Particle Swarm Optimization Applied to Document Clustering

Carla Fernández González

Universidad Politécnica de Madrid, Madrid, Spain
carla.fernandez.gonzalez@alumnos.upm.es

Abstract. In this paper, a method to cluster documents using the Particle Swarm Optimization algorithm is presented. This method is evaluated by clustering documents from the Enron Spam and BBC News datasets. Finally, a study on the optimal hyperparameters for the algorithm is performed.

Keywords: Particle Swarm Optimization · Document clustering.

1 Introduction

Document classification is a topic that has been attracting much attention in recent years. The unsupervised classification of documents, in particular in the form of clustering, is a problem that has seen many applications such as detecting relevant documents during computer forensic analyses [1] or identifying descriptive titles from biomedical collections [2]. In the following sections, a specialized implementation of Particle Swarm Optimization (PSO) for document clustering, based on the approach by Cui *et al.* [3], will be explained and evaluated using subsets of the Enron Spam[1] and BBC News[2] datasets. All the code for this paper has been made publicly available at [3].

2 Algorithm

2.1 Document processing

As a way to encode the feature vectors, the Vector Space Model using Term Frequency-Inverse Document Frequency (TF-IDF) [4] was chosen. This was partly due to its use in [3] and partly because of its popularity and understandability. We define an $D \times W$ matrix, where D is the number of documents in the dataset and W is the number of words in the vocabulary. Each item in the matrix will represent the TF-IDF value for a specific word in a specific document, providing an intuition of how relevant it is in that context. Preprocessing

[1] http://www2.aueb.gr/users/ion/data/enron-spam/
[2] http://mlg.ucd.ie/datasets/bbc.html
[3] https://github.com/CarlaFernandez/PSO-Clustering

was applied in order to reduce the size of the dataset's vocabulary, first by performing stemming (with Porter's algorithm [5]) and stopword removal, and then by considering only those words whose frequency was at least 2 in the whole collection.

2.2 Particle Swarm Optimization

In this specific version of PSO, a particle holds as its position a matrix containing the cluster centroid vectors $C_1, C_2, ..., C_i, ..., C_k$, where the length of each C_i is the size of the vocabulary, thus representing a solution to the clustering problem. With each iteration of the algorithm, the particle adjusts the position of the centroid vectors according to its velocity, its position and the position of its neighboring particles. The objective function used was that of [3], which measures the average dispersion of documents inside their assigned clusters by the following equation:

$$f = \frac{\sum_{i=1}^{N_c} \left\{ \frac{\sum_{j=1}^{p_i} d(o_i, m_{ij})}{p_i} \right\}}{k} \tag{1}$$

where m_{ij} denotes the jth document vector which belongs to cluster i, O_i is the centroid vector of the ith cluster, $d(o_i, m_{ij})$ is the distance between document m_{ij} and the cluster centroid O_i, p_i stands for the number of documents which belongs to cluster C_i and k stands for the number of clusters. Thus, the goal of the algorithm is to minimize this fitness function.

The metric chosen to calculate $d(o_i, m_{ij})$ was cosine similarity, a popular alternative to measure both similarity between documents and cohesion within clusters [6], calculated as:

$$\cos(d_1, d_2) = \frac{d_1 d_2}{|d_1||d_2|} \tag{2}$$

where $d_1 d_2$ is the dot product of document vectors d_1 and d_2 and $|\cdot|$ denotes the length of the vector [3].

The algorithm works in the following way:

```
input  : TF-IDF document matrix, max_iter.
output: Assignment of documents to clusters.
for cluster k in particle ∈ swarm do
    cluster.centroid ⟵ random(doc_vec);
end
current_iter = 0;
while current_iter < max_iter do
    for particle ∈ swarm do
        assignClosestCentroid(doc_vec, particle.clusters);
        calculateFitness(particle);
        move(particle);
        current_iter = current_iter + 1;
    end
end
return doc_cluster_assignment;
```

Algorithm 1: PSO for document clustering.

3 Evaluation

3.1 Datasets

Class-balanced subsets of the Enron Spam and BBC News datasets were selected at random for the evaluation of the algorithm. Table 1 shows the distribution of documents and their classes. In the case of the BBC News dataset, text encoding problems reduced the number of documents available for the creation of a balanced subset. For this reason, the final number of documents in this dataset was significantly smaller than that in the Enron dataset.

Table 1. Distribution of document classes.

Dataset	Classes	Distrib.	Num. Docs.	Num. Terms
Enron Spam	Spam	500	1000	8176
	Legitimate	500		
BBC News	Business	40	200	3419
	Entertainment	40		
	Politics	40		
	Sport	40		
	Tech	40		

3.2 Results

Results from the evaluation of the algorithm and variations of its hyperparameters are shown in Table 2 and Table 3. Each row contains results of the best run

of the algorithm in terms of the fitness function among 5 executions. It should be noted that fitness values are generally small due to the use of cosine correlation as distance metric.

Table 2 shows the results from clustering the documents of the Enron Spam dataset. It can be seen that the best combination of hyperparameters occurs using the highest value for the social factor, which strenghtens the explorative nature of the algorithm and eases the escape of particles from local maxima or minima. Further, a good solution is also achieved using 15 particles. This may indicate that the problem is a complex one, which benefits from a larger swarm in order to better explore the search space. Overall, these results show that the algorithm has difficulties delimiting the clusters, even though generally more documents of the spam type are assigned to a cluster while legitimate documents are assigned to the other one.

