A graph matching method based on leading Eigenvector and Sinkhorn-Knopp algorithm

Romain Raveaux[1]

[1]Université de Tours, Laboratoire d'Informatique Fondamentale et Appliquée de Tours (LIFAT - EA 6300), 64 Avenue Jean Portalis, 37000 Tours, France

October 27, 2021

Abstract

The goal of the report is to present a graph matching method based on the leading Eigenvector and Sinkhorn-Knopp algorithm. This method is not new and is reported in [6].

1 Introduction

The goal of the report is to present a graph matching method based on the leading Eigenvector and Sinkhorn-Knopp algorithm. A tutorial and codes are provided on :
https://gitlab.com/romain.raveaux/a-graph-matching-method-based-on-the-leading-eigen-vector

2 The graph matching problem

The objective of graph matching is to find correspondences between two attributed graphs G_1 and G_2. A solution of graph matching is defined as a subset of possible correspondences $\mathcal{Y} \subseteq V_1 \times V_2$, represented by a binary assignment matrix $Y \in \{0,1\}^{n1 \times n2}$, where $n1$ and $n2$ denote the number of nodes in G_1 and G_2, respectively. If $u_i \in V_1$ matches $v_k \in V_2$, then $Y_{i,k} = 1$, and $Y_{i,k} = 0$ otherwise. We denote by $y \in \{0,1\}^{n1.n2}$, a column-wise vectorized replica of Y. With this notation, graph matching problems can be expressed as finding the assignment vector y^* that maximizes a score function $S(G_1, G_2, y)$ as follows:

Problem 1. *Graph matching model (GMM)*

$$y^* = \underset{y}{\text{argmax}} \quad S(G_1, G_2, y) \tag{1a}$$

$$\textit{subject to} \quad y \in \{0,1\}^{n1.n2} \tag{1b}$$

$$\sum_{i=1}^{n1} y_{i,k} \leq 1 \quad \forall k \in [1, \cdots, n2] \tag{1c}$$

$$\sum_{k=1}^{n2} y_{i,k} \leq 1 \quad \forall i \in [1, \cdots, n1] \tag{1d}$$

where equations (1c),(1d) induces the matching constraints, thus making y an assignment vector.

The function $S(G_1, G_2, y)$ measures the similarity of graph attributes, and is typically decomposed into a first order dissimilarity function $s(u_i, v_k)$ for a node pair $u_i \in V_1$ and $v_k \in V_2$, and a second-order similarity function $s(e_{ij}, e_{kl})$ for an edge pair $e_{ij} \in E_1$ and $e_{kl} \in E_2$. Thus, the

objective function of graph matching is defined as:

$$\begin{aligned} S(G_1, G_2, y) =& \sum_{i=1}^{n1}\sum_{k=1}^{n2} s(u_i, v_k)\cdot y_{i,k} + \sum_{i=1}^{n1}\sum_{j=1}^{n1}\sum_{k=1}^{n2}\sum_{l=1}^{n2} s(e_{ij}, e_{kl})\cdot y_{ik}\cdot y_{jl} \\ =& \sum_{i=1}^{n1}\sum_{j=1}^{n1}\sum_{k=1}^{n2}\sum_{l=1}^{n2} K_{ik,jl}\cdot y_{i,k}\cdot y_{j,l} \\ =& y^T K y \end{aligned} \tag{2}$$

Similarity functions are usually represented by a symmetric similarity matrix $K \in \mathbb{R}^{n1n2\times n1n2}$. A non-diagonal element $K_{ik,jl} = s(e_{ij}, e_{kl})$ contains the edge similarity and a diagonal term $K_{ik,ik} = s(u_i, v_k)$ represents the vertex similarity. K is called an affinity matrix or compatibility matrix. In this representation, $K_{ij,kl} = 0$ means an impossible matching or a very dissimilar matching. In essence, the score accumulates all the similarity values that are relevant to the assignment. The formulation in Problem 1 is referred to as an integer quadratic programming. This integer quadratic programming expresses the quadratic assignment problem, which is known to be NP-hard [1]. This model was proposed in [3].

