Ant Colony Optimisation-Based Classification Using Two-Dimensional Polygons

Morten Goodwin[1](✉) and Anis Yazidi[2](✉)

[1] Deptartment of ICT, Institute for Technology and Sciences,
University of Agder, Agder, Norway
morten.goodwin@uia.no

[2] Department of Computer Science,
Akershus University College of Applied Sciences, Oslo, Norway
anis.yazidi@hioa.no

Abstract. The application of Ant Colony Optimization to the field of classification has mostly been limited to *hybrid approaches* which attempt at boosting the performance of existing classifiers (such as Decision Trees and Support Vector Machines (SVM)) — often through guided feature reductions or parameter optimizations.

In this paper we introduce PolyACO: A novel Ant Colony based classifier operating in two dimensional space that utilizes ray casting. To the best of our knowledge, our work is the first reported Ant Colony based classifier which is *non-hybrid*, in the sense, that it does not build on any legacy classifiers. The essence of the scheme is to create a separator in the feature space by imposing ant-guided random walks in a grid system. The walks are self-enclosing so that the ants return back to the starting node forming a closed classification path yielding a many edged polygon. Experimental results on both synthetic and real-life data show that our scheme is able to perfectly separate both simple and complex patterns, without utilizing "kernel tricks" and outperforming existing classifiers, such as polynomial and linear SVM. The results are impressive given the simplicity of PolyACO compared to other approaches such as SVM.

1 Introduction

Supervised Learning is one of the most central tasks in Machine Learning and Pattern Recognition. However, it becomes intrinsically challenging whenever the data to be classified is not easily separable in the feature space. A myriad of classification algorithms have been proposed in the literature with a variety of behaviors and limitations [11]. Examples of these algorithms include Neural Networks, SVM and Decision trees.

Common trends in research is to apply Ant Colony Optimization (ACO) as rule based variants or as a way to enhance some of the state-of-the-art classifiers. The latter work as optimisers for classifiers such as decision trees or neural networks [3,12]. To the best of our knowledge, there is no similar work on non-hybrid Ant Colony based classifiers that solely resorts to ACO without the aid of any other legacy classifier.

M. Dorigo et al. (Eds.): ANTS 2016, LNCS 9882, pp. 53–64, 2016.
DOI: 10.1007/978-3-319-44427-7_5

A broad class of classification algorithms such as SVM and perception rely upon defining mathematical functions with weights that efficiently can separate two or more classes of data where unknown weights are learned based the training data. Often, the "best" hyperplane[1] to separate classes does not follow the mathematical properties of a function. The "best" line can for example be a polygon encircling certain data points, which is not a function and therefore cannot straightforward be outputted by SVM or similar classifiers.

Figure 1 shows an example of labeled data where it is not possible to perfectly separate the data with one function simply because any line separating the data perfectly will have multiple $y-$values of some of the $x-$values — which defies the definition of mathematical functions. SVM deals with this by projecting the data in high dimensional space or using the "kernel trick".

Many kernels are available for SVM as a way to provide a "shape" of the separator which is not limited to linear or polynomial functions. The kernel yields an equivalent functionality as to transposing the data to many dimensions. However, the accuracy of the SVM is dependent on the right choice of the kernel function, as well as several other parameters, which is not an easy task given the unlimited number of available kernels. It is often based on trial and error.

This paper introduces PolyACO, a novel classification scheme operating in two dimensions using ACO that does not involve a "kernel trick" whenever the data is not easily separable. The presented approach deals with classification problems in two-dimensional Euclidean space by building separators with many-sided polygons. The polygons are extrapolated from pheromone trails of ants walking with a preference towards encapsulating of all items from one class and excluding any items from other classes from the encapsulation. This way, emerging polygons encapsulate each class in such a way that they can be used as classifiers. The classification takes place by resorting to ray casting unknown items which identifies whether an item is part of a polygon, and each item is labeled accordingly.

