Stochastic Aware Random Forests — A Variation Less Impacted by Randomness

Paulo Fernandes * and **Lucelene Lopes** and **Silvio Normey** and **Duncan Ruiz**
Av. Ipiranga, 6681 - 90.619-900 - Porto Alegre - Brazil
{ *paulo.fernandes, lucelene.lopes, silvio.normey, duncan.ruiz*}*@pucrs.br*

Abstract

The impact of random choices is important to many ensemble classifiers algorithms, and the Random Forests is particularly sensible to pseudo-random number generation decisions. This paper proposes an extension to the classical Random Forests method that aims to reduce its sensibility to randomness. The benefits brought by such extension are illustrated by a large number of experiments over 32 different public data sets.

The effectiveness of ensemble classifiers for classification tasks in the machine learning area is a known fact. Classical methods as Bagging (Breiman 1996) and Random Forests (Breiman 2001) are widely spread in both researchers and practitioners communities. However, all ensemble classifiers rely on pseudo-random choices to generate randomly altered versions of the original data sets (Banfield et al. 2007), and those choices may have a non-negligible impact on accuracy, as was demonstrated for Bagging and Boosting (Fernandes, Lopes, and Ruiz 2010).

The data mining research area is rich in propositions to enhance well establish methods by means of efficient algorithms to implement classical methods, or by proposing extensions to the methods themselves. Examples of efficient algorithm efforts are the works of Kulkarni (2012), as well as all the implementations within the WEKA software tool (Hall et al. 2009).

Different approaches are related to extend existing methods by empirical changes. Guo and Fan (2011) present an extension to ensemble classifiers methods that is based on selection of classifiers according to Ensemble Pruning via Based-Classifier Replacement (EPR). Their technique aims to reduce ensemble size (less classifiers) while increase accuracy. DeBarr and Wechsler (2012) work is an applied effort that uses some popular ensemble classifiers to spam detection, but it also proposes an extension, called Random Boost, that adds to the traditional Random Forests some Logit Boost techniques (*e.g.*, weighting on classifiers).

*Corresponding Author. The order of authors is merely alphabetical. Paulo Fernandes is partially funded by CNPq-Brazil (PQ 307284/2010-7). Lucelene Lopes is funded by FAPERGS/CAPES agreement (DOCFIX 07/2012).

In such context, the our paper objective is to analyze the Random Forests classifier in respect to its vulnerability to random choices. In such case, the effectiveness of the method, or its ineffectiveness, may be a consequence of a lucky, or unlucky, choice of seed.

Established the impact of randomness, we propose an extension to cope with this vulnerability without having a negative impact on its accuracy. In order to do so, we propose an extension called Stochastic Aware Random Forests (SARF). This extension effectiveness is empirically illustrated through the application of the classical Random Forests and the novel SARF to 32 public data sets.

This paper is organized as follows: the second section describes the data sets used in this paper, and it shows the impact of random choices on classical Random Forests compared to the impact on Bagging method; The proposed extension, called SARF, is detailed presented in the third section, and different options of SARF are experimented over the 32 data sets; Finally, the conclusion summarizes this paper contribution and proposes future works.

The impact of randomness in Random Forests

The motivation to propose our extension to Random Forests is its vulnerability to decisions affected by pseudo-random number generation within the implementation. Therefore, this section goal is to illustrate such motivation showing that Random Forests implementation is more vulnerable to random choices, than other methods. To do so, we apply the Bagging and Random Forests implementations found in WEKA version 3.4.19 (Hall et al. 2009) to 32 different data sets, using 100 different random seeds.

The experiments of this paper were conducted over 32 data sets that came from public repositories (Table 1) from areas like medical information, software development, wine recognition, biology, chemistry and physical phenomena. The first column of this table indicates the data base name and its original repository: $\triangle$ for University of California Irvine (Asuncion and Newman 2007); and ∇ for University of West Virginia (Boetticher, Menzies, and Ostrand 2007). The second column states information about the data: the number of attributes excluding the class attribute (nb_a), the number of instances (nb_i), and the percentile of missing values (*miss.*) over the total data ($nb_a \times nb_i$).

Table 1: Data bases characteristics.