3 Relaxed graph matching problems

The NP-hard graph matching problem is relaxed by dropping both the binary and the mapping constraints. The quadratic assignment problem is relaxed to the continuous domain and the matching constraints are replaced by a ball constraint forcing the solution to be at the surface a ball of radius one.

The model to be solved is then:

Problem 2. *Graph Matching Relaxed by L2-norm*

$$y^* = \underset{y}{\text{argmax}} \quad y^T K y \tag{3a}$$

$$\textit{subject to} \quad y \in [0,1]^{|V_1|\cdot|V_2|} \tag{3b}$$

$$\|y\|_2 = 1 \tag{3c}$$

4 Relation between Graph Matching Relaxed by L2-norm and Lagrangian relaxation

4.1 Lagrangian relaxation problem

Lagrangian relaxation is a method to solve optimization problem under constraints. The main idea is to inject the constraints into the objective function as penalties. The penalties increase as the solution breaks the constraints. The penalties are proportional to the amount of violation of the constraints.

Our Problem 2 becomes :

Problem 3. *Lagrangian relaxation of the Relaxed Graph Matching*

$$y^* = \underset{y\in[0,1]^{|V_1|\cdot|V_2|}}{\text{argmax}} \quad y^T K y + \lambda(1 - \|y\|_2) \tag{4a}$$

The objective function is composed of two terms. 1) The score of the graph matching $y^T K y$ and 2) The penalty $(1 - \|y\|_2)$ that gauges the gap between the magnitude of y and 1. $\lambda \in \mathbb{R}$ is a Lagrange multipliers connecting/weighting the two terms. Let us denote by l_1 and l_2 the two terms such that $l_1 = y^T K y$ and $l_2 = 1 - \|y\|_2$. Let us denote by $L(\lambda, y)$ the function to be maximized:

$$L(\lambda, y) = l_1(y) + \lambda l_2(y) = y^T K y + \lambda(1 - \|y\|_2 \tag{5}$$

4.2 Solving the Lagrangian relaxation

To solve Problem 3, we need to compute where the derivative of L with respect y equals 0.

$$\frac{\partial L(y)}{\partial y} = 0 \tag{6}$$

Let us calculate the gradient of L:

$$\frac{\partial L(y)}{\partial y} = 2K.y + \lambda(-2y) \tag{7}$$

Let us set $\frac{\partial L(y)}{\partial y}$ to zero and isolate λ.

$$\frac{\partial L(y)}{\partial y} = 0 \tag{8a}$$
$$2K.y + \lambda(-2y) = 0 \tag{8b}$$
$$2K.y - 2\lambda y = 0 \tag{8c}$$
$$K.y - \lambda y = 0 \tag{8d}$$
$$K.y = \lambda y \tag{8e}$$

The equation 8e is exactly the definition of an eigenvector of the matrix K. So y is an eigenvector and λ is an eigenvalue.

4.3 Which eigenvector is y ?

Let us notice that $\|y\|_2 = y^T y = 1$. Now let us multiply the equation 8e by y^T.

$$y^T.K.y = \lambda y^T.y \tag{9a}$$
$$y^T.K.y = \lambda \tag{9b}$$

The left part of the equation 9b is exactly our graph matching objective function (Equation 2). The right part of the equation 9b is just λ. We want to maximise our objective function so we want λ with the maximum value. λ is an eigenvalue so we want the principal eigenvector. The eigenvector that corresponds the largest eigenvalue (λ).

y is the leading eigenvector of matrix K.

