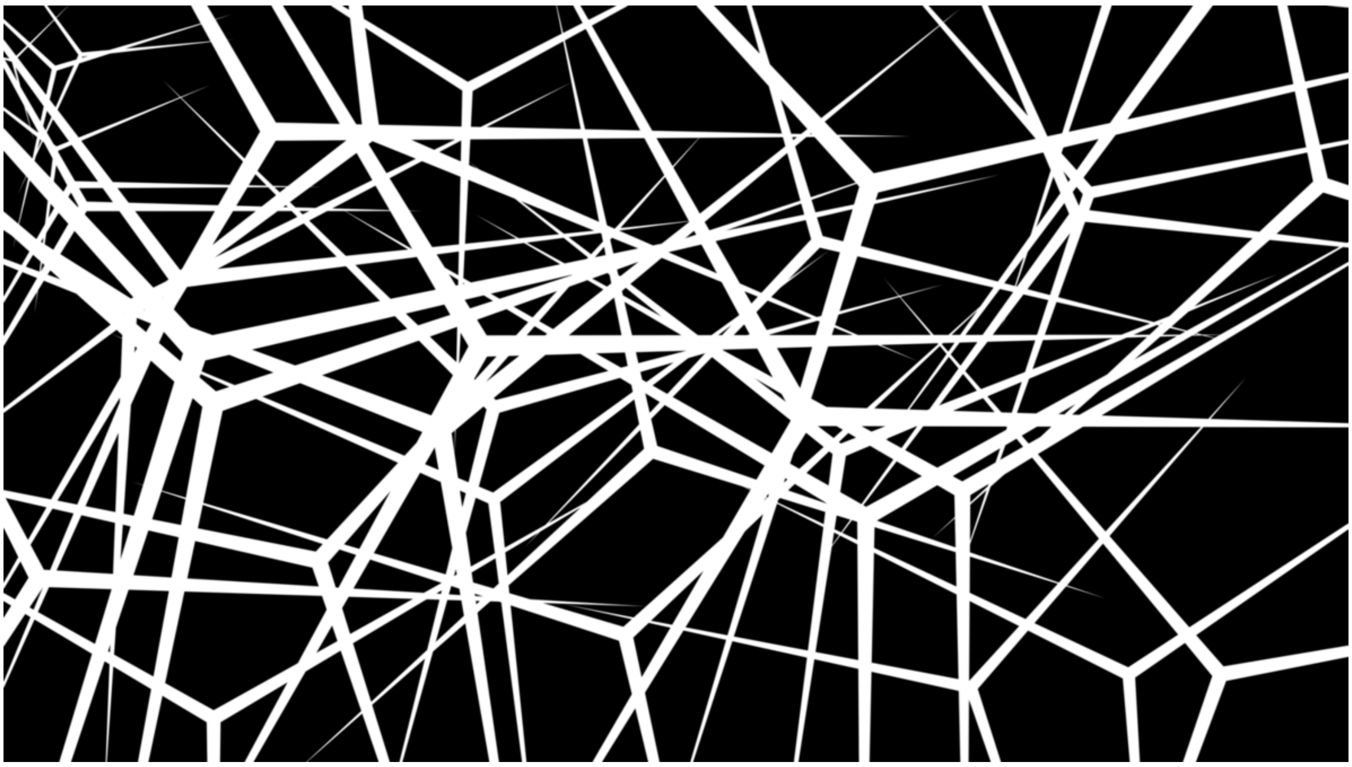
Introduction to Label Propagation with NetworkX — Part 1

Graph ML
Sep 7, 2018 · 7 min read ★



Networks tell us a lot. One interesting phenomenon that networks illustrate is that similar objects are likely to form a connection to each other. This indeed can be seen in our social network. Two individuals who have similar hobbies are likely to become friends with each other. This observation gives us an ability to predict one's hobby based on the friendship relationships in our social network. It's safe to say that a person who has a lot of friends who play tennis is likely to play tennis, isn't it?