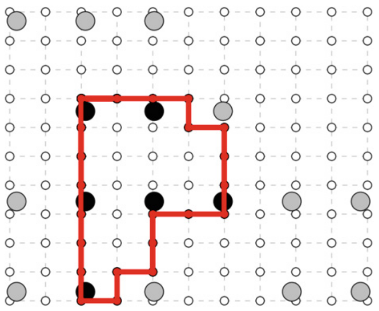


Fig. 1. Example of simple two class classification scenario with the classes Black (T_1) and Gray (T_2) (Color figure online)

Classification of unknown items based on labeled data is a supervised learning problem. Hence, in line with common practice, the problem is divided into two phases, namely (1) training and (2) classification:

1. **Training phase:** The aim is to create a polygon that encircles classes of items so that the polygons separate the training classes from each other.
2. **Classification phase:** In this phase, we use the polygon as a basis to determine which class a new unknown item to be classified belongs to.

[1] The hyperplane is a line in a two-dimensional space.

The overall aim of the training phase can therefore be stated as to find a polygon $s*$, consisting of vertices and edges, that minimises $f(s*)$ where $f(s*)$ tells how well the polygon $s*$ separate the items in the training. Thus, formally, we aim to find an $s* \in \mathbf{S}$ so that $f(s*) \leq f(s) \in \mathbf{S}$. For this we use ACO explained in Sect. 2. In turn, the classification determining whether the item to be classified is within or outside of the polygon $s*$.

The paper is organised as follows. Section 2 reviews the-state-of-the-art in the area of swarm intelligence based classifiers with special focus on ACO. Section 3 continues with introducing our solution: PolyACO as a method for creating polygons for classification with two classes and corresponding results. Finally, in Sect. 4, we draw conclusions and gives insights into future work.

2 Ant Colony Optimisation (ACO)

For completeness of the paper, we will briefly discuss variants of ACO. Details of ACO, including updating rules, are presented in many other papers [4]. We therefore include elements relevant to PolyACO are included.

2.1 Standard ACO

Swarm intelligence denotes a set of nature-inspired paradigms that have received a lot of attention in computer science due to its simplicity and adaptability [20]. ACO figures among the most popular swarm intelligence algorithms due to its ability to solve many optimization problems. ACO involves artificial ants operating a reinforced random walk over a graph. The ants release pheromones in favorable paths which subsequent ant members follow creating a reinforcement learning based behavior. The colony of ants will thus concentrate its walk on the most favorable paths and in consequence iteratively optimize the solution [4].

Finding the shortest path in a bidirectional graph with vertices and edges $G(V, E)$ using ACO in its simplest form works as follows. Artificial ants move from vertex to vertex. An ant that finds a route s from the source v_s to the sink v_t will release pheromones $\tau_{i,j}$ corresponding to all edges $e_{i,j} \in s$, and $\Delta\tau^k_{i,j}$ corresponds to the change in pheromones for ant k . The pheromones for all ants m is defined as:

$$\tau_{i,j} \leftarrow (1-p)\tau_{i,j} + \sum_{k=1}^{m} \Delta\tau^k_{i,j} \tag{1}$$

2.2 $\mathcal{MAX} - \mathcal{MIN}$ ACO

In order to improve the convergence performance of ACO a special greedier variant called $\mathcal{MAX} - \mathcal{MIN}$ ACO was introduced [18,19]. Another variant referred to as $\mathcal{MMAS}$ was reported in [13]. These variants are greedier in the sense that they only spread out pheromones for the best solutions, and in this way subsequent ants will converge faster than with the traditional variants. Certain setups even have theoretically guaranteed convergence [5].

The improvement is summarized as follows [19]: (1) Only the global best ant is allowed to release pheromones. (2) The pheromones on each edge are limited to an interval $[\tau_{min}, \tau_{max}]$ to avoid stagnation. (3) All edges are initiated with τ_{max}. This is to achieve high exploration in the beginning of the search.