Data Bases	*id*	Information Data			Class Data	
		nb_a	nb_i	*miss.*	nb_c	u
Abalone$^{\triangle}$	B01	8	4,177	0.00%	28	0.071
Arrythmia$^{\triangle}$	B02	279	452	0.32%	13	0.268
Audiology$^{\triangle}$	B03	69	226	2.03%	24	0.103
Balance$^{\triangle}$	B04	4	625	0.00%	3	0.146
Breast cancer$^{\triangle}$	B05	9	286	0.39%	2	0.165
Car Evaluation$^{\triangle}$	B06	6	1,728	0.00%	4	0.390
CM1 software defect$^{\triangledown}$	B07	21	498	0.00%	2	0.645
Datatrieve$^{\triangledown}$	B08	8	130	0.00%	2	0.690
Desharnais$^{\triangledown}$	B09	11	81	0.00%	3	0.150
Ecoli$^{\triangle}$	B10	8	336	0.00%	8	0.168
Echo cardiogram$^{\triangle}$	B11	11	132	5.10%	3	0.054
Glass$^{\triangle}$	B12	10	214	0.00%	6	0.116
Heart(Cleveland)$^{\triangle}$	B13	13	303	0.38%	2	0.008
Heart statlog$^{\triangle}$	B14	13	270	0.00%	2	0.012
Hepatitis$^{\triangle}$	B15	19	155	5.70%	2	0.345
JM1 software defect$^{\triangledown}$	B16	21	10,885	0.00%	2	0.376
Kr-vs-kp$^{\triangle}$	B17	36	3,196	0.00%	2	0.002
MW1 software defect$^{\triangledown}$	B18	37	403	0.00%	2	0.716
Pima-diabetes$^{\triangle}$	B19	8	768	0.00%	2	0.091
Post-operative$^{\triangle}$	B20	8	90	0.42%	3	0.366
Primary-tumor$^{\triangle}$	B21	17	339	3.92%	21	0.066
Reuse$^{\triangledown}$	B22	27	24	0.93%	2	0.063
Solar Flare$^{\triangle}$	B23	12	1,389	0.00%	8	0.682
Tic-Tac-Toe Endgame$^{\triangle}$	B24	9	958	0.00%	2	0.094
Thyroid(Allhyper)$^{\triangle}$	B25	29	2,800	5.61%	4	0.928
Thyroid(Hypothyroid)$^{\triangle}$	B26	29	3,772	5.54%	4	0.807
Thyroid(Sick euthyroid)$^{\triangle}$	B27	25	3,163	6.74%	2	0.664
Wbdc$^{\triangle}$	B28	30	569	0.00%	2	0.065
Wisconsin breast cancer$^{\triangle}$	B29	9	699	0.25%	2	0.096
Wine recognition$^{\triangle}$	B30	13	178	0.00%	3	0.013
Yeast$^{\triangle}$	B31	9	1,484	0.00%	10	0.137
Zoo$^{\triangle}$	B32	17	101	0.00%	7	0.114

The last column contains the information about class distribution of each data base: the number of different classes (nb_c) and a rate indicating how unbalanced these classes are (u). This rate was proposed by Fernandes *et. al.* (2010) and it is computed as the ratio between the standard deviation of the number of instances in each class (std) by a completely balanced distribution of instances among the classes (nb_i/nb_c), divided by the square root of the number of classes. Therefore, the index u varies from 0 (completely balanced) asymptotically towards 1 (as unbalanced as possible). Its numerical value is computed by:

$$u = \left(\frac{std}{nb_i/nb_c} \right) / \sqrt{nb_c}$$

Every ensemble classifier method[1] needs random choices to produce different data sets in order to generate distinct classifiers, that are as uncorrelated as possible. It is a known fact that these random choices have an non-negligible impact on the precision of Bagging (Fernandes, Lopes, and Ruiz 2010).

[1]According to WEKA menus to choose classifier methods, Random Forests is not an ensemble method, but a tree method. However, following Breiman (2001) definition: "*Random forests are a combination of tree predictors such that each tree depends on the values of a random vector sampled independently and with the same distribution for all trees in the forest*" and, therefore we assume it to be an ensemble classifier.