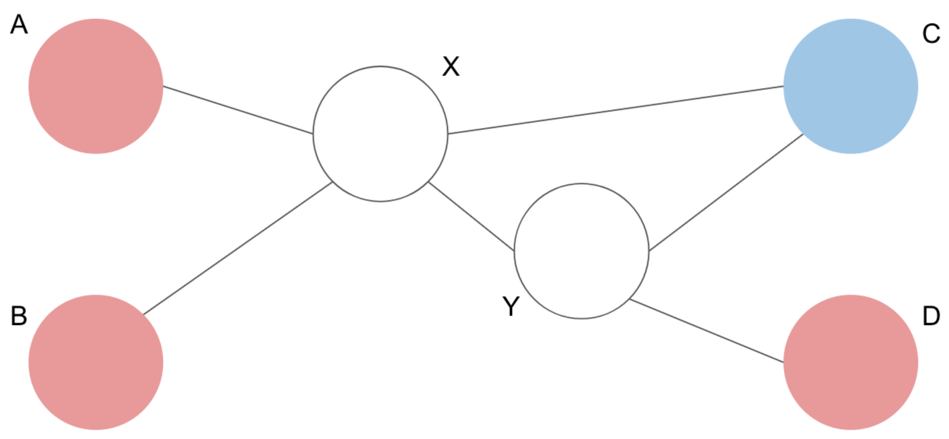
In this series, you will learn a machine learning algorithm called **Label Propagation**. This algorithm does the above prediction in more mathematical way. This post is Part1 of the series that shows you the basic concept of Label Propagation. In Part2, we will try Label Propagation using a Python library called NetworkX, which is one of the famous Python libraries for manipulating network data.

Introduction to Label Propagation with NetworkX — Part 1 (this post)

Introduction to Label Propagation with NetworkX — Part 2

. . .

Let’s begin with the definition of the problem we are trying to solve. This problem is called **Node Classification**. In this problem, we are given a network that is composed of a set of nodes and a set of edges. The important aspect of this problem is that some part of nodes are given their **labels**. Let’s illustrate the ingredients we are given:



We are given this.

Some nodes (A, B, C, and D) are colored red or blue, whereas some (X and Y) are not. Let’s suppose that the color represents the label. What we want to do here is to predict the colors (labels) of the unlabeled nodes (X and Y). This is the problem that Label Propagation can solve.

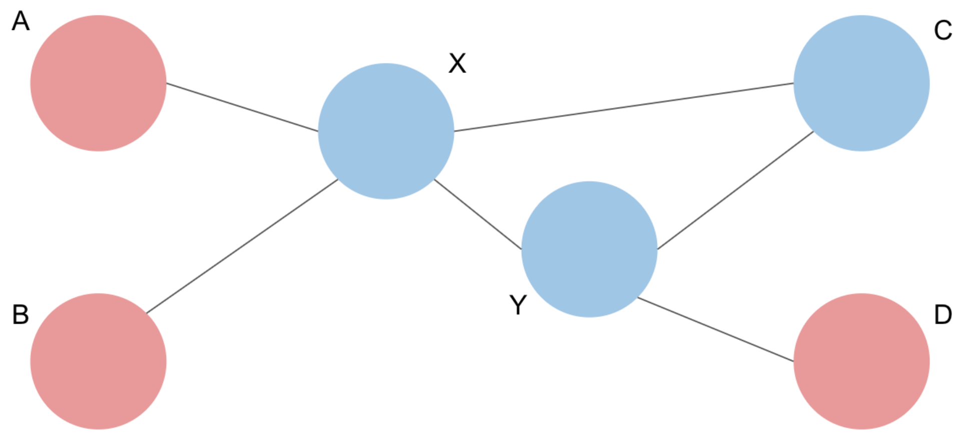
So how can we solve this problem? Label Propagation tries to find the optimal coloring that satisfies the following two constraints:

Constraint1: A node that is given its label must not change its label.

Constraint2: A pair of connected nodes should have the same labels.

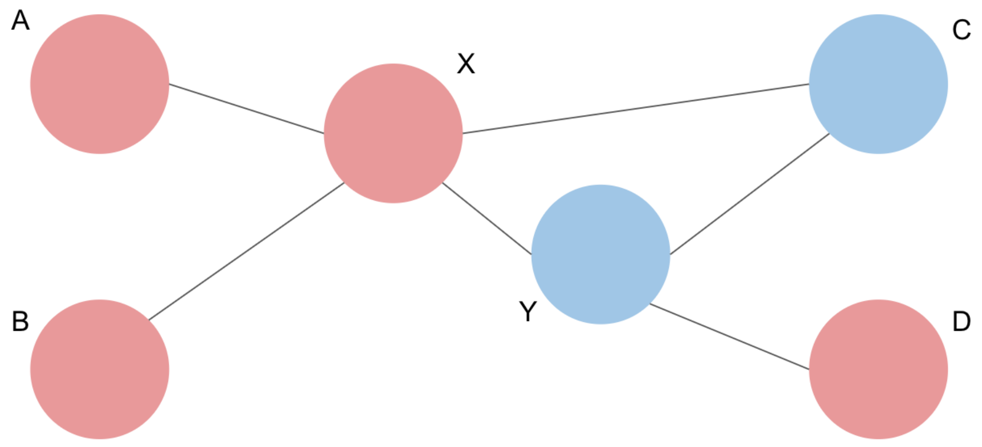
Constraint1 is simple. It just states that blue is blue, red is red. The colors given must not change in the output of Label Propagation. Constraint2 is the core of this algorithm. If a node has a connection with another node whose color is red, the first node should also be red. Note that the first constraint is “must”, but the second constraint is “should”. So we will find the coloring that **minimizes the number of violations of Constraint2** without no violations of Constraint1.

