University of Illinois Fall 2018

ECE 544NA: Pattern Recognition
Lecture 13: Oct 16

Lecturer: Alexander Schwing Scribe: Ao Li

1 Introduction

Structured Prediction is estimating a complex project, i.e., the output stage is really large. Image segmentation, sentence parsing and stereo vision are examples of structured prediction. The inference of structured prediction is to find the most possible configuration, which is the highest scoring configuration. In the last lecture, we talked about 2 ways to implement the structured prediction – Exhaustive Search and Dynamic Programming. However, Exhaustive Search runs quite slow, and Dynamic programming works only for tree structured conditional random field. In this article, we will discuss about several other ways to perform the structured prediction – Integer linear program, Linear programming relaxation, Message passing, Graph-cut [2].

2 Integer linear program

The goal of the inference program is to minimize

$$\mathbf{y}^\star = \arg\max_{\mathbf{y}} \sum_r f_r(\hat{\mathbf{y}}_r) \tag{1}$$

For integer linear program, we introduce several optimization variables and turns (1) into another summation. Taking the following graph as example.

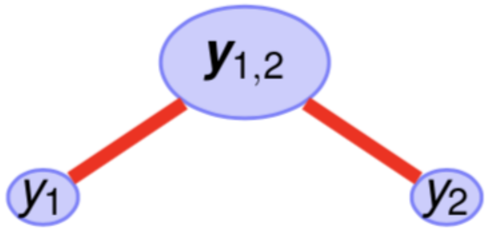


Figure 1: Dependency graph

There are three nodes in the above graph, y_1 y_2 and $y_{1,2}$. The possible values for $y1$ and $y2$ are $\{1, 2\}$ These three nodes stands for the three variables. For each node or variable, we can use several binary variables to show its assigned value. For example, if $b_1(1) = 1$ and $b_1(2) = 0$, then we can say that the variable y_1 is assigned to label 1. Therefore, the whole assignment could be depicted in the following way:

$$\begin{bmatrix} b_1(1) & b_1(2) & b_2(1) & b_2(2) & b_{12}(1,1) & b_{12}(2,1) & b_{12}(1,2) & b_{12}(2,2) \end{bmatrix} \begin{bmatrix} f_1(1) \\ f_1(2) \\ f_2(1) \\ f_2(2) \\ f_{12}(1,1) \\ f_{12}(2,1) \\ f_{12}(1,2) \\ f_{12}(1,2) \end{bmatrix} \tag{2}$$

Our goal is to find the appoporiate b_1, b_2 and b_{12} to solve the above equation. i.e., we need to find b_1, b_2 and $b_{1,2}$ that maximizes

$$\max_{b_1,b_2,b_{12}} \begin{bmatrix} b_1(1) & b_1(2) & b_2(1) & b_2(2) & b_{12}(1,1) & b_{12}(2,1) & b_{12}(1,2) & b_{12}(2,2) \end{bmatrix} \begin{bmatrix} f_1(1) \\ f_1(2) \\ f_2(1) \\ f_2(2) \\ f_{12}(1,1) \\ f_{12}(2,1) \\ f_{12}(1,2) \\ f_{12}(1,2) \end{bmatrix} \tag{3}$$

In the Integer Linear Program Model, the optimization variables b_1, b_2 and b_12 has to follow the following constraints.

$$\begin{cases} b_r(\mathbf{y}_r) \in \{0,1\} \\ \sum_{\mathbf{y}_r} b_r(\mathbf{y}_r) = 1 \\ \sum_{\mathbf{y}_p \setminus \mathbf{y}_r} b_p(\mathbf{y}_p) = b_r(\mathbf{y}_r) \end{cases}$$

The reason we need to ensure $\sum_{\mathbf{y}_p \setminus \mathbf{y}_r} b_p(\mathbf{y}_p) = b_r(\mathbf{y}_r)$ is that we need to ensure the consistency constraints of y_1 and y_2 on variable $y_{1,2}$. In the above example, if we marginize $b_{1,2}$ over $y_1 = 1$, the we must obtain $b_1(1) = b_{1,2}(1,1) + b_{1,2}(1,2)$

The advantage of this method it that it is very slow for large problems. After all, it's a combinational constraints instead of a continuous one.