It follows that the pheromone trails are updated according to the function 2 (an update of Eq. 1):

$$\tau_{i,j} \leftarrow (1 - p)\tau_{i,j} + \Delta\tau_{i,j}^{best} \tag{2}$$

$\mathcal{MAX} - \mathcal{MIN}$ ACO is shown to be a good alternative to existing algorithms when solving NP-hard combinatorial optimization problems such as Traveling Salesman where it performs at the same level as comparative algorithms. In the asymmetric variants, it outperforms other known approaches [1].

2.3 ACO for Classification

A considerable amount of work for Swarm Intelligence classification tasks, including ACO, is reported in the literature. The existing approaches fall into three main categories: (1) Rule based extractors, and (2) Hybrid approaches involving ACO that attempt to enhance the quality of existing classifiers, (3) Clustering based approaches using Swarm Intelligence.

The rule based classifiers [10,12,14] construct graphs by letting the ants walk with preference towards common occurring examples so that strong pheromone trails are used as rules in the classifier, or a set of IF-THEN rules. Probably, the most notable rule based ACO classifiers are the AntMiner series [7] including: AntMiner [14], AntMiner2, AntMiner3, AntMiner+ [12] and new variantssuch as MAnt-Miner [6]. All the aforementioned AntMiner variants rely on the idea of letting the ant walk "on" examples so that the pheromone trails can yield usable rules.

The hybrid approaches use ACO to improve the performance of legacy classifiers, for example in feed forward neural networks [16]. In the latter case, this is achieved by letting the ants minimize a function consisting of a set of decision variables corresponding to the neuron parameter weights. Furthermore, there is a multitude of hybrid ACO variants for Bayesian networks [3], multi-net classifiers [17], and rule pruning [2].

The clustering based swarm intelligence rely techniques such as Particle Swarm and Artificial Bee Colony to cluster data in an unsupervised manner [8]. Even though these have similarities with PolyACO, they cannot be directly applied for supervised learning such as classification.

3 PolyACO

This section presents the novel algorithm PolyACO. For the training phase, it maps the $\mathcal{MAX} - \mathcal{MIN}$ ACO to the problem area by formally specifying an appropriate cost function that encircles one class. PolyACO trains the classifier by defining a polygon s. Subsequently, it uses s with ray casting to find if an item is part of the s.

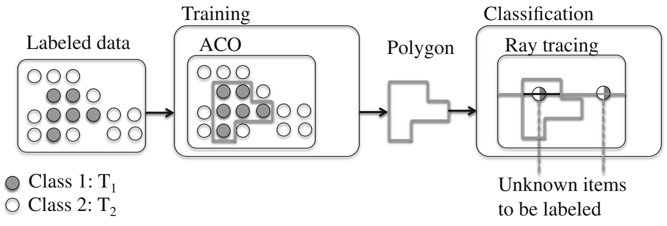


Fig. 2. Overview of PolyACO

Figure 2 presents an overview of the approach. The data is separated using ACO yielding a polygon. Next, the polygon is used in the classification with ray casting. In this example, the first item to be labeled will be classified as a T_1 ("Class 1") since it is shown to be inside the polygon, while the second item will be classified as T_2 ("Class 2") since it is outside the polygon.

In order to use ACO for encircling points into polygons, we extend the ACO $\mathcal{MAX}-\mathcal{MIN}$ update function (Eq. 2) with a cost function that measures the quality of PolyAco solution. In order to find whether a point is within a polygon we use ray casting.

3.1 Ray Casting

Vertical ray casting is used to consider whether an item is within or outside a many edged polygon [15]. Ray casting is a simple algorithm that determines where a virtual ray enters and exits a given solid.