5 Solving the relaxed graph matching problem

The optimal y^* of Problem 2 is then given by the leading eigenvector of the matrix K. Therefore, the eigenvalues of K are values of λ that satisfy the equation:

5.1 Computing the Eigenvectors

First, we need to find the eigenvalues $\lambda \in \mathbb{R}$ such that :

$$det(K - \lambda I) = 0 \tag{10}$$

I is the identity matrix. $det()$ is a function computing the determinant of a matrix. There is not only one values that satisfies the Equation 10. So potential values of λ are denoted λ_i. $P = det(K - \lambda I)$ is a polynomial of the variable λ. Setting the polynomial (P) equal to zero: $P = 0$, we need to find values of λ that set P to zero. Such values are called root of the polynomial. The roots of the polynomial at $\lambda_i \quad \forall i \in [1, \cdots, n1.n2]$. λ_i are the eigenvalues of K. The eigenvector corresponding to each eigenvalue can be found by solving for the components of m_{λ_i} in the equation:

$$(K - \lambda I)m_{\lambda_i} = 0 \quad \forall i \in [1, \cdots, n1.n2] \tag{11}$$

The eigenvector associated to the highest eigenvalue is the solution of the Problem 2.

$$k^* = arg \max_{k \in [1, \cdots, n1.n2]} \lambda_k \tag{12}$$

$$y^* = m_{\lambda_{k^*}} \tag{13}$$

5.2 Power iteration

Computing the leading eigenvector y^* of the affinity matrix K can be approximated by using power iterations.

$$m^{(k+1)} = \frac{Km^{(k)}}{\|Km^{(k)}\|_2} \tag{14}$$

$m_0 = 1$ is initialize to 1. N iterations are run to output the vector $y^* = m^{(N)}$.

The time complexity of this algorithm per power iteration is $O(n1^2.n2^2)$ when the matrix K is dense. The power iteration algorithm is presented in [6].

6 Refined the relaxed graph matching problem by adding of the one-to-one mapping constraints

The solution y^* of the relaxed graph matching problem (Problem 2) is reshaped to become a matrix $Y^* \in \mathbb{R}^{n1 \times n2}$. Then, the matrix Y^* is modified to become a doubly stochastic matrix. In the graph matching context, this modification represents the addition of the one-to-one mapping constraints (L1 constraints) $\forall k, \sum_i Y_{i,k} = 1$ and $\forall i, \sum_k Y_{i,k} = 1$. To address this problem the let us first define the Kantorovitch problem that contains such one-to-one mapping constraints.

6.1 Kantorovitch Problem (KP)

$X = \{x_i\}_{i=1}^{|X|}$ and $Z = \{z_j\}_{j=1}^{|Z|}$ are two sets with x_i and $z_j \in \mathbb{R}^d$. Associated with these two sets, we denote by $Pr(x_i) = a_i$ and $Pr(z_j) = b_j$. $Pr(x_i)$ is the probability to observe x_i. A solution of KP is matrix $\mathbf{P} \in \mathbb{R}_+^{|X| \times |Z|}$. We denote by $\mathbf{p} \in \mathbb{R}_+^{|X|.|Z|}$, a column-wise vectorized replica of $\mathbf{P}$. The KP is defined as follows :

Problem 4. *Kantorovitch Problem (KP)*

$$\hat{\boldsymbol{p}} = \underset{\boldsymbol{p}}{\operatorname{argmin}} \sum_{x_i \in X} \sum_{z_j \in Z} c(x_i, z_j) \cdot \boldsymbol{p}_{i,j} \tag{15a}$$

$$\textit{subject to} \quad \boldsymbol{p} \in \mathbb{R}_+^{|X|.|Y|} \tag{15b}$$

$$\sum_{a_i \in X} \boldsymbol{p}_{i,j} = b_j \quad \forall b_j \in Z \tag{15c}$$

$$\sum_{b_j \in Z} \boldsymbol{p}_{i,j} = a_i \quad \forall a_i \in X \tag{15d}$$

$$\sum_{a_i \in X} a_i = \sum_{b_j \in Z} b_j \tag{15e}$$

There exists a solution to KP only if $\sum_{a_i \in X} a_i = \sum_{b_j \in Z} b_j$.
Let us define a special case of KP where:

1. $|X| = |Z| = N$
2. $a_i = b_j = 1 \quad \forall a_i \in X \quad \forall b_j \in Z.$

Now we define the Kantorovitch Problem in this special case.