Ignoring other details of Bagging method, we notice that the random choices to take are restricted to the sampling procedure over the instances of the original training set. Observing the Random Forests method (Breiman 2001), we notice additional random choice points. Not only an instance sampling with repetition is made (just like Bagging), but random choices are also responsible to chose k attributes among the total nb_a attributes of the data set. In WEKA implementation, this second kind of random choice is encapsulated inside the tree builder procedure called *RandomTree*.

As stated by Breiman (Breiman 2001), it is true that Random Forests is much lighter than Bagging, since it has simpler tree classifiers to build[2]. However, it has more random choices to make, so the impact of the random choices in Random Forests is likely to be greater than in Bagging.

To confirm that, we have ran in WEKA 3.4.19 a set of experiments for Bagging and Random Forests (Hall et al. 2009). For Bagging we use J48 as tree builder and we generate 30 classifiers. For Random Forests we use RandomTree (Dietterich 2000) to build 30 trees, and the number of features was the recommended $k = (\log_2 nb_a) + 1$. We have ran these experiments for the 32 data sets presented in Table 1, and we have observed the accuracy achieved with a ten-fold cross-validation.

However, for both methods, instead of using a default random seed, we did 100 different experiments to each data set and method using a different random seed each time (we use seeds 0, 1, 2, ..., 99). Then, we discarded the 5 lowest accuracy values, as well as the 5 highest ones. For the remaining 90 results we have computed average (*avg.*) and standard deviation (*st.dv.*) among these 90 values.

The first two double columns of Table 2 present these results. There it can be noticed that Bagging and Random Forests are somewhat balanced concerning the more accurate method, since on 21 data sets Bagging was more accurate, while Random Forests was better on 11 data sets.

On the contrary, concerning the impact of randomness, Random Forests is undoubtedly more vulnerable than Bagging, since it delivers larger standard deviation values for 28 out of 32 data sets. To facilitate this observation the highest accuracy average values, and the lowest standard deviations were boldfaced in the first two double columns of Table 2.

SARF - Stochastic Aware Random Forests

It is a fact that Random Forests is more vulnerable to random choices than, for instance, Bagging, but, it is not obvious why this happens. According to Breiman (2001), the use of a small number of attributes and randomly selected them guarantees correlation free ensemble of trees. However, if we dare to be so bold, we believe this is only true if you have theoretical random choices, which is never the case for a computer implementation.

[2]It is worthy mention that some important characteristics of Bagging and Random Forests methods, *e.g.*, "out-of-the-bag" classifier drop out, or classifier voting, are not detailed here, since they are deterministic steps. The reader is invited to consult the references for detailed information.

Table 2: Accuracy results applied with 100 different seeds for Bagging, Random Forests and SARF variations.

