Analyzing and Inferring Distance Metrics on the Particle Competition and Cooperation Algorithm

Lucas Guerreiro* and Fabricio Breve

São Paulo State University (UNESP), Rio Claro, SP, Brazil

guerreiroluc@gmail.com, fabricio@rc.unesp.br

Abstract. Machine Learning is an increasing area over the last few years and it is one of the highlights in Artificial Intelligence area. Nowadays, one of the most studied areas is Semi-supervised learning, mainly due to its characteristic of lower cost in labeling sample data. The most active category in this subarea is that of graph-based models. The Particle Competition and Cooperation in Networks algorithm is one of the techniques in this field, which has always used the Euclidean distance to measure the similarity between data and to build the graph. This project aims to implement the algorithm and apply other distance metrics in it, over different datasets. Thus, the results on these metrics are compared to analyze if there is such a metric that produces better results, or if different datasets require a different metric in order to obtain a better correct classification rate. We also expand this gained knowledge, proposing how to identify the best metric for the algorithm based on its initial graph structure, with no need to run the algorithm for each metric we want to evaluate.

Keywords: Artificial intelligence • Semi-supervised learning • Distance metrics • Graphs

1 Introduction

Machine Learning (ML) is a field in Artificial Intelligence which can be applied to many areas. ML aims at developing algorithms and techniques based on previous knowledge, in an analog way to how the human brain works when acquiring some new knowledge [1-2]. In ML we can highlight the semi-supervised learning class of algorithms. In this group of techniques, we use a small portion of labeled data, and the rest in unlabeled data, which we want to label or classify. The goal here is to use a combination of these data in order to achieve an efficient classification or labeling, taking into account that the labeling of training data is a task with high-costs and is slow. In this class of algorithms, we can talk about the group of graph-based algorithms, which has been the area with more studies in this type of algorithms [3]. We will work on an algorithm in this area, which is the Particle Competition and Cooperation algorithm (PCC) [4]. The premise of the algorithm is to construct a graph based on the dataset, then label a small portion of the graph nodes, and insert "particles" on these labeled points. These particles will walk through the graph and broad-

cast their labels on the unlabeled data; their labels are based on their original source node. Particles of the same class will cooperate to gain more territory when walking on the graph, making their class stronger. Particles of different classes will compete to gain territory, trying to strength nodes of its class and weakening nodes of different classes. When a particle visits one node of the same class as it, the particle gets stronger and raises the dominance of the node. When the particle visits a node of a different class, it gets weaker and the dominance level of the node is decreased. On the end of the execution, we attribute the class with the higher dominance on each node as its class and the algorithm is finished. Such as in many different graph-based algorithms, the initial structure of the graph is very important in determining a good or bad classification rate. In this algorithm, we create a graph at first, based on the distance between the parameters of each pair of nodes, and then we use the k-nearest-neighbors to determine if there is an edge between two nodes or not. The algorithm since its conception has always used the Euclidean Distance to construct the graph. In this work, we implement other 6 different metric distances to evaluate if we can get better results. We also propose a method to determine the best possible distance metric without running the algorithm for all the seven different metrics we are using.

2 Distance Metrics

For our experiments we are using seven of the most recommended distance metrics in the literature, these metrics are seen in Table 1.

Table 1. Distance Metrics

Distance	***Equation***
Euclidean [5]	$d(x,y) = \sum_i \sqrt{(x_i - y_i)^2}$
Mahalanobis [5]	$d(\vec{x},\vec{y}) = \sqrt{(\vec{x}-\vec{y})^T S^{-1}(\vec{x}-\vec{y})}$
City Block [5]	$d(x,y) = \sum_{i=1}^{n} \lvert x_i - y_i \rvert$
Chebyshev [6]	$d(x,y) = max\lvert x_i - y_i \rvert$
Minkowski [5]	$d(x,y) = \sqrt[m]{\sum_{i=1}^{n} \lvert x_i - y_i \rvert^m}$
Bray-Curtis [7]	$d(x,y) = \frac{\sum \lvert x_i - y_i \rvert}{\sum (x_i + y_i)}$
Canberra [8]	$d(x,y) = \sum_{i=1}^{n} \frac{\lvert x_i - y_i \rvert}{\lvert x_i \rvert + \lvert y_i \rvert}$

3 Methodology

In this work we propose the use of different distance metrics on the PCC algorithm, building different graphs. Considering the "walking" property, we want to evaluate whether we can achieve better results with these changes or not. In order to analyze such suppositions, we are implementing and testing these concepts on three UCI [9] datasets: Iris, Wine and Banknote Authentication. Iris dataset has 150 instances, with three parameters and 3 classes. Wine dataset has 178 instances, with 13 parameters and 3 classes. Banknote Authentication dataset has 1372 instances, with 4 parameters and 2 classes. After that, we want to find an equation which can describe a method to find the best metric based on the graph structure.

