Week 11

Graph Cut, Pseudo Boolean Function and Optimisation*

The lecture encompasses the concept of optimisation of pseudo boolean functions using graph cut framework and application of graph cut in image segmentation.

11.1 Pseudo Boolean Function

The section elaborates on Pseudo Boolean Function along with examples and optimisation using brute force and using graph cut framework.

11.1.1 Definition

Pseudo Boolean Function is a function from Boolean numbers to Real numbers. Domain of these function is Boolean numbers and Range is Real numbers.

The function is defined as $f : B \to R$ or $f : \{0, 1\} \to R$.

11.1.2 Example

Consider an example of a Pseudo Boolean function $f(x) = 2x_1 + 3\overline{x_1} + 4x_2 + 2\overline{x_2} + 2\overline{x_1}x_2$ where, $x_1, x_2 \in \{0, 1\}$

Now we need to optimise this function or say we have to find the global minimum value of this Pseudo Boolean function $f(x)$. One way to find the global minimum is using brute force approach. In brute force method we have to prepare a truth table for the function then check for the minimum value of the above function.

Here, $x_1, x_2 \in \{0, 1\}$ thus x_1 can take two values and x_2 can take two values. Considering that the Truth table of $f(x)$, with four possible assignments, has been shown below.

*Lecturer: Anand Mishra. Scribe: Neha Soni (B19CSE058).

x1	x2	f (x)
0	0	0 + 3 + 0 + 2 + 0 = 5
0	1	0 + 3 + 4 + 0 + 2 = 9
1	0	2 + 0 + 0 + 2 + 0 = 4
1	1	2 + 0 + 4 + 0 + 0 = 6

Figure 11.1: Truth table *f(x)*

In the example Figure 11.1, we can observe that the minimum value of $f(x)$ comes out to be 4 when $x_1 = 1$ & $x_2 = 0$.

This method of brute force is not feasible when the number of variable increases. As each variable can take either of the two values, 0 or 1, thus when we have n variables with us, then the time complexity of brute force will come out to be 2^n *i.e.* exponential.

11.1.3 Optimisation

Now we'll look into optimisation of the above Pseudo Boolean function $f(x) = 2x_1 + 3\overline{x_1} + 4x_2 + 2\overline{x_2} + 2\overline{x_1}x_2$ where, $x_1, x_2 \in \{0, 1\}$ using graph cut framework.

Construction The flow graph is constructed in a way where we have two special nodes - Source node and Target node, and for every variable in the function we have a node.

Below is the constructed flow graph for the function $f(x)$.

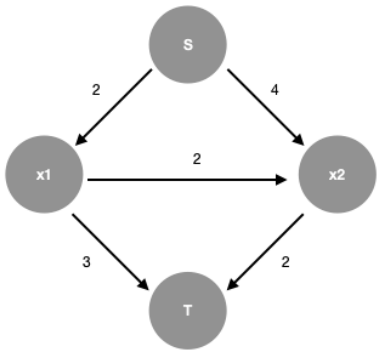

Figure 11.2: Flow graph of *f(x)*

Optimisation Now, we will find the min cut of the flow graph shown in Figure 11.2 using Ford-Flukerson method. This min cut will give us the assignment for minimum value of the Pseudo Boolean function. Initially the flow is zero, *i.e.* $flow = 0$

Step 1 Choose the path $S \to x_2 \to T$.

The maximum we can flow through this path is 2. Thus, $flow = 0 + 2 = 2$.

The new flow graph is shown below.

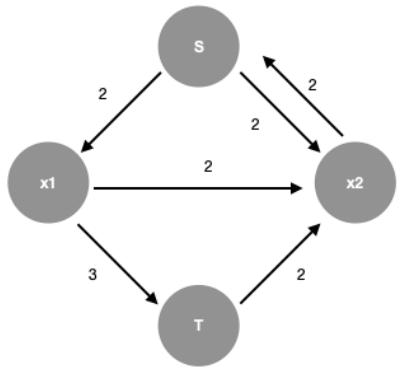

Figure 11.3: Flow graph of *f(x)*

Step 2 Choose the path $S \to x_1 \to T$.