3 Linear programming relaxation

One way to improve the above integer linear program is through linear programming relaxation. The difference between the Linear programming relaxation and the above formula is that it does not have the constraint $b_r(\mathbf{y}_r) \in \{0,1\}$ By this way, we made b_r the local probability. And we turn the constraints continuous. However, if we apply relaxation, the optimum value of will turn out to be greater than the original one since the constraints are relaxed.

4 Message Passing

Another way to do this is via massage passing. We can just compute the dual of the above problem and then find the maximum value of the dual function. As described above, our optimization problem is like follows:

$$\max_{b} \sum_{r,\mathbf{y}_r} b_r(\mathbf{y}_r) f_r(\mathbf{y}_r) s.t. \begin{cases} b_r(\mathbf{y}_r) >= 0 \\ \sum_{\mathbf{y}_r} b_r(\mathbf{y}_r) = 1 \\ \sum_{\mathbf{y}_p \backslash \mathbf{y}_r} b_p(\mathbf{y}_p) = b_r(\mathbf{y}_r) \end{cases} \tag{4}$$

In order to make each Lagrange multiplier corresponds to the weight of the dependency graph, when we do the Lagrange, we only consider the last constraint, which is $\sum_{\mathbf{y}_p \backslash \mathbf{y}_r} b_p(\mathbf{y}_p) = b_r(\mathbf{y}_r)$. For the rest two, we consider them while maximizing the Lagrangian.

$$L() = \sum_{r,\mathbf{y}_r} b_r(\mathbf{y}_r), f_r(\mathbf{y}_r) + \sum_{r,p \in P(r),\mathbf{y}_r} \lambda_{r \to p}(\mathbf{y}_r)(\sum_{y_p \backslash \mathbf{y}_r} b_p(y_p) - b_r(\mathbf{y}_r)) \tag{5}$$

Using some technique, the above formula could be written like the following. The details about this is in the appendix

$$L() = \sum_{r,\mathbf{y}_r} b_r(\mathbf{y}_r)(f_r(\mathbf{y}_r) - \sum_{p \in P(r)} \lambda_{r \to p}(\mathbf{y}_r) + \sum_{c \in C(r)} \lambda_{c \to r}(\mathbf{y}_c)) \tag{6}$$

Then we just need to maximize the Lagrangian while considering the following constraints, which means we need to

$$\max_{b} L() s.t. \begin{cases} b_r(\mathbf{y}_r) > 0 \\ \sum_{\mathbf{y}_r} b_r(\mathbf{y}_r) = 1 \end{cases} \tag{7}$$

And the dual function of the above Lagrange program is

$$g(\lambda) = \sum_{r} \max_{\mathbf{y}_r}(f_r(\mathbf{y}_r) - \sum_{p \in P(r)} \lambda_{r \to p}(\mathbf{y}_r)) + \sum_{c \in C(r)} \lambda_{c \to r}(\mathbf{y}_c) \tag{8}$$

What we need to do is to minimize the dual program, that is to say we need to find $\min_{\lambda} g(\lambda)$

5 Graph-cut Solvers

5.1 Solving the minimization problem

Since we already have efficient algorithms which can compute the minimum cost of a weighted graph [1] we can convert our optimization problem to finding the minimum cut. To achieve this, we need an auxiliary graph, note that the auxiliary graph is not the variable dependency graph shown above. Taking the following auxiliary graph as an example.

