

A web application to demonstrate the properties of Label Propagation algorithms*

Paolo Ceravolo^{1,*}, Samira Maghool^{1,*} and Andrei Georgiani Talpalaru¹

¹Department of Computer Science, Università degli Studi di Milano, Via Celoria 18, Milano, Italy

Abstract

Vector Label Propagation (VLP) is a machine learning technique used for tasks like semi-supervised learning and graph-based analysis. It is valuable in various fields, including social network analysis and recommendation systems, where each data point has multiple attributes. This paper discusses the development of a web application aimed at demonstrating the Agent-based vector-label propagation algorithm's (AVPRA) operation. Prior to this application, AVPRA execution and visualization were challenging for non-experts, relying on scripts. The web app offers a user-friendly, graphical interface, enabling real-time tracking and transparent visualization of the algorithm's iterations.

Keywords

Social Network Analysis, Vector Label propagation, Agent-based simulation, Web Application

1. Introduction

Social Network Analysis (SNA) explores social structures through networks and graph theory. Thanks to an abstract representation, elements of a system are considered as nodes (or agents when they are capable of making choices or active actions) of a graph while the connections between these nodes are edges. Using this representation, a multitude of structures and relationships such as *community* structure, which represents the tendency of nodes to be aggregated into distinct groups, can be described [1].


Label Propagation represents a family of algorithms used to predict the labels of nodes in a network and potential communities. Differently from other SNA algorithms, instead of taking a global view of the network, the decisions made by the algorithm are made with respect to local properties, thus Label Propagation algorithms take a local view of the network [2]. In an extended version, called *Vector Label Propagation*, the algorithms have been generalized to allow the identification of overlapping communities [1]. *Agent-based propagation* considers semi-intelligent agents acting with respect to the encountered situation supposing that they are located on a structured underlying network spreading the features through the edges. This way, the algorithm can control global effects, such as the termination condition using basic rules. The update rule defines how the labels are updated at each iteration.


CoopIS 2023 Demo Track

*Corresponding author.

✉ paolo.ceravolo@unimi.it (P. Ceravolo); samira.maghool@unimi.it (S. Maghool); andrei.talpalaru@unimi.it (A. G. Talpalaru)

ORCID 0000-0003-4473-6258 (P. Ceravolo); 0000-0001-8310-2050 (S. Maghool)

 © 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

Agent-based vector-label propagation algorithm (AVPRA) is an algorithm that allows the implementation of the update rule with respect to multiple factors, such as domain properties. Each agent is labeled with a d -dimensional vector where d is the number of the unique labels in the network \mathcal{L} . The output of the algorithm is a vector compatible with the input format used in most of the Machine Learning algorithms. The membership coefficient in the case of the AVPRA algorithm is a diffusion/accumulation of node labels, dependent on the distance to other nodes and their frequency. To avoid conflicts in the updates, the agents' statuses are updated synchronously. In AVPRA, the update function is as follows:

$$\mathbf{VL}_i[l](t) = w_1 \mathbf{VL}_i[l](t) + w_2 \sum_{j \in \Gamma(i)} \mathbf{VL}_{j \in \Gamma(i)}[l](t-1) \quad (1)$$

At each time step t , the $\mathbf{VL}_i[l](t)$ can be updated by aggregating the k neighbors' $\mathbf{VL}_{j \in \Gamma(i)}[l](t-1)$, where the w_1 and w_2 are the weights of current and previously assigned labels. The iterations continue till reaching a stationary state which implies that the changes in the membership coefficients must be smaller than a parameter p which is called the neglect threshold [3].

This paper is divided into the following sections. In Section 2, we discuss the application development and used technologies. In continuation, the application's properties are discussed in Section 1. Using a real dataset, the usability of the application is evaluated. In the last section, this paper is concluded with some final remarks and future plans.

