Beyond Coreset Discovery: Evolutionary Archetypes

Pietro Barbiero
Politecnico di Torino
Torino, Italy
pietro.barbiero@studenti.polito.it

Giovanni Squillero
Politecnico di Torino
Torino, Italy
giovanni.squillero@polito.it

Alberto Tonda
INRA, Université Paris-Saclay
Thiverval-Grignon, France
alberto.tonda@inra.fr

ABSTRACT

In machine learning a coreset is defined as a subset of the training set using which an algorithm obtains performances similar to what it would deliver if trained over the whole original data. Advantages of coresets include improving training speed and easing human understanding. Coreset discovery is an open line of research as limiting the training might also impair the quality of the result. Differently, virtual points, here called *archetypes*, might be far more informative for a machine learning algorithm. Starting from this intuition, a novel evolutionary approach to archetype set discovery is presented: starting from a population seeded with candidate coresets, a multi-objective evolutionary algorithm is set to modify them and eventually create archetype sets, to minimize both number of points in the set and classification error. Experimental results on popular benchmarks show that the proposed approach is able to deliver results that allow a classifier to obtain lower error and better ability of generalizing on unseen data than state-of-the-art coreset discovery techniques.

CCS CONCEPTS

• **Computing methodologies** → **Supervised learning**; **Genetic algorithms**.

KEYWORDS

Archetype sets; Classification; Coresets; Coreset discovery; Evolutionary algorithms; Explain AI; Machine learning; Multi-objective

ACM Reference Format:
Pietro Barbiero, Giovanni Squillero, and Alberto Tonda. 2019. Beyond Coreset Discovery: Evolutionary Archetypes. In *Proceedings of the Genetic and Evolutionary Computation Conference 2019 (GECCO '19).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3319619.3326789

1 PROPOSED APPROACH

In machine learning a coreset is defined as a subset of the training set using which an algorithm obtains performances similar to what it would deliver if trained over the whole original data. Advantages of coresets include improving training speed and easing human understanding. Coreset discovery is an open line of research as limiting the training might also impair the quality of the result.

An *archetype* is defined here as a virtual data point, not originally present in a given dataset, but created from scratch, taking into account both the data distribution and the information contained in the original dataset. If correctly crafted, archetypes can be more instructional for a ML algorithm, potentially requiring less data points to obtain the same results of a comparable coreset, and/or allowing for better generalization. Searching for sets of archetypes, or *archetype sets*, can also be seen as a relaxation of the initial coreset discovery problems, as an optimization algorithm becomes less constrained, being no longer forced to pick only from a discrete set of alternatives.

As the search space of all possible archetype sets of varying size for a given problem is clearly vast, it is necessary to resort to stochastic optimization to efficiently explore it. It must also be noted that the problem of coreset/archetype discovery is inherently multi-objective, as the two conflicting aims are to find a set of data points that delivers the best possible ML results, while being as compact as possible. Evolutionary algorithms (EAs) [7] are a technique with a long track record of successes, able to deliver good results in a reasonable amount of time. In particular, multi-objective EAs (MOEAs), such as NSGA-II [8], represent the state of the art in multi-objective optimization. Building on previous works on evolutionary coreset discovery [1], the algorithm proposed here extends coreset discovery to archetype set discovery, using a MOEA.

The genotype of a candidate solution is represented by a matrix, where each row encodes the real-valued features of one virtual data point, plus an integer that associates the virtual point to one of the known classes. The two fitness functions used in this multi-objective problem are the size of the archetype set (to be minimized), and the error of the target classifier trained on the archetype set, evaluated on the original dataset (to be minimized). The starting population of the MOEA is initialized with candidate coresets of random size. The minimum size corresponds to the number of classes in the problem, so that each candidate solution has at least one data point associated to each class; the maximum size is defined as a 1/10 of the original dataset size. Data points in each of such coresets are randomly drawn from the original dataset. When generating new candidate solutions, the MOEA selects one of the following operators, with flat probability: i. mutate one feature of one archetype in an archetype set, using a Gaussian mutation with $\mu = 0$ and $\sigma = 0.1$; ii. mutate the class associated to one archetype, changing it to a different valid class in the problem; iii. remove one archetype from the archetype set, checking that the solution is still valid, containing at least one archetype per class; iv. add one random sample from the original training set to an archetype set; v. recombine two candidate solutions, by randomly distributing each archetype contained in both between two children solutions.

GECCO '19, July 13–17, 2019, Prague, Czech Republic

ACM ISBN 978-1-4503-6748-6/19/07…$15.00
https://doi.org/10.1145/3319619.3326789

Table 1: Iris dataset. Coreset size, accuracy on test set and running time (seconds) of the considered classifiers and coreset algorithms.

	RandomForest			Bagging			SVC			Ridge		
algorithm	size	accuracy	avg time	size	accuracy	avg time	size	accuracy	avg time	size	accuracy	avg time
all samples	99	**0.9608**		99	0.9254		99	0.9412		99	0.8824	
EvoArch	5	**0.9608**	914.3	16	**0.9608**	838.5	6	0.9412	194.9	3	**0.9804**	149.5
GIGA	7	0.9216	0.01	7	0.6667	0.01	7	**0.9804**	0.01	7	0.8431	0.01
FW	15	0.8824	3.6	15	0.8627	3.6	15	0.9412	3.6	15	0.8235	3.6
MP	14	0.9412	4.8	14	0.8627	4.8	14	0.9216	4.8	14	0.7255	4.8
FS	7	0.6667	4.5	7	0.7059	4.5	7	0.6471	4.5	7	0.6275	4.5
OP	5	0.7059	0.01	5	0.5294	0.01	5	0.7843	0.01	5	0.8235	0.01
LAR	4	0.5294	22.2	4	0.6863	22.2	4	0.6471	22.2	4	0.7059	22.2

