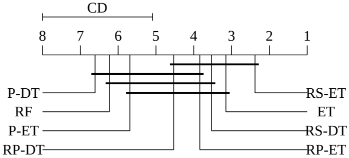
For both methods, K features are re-drawn locally at each node. By contrast, in Random Patches, $p_f N_f$ features are drawn **once**, globally.

How such ensembles compare in terms of accuracy ?

Experimental results



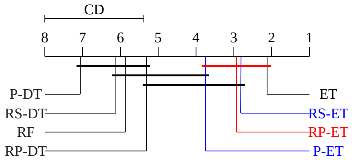
Results on small datasets.



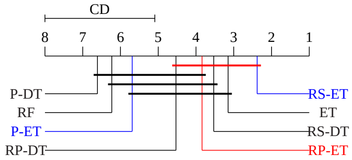
Results on larger datasets.

- Full comparison of 8 methods on 16+13 datasets, using either standard decision trees (-DT) or randomized decision trees (-ET) as base estimators.

Experimental results



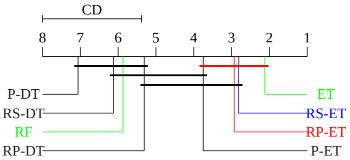
Results on small datasets.



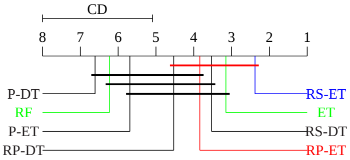
Results on larger datasets.

- 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.

Experimental results



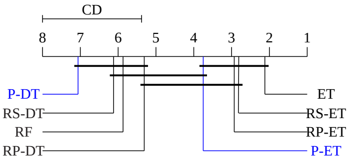
Results on small datasets.



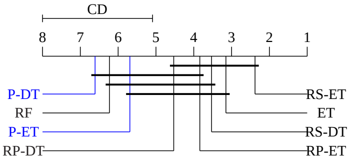
Results on larger datasets.

- 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.

Experimental results



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.

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.

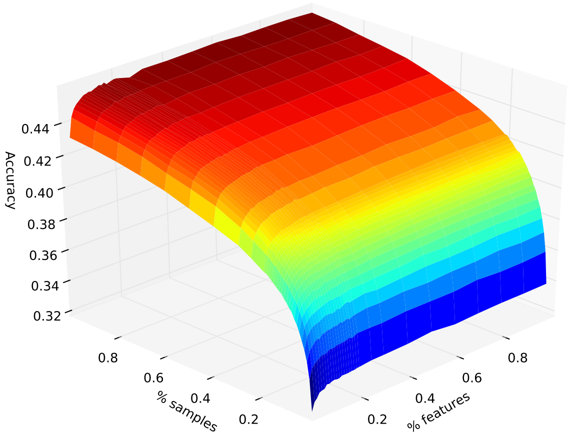
Conclusions (I)

- 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.

Why tuning both p_s and p_f ?

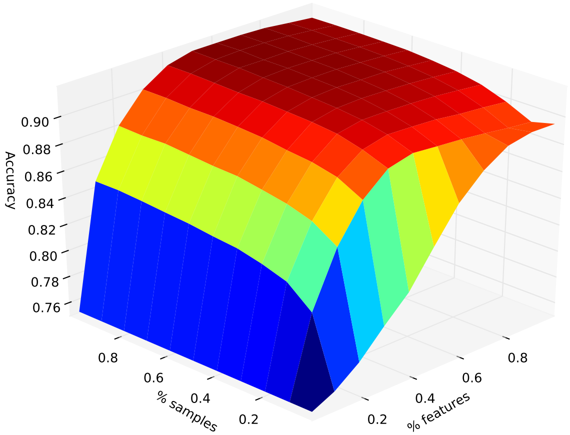
3. Sensitivity
   - Sensitivity to p_s
   - Sensitivity to p_f
   - Sensitivity to p_s and p_f
   - Plateaus
   - Conclusions

Sensitivity to p_s



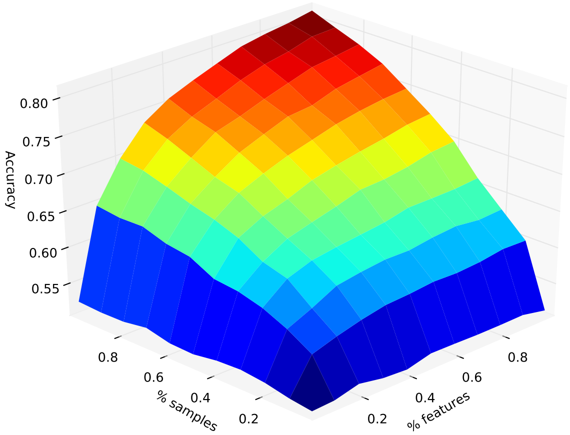
On some datasets, accuracy mainly increases with p_s, while p_f has a limited effect.