The maximum we can flow through this path is 2. Thus, $flow = 2 + 2 = 4$.

The new flow graph is shown below.

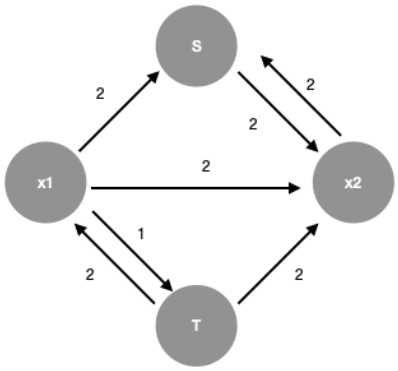

Figure 11.4: Flow graph of *f(x)*

Step 3 Now, no more augmenting path left from S to T. Thus, we stop here.

The maximum flow has come out to be $flow = 4$ and the min cut is shown below.

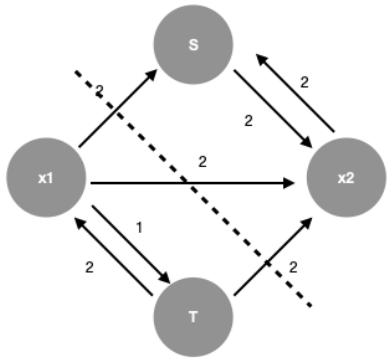

Figure 11.5: Flow graph of *f(x)*

From Figure 11.5, we can see that the two sets are : $S = \{s, x_2\}$ & $T = \{t, x_1\}$.

Now, for the assignment purpose, we will assign the vertices belonging to set S as 0 and vertices belonging to set T as 1. Thus, x_1 & x_2 are assigned as $x_1 = 1$ & $x_2 = 0$.

This assignment will give us the global minimum of the Pseudo Boolean function. Also, it can be observed and verified that the brute force method has also given us the same result.

11.2 Graph Cut in Image Segmentation

The section encompasses the application of graph cut approach in image segmentation.

11.2.1 Motivation

We are given with an image and then we have to represent it as Pseudo Boolean function, say energy function. Then we have to separate the foreground from the background. Consider the below image segmentation.

Figure 11.6: Image Segmentation

We will represent this image shown on left side in Figure 11.6 as an enery function, $E(x)$

$$E(x) = \sum_i c_i x_i + \sum_{i,j} c_{i,j} x_i (1 - x_i) \tag{11.1}$$

Here, each $x_i^{'}s$ represent each of the pixel locations in the image. c_i represents the cost of assigning it to foreground or background and $c_{i,j}$ represents the cost of assigning different labels to the neighbours where x_i, x_j are neighbours. In simple words $c_{i,j}$ represents if x_i is foreground and x_j is background then what is the cost.

One way to get this cost is by some sort of human interaction like drawing the bounding box on the image and saying that everything outside the bounding box belongs to the background and everything inside the bounding box could be foreground or background. Then, lots of supervision compute the costs like if a pixel is closer to foreground then it is foreground and if a pixel is far from foreground then it is background and we compute costs.

Once we declare the function, we assign each $x_i^{'}s$ with 0 or 1 and we find the overall minimum cost by using graph cut framework.

11.2.2 Construction

Construction of the graph is done as follows,

1. Any cut corresponds to an assignment of x
2. The cost of the cut is equal to energy of x : $E(x)$

Consider an Energy function as,

$$E(a_1, a_2) = 2a_1 + 5\overline{a_1} + 9a_2 + 4\overline{a_2} + 2a_1\overline{a_2} + \overline{a_1}a_2 \tag{11.2}$$

where, $a_1, a_2 \in \{0, 1\}$.

The flow graph for this energy function is constructed as below.

