Making Sense of Economics Datasets with Evolutionary Coresets

Pietro Barbiero, Alberto Tonda

Making Sense of Economics Datasets With Evolutionary Coresets

Pietro Barbiero[1] and Alberto Tonda[2]

[1] Politecnico di Torino, Torino, Italy
`pietro.barbiero@studenti.polito.it`
https://orcid.org/0000-0003-3155-2564
[2] Université Paris-Saclay, INRA, UMR 782 GMPA, 78850, Thiverval-Grignon France
`alberto.tonda@inra.fr`

Abstract. Machine learning agents learn to take decisions extracting information from training data. When similar inferences can be obtained using a small subset of the same training set of samples, the subset is called *coreset*. Coresets discovery is an active line of research as it may be used to reduce the training speed as well as to allow human experts to gain a better understanding of both the phenomenon and the decisions, by reducing the number of samples to be examined. For classification problems, the state-of-the-art in coreset discovery is *EvoCore*, a multi-objective evolutionary algorithm. In this work EvoCore is exploited both on synthetic and on real data sets, showing how coresets may be useful in explaining decisions taken by machine learning classifiers.

Keywords: classification, coreset discovery, EvoCore, evolutionary algorithms, explain AI, machine learning, multi-objective

1 Introduction

Machine learning (ML) algorithms have recently emerged as an extremely effective set of technologies in addressing real world problems, both on structured and unstructured data [1]. Such progress may be explained as a combination of several factors, including an increasing availability of data and the diffusion of high-performance computing platforms. The main advantage of such models consists in their capacity, as they are composed of thousands or even millions of parameters. Such peculiarity makes it possible for ML models to fit almost any kind of data distributions [2], thus providing effective solutions to problems that could not have been tackled before. Addressing complex problems requires both a detailed and possibly large set of data and a model with sufficient capacity. However, even if accurate, ML decisions are often completely incomprehensible even for human experts, as they involve a combination of large sets of (i) variables, (ii) samples, and (iii) model parameters. As the amount of available data grows, it takes more time for algorithms to be trained, and it becomes harder for domain experts to make sense of the data itself. One possible solution to such issues is *coreset discovery*, extracting a subset of the original samples that can approximate the original data distribution. Termed *coreset* [3], such set of samples can be used both to speed up training, and to help human experts making sense of large amounts of data. With regards to coreset discovery for classification, the state-of-the-art is represented by EvoCore [4][5], an evolutionary approach to classification tasks, exploiting a state-of-the-art multi-objective evolutionary algorithm, NSGA-II [6]. EvoCore finds the best trade-offs between amount of samples in the coreset (to be minimized) and classifier error (to be minimized), for a specific classification algorithm. The resulting Pareto front includes different coresets, each one representing an optimal compromise between the two objectives. A human expert would then be able to not only select the coreset more suited for his needs, but also obtain extra information on the ML algorithm's behavior, by observing its degradation in performance as the number of coreset points in Pareto-optimal candidate solutions decreases. Alternatively, a candidate coreset on the Pareto front can be automatically selected depending on its performance with respect to an unseen validation set. In this work, EvoCore is employed (i) on a toy problem, to provide the reader with an intuitive assessment of its capabilities, and (ii) on a dataset related to credit risk. A short analysis of the samples found in the coresets for the credit risk dataset shows how such points can be extremely informative for a human expert, probably representing different typologies of ideal/not ideal customers.

2 Background

2.1 Machine learning and classification

ML algorithms are able to *improve their performance on a given task over time through experience* [7]. Such techniques automatically create models that, once trained on user-provided (training) data, can then provide predictions on unseen (test) data. In essence, ML consists in framing a learning task as an optimization task and finding a near-optimal solution for the optimization problem, exploiting the training data. Popular ML algorithms range from decision trees [8], to logistic regression [9], to artificial neural networks [10]. *Classification*, a classic ML task, consists in associating a single instance of measurements of several *features*, called *sample*, to one (or more) pre-defined *classes*, representing different groups. ML algorithms can position hyperplanes (often called *decision boundaries*) in the feature space, and later use them to decide the group a given sample belongs to. The placement of decision boundaries is set to maximize technique-specific metrics, whose value depend on the efficacy of the boundary with respect the (labeled) training data. Decision boundaries inside a classifier can be represented explicitly, for example as a linear or non-linear combination of the features, or implicitly, for example as the outcome of a group of decision trees or other weak classifiers.