In a two-dimensional XY-plane, a ray is sent with a $y-$coordinate and a bit starting at 0 and is increased by one very time an edged is passed. When the ray hits the item to be labelled, whether it is inside or outside the polygon is determined by reading the bit. An even number means outside while an odd number means inside. Formally, for node t_i and a polygon s, we get $h(t_i, s)$ representing to what extent it is inside or outside of the polygon as follows:

$$h(t_i, s) = \begin{cases} 1 \; \textit{if} \; t_i \in T_1 \; \textit{and is inside of} \; s. \\ 0 \; \textit{otherwise} \end{cases} \tag{3}$$

$h(t_i, s)$ gives 1 if t_i is correctly inside of the polygon, 0 otherwise. Note that the cost function $f(s)$ in Eq. 4 handles both items correctly inside and correctly outside of polygons.

3.2 Cost Function

Equation 4 presents the cost function. The cost function takes into account the information about whether an item t_i is inside or outside of a polygon s. It measures how good a polygon s is at encircling and isolating one class in the training data and is defined as:

$$f(s) = \frac{\sum_{t_i \in T_1} h(t_i, s) + \sum_{t_j \notin T_1} (1 - h(t_j, s))}{|T|}. \tag{4}$$

In layman's terms; function 4 gives the percentage of items that are either correctly inside or correctly outside of the polygon. From the example in Fig. 1 the red polygon s correctly encircles all items of class T_1, while correctly avoiding to encircle any other items from the other class T_2. Since s is a polygon that perfectly separates the two classes, it gives $f(s) = 1$.[2]

Hence, the pheromones update obeys the following function, combining Eqs. 2 and 4.

$$\Delta\tau_{i,j} = \frac{f(s)}{|s|} \tag{5}$$

The problem reduces to optimising $f(s)$, given the training data T, subject to the search space $\mathbf{S}$ — which is equivalent to finding an $s* \in \mathbf{S}$ so that $f(s*) \leq f(s) \in \mathbf{S}$.

3.3 Training Phase

The classifier is trained using a guided walk with $\mathcal{MAX} - \mathcal{MIN}$ ACO optimising for the score function $f(s)$ in order to create a polygon. Since new ants released walk with a preference towards high pheromone areas, the ant will converge towards a polygon that is a good separator. This polygon is the key to the classification. Hence, the pheromones on the path are deposited directly in accordance with a score function over the length of the path.

Note that the classifier, implicitly, performs optimization according to two properties of the data: the score function $f(s)$ and the length of the path, $|s|$. A shorter path will give larger amounts of pheromones per edge than a longer path because a shorter path gives pheromones over fewer edges.

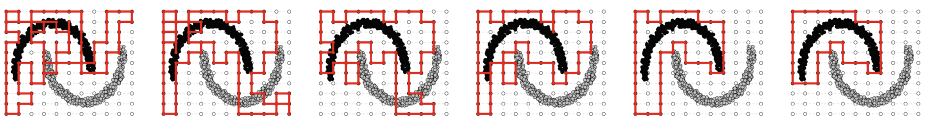
The classifier can therefore be considered as a many-edged polygon with only vertical or horizontal edges. The ants are not allowed to walk on nodes that has previously been selected, except for the initial starting node.

Convergence. Figure 3 shows an example for optimisation over time, i, during the training phase. The figure clearly depicts that in the beginning of the optimisation, after 50 ants, $i = 50$, PolyACO has already found an acceptable but imperfect solution that gives a score function $f(s) = 0.830$. The polygon s is continuously improved according to two ways: (1) The result from the score function increases and reaches close to 1 at $i = 850$. It reaches 1 at $i = 2000$. (2) The polygon becomes shorter over time. Consequently, the polygon is increasingly better fitting the data as i increases in this example.

The algorithm includes some off-the-shelf features to aid the convergence of the training, namely: graph pruning to help stuck ants, random start, and pheromone evaporation.

Pruning: To improve the performance, the graph is after each iteration pruned for indisputable simplifications. An example is a solution that goes directly from

[2] Note that s is one of the possible polygons with the shortest circumference that is able to perfectly separate the data. The reason for this is explained in Sect. 3.3.