Data	Bagging		Random Forests		SARF 2		SARF 6		SARF 10		SARF 15		SARF 30	
Sets	*avg.*	*st.dv.*	*avg.*	*st.dv.*	*avg.*	*st.dv.*	*avg.*	*st.dv.*	*avg.*	*st.dv.*	*avg.*	*st.dv.*	*avg.*	*st.dv.*
B01	**23.79%**	**0.309**	23.69%	0.335	23.40%	0.33	23.51%	0.304	23.53%	0.354	23.63%	0.271	23.58%	**0.219**
B02	**73.97%**	**0.500**	67.91%	0.656	68.08%	0.749	68.22%	0.667	68.17%	**0.619**	67.94%	0.655	67.95%	0.69
B03	**80.86%**	**0.628**	79.46%	0.81	79.49%	0.876	79.57%	0.993	79.45%	1.096	79.61%	1.141	79.43%	**0.591**
B04	**19.08%**	0.677	5.03%	**0.278**	5.12%	0.299	5.08%	0.274	5.03%	0.322	4.93%	**0.206**	4.77%	0.301
B05	**74.58%**	**0.553**	67.93%	0.845	68.32%	1.043	68.53%	0.948	68.68%	0.844	68.89%	0.725	68.93%	**0.628**
B06	93.66%	**0.176**	**94.36%**	0.266	94.39%	0.254	94.30%	0.264	94.30%	0.222	94.31%	0.219	94.41%	**0.179**
B07	**88.67%**	0.3	88.59%	**0.292**	88.69%	0.32	88.51%	**0.211**	88.45%	0.275	88.44%	0.301	88.12%	0.297
B08	**90.27%**	**0.565**	89.42%	0.631	89.49%	0.669	89.17%	0.681	89.16%	0.645	89.47%	0.558	89.74%	**0.525**
B09	70.92%	**1.563**	**73.98%**	1.777	73.59%	**1.558**	74.17%	2.035	74.09%	2.11	73.87%	1.833	73.61%	1.6
B10	**85.15%**	**0.466**	66.00%	1.265	65.25%	**1.107**	65.69%	1.229	65.98%	1.109	65.80%	1.245	66.36%	1.215
B11	**60.39%**	**1.439**	58.75%	1.96	58.41%	1.907	58.70%	1.725	59.02%	1.723	59.60%	1.923	59.90%	**1.697**
B12	97.10%	**0.308**	**98.54%**	0.354	98.39%	0.424	98.48%	0.396	98.60%	0.334	98.57%	0.333	98.53%	**0.311**
B13	79.83%	**0.745**	**81.44%**	0.777	81.38%	0.884	81.68%	0.873	81.69%	0.919	81.60%	**0.730**	81.02%	1.133
B14	81.20%	**0.761**	**81.31%**	0.861	81.25%	0.788	80.98%	**0.694**	80.74%	0.697	81.31%	0.814	81.74%	1.169
B15	82.72%	**0.731**	**84.47%**	1.005	84.16%	1.08	84.12%	0.831	84.09%	0.99	84.45%	0.887	84.46%	**0.803**
B16	**81.90%**	**0.096**	81.35%	0.101	81.33%	0.127	81.27%	0.108	81.28%	0.126	81.34%	0.108	81.33%	**0.088**
B17	**99.46%**	**0.034**	99.20%	0.074	99.19%	0.072	99.19%	0.069	99.18%	0.064	99.18%	0.062	99.20%	**0.052**
B18	**91.93%**	**0.402**	90.39%	0.418	90.27%	0.392	90.27%	0.45	90.39%	0.405	90.41%	0.355	90.38%	**0.288**
B19	**75.50%**	**0.537**	75.09%	0.548	75.15%	0.578	75.08%	0.508	75.15%	0.59	75.27%	0.528	75.15%	**0.406**
B20	**69.64%**	**0.934**	62.28%	1.512	62.41%	1.456	62.38%	1.321	62.62%	1.667	62.88%	1.17	63.16%	**1.170**
B21	**43.90%**	**0.691**	42.72%	0.709	42.63%	0.645	42.62%	0.635	42.76%	0.596	42.83%	0.59	42.62%	**0.494**
B22	**95.83%**	**0.000**	89.95%	3.773	90.69%	**2.850**	90.65%	2.867	89.86%	3.23	90.37%	2.753	91.62%	2.88
B23	**84.31%**	**0.000**	81.16%	0.202	81.13%	0.212	81.20%	0.192	81.15%	0.166	81.11%	0.199	81.17%	**0.132**
B24	94.12%	**0.318**	**95.34%**	0.469	95.33%	0.442	95.35%	0.404	95.52%	**0.348**	95.42%	0.361	95.66%	0.585
B25	**99.62%**	**0.026**	99.37%	0.066	99.35%	0.061	99.37%	**0.055**	99.38%	0.062	99.40%	0.072	99.37%	0.064
B26	**98.65%**	**0.048**	98.47%	0.07	98.49%	0.077	98.46%	0.065	98.47%	0.059	98.47%	0.056	98.44%	**0.050**
B27	**97.91%**	**0.039**	97.86%	0.068	97.84%	0.074	97.84%	**0.067**	97.83%	0.074	97.84%	0.082	97.82%	0.07
B28	95.67%	0.351	**96.41%**	**0.317**	96.52%	0.31	96.46%	0.291	96.46%	0.269	96.42%	**0.224**	96.32%	0.272
B29	95.86%	**0.233**	**96.21%**	0.252	96.11%	0.226	96.15%	**0.217**	96.12%	0.225	96.11%	0.255	96.12%	0.308
B30	95.86%	0.493	**97.82%**	**0.406**	97.62%	0.459	97.77%	0.422	97.73%	0.513	97.76%	0.513	97.94%	**0.324**
B31	**52.11%**	**0.224**	48.20%	1.467	48.34%	**1.194**	48.15%	1.326	48.28%	1.345	48.98%	1.445	48.49%	1.323
B32	92.64%	**0.609**	**93.71%**	1.004	93.64%	1.022	93.69%	1.325	93.67%	1.276	93.31%	1.298	93.71%	**0.914**
avg.	**80.22%**	**0.461**	78.32%	0.737	78.30%	0.703	78.33%	0.701	78.34%	0.727	78.42%	0.685	78.47%	**0.649**

