Random Subspace with Trees for Feature Selection Under Memory Constraints
Supplementary materials

Antonio Sutera[1] **Célia Châtel**[2] **Gilles Louppe**[1,3] **Louis Wehenkel**[1] **Pierre Geurts**[1]
[1]University of Liège, Belgium [2]Aix-Marseille University, France [3]New York University, USA

A Proof of Theorem 2

Theorem 2. *$\forall \alpha, K$, if $r \leq q$:*

$$X \textit{ strongly relevant} \Rightarrow X \in F_{q,\infty}^{K,\alpha}$$

Proof. By definition, X belonging to $F_{q,\infty}^{K,\alpha}$ means that there is at least one tree (grown with parameters q, α and K) in which X receives a strictly positive score for its split, i.e. such that Y depends on X conditionally to the variable assignement defined by the path from the root node to the node where X is used to split. Let us show that one such tree always exists whatever K and α when X is strongly relevant and $r \leq q$.

Within the infinite ensemble, let us consider only the trees grown using all r relevant variables (and $q - r$ irrelevant ones randomly selected). Given that $r \leq q$ and given that only relevant features can be kept in memory, these trees are always explored whatever the value of $\alpha \geq 0$. Among these trees, let us furthermore only consider those such that irrelevant variables are tested in each branch only when all relevant variables (including X) are exhausted. These trees are always explored whatever the value of K. This derives from the fact that a relevant variable can always be picked with non zero probability at any tree node, except if all relevant variables have been tested above that node. Indeed, except in this latter case, the K tested variables can always include at least one relevant variable. If some relevant variable gets a non zero score, one relevant variable will be automatically used to split since irrelevant variables can only get zero scores. Even when all tested relevant variables get a zero score, one of them can still be selected instead of an irrelevant one given that tie are resolved by randomization.

Let us denote by τ_R the set of trees as just defined and let us show that X gets a non zero score in at least one tree in τ_R.

By definition 2 and property 1, X strongly relevant implies that there exists at least one assignement of values to all relevant variables but X such that conditionally to this assignement, Y is dependent on X. In each tree in τ_R, there is a path from the root node to a node where X is used to split that is compatible with this assignement. Let us assume that X always gets a zero score in all these compatible paths and show that this leads to a contradiction.

If all relevant variables are tested above X in a compatible path then X should receive a non zero score at its node, which would contradict our hypothesis. Thus, X can only be tested in a compatible path before all relevant variables have been tested. Given our hypothesis that X only gets zero scores, if X is used to split in one compatible path, then there exists another tree in τ_R with the same splits above X in the compatible path and with the split on X replaced by a split on another relevant variables (because of tie randomization or because of the randomization due to $K < q$). In this new tree, X is thus used to split at least one level below in the compatible path. Applying this argument recursively, one can thus show that there is at least one tree in τ_R where X is the last variable used to split in the compatible path. In this tree, X thus gets a non zero score, which contradicts the hypothesis and therefore concludes the theorem. □

B Convergence analysis

B.1 Simplifying assumptions

Below, we compute analytically the average number of trees needed to find all relevant variables in the chaining and clique scenarios and we derive transition matrices of Markov chains that model the evolution of the number of variables found through the iterations in the three scenarios. These results are obtained assuming $K = q$ and $r \leq q$, and with either $\alpha = 0$ (RS) or $\alpha = 1$ (SRS).

To make these derivations possible and independent of a particular data distribution, one needs furthermore to simplify the decision tree growing algorithm in the case of the chaining and clique scenarios. In what follows, trees are thus assumed to be grown such that a unique variable is selected at each tree level and this variable is selected at random among all variables X such that $Y \not\perp X|B$ where B is the set of all variables tested at previous levels.

In the clique scenario, this assumption implies that only one variable of the clique will get a non-zero importance when all clique variables are selected at one iteration of RS/SRS (since only the last variable of the clique tested along a tree branch can get a non-zero score and this variable is the same in each branch given our tree growing assumption). This corresponds to a pessimistic scenario. Indeed, with standard unconstrained trees, several relevant variables could be found at one iteration given that the ordering of the variables, and thus the last variable of the clique tested, might differ from one tree branch to another. As a consequence, the tree growing assumption will lead to an overestimation of the number of trees needed to reach convergence. In the chaining scenario, the simplified tree growing algorithm implies that all relevant variables selected at one iteration of RS/SRS together with their minimal conditioning will get a non-zero importance. This corresponds this time to an optimistic scenario, as, with unconstrained trees, such variable might not be detected at one iteration depending on the exact data distribution. This will thus lead this time to an underestimation of the number of trees needed to reach convergence. Note however that, in both cases, these over/under-estimations will affect both RS and SRS in the same proportion and thus our assumption will not impact their relative performance.