Therefore, we divide our work to two experiments: i) Evaluating the influence of different distance metrics on PCC algorithm; ii) Finding the best distance metric based on the graph structure.

For the experiment i) we firstly implement the original algorithm [4], then we apply the modifications, by providing all the seven distance metrics, and then we compare the classification results of the Euclidean distance and the others in order to evaluate whether we achieve some gains. We run the algorithm 100 times and take the average since it is stochastic. At the end of the experiment for each dataset, we will have the results for each distance metric and the best metric for that dataset. For the experiment ii) we work in an analog way to the first one, but we are just concerned with the assertive of best metric for the datasets and not the results themselves.

We apply the *z-score* normalization for the Wine and Banknote Authentication datasets, in order to get more consistent results. The Iris dataset already comes normalized so we did not need to apply any normalizations. To note this classification difference we applied the experiment for normalized and non-normalized data. Also, since Minkowski is a generalized equation, sensitive to the parameter m, we want to use a value that can generate interesting results. Considering we are already using $m = 1$ (City-Block), $m = 2$(Euclidean) and $m = \infty$(Chebyshev), we decided to use $m = 4$ in order to use a different distance metric. Therefore, when we say Minkowski in this work, we are referring to its form with $m = 4$.

Parameters

The PCC algorithm is sensitive to its initial parameters [4]. The aim of this word is to achieve good classification results, evaluating the influence of different distance metrics; therefore, it is important to optimize such parameters. To analyze and identify the best parameters we used the Iris dataset and applied all the definitions on it, fixing 2 parameters and varying the others. The main parameters in the PCC algorithm are: learning rate (Δv), the number of particles (p), the number of neighbors for each node (k), the number of iterations (z) and greedy walk rate (p_{grd}). The least sensitive among these five parameters is the number of iterations, considering a large number can produce good results, so we fixed it on 100,000 iterations. The number of particles has an influence on the final results as well; however, since we are using a semi-supervised algorithm and the labeled data on the datasets are random, we are using 10% of nodes as particles for all the cases. We have left the parameters we are investigating: Δv, k e p_{grd}.

Firstly, we have fixed k and p_{grd} to find Δv, we have varied its value between 0.05 and 0.5, in 0.05 intervals, we did not use bigger values to avoid overfitting the model and also because tests have shown it did not improve the results. We fixed the other parameters as recommended by the original algorithm, with k = 10 and p_{grd} = 0.5. These results are seen in Fig. 1.

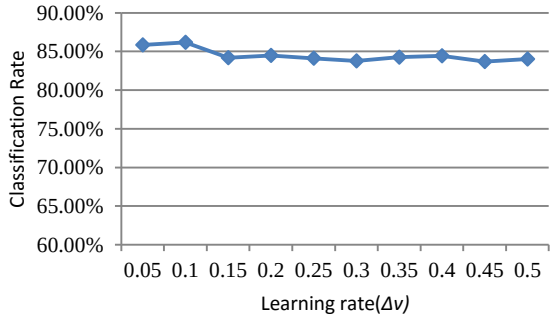


Fig. 1. Graph of classification rate versus learning rate, Δv

As we can observe the best learning rate was 0.1. We can also notice that learning rates larger than 0.1 produce worse results due to some overfitting. Therefore, we have chosen learning rate as 0.1.

The next parameters to be explored was the number of neighbors. We have varied k from 5 to 20, or proportionally to n the variation was from 3% to 13%, in intervals of 3. The other parameters were fixed as before. These test results are shown in Fig. 2.