OK, let’s find the optimal coloring for the above example. Since it’s your first trial, let’s take time to try all the cases of coloring. First, we’ll try painting both unlabeled nodes blue (shown below).



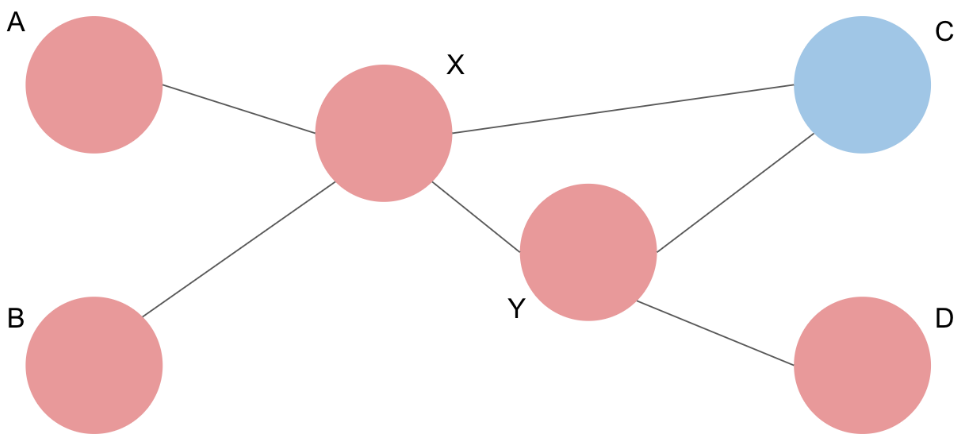
(X = blue and Y = blue) → 3 violations!

Once we paint all unlabeled nodes, it’s easy to count the number of violations of Constraint2. As shown in the above figure, the number of violations is 3. OK, it seems a good starting point. Let’s try different painting.



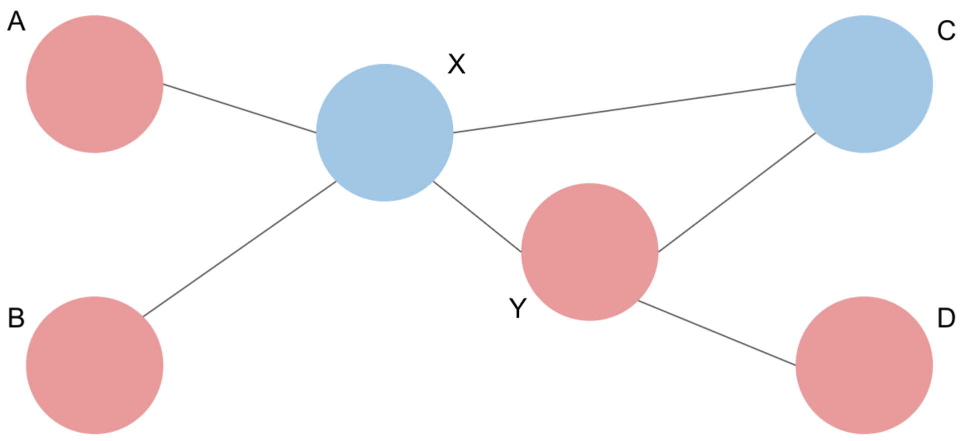
(X = red and Y = blue) → 3 violations!

This time, we painted X red, and Y blue. The number of violations in this case is also 3. The violations on the connections (A, X) and (B, X) are resolved, but new violations on (X, C) and (X, Y) are introduced. OK, let’s try the next coloring.



(X = red and Y = red) → 2 violations!

Now we painted both unlabeled nodes red as shown in the above figure. Wow, the number of violations is now 2. The result improved! Before concluding this is the optimal solution, let's try the last combination of coloring.



(X = blue and Y = red) → 4 violations!

As you might guess, we got worse. Now we have 4 violations.

OK, as a result, the optimal coloring can be obtained by the third trial (X = red, Y = red), which is exactly the output of Label Propagation! It's really simple, isn't it?