Note that in the marginal-only scenario, given that all relevant variables are marginally and strongly relevant, they will always get a non-zero importance as soon as they are selected at one iteration. Our estimations below are thus not impacted by the simplification of the tree growing algorithm.

B.2 Average times

Chaining. Let us denote by $T^{RS}_{chain}(i,p,q)$ ($1 \leq i \leq r$) the average number of iterations needed to find the feature X_i of degree $i-1$ and by $T^{SRS}_{chain}(i,p,q)$ the average number of iterations needed to find the same feature with the SRS algorithm (that forces the selection of already found relevant variables). Given our assumptions above, each tree will be able to identify all relevant variables X it gets as soon as it gets also the relevant variables in its minimal conditioning. Note that $T^{RS/SRS}_{chain}(i,p,q)$ can also be interpreted as the average time needed to find the first i relevant features, given that one can not find X_i without finding all features X_j with $1 \leq j < i$. $T^{RS/SRS}_{chain}(r,p,q)$ also represents the average number of iterations needed to find all relevant variables under the chain assumption.

Theorem 4. *Under our assumptions, the T^{RS}_{chain} function can be computed as follows:*

$$T^{RS}_{chain}(i,p,q) = \prod_{l=0}^{i-1} \frac{p-l}{q-l} \tag{1}$$

Proof. Indeed, $T^{RS}_{chain}(i,p,q)$ is the mean of a geometric distributed random variable with a probability of success defined as the probability of drawing the $i-1$ variables in X_i's conditioning and X_i at the same time, which is given by:

$$\frac{\binom{p-i}{q-i}}{\binom{p}{q}} = \prod_{l=0}^{i-1} \frac{q-l}{p-l}. \tag{2}$$

□

Theorem 5. *Under the same assumption, $T^{SRS}_{chain}(i,p,q)$ can be computed as follows:*

$$T^{SRS}_{chain}(i,p,q) = \sum_{l=0}^{i-1} \frac{p-l}{q-l} - (i-1) \tag{3}$$

Proof. Let us show this by induction on i. The base case corresponds to $i=1$. In this case, we have:

$$T^{SRS}_{chain}(1,p,q) = T^{RS}_{chain}(1,p,q) = \frac{p}{q},$$

which satisfies Eqn (3). Let us assume that Eqn. (3) is satisfied for $i < i'$ and let us show that it is satisfied for $i = i'$. $T^{SRS}_{chain}(i',p,q)$ can be defined as follows:

$$\begin{aligned} T^{SRS}_{chain}(i',p,q) = \frac{q}{p} T^{SRS}_{chain}(i'-1,p-1,q-1) + \\ (1-\frac{q}{p})(1+T^{SRS}_{chain}(i',p,q)). \end{aligned} \tag{4}$$

One can indeed distinguish two cases:

- X_1 is selected at the first iteration (this happens with probability q/p): the average time needed to find feature $X_{i'}$ of degree $i'-1$ then becomes the time needed to find a feature of degree $i'-2$ when one is allowed to draw $q-1$ features among $p-1$, which is $T^{SRS}_{chain}(i'-1,p-1,q-1)$
- X_1 is not selected at the first iteration (this happens with probability $1-q/p$): in this case, the first iteration is useless and thus the number of iterations needed will be $1+T^{SRS}_{chain}(i',p,q)$.

Eqn. (4) can be used to compute T^{SRS}_{chain} recursively:

$$T^{SRS}_{chain}(i',p,q) = T^{SRS}_{chain}(i'-1,p-1,q-1) + (\frac{p}{q}-1). \tag{5}$$

Deriving Eqn. (3) from Eqn. (5) is then straightforward, which concludes the proof by induction. □

Eqn. (3) shows that the average time needed to find the i first features is equal to the sum of the time needed to find all features individually minus the number of features. This last term takes into account the fact that by chance, one might find several features at once.

