Binary Classification in Unstructured Space With Hypergraph Case-Based Reasoning (Additional Material)

Alexandre Quemy

IBM Krakow Software Lab, Cracow, Poland

Faculty of Computing, Poznań University of Technology, Poznań, Poland

Introduction

This document contains additional material for the paper 'Binary Classication in Unstructured Space With Hypergraph Case-Based Reasoning', accepted for publication in the Information Systems special issue from DOLAP 2018.
This material includes:

- Section 1: A formal proof of the training algorithm convergence,
- Section 2: The average confusion matrix obtained in the experiments on structured datasets,
- Sections 3 and 4: The detailed comparison of HCBR with other methods (w.r.t. accuracy and MCC),
- Section 5: The evolution of MCC depending on the training set size,
- Section 6: A discussion on the learning curves,
- Section 7: The hyperparameters for the genetic algorithms used to assess the model space limitation,
- Section 8: A discussion on the model locality.

Email address: `aquemy@pl.ibm.com` (Alexandre Quemy)

1. On the convergence of the training algorithm

For a training iteration k, let assume a case $\mathbf{x}_i$ with a strength $S_k(\mathbf{x}_i)$ that is wrongly classified. The update rule on μ implies that $S_{k+1}(\mathbf{x}_i) \in [-|S_k(\mathbf{x}_i)|, S_k(\mathbf{x}_i)]$. Any case $\mathbf{x}_j$ such that $\mathbf{x}_i \cap \mathbf{x}_j \neq \emptyset$ is modified, and in the same direction (toward the same class). Let us assume that $S_k(\mathbf{x}_i) < 0$ such that after the update rule $S_{k+1}(\mathbf{x}_i) > S_k(\mathbf{x}_i)$. Then, $S_{k+1}(\mathbf{x}_j) > S_k(\mathbf{x}_j)$.

The only problematic case is when $y_j = 0$, $S_k(\mathbf{x}_j) < 0$ but $S_{k+1}(\mathbf{x}_j) > 0$ (that is to say that $\mathbf{x}_j$ become wrongly classified due to the modification of μ involved by $\mathbf{x}_i$).

Let us consider any μ_l for a e_l that is included in $\mathbf{x}_i$ and $\mathbf{x}_j$. When μ_l is modified, then $|S(x_j)|$ is smaller because by definition $|w(e_l, x_j)\mu_l| \leq |S(x_j)|$.

As a result, both the case that *triggers* the modification of μ and the cases that are consequently modified have a strength that is closer to 0 than before the modification.

Therefore, there are only two possible cases:

- All cases become correctly classified and the process stops.
- Some cases cannot be properly classified within the model space and switch iterations after iterations between classes. Their strength converges toward 0. It does not imply that the process converges toward the best possible accuracy.

2. Average confusion matrix obtained with a 10-fold cross-validation.

Table 1: Average confusion matrix obtained with a 10-fold cross-validation.

		TP	FN	FP	TN
`adult`	without tuning	2182.4	295.3	288.5	488.8
	with tuning	2226.6	245.3	311.4	472.7
`breast`	without tuning	23.0	1.4	0.7	43.9
	with tuning	23.5	1.1	0.4	44.0
`heart`	without tuning	12.4	1.8	1.9	9.9
	with tuning	13.5	1.0	1.4	10.1
`mushrooms`	without tuning	390.6	0.0	0.0	420.4
`phishing`	without tuning	595.4	23.8	19.8	465.0
	with tuning	599.1	19.2	15.9	468.9
`skin`	without tuning	4886.3	132.4	199.4	19286.9
	with tuning	4888.3	130.4	194.1	19292.2
`splice`	without tuning	155.7	9.1	8.5	142.7
	with tuning	156.2	8.6	6.9	144.3

3. Comparison of HCBR with several methods (Scikit-Learn implementation) w.r.t. MCC.

Table 2: Comparison of HCBR with several methods (Scikit-Learn implementation) w.r.t. MCC.

