Graph Clustering with Belief Propagation

Wei Dong

July 8, 2013

1 Introduction

1.1 Exemplar-Based Graph Clustering

Given a set of points V equipped with a distance metric $d : V \times V \to \mathbb{R}$. An exemplar-based clustering algorithm tries to find a subset C of V as cluster centers, so that the sum of distance between every point and its nearest cluster center is minimized:

$$\max \quad -\sum_{v \in V} d(v, C).$$

Without any constraints, the minimum can be reached when $C = V$, which does not provide any useful information. Typically, either of the following formulation are used to control the granularity of clustering:

- Constraint on cluster size.

  $$\begin{aligned} \max \quad & -\sum_{v \in V} d(v, C), \\ \text{s.t.} \quad & |C| = k. \end{aligned}$$

  A typical example is the k-medoid algorithm.

- Regulation

  $$\max \quad -\sum_{v \in V} d(v, C) - \sum_{c \in C} p(c). \tag{1}$$

  where $p(v)$ is the penalty if vertex v is chosen as a cluster center. We use the regulation approach in this paper. Affinity-propagation falls in this category.

In this paper, we are mostly interested in clustering the vertices of a graph. In such a case, the graph distance metric is used, where the distance between two vertices is given by the summation of the weights of edges along the shortest path connecting the two vertices.

1.2 Affinity Propagation

Affinity Propagation (AP) is a clustering algorithm based on belief propagation that has been shown to be more accurate than the standard k-medoid algorithm, and has seen many applications recently. The problem of AP is that it works better with a complete graph, costs $O(n^2)$ time in each iteration. Such complexity is not tolerable for very large datasets. AP requires that each vertex is directly connected to its exemplar, so when the graph is sparse, many small clusters will be created. An extreme example is to cluster vertices on a linear list of size n: at least $\lceil n/3 \rceil$ clusters will be generated by AP.

In this paper, we are interested in developing a generalization to the AP algorithm to allow clustering sparse graphs at coarse granularities.

2 Background

2.1 Factor Graph

A factor graph is a bipartite graph that represents a function that can be factorized. Given a function $g : X_1 \times X_2 \times \cdots \times X_n \to \mathbb{R}$ that can be factorized[1]:

$$g(x_1, x_2, \cdots, x_n) = \sum_{j=1}^{m} f_j(S_j),$$

where $S_j \subset \{x_1, x_2, \cdots, x_n\}$, the corresponding factor graph is $\langle X, F, E \rangle$, where $X = \{x_1, x_2, \cdots x_n\}$, $F = \{f_1, f_2, \cdots, f_m\}$ and $E = \bigcup_{j=1}^{m} S_j \times \{f_j\}$.

2.2 Belief Propagation

Belief propagation is a randomized algorithm to maximize the target function g by passing messages along the edges of the corresponding factor graph. A message passed between a variable node x_i and a factor node f_j, in either direction, is a real-valued function $X_i \to \mathbb{R}$.

A message from variable x_i to factor f_j is

$$\mu_{i \to j}(x_i) = \sum_{j' \in N(i) \setminus \{j\}} \mu_{j' \to i}(x_i).$$

A message from factor f_j to variable x_i is

$$\mu_{j \to i}(x_i) = \max_{S_j \setminus \{x_i\}} \left\{ f_j(S_j) + \sum_{i' \in N(j) \setminus \{i\}} \mu_{i' \to j}(x_{i'}) \right\}.$$

[1] We work in the logarithm domain, so all products are converted to summations.

3 The Proposed Algorithm

3.1 Factor Graph Formulation

Given an undirected input graph $G = \langle V, E\rangle$, the clustering result is a directed acyclic graph $\mathcal{C}(G) = \langle V, A\rangle$, where A is obtained by setting directions to a subset of E, such that each vertex has either zero or one outgoing edge. The clustering affiliation can be read off $\mathcal{C}(G)$ using the following two rules:

- If there is no outgoing edges from a vertex v, then v is a cluster center;
- Each directed edge $\langle u, v\rangle \in A$ means that u and v belong to the same cluster.

we convert G into an equivalent factor graph using the following protocol:

1. Making G a directional graph by assign an arbitrary direction to each edge $e_i \in E$.
2. Create a variable node $x_i \in \mathbb{Z}$ for each edge e_i.
3. Create a factor node w_i for each variable.
4. Create a factor node f_j for each node $v_j \in V$.
5. For each edge $e_i = \langle v_j, v'_j\rangle$ in G, with direction set in step 1, add two weighted edges to the factor graph: $\langle x_i, f_j, s_{ij} = -1\rangle$ and $\langle x_i, f_{j'}, s_{ij} = +1\rangle$. Note that the two edges incident to a variable node always have the opposite signs.