$i = 50$, $f(s) = 0.830$; $i = 350$, $f(s) = 0.869$; $i = 550$, $f(s) = 0.873$; $i = 850$, $f(s) = 0.999$; $i = 2000$, $f(s) = 1.0$; $i = 2500$, $f(s) = 1.0$

Fig. 3. Example of best known polygon, $s*$ over training periods.

node v_i to v_j, and directly back to v_i. Such a solution is automatically pruned by removing v_i from the solution s.

Stuck ants: Since the ants cannot walk on previously visited nodes, they can easily get stuck. Any solution with stuck ants are simply ignored since the solution is obviously not correct.

Random starts: The ants start at random positions in the grid. A natural strategy would have been to always start the ants from the same node, say v_1. Nevertheless, the "best" polygon $s*$ harvesting the highest output from the score function may not include v_1. Accordingly, the ants start at a random node.

Pheromone evaporation: Pheromones are evaporated with 1 % probability. This means that 1 % of the pheromones are after each iteration dropped to avoid too early convergence and at the same time enable exploration. This is a balance between exploration and convergence in line with the literature [19].

3.4 Classification Phase

The training phase produces a polygon $s*$ that separates the training data in two sets. For any new unknown item t_i, the ray casting function from Eq. 3 is used. If $h(t_i, s) = 1$, t_i is classified as T_1, otherwise it is classified as T_2. This section presents results from various scenarios ranging from simple solutions with easily separable data to more complex settings with noisy data both real-life and synthetic. For each generated scenario, 1000 data points per class are generated or extracted. For the real scenarios, all data is used. In all cases, half of the data is used for training and the other half for classification as cross validation. All scenarios are run with 10000 ants unless otherwise explicitly specified. Table 1 presents a summary of the results.[3]

Simple Environments. We shall present a simple experimental settings as proof of concept of PolyACO. This section empirically shows that the approach works in a simple environment with two easily separable sets of data. The data is composed of two blocks of data: T_1 and T_2. Figure 4(a) shows the pheromone trails after the training phase in this environment. The pheromones have built

[3] Many more data sets where tested, but due to the limited space in the paper only the most interesting results are included.

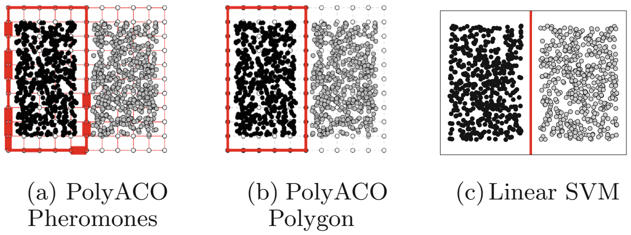


Fig. 4. Example of solutions based on easily separable items with the classes Black (T_1) and Gray (T_2).

a rectangular polygon encircling all items in T_1, but none of the items in T_2. Figure 4(b) presents the best known polygon s based on the pheromone trail. Since this is a polygon that perfectly separates the classes, it yields $f(s) = 1$. The mapping from pheromones to polygon in this example is quite straight forward. Lastly, for comparison purposes, Fig. 4(c) depicts the corresponding linear SVM. It is interesting to observe that PolyACO and SVM find the same boundaries.

This example strongly indicates that, for this particular example with easily separable data, the result of the PolyACO is equivalent to that of a linear SVM — both can be interpreted as perfect classifiers. Table 1 shows an overview of the classification results. Both PolyACO and SVM reach an accuracy of 1.0 — which is not surprising given the simplicity of the classification task.

Overlapping Data. In the above scenario, the data is organized so that it is perfectly separable. In contrast, the data in Fig. 5 is more challenging because it is overlapping — no line or polygon can perfectly separate the data sets.