2 EXPERIMENTAL RESULTS

All the experiments presented in this section exploit 4 classifiers, representative of both hyperplane-based and ensemble, tree-based classifiers: `Bagging` [3], `RandomForest` [4], `Ridge` [15], and `SVC` (Support Vector Machines) [11]. All classifiers are implemented in the `scikit-learn`[1] [14] Python module and use default parameters. The code for reproducing the following experiments is freely available in a BitBucket public repository[2]. For the sake of comparison, it is important that the classifier will follow the same training steps, albeit in different conditions. For this reason a fixed seed has been set for all those that exploit pseudo-random elements in their training process.

The experiments are performed on well-known data sets publicly available in the `scikit-learn`: i. Blobs, three isotropic gaussian blobs (3 classes, 400 samples, 2 features); ii. Circles, a large circle containing a smaller one (2 classes, 400 samples, 2 features); iii. Moons, two interleaving half circles (2 classes, 400 samples, 2 features); iv. Iris [10] (3 classes, 150 samples, 4 features). For each case study, samples are randomly split between the original training set **Tr** (66%) and test set (33%).

The results obtained by the proposed approach are then compared against the 6 coreset discovery algorithms GIGA [5], FW [6], MP [13], OMP [13], LAR [9][2], and FS [12]. The comparison is performed on three measures: i. the coreset size (low is better); ii. the classification accuracy on the test set (high is better); iii. the running time of the algorithm (low is better). Table 1 shows the results of the comparison on the Iris data set, where the proposed approach is labeled *EvoArch*. Text in **bold** highlights the highest accuracy for each classifier on the test set. Figure 1 shows the details of the evolved solutions on the Moons data set using SVC.

Experimental results suggest that the performance of ML classifiers would not be a function of the *size* of the training set, but rather a function of the *mutual position* of the training samples in the feature space. By exploiting the original training set and by relaxing the constraint of sample positions, EvoArch generates a new smaller data set suited for each classifier in order to provide the best generalisation ability.

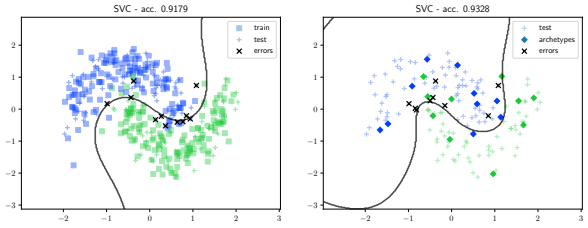

Figure 1: SVC classifier on the Moons data set. Pareto front (Left), decision boundary using all training samples (Center), and decision boundary using an archetype set (Right).

REFERENCES

[1] Pietro Barbiero and Alberto Tonda. 2019. Fundamental Flowers: Finding Core Sets for Classification using Evolutionary Computation. In *EvoApplications 2019.*

[2] Christos Boutsidis, Petros Drineas, and Malik Magdon-Ismail. 2013. *Near-optimal Coresets For Least-Squares Regression.* Technical Report. arXiv:1202.3505v2 https://arxiv.org/pdf/1202.3505.pdf

[3] Leo Breiman. 1999. Pasting small votes for classification in large databases and on-line. *Machine Learning* 36, 1-2 (1999), 85–103.

[4] Leo Breiman. 2001. Random forests. *Machine learning* 45, 1 (2001), 5–32.

[5] Trevor Campbell and Tamara Broderick. 2018. Bayesian Coreset Construction via Greedy Iterative Geodesic Ascent. In *International Conference on Machine Learning (ICML).* arXiv:arXiv:1802.01737v2 https://arxiv.org/pdf/1802.01737.pdf

[6] Kenneth L Clarkson. 2010. Coresets, Sparse Greedy Approximation, and the Frank-Wolfe Algorithm. In *ACM Transactions on Algorithms.* http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.145.9299

[7] Kenneth A De Jong. 2006. *Evolutionary computation: a unified approach.* MIT press.

[8] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation* 6, 2 (2002), 182–197.

[9] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. 2004. Least Angle Regression. *The Annals of Statistics* 32, 2 (2004), 407–451. https://doi.org/10.1214/009053604000000067

[10] Ronald A Fisher. 1936. The use of multiple measurements in taxonomic problems. *Annals of eugenics* 7, 2 (1936), 179–188.

[11] Marti A. Hearst, Susan T Dumais, Edgar Osman, John Platt, and Bernhard Scholkopf. 1998. Support vector machines. *IEEE Intelligent Systems and their Applications* 13, 4 (1998), 18–28.

[12] Efroymson M. A. 1960. Multiple Regression Analysis. *Mathematical Methods for Digital Computers* (1960).

[13] Y.C. Pati, R. Rezaiifar, and P.S. Krishnaprasad. 1993. Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers* (1993), 40–44. https://doi.org/10.1109/ACSSC.1993.342465 arXiv:1108.3326

[14] 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. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[15] Andrey Nikolayevich Tikhonov. 1943. On the stability of inverse problems. In *Dokl. Akad. Nauk SSSR*, Vol. 39. 195–198.

[1] scikit-learn: Machine Learning in Python, http://scikit-learn.org/stable/

[2] Evolutionary Discovery of Archetypes, https://bitbucket.org/evomlteam/evolutionary-archetypes/src/master/