Clique. Let us denote by $T^{RS}_{cl}(i,p,q)$ and $T^{SRS}_{cl}(i,p,q)$, the average time needed to find i features (among r) from the clique respectively with the RS and the SRS algorithm. Given our assumptions above, when the tree growing algorithm is given all r relevant features, it will be able to identify one (and only one) feature from the clique at random. If it has already found i features from the clique, the chance to get a new one, when all r features are selected among the q ones, will thus be $(r-i)/r$, i.e., the probability to test one of the $r-i$ not yet found features after all other r features from the clique.

Theorem 6.

$$T^{RS}_{cl}(i,p,q) = \left(\prod_{l=0}^{r-1}\frac{p-l}{q-l}\right)\cdot\left(\sum_{l=0}^{i-1}\frac{r}{r-l}\right) \tag{6}$$

Proof. The first factor in Eqn.(6) is the inverse of the probability of selecting all r relevant features at once. Each term of the sum in the second factor corresponds to the inverse of the probability of testing a new relevant variables, not yet found, at the bottom of the tree. As discussed above, this probability is $\frac{r-l}{r}$ when we have already found l features from the clique. □

Theorem 7.

$$T^{SRS}_{cl}(i,p,q) = \sum_{l=0}^{i-1}\frac{r}{r-l}\prod_{m=l}^{r-1}\frac{p-m}{q-m} \tag{7}$$

Proof. Each term of the sum represents the average time needed to find a new clique feature given that we have already found l features. This time is equal to one over the probability of finding a new feature when we have already found l of them. This latter is the probability of selecting among q the $r-l$ missing relevant features (i.e., $\prod_{m=l}^{r}\frac{q-m}{p-m}$) times the probability of testing one of the missing relevant features at the bottom of the tree (i.e., $(r-l)/r$). □

When $i=1$, $T^{SRS}_{cl}(1,p,q) = T^{RS}_{cl}(1,p,q)$. Intuitively, it indeed takes the same time for the RS and the SRS algorithms to find the first relevant features. When i increases however, the SRS algorithm becomes faster and faster than the RS algorithm. Indeed, the RS algorithm always needs to find all r clique features, while the SRS one only needs to find the $r-i$ missing relevant features.

B.3 Markov chain interpretation

Let us denote by $N^{X,Y}_t$ the number of variables found for t iterations, with $X=c$, $X=g$, and $X=m$ respectively for the chain hypothesis, the clique hypothesis and the marginal only hypothesis (as defined in the first section of this document) and $Y=n$ and $Y=s$ respectively for the RS and SRS algorithms. All these random variables follow order 1 Markov chains. The transition probabilities are provided below for each chain (without proof), under the assumptions given in Section B.1.

Chain hypothesis.

$$P(N^{c,n}_t = l_1 | N^{c,n}_t = l_2) = \begin{cases} 0 & \text{if } l_1 < l_2 \\ \frac{\binom{p-r}{q-l_1}}{\binom{p}{q}} & \text{if } l_1 > l_2 \\ 1-\sum_{i=l_2+1}^{r}\frac{\binom{p-r}{q-i}}{\binom{p}{q}} & \text{if } l_1 = l_2 \end{cases} \tag{8}$$

$$P(N^{c,s}_t = l_1 | N^{c,s}_t = l_2) = \begin{cases} 0 & \text{if } l_1 < l_2 \\ \frac{\binom{p-r}{q-l_1}}{\binom{p-l_2}{q-l_2}} & \text{if } l_1 > l_2 \\ 1-\sum_{i=l_2+1}^{r}\frac{\binom{p-r}{q-i}}{\binom{p-l_2}{q-l_2}} & \text{if } l_1 = l_2 \end{cases} \tag{9}$$

Clique hypothesis.