2. Application Development

The **Social Networks Viewer** is a web application that allows the user to run and view the results of the algorithm in an interactive way in the web browser. The development phase of this application contains two parts: **frontend** and *backend*. The frontend component allows the user to interactively visualize the situation, at the current iteration, of the network and its vector labels. The backend component of the application with which the frontend interacts and is responsible for exposing an API that performs the required functionality. Backend should be able to execute the given AVPRA algorithm through a Python script and expose the results on the API. For the development of the frontend part of the application, containing user interaction and communication with the backend, several technologies and libraries are used which are presented as follows:

React.js¹: The main components of this library are the parts that allow the user interface to be organized in React and are defined in two modes: The component functions and the component classes. UI² state updates in basic JavaScript happen by finding the element in the DOM and updating it manually.3) Hooks³ also are added to allow state and other functions that existed in Component Classes to be used in Component Functions as well. JSX⁴ represents an extension of JavaScript that allows HTML to be written directly within JavaScript.

¹<https://it.reactjs.org/docs/react-component.html>

²<https://it.reactjs.org/docs/state-and-lifecycle.html>

³<https://it.reactjs.org/docs/hooks-intro.html>

⁴<https://it.reactjs.org/docs/introducing-jsx.html>

Also we have taken advantage of other libraries such as **Type-script**⁵, **Next.js**⁶, **D3.js**⁷, **Katex**⁸, **Tailwind CSS**⁹. For developing the backend part of the application, containing the API that has the ability to respond to frontend requests, several technologies and libraries are used. **Flask**¹⁰, **Sympy**¹¹, and **CDlib**¹² are the important ones.

2.0.1. Application Presentation

In this section, we present some of the principal features of the implemented web application. **Main Page** is used to allow the user to choose between two different modes of testing and viewing the results of the algorithm (Fig. 1. a).

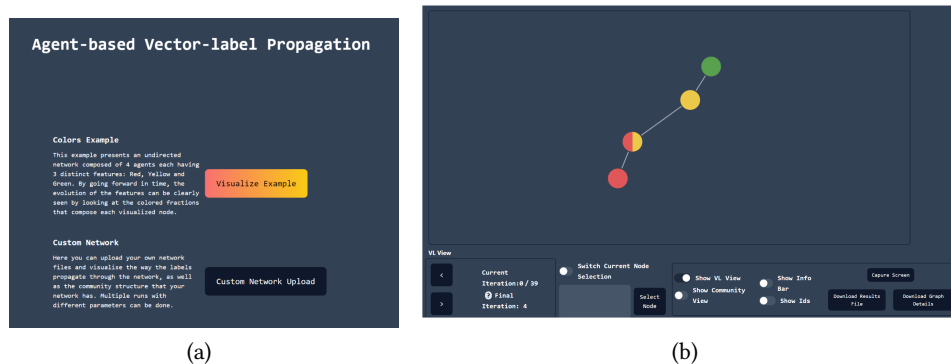


Figure 1: The screenshots of the application, in a) the main page and in b) the page with the control bar are presented respectively.

Colors Example can be reached by pressing the "Visualize Example" button next to "Colors Example" on the main page (Figure 1. a).

The control bar contains all the modes that can be used to control the network display and the iteration. It also contains the node selection mode and displays layout and some buttons for capturing the configuration of the interface, downloading the algorithm results file, downloading the file containing the graph details, and downloading a file containing the comparison of two selected nodes (Figure 1. b).

The VL View represents the view that visualizes the network from the perspective of the label vectors. All coefficient belongings sum to 1 and thus each label is represented by a color that occupies the specific percentage on the node (Fig. 2. a).

⁵<https://www.typescriptlang.org/>

⁶<https://nextjs.org/learn/foundations/about-nextjs/what-is-nextjs>

⁷<https://d3js.org/>

⁸<https://www.npmjs.com/package/katex>

⁹<https://tailwindcss.com/>

¹⁰<https://flask.palletsprojects.com/en/2.1.x/>

¹¹<https://www.sympy.org/en/index.html>

¹²<https://cdlib.readthedocs.io/en/latest/>

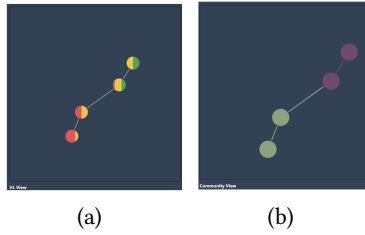


Figure 2: The possible view of the web application, a) the VL view and b) the community view.