Dataset	Method	MCC	#
adult	**HCBR**	0.5146	3
	AdaBoost	0.5455	1
	k-NN	0.4785	7
	Linear SVM	0.4918	5
	RBF SVM	0.5065	4
	Decision Tree	0.4821	6
	Rand. Forest	0.3776	8
	Neural Net	0.5349	2
	Naive Bayes	0.2493	9
	QDA	0.4785	7
breast	**HCBR**	0.9222	3
	AdaBoost	0.9023	6
	k-NN	0.9163	4
	Linear SVM	0.9126	5
	RBF SVM	0.8829	8
	Decision Tree	0.8760	9
	Rand. Forest	0.9296	1
	Neural Net	0.9280	2
	Naive Bayes	0.8991	7
	QDA	0.8616	10
heart	**HCBR**	0.7082	1
	AdaBoost	0.5972	6
	k-NN	0.5879	7
	Linear SVM	0.6849	4
	RBF SVM	0.6287	5
	Decision Tree	0.5763	8
	Rand. Forest	0.5703	9
	Neural Net	0.6995	2
	Naive Bayes	0.6932	3
	QDA	0.4500	10
mushrooms	**HCBR**	0.9995	2
	AdaBoost	1.0000	1
	k-NN	0.9993	3
	Linear SVM	1.0000	1
	RBF SVM	0.9990	5
	Decision Tree	0.9991	4
	Rand. Forest	0.8840	7
	Neural Net	1.0000	1
	Naive Bayes	0.9767	6
	QDA	1.0000	1
phishing	**HCBR**	0.9191	1
	AdaBoost	0.8637	6
	k-NN	0.9138	4
	Linear SVM	0.8740	5
	RBF SVM	0.9286	2
	Decision Tree	0.8585	7
	Rand. Forest	0.7582	8
	Neural Net	0.9448	1
	Naive Bayes	0.5292	10
	QDA	0.5872	9
skin	**HCBR**	0.9551	4
	AdaBoost	0.8552	8
	k-NN	0.9982	1
	Linear SVM	0.8090	9
	RBF SVM	0.9950	3
	Decision Tree	0.9544	5
	Rand. Forest	0.9539	6
	Neural Net	0.9967	2
	Naive Bayes	0.7600	10
	QDA	0.9483	7
splice	**HCBR**	0.8857	2
	AdaBoost	0.8801	3
	k-NN	0.6072	9
	Linear SVM	0.7282	8
	RBF SVM	0.8461	4
	Decision Tree	0.8998	1
	Rand. Forest	0.5925	10
	Neural Net	0.8390	5
	Naive Bayes	0.7595	7
	QDA	0.8251	6

4. Comparison of HCBR with several methods (Scikit-Learn implementation) w.r.t. accuracy.

Table 3: Comparison of HCBR with w.r.t. accuracy.

Dataset	Method	Acc.	#
`adult`	**HCBR**	0.8232	5
	AdaBoost	0.8444	1
	k-NN	0.8156	7
	Linear SVM	0.8274	4
	RBF SVM	0.8327	3
	Decision Tree	0.7918	8
	Rand. Forest	0.8223	6
	Neural Net	0.8378	2
	Naive Bayes	0.4675	10
	QDA	0.7528	9
`breast`	**HCBR**	0.9833	1
	AdaBoost	0.9563	4
	k-NN	0.9614	3
	Linear SVM	0.9614	3
	RBF SVM	0.9457	7
	Decision Tree	0.9429	9
	Rand. Forest	0.9543	5
	Neural Net	0.9671	2
	Naive Bayes	0.9533	6
	QDA	0.9430	8
`heart`	**HCBR**	0.8538	1
	AdaBoost	0.8037	6
	k-NN	0.7926	7
	Linear SVM	0.8444	3
	RBF SVM	0.8148	5
	Decision Tree	0.7556	9
	Rand. Forest	0.7741	8
	Neural Net	0.8519	2
	Naive Bayes	0.8444	3
	QDA	0.8185	4
`mushrooms`	**HCBR**	0.9998	2
	AdaBoost	1.0000	1
	k-NN	0.9996	3
	Linear SVM	1.0000	1
	RBF SVM	0.9995	4
	Decision Tree	0.9996	3
	Rand. Forest	0.9582	6
	Neural Net	1.0000	1
	Naive Bayes	0.9882	5
	QDA	1.0000	1
`phishing`	**HCBR**	0.9645	3
	AdaBoost	0.9477	8
	k-NN	0.9505	7
	Linear SVM	0.9532	6
	RBF SVM	0.9550	5
	Decision Tree	0.9625	4
	Rand. Forest	0.9738	1
	Neural Net	0.9726	2
	Naive Bayes	0.7062	10
	QDA	0.7656	9
`skin`	**HCBR**	0.9847	4
	AdaBoost	0.9399	7
	k-NN	0.9994	1
	Linear SVM	0.9297	8
	RBF SVM	0.9984	3
	Decision Tree	0.9456	5
	Rand. Forest	0.9415	6
	Neural Net	0.9989	2
	Naive Bayes	0.8802	9
	QDA	0.8978	10
`splice`	**HCBR**	0.9430	3
	AdaBoost	0.9528	2
	k-NN	0.7843	10
	Linear SVM	0.8645	9
	RBF SVM	0.9230	7
	Decision Tree	0.9415	4
	Rand. Forest	0.9399	5
	Neural Net	0.9195	8
	Naive Bayes	0.9245	6
	QDA	0.9838	1