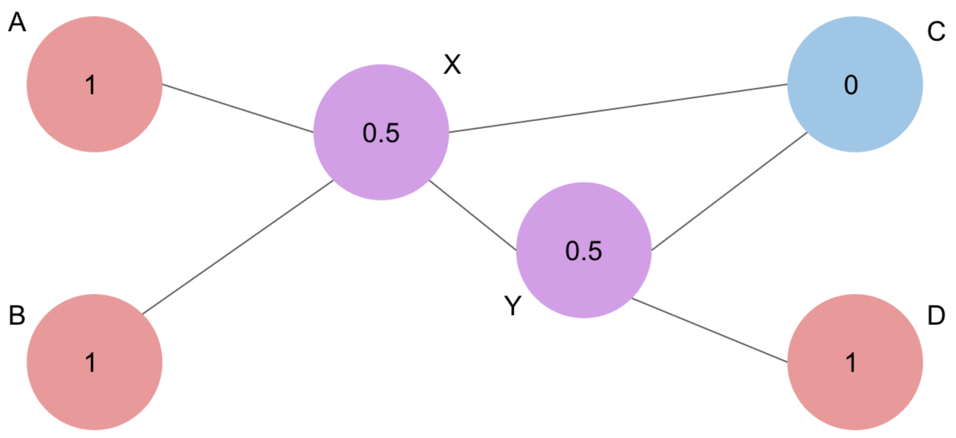
. . .

Is this the end of the story? — Wait! One question naturally comes up. Do we need to try all the combinations? What if the number of unlabeled nodes is, let's say, a million? It's obviously impossible to try out all the possible colorings. How can we obtain the optimal solution then? Unfortunately, it can be proved that we have to try all the combinations to find the optimal solution.

Does that mean Label Propagation is a garbage? Definitely not. Label Propagation can get an approximated solution, which is a reasonable, and a good enough solution in most cases. Let's see how to obtain the approximated solution.

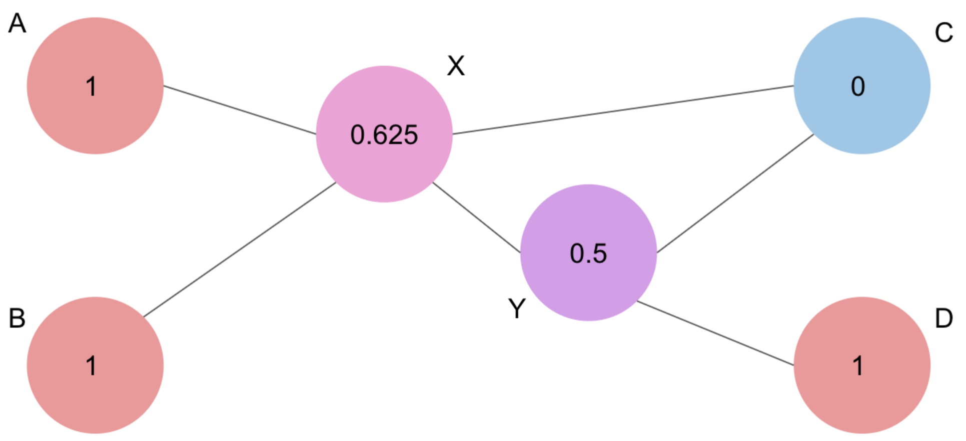
Label Propagation considers that the labeled nodes are given the number 0 or 1. The red labeled node is given 1, and the blue labeled node is given 0. So here the number indicates the color. Let's say it's "redness". In other words, the number close to 1 represents the color close to red. The goal here is to assign the "optimal" numbers from 0 to 1 to unlabeled nodes. If an unlabeled node is assigned a number close to 1 (like 0.78), which means it should be colored red!

To obtain those numbers, Label Propagation "propagates" the numbers from node to node along the edges. This is why this algorithm is called Label Propagation. Let's see what Label Propagation does step by step.



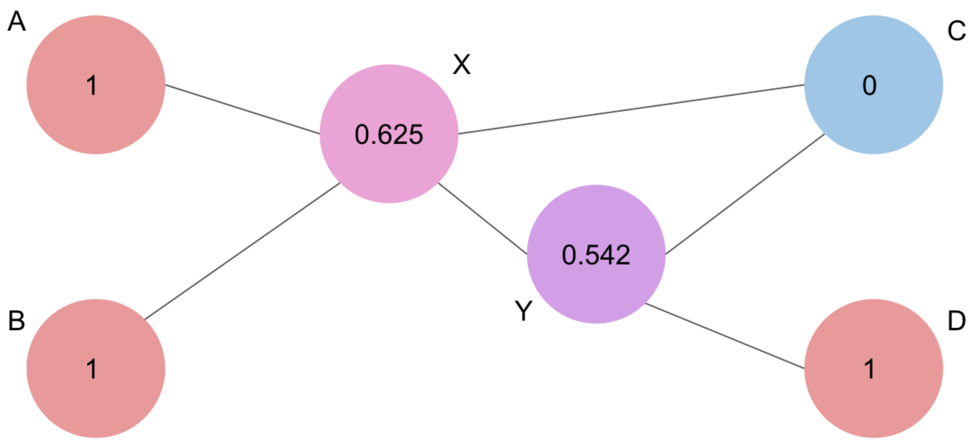
The labeled nodes are given 0 or 1. The unlabeled nodes are given 0.5 because we know nothing on their color.