Figure 3: The information bar extracted for nodes of example network at timestamp = 2.

The Community View represents the view that visualizes the network from the perspective of the communities that make up the network (Fig. 2. b).

The Information Bar displays general information regarding the network, such as network type, number of nodes, number of edges, and diameter. Once a node is selected, it displays the degree, the community belonging to it, its color, and the vector of labels with the belonging coefficients of each label (Figure 3).

Taking advantage of **Custom Social Networks Upload Page** feature, the user can upload the datasets from the main page, view AVPRA results, and customize several options.

As a case scenario, we have used the POPULITE (POPulist Language in ITALian political Elites) dataset in which each node represents a politician who belongs to one party. Implementing the AVPRA on this data set, Figure 4 demonstrates nodes aggregate into different communities. The network diameter of this dataset is 5 equal to the number of iterations needed to arrive at the stationary state. A neglect threshold of 0.1 is used in the presented case [4].

3. Conclusion

The **Social Networks Viewer** for AVPRA is a web application that allows the user to run and view the results of the algorithm in an interactive way in the web browser. It is capable of receiving many parameters permitting the user to control the behavior of the algorithm and input. Moreover, through this application is possible to export the network and present the information on the screen in PNG format and in JSON format. Individual nodes can be selected in order to compare and export more details and measurements regarding that specific node even between two different nodes in the network. It supports the display in *Vector Label View*

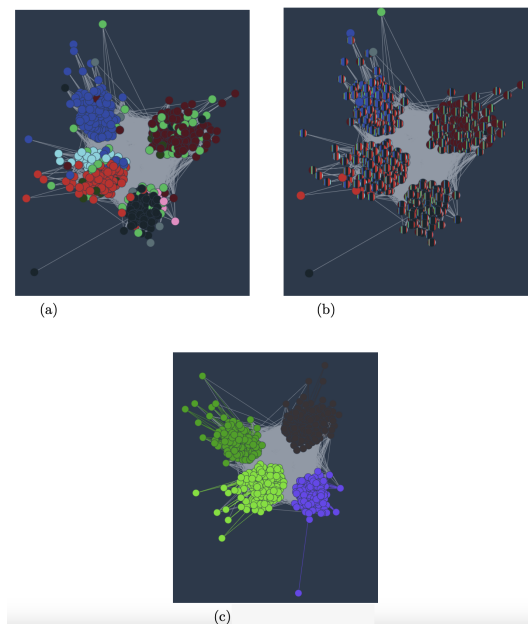


Figure 4: Community representation of the POPULITE dataset using the AVPRA implemented by the web application in different iterations.

and *Community View*. Through the application, it is possible to monitor the iteration's outcome. The color palette is also changeable using an additional file in the loading phase of the network. Besides the capabilities of this platform, it also has some limitations such as keeping the user datasets and the results of algorithm executions on the application server, albeit for a limited time. Also, low performance while using very large networks.

References

- [1] S. Gregory, Finding overlapping communities in networks by label propagation, *New journal of Physics* 12 (2010) 103018.
- [2] U. N. Raghavan, R. Albert, S. Kumara, Near linear time algorithm to detect community structures in large-scale networks, *Physical review E* 76 (2007) 036106.
- [3] V. Bellandi, P. Ceravolo, E. Damiani, S. Maghool, Agent-based vector-label propagation for explaining social network structures, in: *International Conference on Knowledge Management in Organizations*, Springer, 2022, pp. 306–317.
- [4] V. Bellandi, E. Damiani, V. Ghirimoldi, S. Maghool, F. Negri, Validating vector-label propagation for graph embedding, in: M. Sellami, P. Ceravolo, H. A. Reijers, W. Gaaloul, H. Panetto (Eds.), *Cooperative Information Systems*, Springer International Publishing, Cham, 2022, pp. 259–276.

4. Online Resources

Github: <https://github.com/georgiani/AVPRANetworkVisualizer/tree/main/Frontend>