2.2 Coreset discovery

In computational geometry, coresets are defined as a small set of points that approximates the shape of a larger point set. The concept of coreset in ML is extended to intend a subset of the (training) input samples, such that a good approximation to the original input can be obtained by solving the optimization problem directly on the coreset, rather than on the whole original set of input samples [3]. Finding coresets for ML problems is an active line of research, with applications ranging from speeding up training of algorithms on large datasets [11] to gaining a better understanding of the algorithm's behavior. Unsurprisingly, a considerable number of approaches to coreset discovery can be found in the specialized literature. Often these algorithms start from the assumption that the single best coreset for a given dataset will be independent from the ML pipeline used, but this premise might not always be correct. Moreover, the problem of finding the coreset, given a specific dataset and an application, can be naturally expressed as multi-objective: on the one hand, the user wishes to identify a set of core samples as small as possible; but on the other hand, the performance of the algorithm trained on the coreset should not differ from its starting performance, when trained on the original dataset. For this reason, multi-objective optimization algorithms could be well-suited to this task.

2.3 Multi-objective evolutionary algorithms

Optimization problems with contrasting objectives have no single optimal solution. Each candidate represents a different compromise between the multiple conflicting aims. Yet, it is still possible to search for *optimal trade-offs*, for which an objective cannot be improved without degrading the others. The set of such optimal compromises is called *Pareto front*. Multi-objective evolutionary algorithms (MOEA) currently represent the state of the art for problems with contradictory objectives, and are able to obtain good approximations of the true Pareto front in a reasonable amount of time. One of the most effective MOEAs is the Non-Sorting Genetic Algorithm II (NSGA-II) [6].

3 EvoCore

Starting from the intuition that coreset discovery can be framed as a multi-objective problem, and that the results could be dependant on the target ML algorithm, a novel evolutionary approach to coreset discovery for classification has been recently proposed in [4][5]. Given a training set **Tr** and a ML classifier, a candidate solution in the framework represents a coreset, a subset of the original training set. Candidate solutions are internally represented as bit strings, of length equal to the size of the training set, where a 1 in position i means that the corresponding sample s_i is retained in the coreset, while a 0 means that the sample is not considered. The classifier is then trained on the candidate coreset, and then an evaluation is performed on two conflicting objectives: number of samples in the coreset (to be minimized), and resulting error of the classifier on the original training set (to be minimized). NSGA-II is then set to optimize the coreset, finding a suitable Pareto front consisting of the best compromises with respect to the two objectives. In case the user wishes to obtain a single solution, the original training set **Tr** can be split into a training set to be used internally **Tr'** and a validation set **V**. At the end of the evolutionary optimization, each candidate coreset on the Pareto front is evaluated on the validation set **V** (unseen by the evolutionary procedure), to find the compromise with the best generality.