The factor w_i is

$$w_i(x_i) = |x_i| \times w(e_i),$$

where $w(e_i)$ is the weight of the edge e_i in the input graph.

For each factor f_j, we partition its neighboring variables $S_j = \{x_i\}$ into three sets according to their signs and the signs, s_{ij}, of the edge:

$$\begin{aligned} S_j^+ &= \{x_i \,|\, x_i \times s_{ij} > 0\}, \\ S_j^0 &= \{x_i \,|\, x_i \times s_{ij} = 0\}, \\ S_j^- &= \{x_i \,|\, x_i \times s_{ij} < 0\} \end{aligned}$$

The factor function f_j is

$$f_j(S_j) = \begin{cases} p_j & \text{if } |S_j^-| = 0, \\ 0 & \text{if } |S_j^-| = 1 \text{ and } \sum_{x_i \in S_j} x_i s_{ij} w(e_i) = \sum_{x_i \in S_j^-} w(e_i), \\ -\infty & \text{otherwise.} \end{cases}$$

Our task is to infer a value for each variable $x_i \in \mathbb{Z}$ for

$$\max g = \sum_i w_i + \sum_j f_j. \tag{2}$$

We can then construct the clustering graph $\mathcal{C}(G)$ with the result using the following rules:

- If $x_i = 0$, then no edge is added in $\mathcal{C}(G)$ for the edge e_i in the input graph.
- If $x_i > 0$, then an edge is added for e_i, with direction the same as that assigned in Step 1.
- If $x_i < 0$, then an edge is added for e_i, with direction opposite from the assignment of Step 1.

3.2 Proof of Correctness

Here we prove that (1) and (2) are equivalent.

3.3 Belief Propagation

There are two kinds of messages actually passed in the algorithm. The message from factor w_i to variable x_i is always

$$\mu_{w_i \to x_i}(x_i) = |x_i| \times w(e_i),$$

and can be implemented by simply attaching $w(e_i)$ to each variable node x_i.

From factor f_j to variable x_i:

$$\mu_{j \to i}(x_i) = \max_{S_j \setminus \{x_i\}} \left\{ f_j(S_j) + \sum_{x_{i'} \in S_j \setminus \{x_i\}} \mu_{i' \to j}(x_{i'}) \right\}$$

From variable x_i to factor f_j:

We use j_i' to denote the factor on the other end of the edge x_i.

$$\mu_{i \to j}(x_i) = |x_i| \times w(e_i) + \mu_{j_i' \to i}(x_i)$$

We use α_{ij} to denote $\mu_{i \to j}$, and β_{ji} to denote $\mu_{j \to i}$, then

$$\alpha_{ij}(x_i) = |x_i| w(e_i) + \beta_{j_i' i}(x_i)$$

$$\beta_{ji}(x_i) = \max_{S_j \setminus \{x_i\}} \left\{ f_j(S_j) + \sum_{x_{i'} \in S_j \setminus \{x_i\}} \alpha_{i' j}(x_{i'}) \right\}$$

so

$$\beta_{ji}(x_i) = \max_{S_j \setminus \{x_i\}} \left\{ f_j(S_j) + \sum_{x_{i'} \in S_j \setminus \{x_i\}} \left[|x_{i'}| w(e_{i'}) + \beta_{j_{i'}'}(x_{i'}) \right] \right\}$$

Therefore, we have obtained an iteration formula that only involve β.

Now we map this iteration formula back to the original input graph. We use i, j to denote two nodes, x_{ij} the variable corresponding edge $\langle i, j \rangle$, and w_{ij} the weight of that edge.

$$\beta_{ij}(x_{ij}) = \max_{x_{ik}:k \neq j} \left\{ f_i(S_i) + \sum_{x_{ik}:k \neq j} [|x_{ik}| w_{ik} + \beta_{ik}(x_{ik})] \right\}$$

3.4 Relation to Affinity Propagation

When the graph is complete and follows triangle inequality, the proposed algorithm is equivalent to affinity propagation.

4 Evaluation