The above figure illustrates the initial condition. The labeled nodes are given numbers 0 or 1. The unlabeled nodes are given 0.5 because we know nothing on their color. Label Propagation propagates the numbers along the edges, and updates the numbers on the unlabeled nodes by **taking the average of the propagated numbers from the neighbors**. The following figure shows what happens after one step performed.



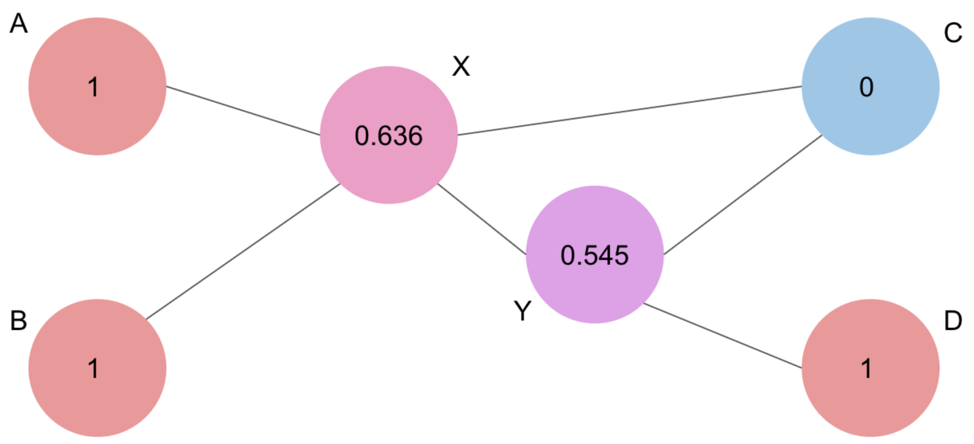
One step performed.

Node X got the number 0.625, because the number 1 is propagated from A and B, 0 is from C, and 0.5 is from Y. Node Y got the number 0.5, because 0 is propagated from C, 1 is from D, and 0.5 is from X. OK, let's perform one more step and see what happens.



One more step performed.

The number on X did not change because the numbers of it's neighbor are the same in the previous step. The number on Y increased from 0.5 to 0.542 because the number propagated from X increased. Label Propagation iterates this step again and again, and the numbers on the unlabeled nodes will be converged to some fixed point. Let's see the final output after a certain number of iterations.



The final result after a certain number of iterations.

In the final result above, both unlabeled nodes are assigned the numbers larger than 0.5, which means that they are more like red than blue. Therefore, Label Propagation concludes that the labels of the unlabeled nodes are both red. It aligns with the optimal result we got before with the brute force solution!

The pros and cons of Label Propagation are:

Pros: it can obtain a reasonable result without trying all the combinations of coloring, which can be billions or even trillions if there are a lot of unlabeled nodes.

Pros: we can tell the "redness" of the unlabeled nodes. In the result above, X is more red than Y. This indicates that we can know the degree of certainty of our prediction.

Pros: it's mathematically proved that the steps of Label Propagation converges regardless of the initial numbers assigned to unlabeled nodes (it was 0.5 in our example). It's practically important because we don't want to use such an algorithm that is not proved to stop.

Cons: The solution of Label Propagation can be different from the optimal solution because it's an approximated solution. But it's not necessarily bad. Sometimes, the approximated solution can be "practically better" than the optimal.

. . .

This is all the basic concept of Label Propagation you need to learn. Although it's not converted in this post, **we can deal with 3 or more labels instead of just 2.** You need a bit of math to see why it can obtain a reasonable result in such a beautiful way. Let’s leave it for more advanced articles. Interested readers can jump into the reference listed up below. OK, next up is, coding with a Python library, NetworkX.

Reference

Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions: this is the paper that first proposed the algorithm that we studied in this post.

Introduction to Semi-Supervised Learning: this is a book explaining Semi-Supervised Learning, which is a category of machine learning algorithms. Label Propagation falls into this category.

Machine Learning Label Propagation Networks Graphs Python