Figure 5(a) shows the pheromones after the training phase. For the left- and lower part of the polygon, the pheromone trail is strong — the lines are thick. In contrast, where the data is overlapping in the diagonal where there are both items from T_1 and T_2, the scheme is less confident the trail is less strong. This indicates that when the confidence of the classifier is strong, PolyACO provides heavy pheromone trails. Figure 5(b) shows the corresponding polygon, and Fig. 5(c) and (d) show corresponding boundaries of linear and polynomial SVM.

From Table 1, we observe that PolyACO reaches an accuracy of 0.852, while linear SVM reaches 0.837, and polynomial SVM reaches 0.842. A conclusion to be drawn from this example is that PolyACO finds a slightly better boundary than the SVM lines, presumably because the rigged lines better fits the data than the straight and polynomial lines. Note that for this example, PolyACO even outperforms SVM with Gaussian kernel.

Circular. Classification in a circular environment is particularly difficult because there does not exist one mathematical function that separate the data

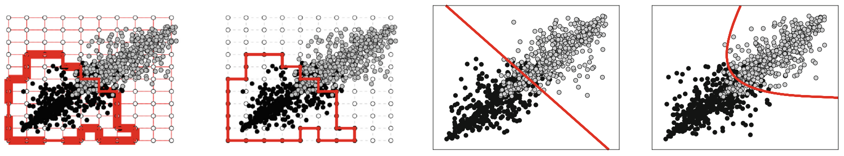


Fig. 5. Example of solutions based on overlapping data with the classes Black (T_1) and Gray (T_2).

without mapping it to multiple dimensions. The data is generated by a Gaussian distribution from two circles with the same center but with two different radius.

Figure 6(a) shows that the polygon is able to perfectly encircle class T_1, which is only matched by SVM with and Gaussian kernel in Fig. 6(c). The linear SVM in Fig. 6(b) and polynomial SVM (not presented as a figure) do not find any viable solution.

By adding 5 % noise to the data, meaning that 5 % of the data is intentionally wrongly labelled, Fig. 6(d) shows that PolyACO is still able to a close to perfect solution despite this discrepancy.

Table 1 shows that PolyACO gets an accuracy of 1 compared to 0.538 for linear SVM and 0.892 for polynomial SVM. The PolyACO accuracy is only matched by Gaussian SVM — which relies upon high dimension space. In the noisy environment, the PolyACO algorithm has only marginally reduced accuracy to 0.948. Correspondingly, the polynomial SVM dropped from 0.892 to 0.778.

Real Data Sets. Figure 7 shows the results from two real data sets from the UC Irvine Machine Learning Repository[4]; namely the Iris Plant Database and the Wine Quality Database. The intention is to show that the proposed scheme works not only in synthetic environments, but with real data.

Figure 7(a) and (b) show the pheromone trail and corresponding polygon for the Iris Plan Database. It is interesting to observe that there is evenly spread

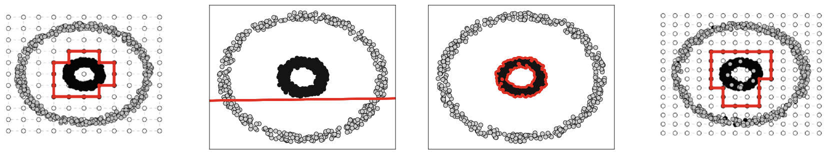


Fig. 6. Example of solutions based on circles with the classes Black (T_1) and Gray (T_2).

[4] http://archive.ics.uci.edu/ml/.

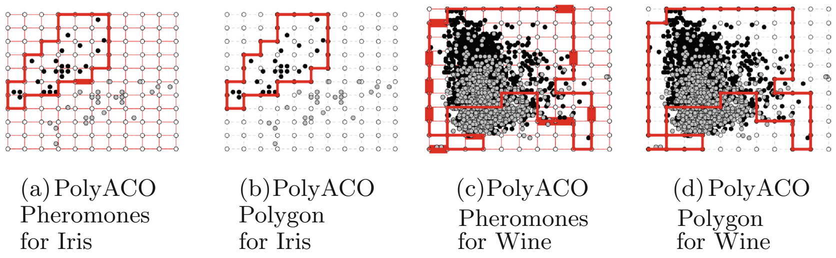

(a) PolyACO Pheromones for Iris (b) PolyACO Polygon for Iris (c) PolyACO Pheromones for Wine (d) PolyACO Polygon for Wine