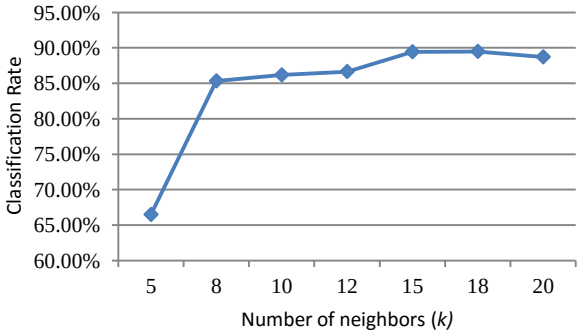


Fig. 2. Graph of classification rate vs number of neighbors, k

We see that the best results have been around $k = n/10$. Greater values have not been tested to avoid too many connections on large datasets. Therefore we have chosen k = 10%, such as the number of particles p.

The last parameter to be found was p_{grd}, which represents the probability between 0 and 1 of having a greedy walk during the algorithm's execution. The original algo-

rithm recommends this value near to 0.5, so we varied it from 0.3 to 0.7. The results are seen below.

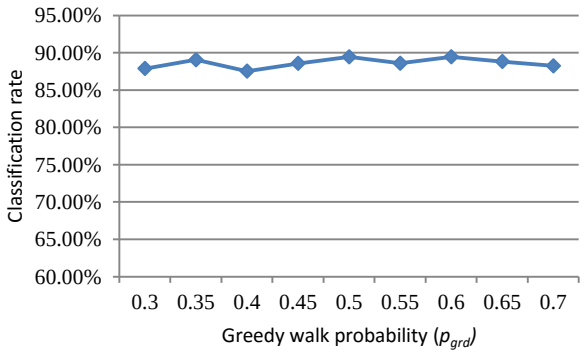


Fig. 3. Graph of classification rate versus greedy walk probability, p_{grd}

We observe that the value recommended on the reference of 0.5 proved to be among the best. Although some values have generated results close to this one, we kept it as 0.5.

4 Results

Firstly, we will discuss the results we have found for the first experiment – executing the algorithm for different distance metrics and not only the Euclidean distance as proposed originally.

Using the parameters and definitions exposed on the previous section, we execute the algorithm, and the results are shown in this section. We show all the results after the average of 100 executions for the first experiment in Table 2.

Table 2. Classification rate for experiment I. Standard deviation represented beween brackets.

Distance Metric	**Classification rate**		
	Iris	**Wine**	**Banknote Authentication**
Euclidean	89.44% (±4.69%)	63.58% (±7.02%)	95.48% (±2.80%)
Normalized Euclidean	-	91.50% (±7.19%)	**96.71% (±0.86%)**
Mahalanobis	**91.17% (±4.06%)**	64.73% (±5.54%	68.63% (±4.99%)
Normalized Mahalanobis	-	90.14% (±12.01%)	79.69% (±1.94%)
City Block	90.37% (±5.51%)	65.40% (±7.30%)	95.84% (±2.46%)
Normalized City Block	-	**93.68% (±8.02%)**	96.49% (±0.66%)
Chebyshev	85.28% (±8.87%)	63.31% (±7.28%)	94.01% (±2.97%)
Normalized Chebyshev	-	84.53% (±15.28%)	94.81% (±1.00%)

Distance Metric	Classification rate		
	Iris	**Wine**	**Banknote Authentication**
Minkowski	87.16% (±5.64%)	64.40% (±6.08%)	94.85% (±2.18%)
Normalized Minkowski	-	86.53% (±10.18%)	95.95% (±0.81%)
Bray-Curtis	89.32% (±8.27%)	65.67% (±6.53%)	64.82% (±19.48%)
Normalized Bray-Curtis	-	34.56% (±6.30%)	51.76% (±3.90%)
Canberra	81.87% (±5.20%)	80.09% (±19.06%)	52.04% (±4.85%)
Normalized Canberra	-	33.72% (±3.12%)	51.47% (±3.41%)

We observe that the best metric for the Iris dataset was the Mahalanobis distance with 91.17% of correct classification. We already can prove that a different metric really improves the results. We also see that City-Block, Euclidean and Bray-Curtis distances did achieve good results as well. This result can also be seen in Fig. 4.

For the Wine dataset, we have run the algorithm for normalized and non-normalized data. For most cases, the normalized data have produced the best results. The best distance metric for this dataset was City-Block normalized, which had 93.68% of correct classification. We can also highlight the Euclidean and Mahalanobis distances, which also got good results. Details are shown in Fig. 5.