Problem 5. *Special Case Kantorovitch Problem (SCKP)*

$$\hat{\boldsymbol{p}} = \underset{\boldsymbol{p}}{\operatorname{argmin}} \sum_{a_i \in X} \sum_{b_j \in Z} c(x_i, z_j) \cdot \boldsymbol{p}_{i,j} \tag{16a}$$

$$\textit{subject to} \quad \boldsymbol{p} \in \mathbb{R}_+^{|X|.|Z|} \tag{16b}$$

$$\sum_{a_i \in X} \boldsymbol{p}_{i,j} = 1 \quad \forall b_j \in Z \tag{16c}$$

$$\sum_{b_j \in Z} \boldsymbol{p}_{i,j} = 1 \quad \forall a_i \in X \tag{16d}$$

$\mathbf{P}$ is a doubly-stochastic matrix. $\mathbf{P}$ is positive and rows and columns sum to one. $\mathcal{P}$ is the set of all doubly-stochastic matrices. $\mathbf{P} \in \mathbb{R}_+^{N \times N}$ is doubly-stochastic matrix where rows and columns sum to one consequently any value $\mathbf{P}_{i,j}$ must be smaller or equal to one. Therefore the domain of $\mathbf{P}$ can be refined to $\mathbf{P} \in [0,1]^{N \times N}$.

The Sinkhorn algorithm [5, 2] is fast heuristic for the KP. Even though the original algorithm assumes only square matrices, the process can be generalized as shown in [4]. Minimizing a cost function C is equivalent to maximize the function $-C$ that is equivalent to maximize the function S.

6.2 Sinkhorn algorithm

The transformation of the matrix Y^* to a doubly stochastic matrix is performed by the Sinkhorn-Knopp algorithm. It is an iterative algorithm. At each iteration the algorithm normalizes the rows of Y^* and then the columns of Y^*. Starting with $M^{(0)} = Y^*$. The algorithm goes as follows:

$$M^{(k+1)} = M_k[1_{n1}^T M^{(k)}]^{-1}$$

$$M^{(k+2)} = [M^{(k+1)} 1_{n2}]^{-1} M^{(k+1)}$$

$$M_{i,j}^{(k+1)} = \frac{M_{i,j}^{(k)}}{\sum_j M_{i,j}^{(k)}} \quad \forall i \in [1, \cdots, n1]$$

$$M_{i,j}^{(k+2)} = \frac{M_{i,j}^{(k+1)}}{\sum_i M_{i,j}^{(k+1)}} \quad \forall j \in [1, \cdots, n2]$$

With : $1_{n1} \in 1^{n1 \times 1}$ and $1_{n2} \in 1^{n2 \times 1}$. The Sinkhorn algorithm is presented in [6].

References

[1] Mohammad Abdulkader Abdulrahim. *Parallel Algorithms for Labeled Graph Matching*. PhD thesis, USA, 1998.

[2] Jason Altschuler, Jonathan Weed, and Philippe Rigollet. Near-linear time approximation algorithms for optimal transport via sinkhorn iteration. *CoRR*, abs/1705.09634, 2017.

[3] M. Cho, K. Alahari, and J. Ponce. Learning graphs to match. In *International Conference on Computer Vision*, 2013.

[4] Timothée Cour, Praveen Srinivasan, and Jianbo Shi. Balanced graph matching. In *NIPS*, 2006.

[5] Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific Journal of Mathematics*, 21:343–348, 1967.

[6] Andrei Zanfir and Cristian Sminchisescu. Deep learning of graph matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.