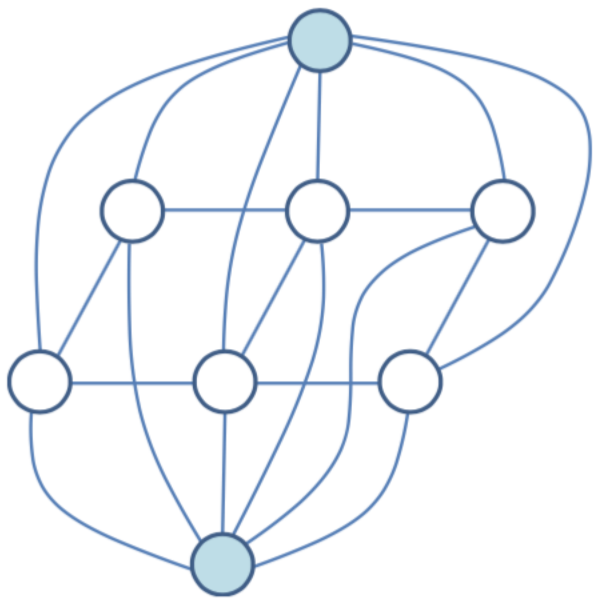


Figure 2: Auxiliary graph

In the above graph, we have a start node, a end node, and the rest stands for the unary variables. If the problems are binary, i.e. each node can take the same two values, then we can use the min-cut to find the most optimal assignment.

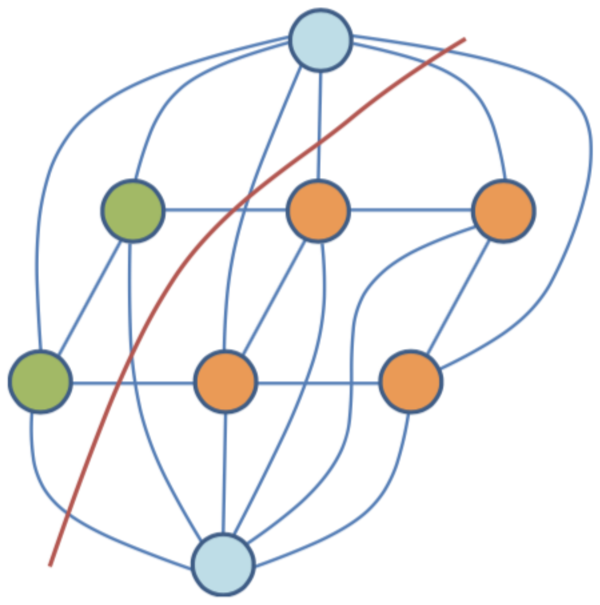


Figure 3: One possible cut

For example, in the above graph, the red cut divide the variables into two parts. Nodes in different parts have different binary values. The next step we are going to do is to assign weights to edges. After assign edge we can run the min-cut algorithm on the graph to find the min-cut. The assignment generate by the min-cut is the assignment that will result in the minimum value. To begin with, we need to write down our local scoring functions in the form of arrays for marginalization. Suppose we only have one variable y_1 which can take binary values of 1 or 2. Then the scoring function could be expressed in the following way

$$\begin{bmatrix} f_1(y_1 = 1) & f_1(y_1 = 2) \end{bmatrix} \tag{9}$$

Now we can assign weights to edges whose one end is connected to the start vertex or end vertex. Here each node y_i shall be assigned to a label in the label set $1, 2$. And we use f_1, f_2 and f_12 to denote the scoring function.

- If $f_1(y_1 = 1) < f_1(y_1 = 2) = 1$, we assign the positive weight $f_1(y_1 = 2) - f_1(y_1 = 1)$ to the edge (s, y_1). Else we assign $f_1(y_1 = 2) - f_1(y_1 = 1)$ to the edge (y_1, v)

Here we take the situation $f_1(y_1 = 1) < f_1(y_1 = 2)$ as an example, then we might obtain the following graph.