All computer languages provide a pseudo-random number generator (Park and Miller 1988), which is the basic tool to implement random decisions in algorithms. The problem with pseudo-random decisions is that, unlike theoretical random decisions, there is always some kind of correlation between successive decisions (Boyar 1989). Specifically in java, the language in which WEKA is implemented, the pseudo-random generation is made through a linear congruence algorithm, which is one of the best approximations to the properties of random generation.

Among these properties, the one that is particularly important to Random Forests is to provide uncorrelated samples, *i.e.*, "*a random vector sampled independently*" (see Footnote 1). One of the options to enhance this independency is to alter the pseudo-random generation procedure. This is the idea of the method extension proposed, which is, therefore, called Stochastic Aware Random Forests (SARF).

The classical Random Forests method, while generating each tree, has two points of random choices: in the first one, k attributes are sampled, and in the second one, nb_i instances are sampled with repetition. Therefore, to each tree generation there is a need to perform $k + nb_i$ pseudo-random number generations, and consequently, the whole execution with T trees demands $T \times (k + nb_i)$ samplings.

The WEKA implementation of Random Forests takes one random seed at the beginning of the algorithm, and all $T \times (k+nb_i)$ samplings are made as the result of a deterministic (pseudo-random) number sequence. Our proposal is to break such sequence by introducing additional points of random seeds input. In such way, the SARF implementation is essentially the same as the classical Random Forests, except for the inclusion of random seeds input points.

Such inclusion of random seeds input points could be made before each tree generation, which would produce T pseudo-random sequences. However, it could be done more sparsely, for instance, at each two tree building, producing $T/2$ pseudo-random sequences. It could go further until having just one additional random seed input point, producing 2 pseudo-random sequences.

These options result in different versions of SARF, according to the number of pseudo-random seeds input points. We will call each of these versions "SARF s", when SARF is implemented including $s - 1$ seed assignment points, *i.e.*, generating s pseudo-random sequences. For instance, "SARF 3" with 30 tree generations ($T = 30$), would generate 10 trees with the first random seed, another 10 trees with the second random seed, and the last 10 trees with the third random seed. In this point of view, classical Random Forests

could be seem as a particular case of SARF, the "SARF 1", *i.e.*, with just one random seed input made at the beginning of the algorithm.

The empirical observation of SARF effectiveness is made through the application of five different versions of SARF with 30 trees ($T = 30$) for the 32 data sets presented previously. As the experiments conducted previously for Bagging and Random Forests, we verify the accuracy achieved using ten-fold cross-validation and 100 different sets of random seeds, discarding the 5 worst and the 5 best accuracy results.

Note that, unlike the experiments for Bagging and Random Forests, just one random seed is no longer enough. For example, for "SARF 3" and $T = 30$, three random seeds are needed, one to each pack of ten tree buildings. Hence, in this example, the random seeds for the first run are (0, 100, 200), for the second run seeds are (1, 101, 201), and so on until the hundredth run that have seeds (99, 199, 299).

Table 2 presents the average accuracy and standard deviation of the 90 middle results for classical Bagging and Random Forests, as well as the following SARF versions: SARF 2, SARF 6, SARF 10, SARF 15 and SARF 30.

The first interesting observation from Table 2 is that the use of SARF extension reduces the impact of randomness (it delivers a smaller standard deviation) in about 63% of the runs (80 out of 128) in comparison with the classical Random Forests. Examining a little closer the results, we notice that at least one version of SARF is always better than classical Random Forests.