$$P(N^{g,n}_t = l_1 | N^{g,n}_t = l_2) = \begin{cases} 0 & \text{if } l_1 < l_2 \\ 1-\frac{\binom{p-r}{q-r}}{\binom{p}{q}}\frac{r-l_2}{r} & \text{if } l_1 = l_2 \\ \frac{\binom{p-r}{q-r}}{\binom{p}{q}}\frac{r-l_2}{r} & \text{if } l_1 = l_2+1 \\ 0 & \text{if } l_1 > l_2+1 \end{cases} \tag{10}$$

$$P(N^{g,s}_t = l_1 | N^{g,s}_t = l_2) = \begin{cases} 0 & \text{if } l_1 < l_2 \\ 1-\frac{\binom{p-r}{q-r}}{\binom{p-l_2}{q-l_2}}\frac{r-l_2}{r} & \text{if } l_1 = l_2 \\ \frac{\binom{p-r}{q-r}}{\binom{p-l_2}{q-l_2}}\frac{r-l_2}{r} & \text{if } l_1 = l_2+1 \\ 0 & \text{if } l_1 > l_2+1 \end{cases} \tag{11}$$

Marginal only hypothesis.

$$P(N^{m,n}_t = l_1 | N^{m,n}_t = l_2) = \begin{cases} 0 & \text{if } l_1 < l_2 \\ \frac{\binom{r-l_2}{l_1-l_2}\binom{p-r+l_2}{q-l_1+l_2}}{\binom{p}{q}} & \text{if } l_1 > l_2 \\ \frac{\binom{p-r+l_2}{q}}{\binom{p}{q}} & \text{if } l_1 = l_2 \end{cases} \tag{12}$$

$$P(N_t^{m,s} = l_1 | N_t^{m,s} = l_2) = \begin{cases} 0 & \text{if } l_1 < l_2 \\ \frac{\binom{r-l_2}{l_1-l_2}\binom{p-r}{q-l_1}}{\binom{p-l_2}{q-l_2}} & \text{if } l_1 > l_2 \\ \frac{\binom{p-r}{q-l_2}}{\binom{p-l_2}{q-l_2}} & \text{if } l_1 = l_2 \end{cases} \tag{13}$$

C Details for Section 5

In this section, we give more details about our practical implementation of SRS and performed experiments.

C.1 On the use of a random probe to distinguish relevant features from irrelevant features.

As explained in Section 5, we add an artificial irrelevant feature in data as a random probe. By comparison with that probe of importances scores, one can distinguish relevant features (better than the probe) from irrelevant features. Through iterations, we can compute a *p-value* score which is the percentage of times a variable has been better than the probe. If the *p-value* is above a given threshold β then the feature is likely relevant. Moreover, a variable has to be sampled more than L times in Q sets to insure that the *p-value* is reliable. Then at each iteration, the variables that satisfy the two criteria are added to F. In the following experiments, we choose arbitrarily $L = 10$ and $\beta = 95\%$.

C.2 On the datasets and on the protocol

We evaluate the accuracy of all these methods on a list of both artificial and real classifications problems (all but madelon are real data) described in Table 1 and publicly available in the UCI machine learning repository Lichman [2013]. For each dataset, we separate it into two random partitions of the same size (i.e., the same number of samples) to have a training set and a test set. There is no optimization of the parameters. For all datasets, the procedure was repeated 50 times, using the same random partitions between all methods. Following results are averages over those 50 runs.

C.3 Detailed results

Table 2 is average accuracy scores obtained on all datasets for each method for some parameters. We consider different sizes of memory (i.e., parameter q) and different value for the parameter α for the SRS algorithm. This allows to consider every behaviour of the SRS algorithm : without memory ($\alpha = 0$) which is equivalent to the Random Subspace method, with a full memory ($\alpha = 1$) and a non-full memory ($\alpha = 0.5$).

Dataset	# samples	# features
arcene	100	10000
breast2	295	24496
cina0	16033	132
isolet	7797	617
madelon	2000	500
marti0	500	1024
reged0	500	999
secom	1567	591
mnist	70000	784
mnist3v8	13966	784
mnist4v9	13782	784
sido0	12678	4932
tis	13375	927

Table 1: Dataset specifications

For both methods (RS and SRS), a single extra-tree is build at each iteration. The randomization parameter of the extra-tree is set to its maximal value (ie., all features). For the tree-based ensemble methods, we consider different values for the randomization parameter. This parameter reduces the ability to consider the whole dataset in once and in that it relates in a way to the size of the memory of SRS. We choose for that parameter values of 0.01, 0.1 and 1 corresponding to considering respectively 1%, 10%, 100% of all features at each node.