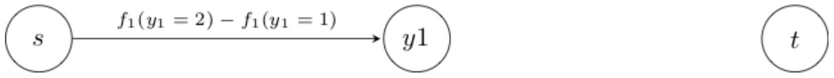
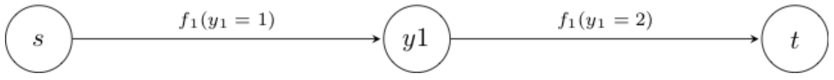


Figure 4: Auxiliary Graph

Since when solving the min-cut algorithm, adding a constant to each edges does not affect the result, therefore we can add $f_1(y_1 = 2)$ to all the edges and obtain the following graph.



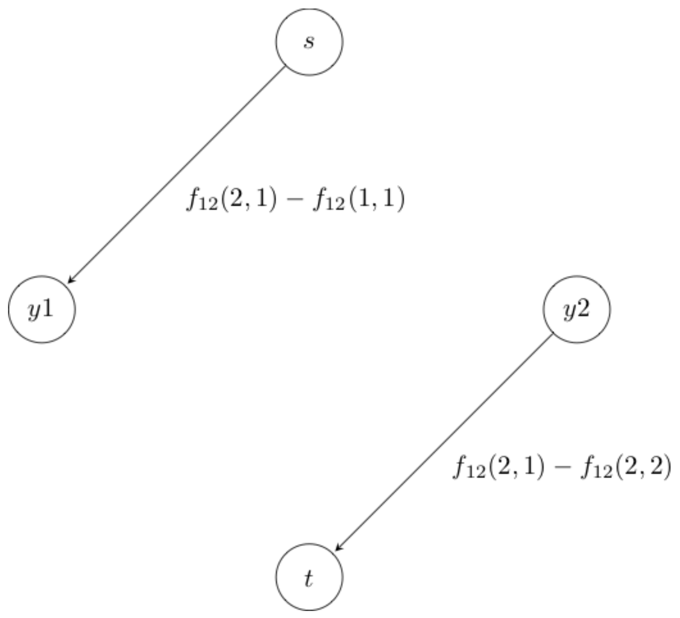
From the above graph, we can see that the min cut is the cut that go through the edge (y_1, t). Therefore the min cut shall go through (y_1, t) and assign the label 1 to variable y_1, which is the same as the expected.

Now we take a look at the following optimization problem:

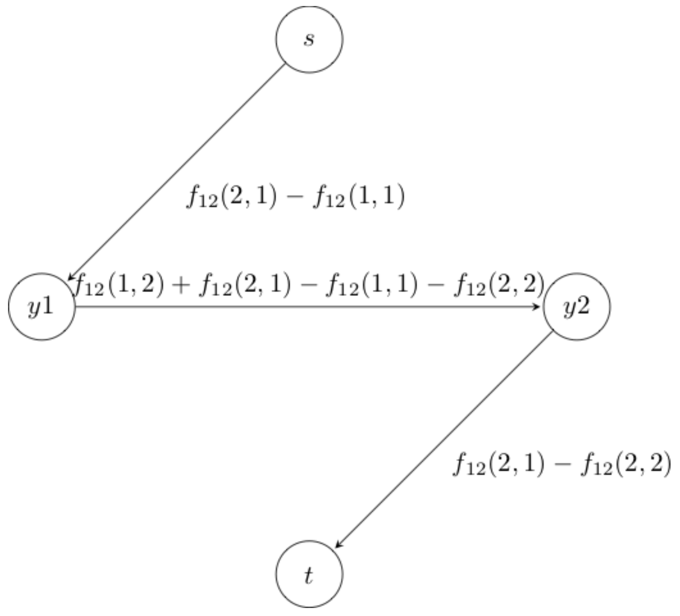
$$\begin{bmatrix} f_{12}(1,1) & f_{12}(1,2) \\ f_{12}(2,1) & f_{12}(2,2) \end{bmatrix}$$