Sensitivity to p_f



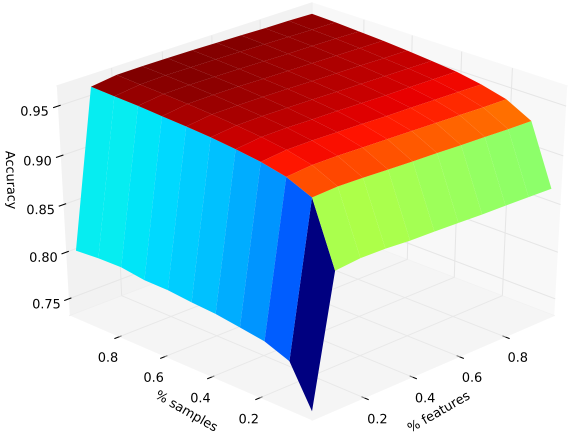
On others, accuracy mainly increases with p_f,
while p_s has a limited effect.

Sensitivity to p_s and p_f



On yet others, accuracy increases with both p_s and p_f.

Plateaus



Finally, accuracy may also plateau with p_s and p_f.

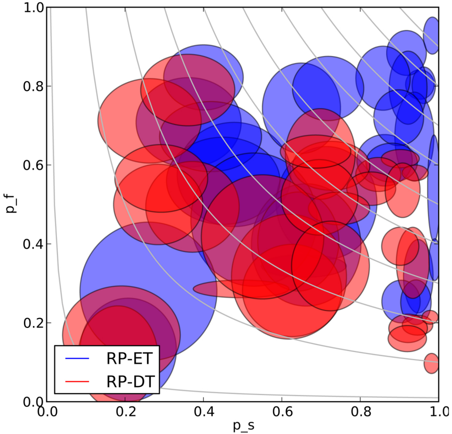
Conclusions (II)

- 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.

What is the optimal size of the patches ?

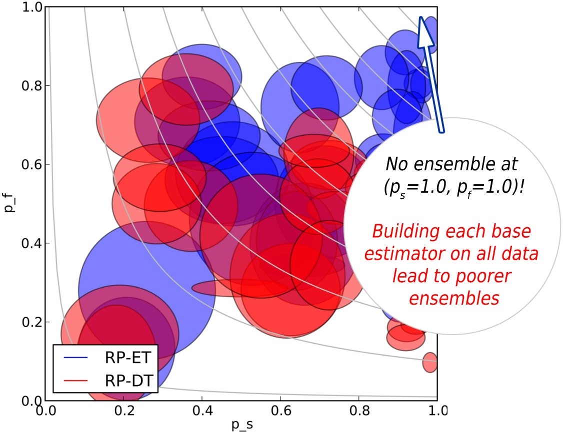
Can they be reduced without affecting (too much) accuracy ?

Optimal size of the patches



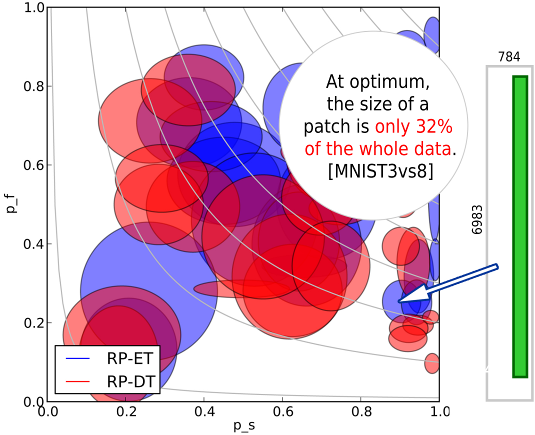
Optimally tuned patch sizes.

Optimal size of the patches



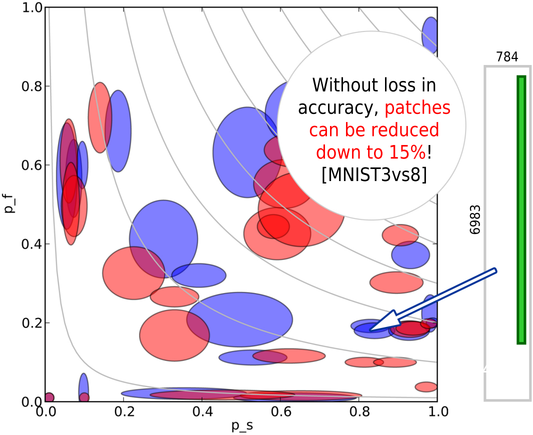
Optimally tuned patch sizes.

Optimal size of the patches



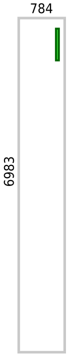
Optimally tuned patch sizes.

Reducing the size of the patches



Minimal size without significant impact on accuracy.

Reducing **further** the size of the patches



On MNIST3vs8, accuracy only drops from 0.986 to 0.970 when the size of a patch is reduced to 1% of the whole data.

- 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.

TABLE: Accuracy at 1% [MNIST3vs8]

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

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.

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.

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.

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.

So what ?

5 Conclusions
- Back to big data
- Future work
- Questions ?

Back to big data

- Assume that your dataset D is much larger than your memory of size M. *How to build a model out of it ?*

Back to big data

- 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.

Future work

- Experiments on giga-scale datasets (ongoing work).
- Automatic tuning of p_s and p_f.
- Theoretical analysis
  - *How small can random patches be ?*
  - *Under which assumptions ?*

Questions ?