4 Experimental results and discussion

In order to prove its efficacy, EvoCore is employed to extract coresets on two benchmark datasets: i. *Moons*, a synthetic data set composed of two interleaving distributions having an half circle shape (2 classes, 400 samples, 2 features); ii. *Credit*, a dataset evaluating credit risk (2 classes, 1000 samples, 20 features). In both cases EvoCore is used to find the coreset for four ML algorithms, representative of both classifiers with explicit hyperplanes (`Ridge` [12], `SVC` Support Vector Machines [13]) and ensemble, tree-based classifiers (`Bagging` [14], `RandomForest` [15]). All classifiers are implemented in the `scikit-learn` [16] Python module and use default parameters. For the sake of comparison, it is important that the classifiers will follow the same training steps, although under different conditions. Therefore, a fixed seed has been used for all algorithms that exploit pseudo-random elements in their training process. All the necessary code for the experiments has been implemented in Python, relying upon the `inspyred` module [17]. The code is freely available in a BitBucket public repository[3]. NSGA-II uses default parameters of the `inspyred` module, with the exception of $\mu = 200$, $\lambda = 400$, stop condition 200 generations, and evolutionary operators bit-flip (probability $p_{bf} = 0.5$) and 1-point crossover (probability $p_c = 0.5$). Parameter values have been defined after a set of preliminary runs. For each case study, samples are randomly split between the original training set **Tr** (66%) and test set (33%). Features of the datasets are normalized using a column statistical scaling (zscore) learned on the training set **Tr**, then eventually applied to the test set. The results obtained by EvoCore are then compared against well-known coreset discovery algorithms GIGA [18], FW [19], MP [20], OMP [20], LAR [21][22], and FS [23]. The comparison is performed on three metrics: i. core set size (lower is better); ii. classification accuracy on the test set (higher is better); iii. running time of the algorithm (lower is better). Results of the comparison are presented in Tables 1 and 2 [4]. Text in **bold** highlights the highest accuracy for each classifier on the test set.

4.1 Meta-analysis

With regards to test accuracy, EvoCore usually bests the other techniques, and sometimes is able to outperform the performance obtained by training the same

[3] Evolutionary Discovery of Coresets, https://bitbucket.org/evomlteam/evolutionary-core-sets/src/master/

[4] The accuracy of EvoCore is related to the coreset along the Pareto front having the highest accuracy on an unseen validation set.

Table 1: *Moons* data set. Training set size, classification accuracy on an unseen test set and running time (in seconds) for different classifiers exploiting both EvoCore and state-of-the-art algorithms for core set discovery.

	RandomForest			Bagging			SVC			Ridge		
algorithm	size	accuracy	avg time	size	accuracy	avg time	size	accuracy	avg time	size	accuracy	avg time
all samples	266	0.9328	-	2661	0.9254	-	266	0.9179	-	266	0.8134	-
EvoCore	10	**0.9403**	440.2 s	30	**0.9478**	415.9 s	24	**0.9403**	126.6 s	2	**0.8209**	139.8 s
GIGA	2	0.4254	0.01 s	2	0.2463	0.01 s	2	0.4701	0.01 s	2	0.4701	0.01 s
FW	6	0.6493	3.6 s	6	0.6493	3.6 s	6	0.5299	3.6 s	6	0.6866	3.6 s
MP	3	0.5149	4.6 s	3	0.5821	4.6 s	3	0.5896	4.6 s	3	0.6642	4.6 s
FS	2	0.5149	4.3 s	2	0.2313	4.3 s	2	0.6119	4.3 s	2	0.6119	4.3 s
OP	2	0.5149	0.01 s	2	0.2463	0.01 s	2	0.6493	0.01 s	2	0.6493	0.01 s
LAR	3	0.5149	24.2 s	3	0.2388	24.2 s	3	0.5224	24.2 s	3	0.5896	24.2 s

Table 2: *Credit* data set. Training set size, classification accuracy on an unseen test set and running time (in seconds) for different classifiers exploiting both EvoCore and state-of-the-art algorithms for core set discovery.

	RandomForest			Bagging			SVC			Ridge		
algorithm	size	accuracy	avg time	size	accuracy	avg time	size	accuracy	avg time	size	accuracy	avg time
all samples	666	0.7275		666	0.7066		666	0.7635		666	0.7695	
EvoCore	223	**0.7395**	735	241	0.7006	538	94	0.7335	173	11	0.7485	161
GIGA	137	0.7305	0.11	137	0.7066	0.11	137	0.7096	0.11	137	0.7156	0.11
FW	537	0.6886	0.87	537	0.6856	0.87	537	**0.7635**	0.87	537	**0.7635**	0.87
MP	528	0.6856	1.99	528	0.7036	1.99	528	0.7515	1.99	528	0.7605	1.99
FS	74	0.7006	1.90	74	**0.7186**	1.90	74	0.7305	1.90	74	0.7455	1.90
OP	20	0.7066	0.09	20	0.7036	0.09	20	0.6617	0.09	20	0.6587	0.09
LAR	21	0.6707	0.68	21	0.6886	0.68	21	0.6407	0.68	21	0.6677	0.68