5. Matthew Correlation Coefficient depending on the training set size.

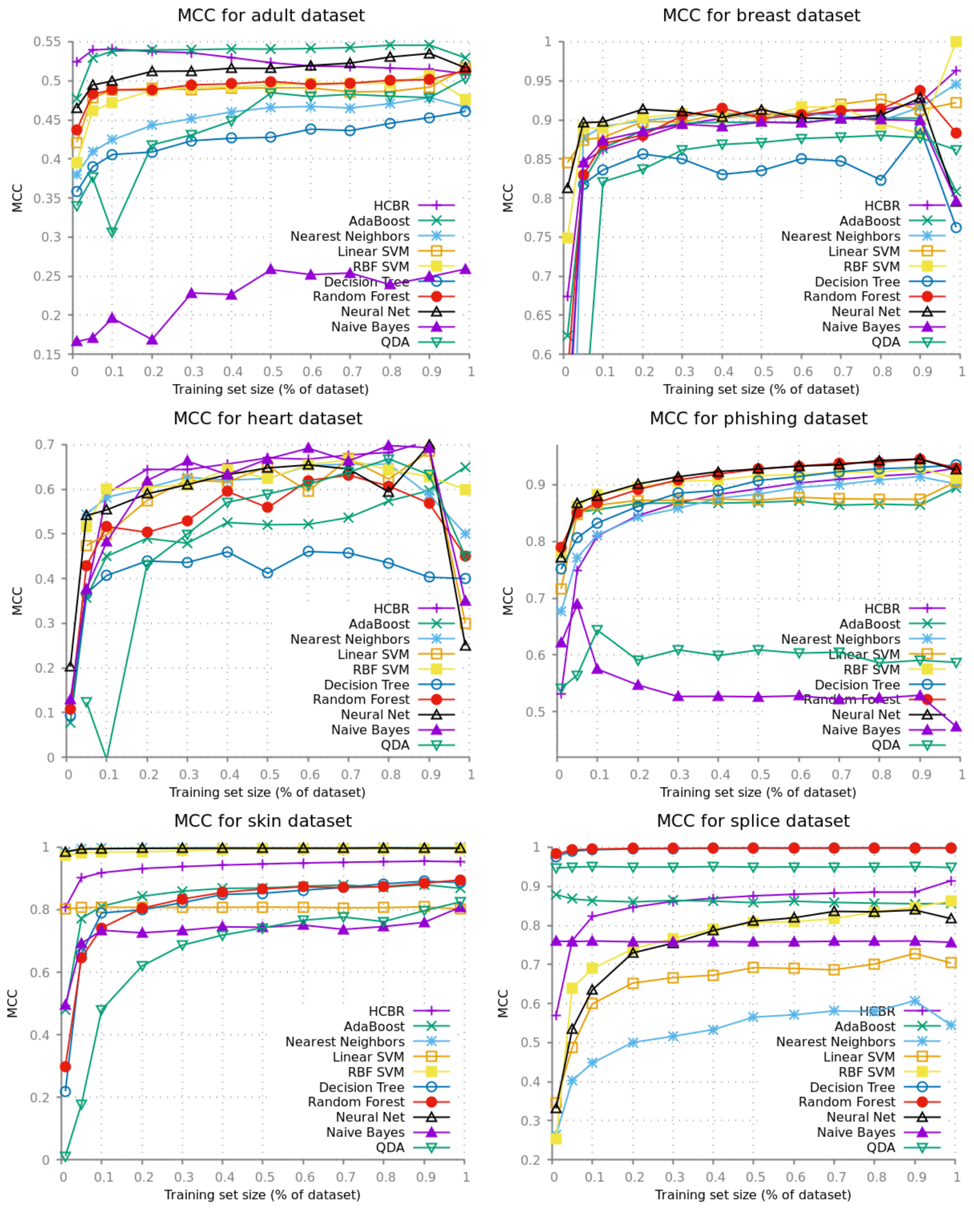

Figure 1: Matthew Correlation Coefficient depending on the training set size.