Using some technique, we can express the above matrix in the following way
$\begin{bmatrix} f_{12}(1,1) & f_{12}(1,2) \\ f_{12}(2,1) & f_{12}(2,2) \end{bmatrix} = \begin{bmatrix} f_{12}(1,1) & f_{12}(1,1) \\ f_{12}(1,1) & f_{12}(1,1) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ f_{12}(2,1) - f_{12}(1,1) & f_{12}(2,1) - f_{12}(1,1) \end{bmatrix} +$
$\begin{bmatrix} 0 & f_{12}(2,2) - f_{12}(2,1) \\ 0 & f_{12}(2,2) - f_{12}(2,1) \end{bmatrix} + \begin{bmatrix} 0 & f_{12}(1,2) + f_{12}(2,1) - f_{12}(1,1) - f_{12}(2,2) \\ 0 & 0 \end{bmatrix}$
By discarding the constant item, we have

$\begin{bmatrix} f_{12}(1,1) & f_{12}(1,2) \\ f_{12}(2,1) & f_{12}(2,2) \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ f_{12}(2,1) - f_{12}(1,1) & f_{12}(2,1) - f_{12}(1,1) \end{bmatrix} +$
$\begin{bmatrix} 0 & f_{12}(2,2) - f_{12}(2,1) \\ 0 & f_{12}(2,2) - f_{12}(2,1) \end{bmatrix} + \begin{bmatrix} 0 & f_{12}(1,2) + f_{12}(2,1) - f_{12}(1,1) - f_{12}(2,2) \\ 0 & 0 \end{bmatrix}$
If we assume that $f_{12}(2,1) - f_{12}(1,1) > 0$ and $f_{12}(2,2) - f_{12}(2,1) < 0$. Then according to the rules, we should assign the weight $f_{12}(2,1) - f_{12}(1,1) > 0$ to the edge (s, y_1) and the weight $f_{12}(2,2) - f_{12}(2,1) < 0$ to the edge (y_2, t)



Now we might add the following edge to the graph to represent the last term


s
$f_{12}(2,1) - f_{12}(1,1)$
$y1$
$f_{12}(1,2) + f_{12}(2,1) - f_{12}(1,1) - f_{12}(2,2)$
$y2$
$f_{12}(2,1) - f_{12}(2,2)$
t

Note that in order to solve the min-cut problem, we must ensure that the weight of each edge is positive. Therefore we must ensure that $f_{12}(1,2) + f_{12}(2,1) - f_{12}(1,1) - f_{12}(2,2) > 0$. This is also called sub-modularity.

5.2 Soving the maximization problem

To solve the maximization problem, we just need to make all the weights opposite and run the same min-cut algorithm, this is because finding the maxinum element in a mateix F is the same as finding the mininum element in $-F$.

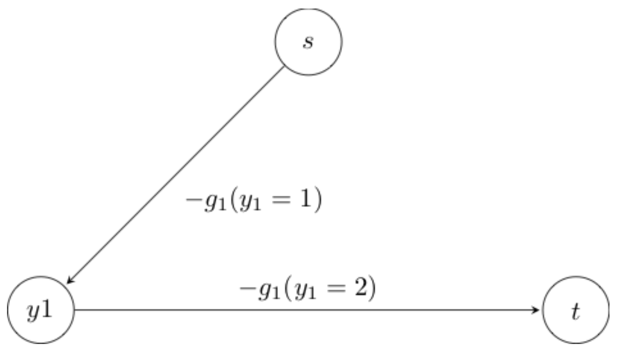
For example, if we want solve the following maximization problem $\begin{bmatrix} f_1(y_1 = 1) & f_1(y_1 = 2) \end{bmatrix}$ First we need to take the opposite of the above matrix