classifier with all the whole original training set. This means that the decision boundaries generated using the evolved core set may generalize even better than those generated using the whole training set. Figure 1 shows the decision boundaries obtained using the whole training set (left column) or the best candidate core set (right column) to train classifiers, on the Moons data set. This result suggest that the performance of ML classifiers would not be a function of the *size* of the training set (as Big Data and Deep Learning often claim) but a function of the *mutual position* of the training samples in the feature space. More generally, for artificial intelligence agents it may be more important to have samples with the right mutual position rather than have a huge data set.

Figure 2 reports a meta-analysis of all the Pareto-optimal candidate core sets found by EvoCore, divided by dataset, considering all classifiers. A few samples clearly appear very often among all candidate core sets, while others almost never do, but overall there is a considerable number of samples that are included with low but non-negligible frequency, indicating that different classifiers indeed exploit core sets of different shape.

4.2 Making sense of coresets and decisions in economics

Focusing on the Credit dataset, a concise analysis of the samples most frequently included in the coresets found by the proposed approach is presented in the

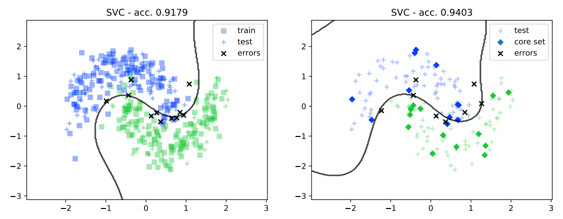

Fig. 1: Decision boundaries on the Moons data set using all the samples in the training set (Left) and only the core set (Right) for training the classifier. Train samples are represented by squares, test samples by crosses, core samples by black diamonds and test errors by 'x'-shapes.

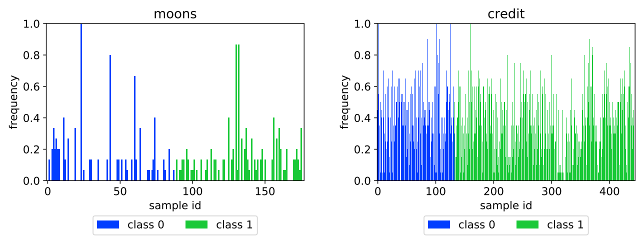

Fig. 2: Frequency of appearance of samples in the Pareto front solutions of all the classifiers.

following. Notice from figure 2 that there are four training points that appear in each core set of the Pareto fronts for all the classifiers. Table 3 lists the features of the most frequent core samples. Interestingly, such samples represent four different customer profiles. **C1** is a 58-years-old married female. She is asking the bank for relatively small credit for a new radio/tv. Currently, she has 4 existing credits at this bank, but her economical status seems stable (long-term employment, house and real estate of ownership), despite her low disposable income (less than 1,000 DMs). **C2** is an aged single male asking a credit for a new car. He is skilled and well paid ($\sim$2,000 DMs). He has recently changed his job, but he lives in its own house and has a life insurance. Overall, he looks like a responsible customer. **C3** is a young and skilled man. He has rented the house where he lives (with his wife, probably) for the last three years. Despite his young age, he owns a real estate, he already paid off previous credits with the bank and his disposable income is very high (more than 15,000 DMs). He is currently asking a remarkable credit of more than 3,000 DMs, but he is very rich and the bank will reserve him a kid-glove treatment. **C4** is 32 years old and wants to buy a new radio/tv. He has had the same employment for the past four years and he owns the house where he lives. However, his saving account is nearly empty and he must provide maintenance for two dependants. Despite their differences, the low-income aged woman (C1), the middle-class aged man (C2), and the wealthy young man (C3) turn out to be good customers for the bank. They probably represent three "ideal" profiles of good customers, as their characteristics suggest

Table 3: Comparison of the fundamental samples found in the credit risk dataset.