6. Discussion on the learning curves

6.1. Heuristic

In almost all datasets, in particular phishing and skin, many input vectors are the same, but with different output (e.g. for skin it's 583 cases i.e. 5.27% of the dataset). By definition, there is no way to distinguish between those vectors, so the best we can do is to assign the class with the highest prevalence.

In most cases the model choses itself the class with the highest prevalence among those redundant input vectors or if it does not, it has very little impact (because the prevalence is close to 0.5 and the size of the redundant vector set is small in comparison to the dataset size).

The heuristic consists in estimating the prevalence of the set of redundant vectors in the training set, and then in bypassing the model prediction with the class associated to this prevalence. For instance, the overall prevalence of phishing is 0.5562 but the prevalence of the redundant vectors is 0.9674. About 5.27% of the test set should consists of cases already in the casebase, among which 0.9674 are of class 1. It represents the insurance of a 5.27 x 0.9674 = 5.20% of the training set correctly labeled.

The reason why the heuristic works in this case is because the prevalence of the redundant vectors set is higher than the accuracy obtained by the model (about 92.5% versus 96%). The gain corresponds exactly to this 5.20% (because without the heuristic, with the grain used by the experiment, the model uses the wrong label).

This heuristic is independant of HCBR as it could be applied to any margin-based discrimative methods (e.g. SVM also cannot discriminate between redundant points with different labels).

6.2. Learning curve

The learning curves for phishing and skin are surprising for two reasons. First, the test accuracy is significantly higher than the training accuracy as depicted by Figure A6.1 and A6.2. The prediction phase on those two datasets uses the heuristic described above. For both datasets, the difference in accuracy between the two curves is much higher than the possible gain due to the heuristic. The learning curves for the same experiment without the heuristic are displayed on Figures A6.3 and A6.4 and it seems to explain perfectly the phenomena. Notice that this is specific to those two datasets that contains several redundant points with the same output. For instance, the heuristic is activated also on splice but as the number of redundant elements is very low w.r.t. the dataset size, the impact on the accuracy is not significant.

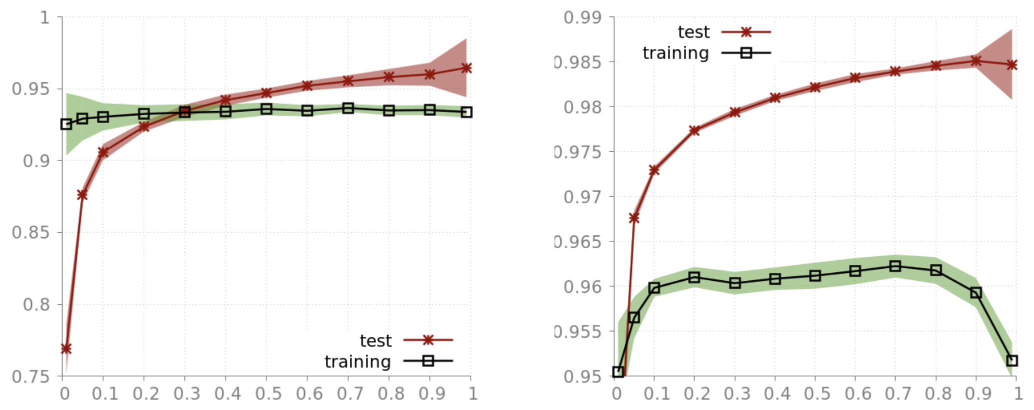

Figure 2: Learning curves on `phishing` and `skin` datasets. The accuracy on the test set is higher than on the training set. On the training set, the accuracy starts by increasing and remains globally stable (with a drop for `skin`).