Table 2. Results from clustering documents from the Enron Spam dataset. In bold are the hyperparameters tested in each case. Clusters show the number of legitimate and spam documents they contain, respectively.

Particles	Inertia	Cognitive	Social	Fitness	Cluster 1	Cluster 2
10	0.15	1.5	2.5	1.07e-11	430, 424	70, 76
10	**0.1**	1.5	2.5	1.26e-11	475, 457	25, 43
10	**0.2**	1.5	2.5	2.73e-11	484, 476	16, 24
5	0.15	1.5	2.5	3.63e-11	431, 384	69, 116
15	0.15	1.5	2.5	8.08e-12	315, 364	185, 136
10	0.15	**1**	**3**	1.14e-12	433, 424	67, 76
10	0.15	**2**	**2**	4.25e-11	482, 468	18, 32
10	0.15	**2.5**	**1.5**	5.59e-11	493, 474	7, 26

Table 3 shows the results from clustering the documents of the BBC News dataset. With this dataset, the algorithm has more difficulty in correctly detecting all the specified clusters. It can be seen that some clusters contain most of the documents while others are left practically empty. Further, it should be noted that the average fitness achieved in the executions is two orders of magnitude below that achieved in the Enron dataset. This is an indicator of the complexity of the problem and, together with the bigger number of clusters specified, could explain the poor classification results. Due to this complexity, the best fitness was achieved with the highest value of the social factor, while the lowest occurred with its lowest value, as happened with the Enron dataset. As expected, using more particles improves results by increasing the explorative power of the algorithm. As for the inertia parameter, it appears that for the BBC News dataset, as opposed to the Enron dataset, bigger values increase the quality of the solution. This may be due to higher velocities being more likely and enabling the particles to escape better from local extrema.

Table 3. Results from clustering documents from the BBC News dataset. Hyperparameters for each row correspond to those found in Table 2. Clusters show the number of business, entertainment, politics, sport and tech documents they contain, respectively.

Fitness	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
4.64e-9	4, 6, 13, 8, 6	12, 19, 6, 1, 16	24, 14, 21, 30, 18	0, 0, 0, 1, 0	0, 1, 0, 0, 0
3.66e-9	14, 2, 12, 3, 10	2, 12, 5, 6, 13	17, 13, 20, 21, 7	7, 13, 3, 8, 10	0, 0, 0, 2, 0
1.98e-9	0, 8, 6, 10, 8	2, 11, 1, 0, 0	20, 12, 24, 23, 16	6, 1, 9, 7, 9	12, 8, 0, 0, 7
4.88e-9	1, 0, 4, 28, 0	38, 40, 36, 9, 40	0, 0, 0, 1, 0	0, 0, 0, 2, 0	1, 0, 0, 0, 0
1.44e-9	17, 25, 11, 24, 15	2, 6, 17, 10, 10	19, 6, 11, 1, 15	1, 3, 1, 5, 0	1, 0, 0, 0, 0
1.35e-9	27, 10, 26, 8, 22	7, 18, 13, 22, 12	4, 9, 1, 5, 4	0, 3, 0, 3, 0	2, 0, 0, 2, 2
4.50e-9	3, 12, 5, 10, 0	37, 26, 32, 29, 38	0, 0, 2, 0, 0	0, 1, 1, 0, 2	0, 1, 0, 1, 0
5.89e-9	40, 39, 37, 21, 40	0, 0, 2, 11, 0	0, 1, 1, 6, 0	0, 0, 0, 1, 0	0, 0, 0, 1, 0

4 Conclusions & Future Research

In this paper, an algorithm for clustering documents using Particle Swarm Optimization has been presented. As opposed to Cui *et al.* [3], we have presented thorough results and discussion of the hyperparameters, specifying the resulting clusters and fitness values. In conclusion, we have proved that PSO can be used to cluster documents, although results suggest there may be other better performing algorithms. The source code and datasets used for this paper have been made publicly available.

In the future, it would be interesting to explore the idea of using different document representations, such as the popular word embeddings [7], in order to account for the context of words and not only their frequency.

References

1. Filipe da Cruz Nassif, L. & Raul Hruschka, E. Document Clustering for Forensic Analysis: An Approach for Improving Computer Inspection. *IEEE Transactions on Information Forensics and Security* **8,** 46–54 (2013).
2. Yeganova, L., Kim, W., Kim, S. & Wilbur, W. J. Retro: Concept-based clustering of biomedical topical sets. *Bioinformatics* **30,** 3240–3248 (2014).
3. Cui, X., Potok, T. & Palathingal, P. *Document clustering using particle swarm optimization* in *Proceedings 2005 IEEE Swarm Intelligence Symposium, SIS 2005* (2005), 185–191.
4. Hiemstra, D. A probabilistic justification for using tf x idf term weighting in information retrieval. *International Journal on Digital Libraries* **3,** 131–139 (2000).
5. Willett, P. The Porter stemming algorithm: Then and now. *Program* **40,** 219–223 (2006).
6. Singhal, A. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.* 1–9 (2001).
7. Mikolov, T., Chen, K., Corrado, G. & Dean, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).