$$\begin{bmatrix} -f_1(y_1 = 1) & -f_1(y_1 = 2) \end{bmatrix} \tag{10}$$

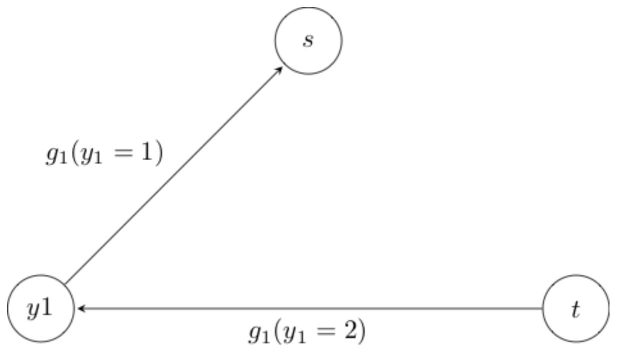
For simplicity, let $g_1 = -f_1$ Then we have

$$\begin{bmatrix} g_1(y_1 = 1) & g_1(y_1 = 2) \end{bmatrix} = \begin{bmatrix} g_1(y_1 = 1) & 0 \end{bmatrix} + \begin{bmatrix} 0 & g_2(y_1 = 2) \end{bmatrix} \tag{11}$$

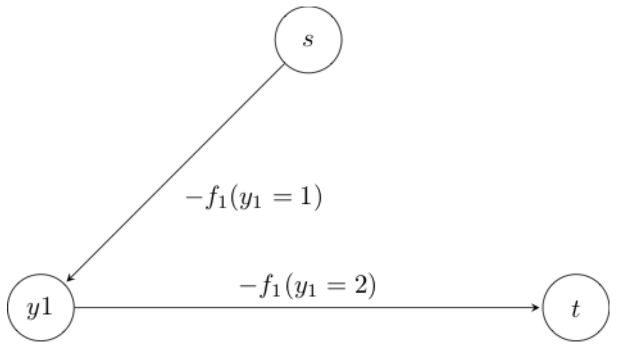
If we assume that $f_1(y_1 = 1) > 0$ and $f_1(y_1 = 2) > 0$, then we have $g_1(y_1 = 1) < 0$ and $g_1(y_1 = 2) < 0$. Therefore according to the rules that we mentioned in 5.1, we can obtain the following graph



Since a directed edge (u, v) with weight w can be replaced by a directed edge (v, u) with weight $-w$, therefore we can convert the above graph to the following form



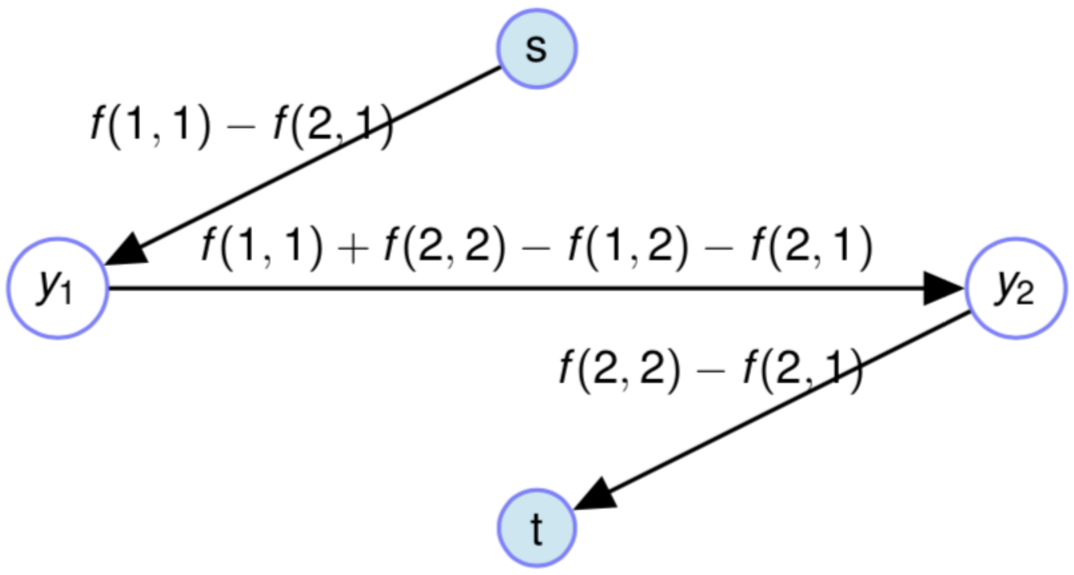
Which is the same as the graph as we discussed in the lecture