In our experiments, we also tested the Banknote Authentication dataset with normalized and non-normalized data. For this dataset, we can see that the Euclidean distance with normalized data achieved the best results with 96.71% of successful classification. City-Block distance achieved good results as well. These results are exposed in Fig. 6.

Therefore, for the three datasets we tested, we had three different metric distances as the best one. We can affirm that there is not one absolute best metric to use in every case. One possible solution is to run the algorithm for every distance metric we have, as we did in this section. However, it is a high-cost operation. We propose, then, a method to evaluate which metric is the best, based on the graph structure; this method and the results are explained in the further sections.

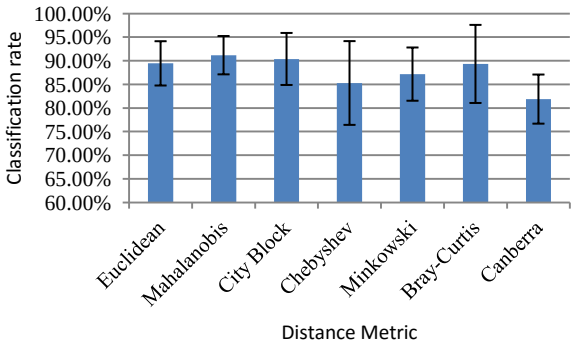


Fig. 4. Graph of classification rate on the Iris dataset

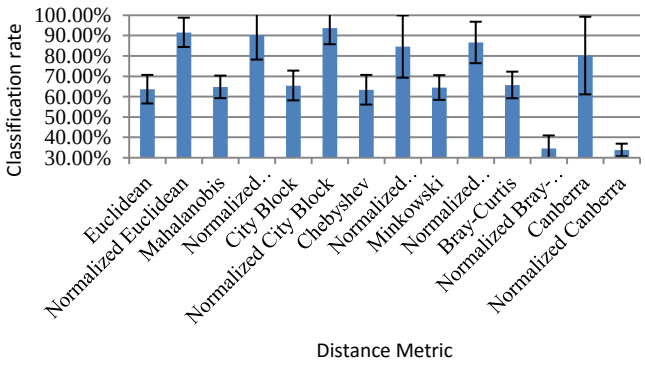


Fig. 5. Graph of classification rate on the Wine dataset

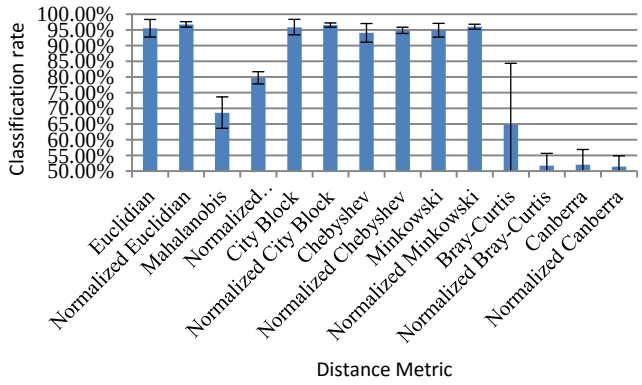


Fig. 6. Graph of the classification rate on the Banknote Authentication dataset

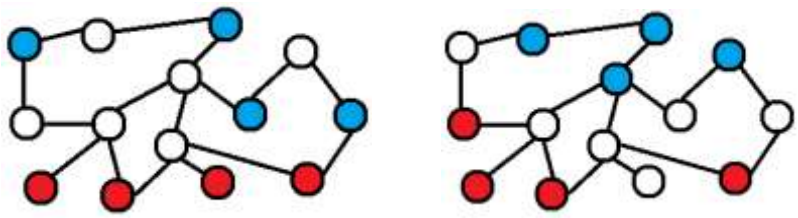
5 Defining the Best Distance Metric

Based on tests and analyzing the algorithm and graph structure, we have got to the equation 1 below.

$$C_m = \frac{\sum_{i=1}^{n}(\max_c A_{p_i} - \sum A_{p_{i-k}})}{\sum_{i=1}^{n} A_{p_i}} \quad (1)$$

With A_{p_i} representing an edge connecting an unlabeled node *i* to a labeled node *p*; $\max_c A_{p_i}$ represents de amount of edges A_{p_i} which have the most classes in common with relation to the node *i*; $A_{p_{i-k}}$ represents edges which are not those with the most common class connected to the node *i*. Therefore, the coefficient C_m of a metric *m*, is the relation for all the unlabeled data, with the difference between the number of edges which link the most recurrent class to a given node and the others classes, for all the

edges which connect labeled and unlabeled data. A coefficient C_m 1, then, represents a "cohesive" graph, with a good split between nodes of different classes, as seen in the example on Fig. 7a; on Fig. 7b we can observe the opposite example with a low value of coefficient C_m. Hypothetically, the distance metric with greater value of C_m would provide a greater classification rate over a metric with low C_m due to the best split on the graph.