Fig. 7. Example of solutions with the classes Black (T_1) and Gray (T_2) for the real data sets: Iris and Wine tasting.

out pheromones resulting in a solid polygon. Similarly, Fig. 7(c) and (d) show the pheromone trail and corresponding polygon for the Wine Quality Database. In this scenario, the data is more chaotic giving polygons with seemingly odd edges. This demonstrates that PolyACO is able to find rather odd and complex patterns. For both scenarios, the accuracy in Table 1 is very close to all variants of SVM. For the Iris Plant Dataset the accuracy for PolyACO is 0.960 compared to 0.980 for all SVM variants. For the Wine Quality Database, PolyACO reaches an accuracy of 0.690 compared to 0.685,0.683, and 0.690 for linear, polynomial and Gaussian SVM. Hence, assuming that SVM is able to classify the data well, so is arguably the PolyACO algorithm.

3.5 Comparisons

Table 1 presents the classification accuracy of PolyACO on the problems introduced in this paper. For comparison purpose SVM (with linear, polynomial and Gaussian kernel) is presented with exactly the same data. To avoid side effects due to the randomness of the data, all results are averages of 1000 runs. This is true even for the real data where the only randomness is which data is used for training and classification.

Table 1. Comparisons of the behaviour of various algorithms through classification for Real or Generated data.

Problem	Real or Generated	PolyACO PolyACO	Linear SVM	Polinomial SVM	Gaussian SVM
Easily separable items	Generated	1.0	1.0	1.0	1.0
Circles	Generated	1.0	0.538	0.892	1.0
Noisy circles	Generated	0.980	0.538	0.892	0.959
Overlapping data	Generated	0.852	0.837	0.842	0.840
Iris	Real	0.960	0.980	0.980	0.980
Wine tasting	Real	0.690	0.685	0.683	0.690

4 Conclusion

In this paper, we introduced PolyACO, a *non-hybrid* Ant Colony Optimisation (ACO) based classifier. To the best of our knowledge, PolyACO is the first *non-hybrid* ACO based classifier reported in the literature. It uses a combination of $\mathcal{MAX} - \mathcal{MIN}$ ACO and Ray Casting. PolyACO is a classification algorithm for data in two dimensions which relies upon encircling items with ant pheromones so that the pheromone trails can be used as polygons in a classification scheme.

We demonstrate that PolyACO gives impressive performance by applying it in many simulated and real environments. In all situations, PolyACO is able to perform equally well or better than state-of-the-art algorithms such as Support Vector Machine with linear, polynomial, and Gaussian kernel. PolyACO does this without relaying upon high dimensional space or the "kernel trick".

Even though PolyACO shows very promising classification performance, several areas need further exploration. As a future work, we aim to extend the current approach to work with more than two classes, potentially as a combination of multiple polygons similar to multiple functions separate many classes in an SVM. Additional exploration when data are represented by more than two features presumably means examining the behavior of PolyACO in more than two dimensions, or as an intelligent combination of several two-dimensional approaches. Furthermore, we plan to examine the fact some areas in the grid could benefit from additional fine tuned resolution while other areas do not, for example using multi-level ACO approach [9].