	SRS									Tree-based ensemble methods					
	q=0.01			q=0.05			q=0.1			RF			ET		
	α									Randomization parameter K					
	0.0	0.5	1.0	0.0	0.5	1.0	0.0	0.5	1.0	0.01	0.1	1	0.01	0.1	1
arcene	0.743	0.717	0.717	0.743	0.743	0.743	0.732	0.732	0.732	0.717	0.706	0.678	0.739	0.729	0.701
breast2	0.649	0.647	0.647	0.651	0.651	0.650	0.654	0.654	0.654	0.646	0.649	0.649	0.650	0.654	0.651
cina0	0.755	0.755	0.777	0.809	0.929	0.873	0.931	0.933	0.921	0.933	0.939	0.939	0.931	0.934	0.934
isolet	0.906	0.899	0.336	0.944	0.945	0.766	0.949	0.950	0.817	0.936	0.940	0.912	0.943	0.951	0.943
madelon	0.558	0.689	0.745	0.639	0.858	0.861	0.673	0.845	0.845	0.620	0.700	0.754	0.608	0.690	0.815
marti0	0.881	0.881	0.881	0.874	0.874	0.874	0.870	0.870	0.870	0.878	0.870	0.866	0.879	0.868	0.854
reged0	0.880	0.966	0.939	0.885	0.974	0.974	0.898	0.974	0.974	0.882	0.963	0.960	0.881	0.948	0.978
secom	0.935	0.935	0.930	0.935	0.931	0.931	0.934	0.932	0.932	0.935	0.933	0.929	0.935	0.930	0.928
mnist	0.564	0.823	0.525	0.959	0.966	0.905	0.968	0.970	0.938	0.964	0.966	0.953	0.966	0.971	0.968
mnist3v8	0.910	0.941	0.828	0.980	0.986	0.958	0.987	0.989	0.975	0.980	0.985	0.978	0.981	0.988	0.987
mnist4v9	0.889	0.957	0.848	0.981	0.986	0.960	0.986	0.988	0.974	0.983	0.984	0.974*	0.985	0.987	0.984*
sido0	0.970	0.972	0.953	0.973	0.968	0.968	0.974	0.969	0.969	0.972	0.973	0.973*	0.973	0.974	0.960*
tis	0.751	0.751	0.757	0.753	0.887	0.888	0.844	0.917	0.915	0.854	0.916	0.913*	0.856	0.906	0.914*

Table 2: Average accuracy scores for all methods with specified parameters on original datasets. SRS and RS were computed with 10000 iterations and RF/ET with 10000 trees.

	RS	SRS		ET	
	$q = 0.1$	$q = 0.1$	$q = 0.1$	$k = 0.1$	$k = 1.0$
$q = 0.1$		$\alpha = 0.5$	$\alpha = 1.0$		
RS	–	1/5/7	**6/4/3**	**7/2/4**	**6/3/4**
$\text{SRS}_{\alpha=0.5}$	**7/5/1**	–	5/8/0	**9/2/2**	**10/2/1**
$\text{SRS}_{\alpha=1.0}$	3/4/6	0/8/5	–	5/2/6	**6/2/5**

(a) $q = 0.1 \times p$

	RS	SRS	
	$q = 0.01$	$q = 0.01$	$q = 0.01$
$q = 0.01$		$\alpha = 0.5$	$\alpha = 1.0$
RS	–	2/2/9	5/2/6
$\text{SRS}_{\alpha=0.5}$	**9/2/2**	–	**7/3/3**
$\text{SRS}_{\alpha=1.0}$	**6/2/5**	3/3/7	–

(b) $q = 0.01 \times p$

Table 3: Pairwise t-test (with a significance level of 0.05) comparisons : each element on line i and column j of the table in terms of Win/Draw/Loss is the result of the comparison for method i vs. method j: the tree values indicate respectively on how many datasets method i is significantly better / not significantly different / significantly worse than method j. All methods were computed with 10000 iterations or trees on all 14 datasets (from Table 1) with parameters specified on columns. In **bold** when the first value is greater than other values.

References

Moshe Lichman. UCI machine learning repository, 2013.
URL `http://archive.ics.uci.edu/ml`.