features	C1	C2	C3	C4
checking account (DMs)	<0	no checking	0<x<200	no checking
credit duration (months)	12	24	18	24
credit history	critical/other existing	existing paid	existing paid	existing paid
credit purpose	radio/tv	new car	radio/tv	radio/tv
credit amount	385	2255	3213	1552
savings account (DMs)	<100	unknown	500<x<1000	<100
employment duration (years)	4<x<7	<1	<1	4<x<7
installment rate	4%	4%	1%	3%
personal status	married female	single male	married male	single male
other debtors	none	none	none	none
residence since (years)	3	1	3	1
property magnitude	real estate	life insurance	real estate	car
age	58	54	25	32
other payment plans	none	none	none	bank
housing	own	own	rent	own
existing credits	4	1	1	1
job	unskilled resident	skilled	skilled	skilled
dependants	1	1	1	2
own telephone	yes	no	no	no
foreign worker	yes	yes	yes	yes
target	good	good	good	bad

economic stability. On the contrary, the middle-class young (and single) man with two dependants to support and low liquidity seems a representative sample of a risky customer.

5 Conclusions

Coreset discovery is a problem of utmost practical importance for decision making, especially when dealing with huge amount of data. This work exploits the EvoCore algorithm to extract coresets from both synthetic and real data sets. The analysis of the results on the economic dataset shows how coresets may be useful in explaining decisions taken by machine learning classifiers.

Future works will extend the coreset analysis to clustering problems in economics, where the objective is to obtain high-quality groups of samples, with no pre-existing information on the number or qualities of the groups.

References

1. Kasey Panetta. Top trends in the gartner hype cycle for emerging technologies, 2017. *Gartner*, pages 1–5, 2017.
2. Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.
3. Olivier Bachem, Mario Lucic, and Andreas Krause. Practical coreset constructions for machine learning. *arXiv preprint arXiv:1703.06476*, 2017.

4. Pietro Barbiero and Alberto Tonda. Fundamental flowers: Evolutionary discovery of coresets for classification. In *Applications of Evolutionary Computation – 22nd International Conference EvoApplications*, page TBA. 2019.
5. Barbiero Pietro and Tonda Alberto. Evolutionary discovery of coresets for machine learning. In *The Genetic and Evolutionary Computation Conference (GECCO)*, 07 2019.
6. Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.
7. Arthur L Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229, 1959.
8. Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and regression trees.* CRC press, 1984.
9. David R Cox. The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 215–242, 1958.
10. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning.* Mit press edition, 2016.
11. Ivor W Tsang, James T Kwok, and Pak-Ming Cheung. Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6(Apr):363–392, 2005.
12. Andrey Nikolayevich Tikhonov. On the stability of inverse problems. In *Dokl. Akad. Nauk SSSR*, volume 39, pages 195–198, 1943.
13. Marti A. Hearst, Susan T Dumais, Edgar Osman, John Platt, and Bernhard Scholkopf. Support vector machines. *IEEE Intelligent Systems and their Applications*, 13(4):18–28, 1998.
14. Leo Breiman. Pasting small votes for classification in large databases and on-line. *Machine Learning*, 36(1-2):85–103, 1999.
15. Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.
16. F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.
17. Aaron Garrett. inspyred (version 1.0.1) inspired intelligence. https://github.com/aarongarrett/inspyred, 2012.
18. Trevor Campbell and Tamara Broderick. Bayesian Coreset Construction via Greedy Iterative Geodesic Ascent. In *International Conference on Machine Learning (ICML)*, 2018.
19. Kenneth L Clarkson. Coresets, Sparse Greedy Approximation, and the Frank-Wolfe Algorithm. In *ACM Transactions on Algorithms*, 2010.
20. Y.C. Pati, R. Rezaiifar, and P.S. Krishnaprasad. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers*, pages 40–44, 1993.
21. Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least Angle Regression. *The Annals of Statistics*, 32(2):407–451, 2004.
22. Christos Boutsidis, Petros Drineas, and Malik Magdon-Ismail. Near-optimal Coresets For Least-Squares Regression. Technical report, 2013.
23. Efroymson M. A. Multiple Regression Analysis. *Mathematical Methods for Digital Computers*, 1960.