References

1. Asmar, D., Elshamli, A., Areibi, S.: A comparative assessment of ACO algorithms within a TSP environment. Dyn. Continous Discrete Impulsive Syst.-Ser. B-Appl. Algorithms **1**, 462–467 (2005)
2. Chan, A., Freitas, A.A.: A new classification-rule pruning procedure for an ant colony algorithm. In: Talbi, E.-G., Liardet, P., Collet, P., Lutton, E., Schoenauer, M. (eds.) EA 2005. LNCS, vol. 3871, pp. 25–36. Springer, Heidelberg (2006)
3. Daly, R., Shen, Q.: Learning Bayesian Network Equivalence Classes with Ant Colony Optimization (2014). arXiv preprint arXiv:1401.3464
4. Dorigo, M., Birattari, M., Stutzle, T.: Ant colony optimization. IEEE Comput. Intell. Mag. **1**(4), 28–39 (2006)
5. Gutjahr, W.J.: ACO algorithms with guaranteed convergence to the optimal solution. Inf. Process. Lett. **82**(3), 145–153 (2002)
6. Hota, S., Satapathy, P., Jagadev, A.K.: Modified ant colony optimization algorithm (MAnt-Miner) for classification rule mining. In: Jain, L.C., Patnaik, S., Ichalkaranje, N. (eds.) Intelligent Computing, Communication and Devices, pp. 267–275. Springer, New Delhi (2015)
7. Junior, I.C.: Data mining with ant colony algorithms. In: Huang, D.-S., Jo, K.-H., Zhou, Y.-Q., Han, K. (eds.) ICIC 2013. LNCS, vol. 7996, pp. 30–38. Springer, Heidelberg (2013)
8. Karaboga, D., Ozturk, C.: A novel clustering approach: Artificial Bee Colony (ABC) algorithm. Appl. Soft Comput. **11**(1), 652–657 (2011)

9. Lian, T.A., Llave, M.R., Goodwin, M., Bouhmala, N.: Towards multilevel ant colony optimisation for the Euclidean symmetric traveling salesman problem. In: Ali, M., Kwon, Y.S., Lee, C.-H., Kim, J., Kim, Y. (eds.) IEA/AIE 2015. LNCS, vol. 9101, pp. 222–231. Springer, Heidelberg (2015)
10. Liu, B., Abbas, H., McKay, B.: Classification rule discovery with ant colony optimization. In: IEEE/WIC International Conference on Intelligent Agent Technology, IAT 2003, pp. 83–88. IEEE (2003)
11. Madjarov, G., Kocev, D., Gjorgjevikj, D., Džeroski, S.: An extensive experimental comparison of methods for multi-label learning. Pattern Recogn. **45**(9), 3084–3104 (2012)
12. Martens, D., De Backer, M., Haesen, R., Vanthienen, J., Snoeck, M., Baesens, B.: Classification with ant colony optimization. IEEE Trans. Evol. Comput. **11**(5), 651–665 (2007)
13. Neumann, F., Sudholt, D., Witt, C.: Analysis of different MMAS ACO algorithms on unimodal functions and plateaus. Swarm Intell. **3**(1), 35–68 (2009)
14. Parpinelli, R.S., Lopes, H.S., Freitas, A., et al.: Data mining with an ant colony optimization algorithm. IEEE Trans. Evol. Comput. **6**(4), 321–332 (2002)
15. Roth, S.D.: Ray casting for modeling solids. Comput. Graph. Image Process. **18**(2), 109–144 (1982)
16. Salama, K.M., Abdelbar, A.M.: Learning neural network structures with ant colony algorithms. Swarm Intell. **1–37**, 229–265 (2015)
17. Salama, K.M., Freitas, A.A.: Ant colony algorithms for constructing Bayesian multi-net classifiers. Intell. Data Anal. **19**(2), 233–257 (2015)
18. Stützle, T., Hoos, H.: MAX-MIN Ant System and local search for the traveling salesman problem. In: IEEE International Conference on Evolutionary Computation, 1997, pp. 309–314. IEEE (1997)
19. Stützle, T., Hoos, H.H.: MAX-MIN ant system. Future Gener. Comput. Syst. **16**(8), 889–914 (2000)
20. Stützle, T., López-Ibáñez, M., Dorigo, M.: A concise overview of applications of ant colony optimization. Wiley Encycl. Oper. Res. Manage. Sci. **26**(2), 25–27 (2011)