(a) (b)

Fig. 7. Hypothetical examples of graphs with: (a) high C_m; (b) low C_m

6 Results

For this experiment, we have used the same parameters defined on the first experiment. However, in order to have more consistency, for each of the randomly generated group of particles, we repeated the experiment 10 times. The experiment for each dataset was repeated 100 times, and taken the average. The results were annotated as 1st and 2nd guess. Being 1st guess the greatest value of C_m and 2nd guess the second greatest value of C_m. The results are shown on Table 3 below.

Table 3. Correct guesses for experiment II

Dataset	1st guess	2nd guess	Total
Iris	91.2%	4%	95.2%
Wine	60%	34%	94%
Banknote Authentication	77.6%	21.2%	98.8%

As seen in the table above, for all the three datasets we have tested, summing up the 1st and 2nd guess we obtained to the best metrics on over 90% of the cases. Therefore, we can assume that using the Equation I we are most of the time certain that we will get to the best distance metric, running the algorithm two times (for the 1st and 2nd guesses) with no need to execute the algorithm for all the metrics we want to evaluate.

7 Conclusions

In this work we had two goals: i) evaluate the influence of different distance metrics on the PCC algorithm and achieve better results with such implementations; ii)

apply a method to identify the best metric previously to the execution of the algorithm in order to find the best distance metric without executing the algorithm several times.

We could see that the first objective was successfully reached, proving we can achieve better results than the original algorithm by applying other distance metrics. We can also notice that there is not a best metric for all the cases, needing to evaluate the metric that is the best for each dataset.

On the second goal, we also can call it successful, based on the results shown. We did infer a method to evaluate the best distance metric for each case. Our results demonstrate that we were able to identify the best metric in two guesses, which is promising considering the time we can save with such premise.

As future work, we propose to use more datasets and evaluate the results on those ones. It is interesting as well to use a more accurate parameter definition, maybe by optimizing those parameters with a genetic algorithm. We also find important to analyze deeply the results of the experiments to understand how the algorithm works to maybe infer those results in a better way.

Acknowledgment

The authors would like to thank the São Paulo Research Foundation - FAPESP (grant #2016/05669-4) and the National Counsel of Technological and Scientific Development - CNPq (grant #475717/2013-9) for the financial support.

References

1. Alpaydin, E.. Introduction to Machine Learning. MIT Press, 2004.
2. Mitchell, T. Machine Learning. McGraw Hill, 1997.
3. Chapelle, O.; Schölkopf, B.; Zien, A. Semi-Supervised Learning, Adaptive Computation and Machine Learning. MIT Press, 2006.
4. Breve, F. A.; Zhao, L.; Quiles, M. G.; Pedrycz, W.; Liu, J. Particle Competition and Cooperation in Networks for Semi-Supervised Learning. IEEE Transactions on Knowledge and Data Engineering, v. 24, p. 1686-1698, 2012.
5. Liu, Q.; Chu, X.; Xiao, J.; Zhu, H. Optmizing Non-orthogonal Space Distance Using PSO in Software Cost Estimation. IEEE Computer Software Applications Conference (COMPSAC), pp. 21-26, 2014.
6. Yang, Z.; Shufan, Y.; Yang, X.; Liqun, G. High-Dimensional Statistical Distance for Object Tracking. International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), vol.2, pp.386-389, 2010.
7. Shyam, R.; Singh, Y. N. Evaluation of Eigenface and Fisherfaces using Bray Curtis Dissimilarity Metric, 9th International Conference on Industrial and Information Systems (ICIIS), pp.1-6, 2014.
8. Kokare, M.; Chatterji, B. N.; Biswas, P. K. Comparison of Similarity Metrics for Texture Image Retrieval, Conference on Convergent Technologies for the Asia-Pacific Region, v. 2, pp.571-575, 2003.
9. Bache, K.; Lichman, M. UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA. University of California, School of Information and Computer Science, 2013