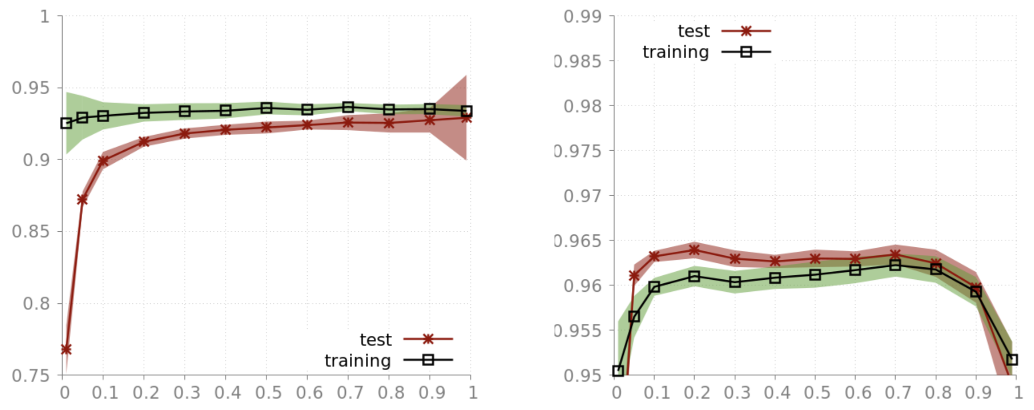

Figure 3: Learning curves on `phishing` and `skin` datasets without the heuristic. Compared to Figure 2, the learning curve behave as expected in theory.

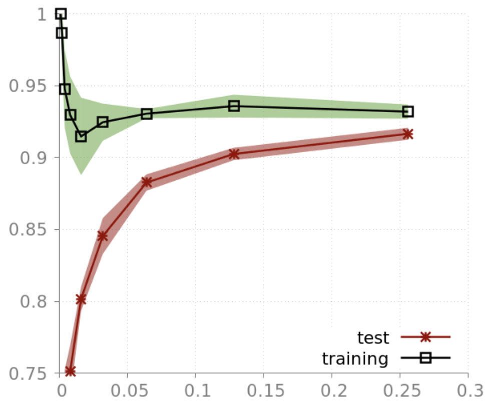

Figure 4: Learning curves on `phishing` for very small training set sizes.

7. Hyperparameters for the genetic algorithm.

Table 4: Hyperparameters for the genetic algorithm. *Mutation* σ *factor* is a factor used to set the standard deviation of the gaussian mutation. We use $\sigma = \frac{\mu^-}{\alpha}$ where α is the factor and μ^- defined by $\min\limits_{i,j} \mu_i - \mu_j$.

Dataset	Generations	Mutation σ factor
`adult`	200	10000
`breasts`	200	10
`heart`	200	10
`phishing`	100	1000
`skin`	100	1000
`splice`	200	100

8. Model Locality

In case $\mathbf{x}$ has too many discretionary features, the classification rule is likely to be irrelevant. Indeed, the intersection between $\mathbf{x}$ and $\mathbb{F}_{\mathbf{X}}$ is to small to hold enough information and make strong analogies with $\mathbf{x}$. To overcome this drawback, $2^{\mathbb{F}}$ is split into two subsets:

- $\mathcal{F}_1 = \{\mathbf{x} \in 2^{\mathbb{F}} \mid |\mathbf{x} \cap \mathbb{F}_{\mathbf{X}}| \geq \delta\},\ \forall \delta \in \mathbb{N}$
- $\mathcal{F}_2 = 2^{\mathbb{F}} \setminus \mathcal{F}_1$

$\mathcal{F}_1$ corresponds to the elements s.t. they share some features with the examples. An alternative may be considered by using $\mathcal{F}_1 = \{\mathbf{x} \in 2^{\mathbb{F}} \mid \frac{D_{\mathbf{x}}}{|\mathbf{x}|} \leq \delta\},\ \forall \delta \in [0, 1]$. In this case, $\mathcal{F}_1$ contains the elements for which we have enough information provided by the examples. From our preliminary tests, the choice depends on the dataset structure.

Finally, the decision rule for new cases is built as follows:

$$\bar{J}(\mathbf{x}) = \begin{cases} \tilde{J}(\mathbf{x}) & \text{if } \mathbf{x} \in \mathcal{F}_1 \\ o_{\mathbf{x}} & \text{if } \mathbf{x} \in \mathcal{F}_2 \end{cases} \tag{R2}$$

where $o_{\mathbf{x}}$ is one draw from a random variable that has Bernoulli law with parameter $p = \frac{|\{\mathbf{x} \in \mathbf{X} \mid J(\mathbf{x}) = 1\}|}{|\mathbf{X}|}$, i.e. the prevalence of class 1 in $\mathbf{X}$. It assumes that the prevalence of $\mathbf{X}$ is close to the prevalence over $2^{\mathbb{F}}$ (or that the prevalence does not change in time for sequential problems in which the new cases are generated by an unknown random measure). The rationale behind is that if for a case $\mathbf{x}$, it is not possible to exploit the model built on the hypergraph, then we can still model J as a Bernoulli random variable and use a maximum likelihood estimation for p. In a sense, it is extending the *local* model to the entire input space $2^{\mathbb{F}}$.