We can also notice that the best standard deviation occurs much more often for SARF 30 version (18 out of 32 data sets). These results indicate that SARF, specially in its version with the largest number of random seeds (SARF 30), has effectively reduced the randomness impact.

Observing the accuracy changes due to SARF 30, we notice a negligible impact (less than 1%) for all 32 data sets. Additionally, comparing it with classical Random Forests, we notice that 14 data sets had an accuracy increase, 15 had accuracy decrease, and 3 data sets did not change the accuracy by using SARF 30. Therefore, we assume that SARF extension does not hinders the accuracy for Random Forests.

Conclusion

This paper proposed an extension to the Random Forests method, called Stochastic Aware Random Forests (SARF), to reduce the variability of accuracy due to random choices. The proposed extension is valid since classical Random Forests is more vulnerable than other ensemble classifier methods, *e.g.*, Bagging.

The average standard deviation for the accuracy achieved with Bagging was 0.461, while classical Random Forests had 0.737. SARF dropped this average standard deviation to 0.649 for the version with a different random seed to each classifier (SARF 30). Hence, it is our opinion that SARF was effective in its goal to reduce the vulnerability to randomness, without reducing the accuracy. The average accuracy for all data sets actually increase a little bit (from 78.32% to 78.47%) with SARF 30 instead of classical Random Forests.

From a practical point of view, we foresee new experiments with SARF changing the random generators themselves, instead of assigning random seeds before each tree building. In our experiments so far, we have inserted random seeds inputs, but instead of this, we could imagine a deeper, and hopefully more effective, change of random generators. For instance, we could change the parameters of linear congruence generators (Viega 2003) before each tree building. It is our belief that it could be even more effective to assure the independence of each random sequence, and, therefore, serve better the Random Forests assumption of uncorrelated samplings among each tree building.

References

Asuncion, A., and Newman, D. J. 2007. Uci machine learning repository. Available at http://www.ics.uci.edu/∼mlearn/MLRepository.html.

Banfield, R.; Hall, L. O.; Bowyer, K. W.; and Kegelmeyer, W. P. 2007. A comparison of decision tree ensemble creation techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29(1):173–180.

Boetticher, G.; Menzies, T.; and Ostrand, T. 2007. Promise repository of empirical software engineering data. available at http://promisedata.org/.

Boyar, J. 1989. Inferring sequences produced by pseudo-random number generators. *Journal ACM* 36(1):129–141.

Breiman, L. 1996. Bagging predictors. *Machine Learning* 24(2):123–140.

Breiman, L. 2001. Random forests. *Machine Learning* 45(1):5–32.

DeBarr, D., and Wechsler, H. 2012. Spam detection using random boost. *Pattern Recognition Letters* 33(10):1237–1244.

Dietterich, T. G. 2000. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning* 40(2):139–157.

Fernandes, P.; Lopes, L.; and Ruiz, D. D. A. 2010. The impact of random samples in ensemble classifiers. In *Proceedings of the 2010 ACM SAC*, SAC '10, 1002–1009. New York, NY, USA: ACM.

Guo, H., and Fan, M. 2011. Ensemble pruning via base-classifier replacement. In *Proceedings of the 12th WAIM*, WAIM'11, 505–516. Springer.

Hall, M.; Frank, E.; Holmes, G.; Pfahringer, B.; Reutemann, P.; and Witten, I. H. 2009. The weka data mining software: an update. *SIGKDD Explor. Newsl.* 11(1):10–18.

Kulkarni, V. 2012. Pruning of random forest classifiers: A survey and future directions. In *Proceedings of 2012 International Conference on Data Science & Engineering (ICDSE)*, 64–68. IEEE Press.

Park, S. K., and Miller, K. W. 1988. Random number generators: good ones are hard to find. *Commun. ACM* 31(10):1192–1201.

Viega, J. 2003. Practical random number generation in software. In *Proceedings of the 2003 Annual Computer Security Applications Conference*, ACSAC 19, 505–516. Silver Spring, MD, USA: ACSAC.