By comparing this graph to figure 3, we can find that the only difference between these two graphs are that the all the weights are opposite. With similar method, when solving the maximization problem discussed during the class

$$\begin{bmatrix} f_{12}(1,1) & f_{12}(1,2) \\ f_{12}(2,1) & f_{12}(2,2) \end{bmatrix}$$

We might obtain the following graph that we have seen in the lecture



Note that here the submodularity should be changed to $f_{12}(1,1)+f_{12}(2,2)-f_{12}(1,2)-f_{12}(2,1) >= 0$

Reference

[1] T. E. W. A. A. Amini and R. C. Jain. Using dynamic programming for solving variational problems in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(9):855–867, Sept. 1990.

[2] V. Kolmogorov and R. Zabin. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2):147–159, Feb. 2004.

Appendix:Proof of forumula 6

Although it is might be a little hard to derive (6) from (5). We can subtract (5) from (6) to see if the result equals zero. First we can write(6) in another way

$$\begin{aligned} L() &= \sum_{r,\mathbf{y}_r} b_r(\mathbf{y}_r)(f_r(\mathbf{y}_r) - \sum_{p\in P(r)} \lambda_{r\to p}(\mathbf{y}_r) + \sum_{c\in C(r)} \lambda_{c\to r}(\mathbf{y}_c)) \\ &= \sum_{r,\mathbf{y}_r} b_r(\mathbf{y}_r) f_r(\mathbf{y}_r) - \sum_{r,\mathbf{y}_r} b_r(\mathbf{y}_r) \sum_{p\in P(r)} \lambda_{r\to p}(\mathbf{y}_r) + \sum_{r,\mathbf{y}_r} b_r(\mathbf{y}_r) \sum_{c\in C(r)} \lambda_{c\to r}(\mathbf{y}_c) \end{aligned} \tag{12}$$

Then, we write(5) also in another way

$$\begin{aligned}L() &= \sum_{r,\mathbf{y}_r} b_r(\mathbf{y}_r), f_r(\mathbf{y}_r) + \sum_{r,p\in P(r),\mathbf{y}_r} \lambda_{r\to p}(\mathbf{y}_r)(\sum_{y_p\setminus \mathbf{y}_r} b_p(y_p) - b_r(\mathbf{y}_r)) \\ &= \sum_{r,\mathbf{y}_r} b_r(\mathbf{y}_r), f_r(\mathbf{y}_r) + \sum_{r,p\in P(r),\mathbf{y}_r} \lambda_{r\to p}(\mathbf{y}_r) \sum_{y_p\setminus \mathbf{y}_r} b_p(y_p) - \sum_{r,p\in P(r),\mathbf{y}_r} \lambda_{r\to p}(\mathbf{y}_r) b_r(\mathbf{y}_r)\end{aligned} \tag{13}$$

Then we subtract (15) from (16), we obtain

$$\sum_{r,\mathbf{y}_r} b_r(\mathbf{y}_r) \sum_{c\in C(r)} \lambda_{c\to r}(\mathbf{y}_c) - \sum_{r,p\in P(r),\mathbf{y}_r} \lambda_{r\to p}(\mathbf{y}_r) \sum_{y_p\setminus \mathbf{y}_r} b_p(y_p) \tag{14}$$

We can see that (17) just expressed the weighted sum of the dependency graph in two ways, so obvious (17) results in 0. Therefore, equation (5) and equation(6) are equal.