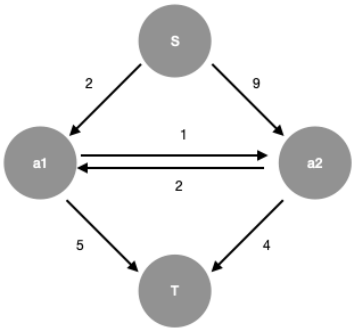

Figure 11.7: Flow graph of *E(x)*

11.2.3 What energy functions can be minimised ?

Not all the energy functions can be minimised using graph cut method.

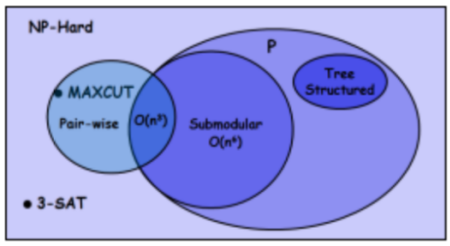

Figure 11.8

1. General energy functions are NP hard to minimize and only approximate solutions are available
2. Easy energy functions (or sub-modular functions) are graph representable and solvable in polynomial time

11.2.4 Sub Modular Function

Let f be a function defined over set of Boolean variables $x = \{x_1, x_2, ..., x_n\}$, then

1. All functions of one Boolean variables are sub modular
2. For a function of two Boolean variable, it is sub modular if,
   $f(0,0) + f(1,1) \leq f(0,1) + f(1,0)$
3. In general a function is sub-modular if all its projection to two variables are sub-modular

If a function is not following the constraints of sub-modularity then that function cannot be represented as graph and cannot be minimised using graph cut.

11.2.5 Multi Label Cases

Now, not everything is just foreground or background in real world scenarios. Not everything is a binary label. For example, in real world we have to distinguish if something is a house or tree or grass etc.

We may wish to pass an object and find different parts of that object. Consider the example of a cow object as below.

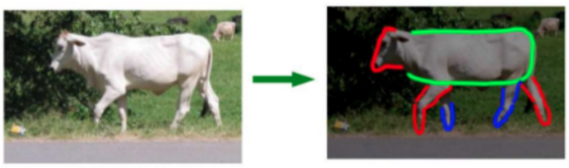

Figure 11.9: Multi label segmentation

In the above image, we may wish to segment out the head of the cow object, its legs, its torso, its body etc.

We can solve this multi label problem using graph cut framework and that too in an efficient manner.

11.2.6 Energy Function

Similar to bi-label cases, the goal here is to find a labelling that assigns each pixel $p \in P$ to a label $f_p \in L$ where, f is both consistent with observed data and piecewise smooth.

Here, f is the labelling and L is the label set. By ***observed data*** we means that in a pixel what do we observe *i.e.* given a pixel what does it look like. If a pixel looks like grass then the cost of that pixel assignment to sky is high and the cost of that pixel assignment to grass is low. By ***piecewise smooth*** we are also looking into the neighbours of the pixels. Just mere observation might result into some noise and to overcome that we need to look into the neighbours.

$$E(f) = E_{data}(f) + E_{smooth}(f) \tag{11.3}$$

We have, $E_{data}(f) = \sum_{p \in P} D_p(f_p)$ where, D_p measures the agreement of inferred label from the observed data. $E_{smooth}(f) = \sum_{(p,q) \in N} V_{pq}(f_p, f_q)$ and this term is used to impose spatial smoothness.

- For, $|L| = 2$, the energy is exactly minimizable
- if $|L| > 2$, then exact minimization is not possible using graph cut as these are NP hard problems. We can only get approximate solution. Although we can get approximate solution with known factor of global minima in case when V_{pq} is either a metric or semi-metric.

11.2.7 Semi Metric and Metric

A function $V(.,.)$ is semi-metric on space of labels $\alpha, \beta \in L$ if it satisfies :

1. $V(\alpha, \beta) = V(\beta, \alpha)$ *i.e.* symmetric
2. $V(\alpha, \beta) = 0 \Leftrightarrow \alpha = \beta$

If $V(.,.)$ also satisfies triangle inequality *i.e.*

$$V(\alpha, \beta) = V(\alpha, \gamma) + V(\gamma, \beta) \tag{11.4}$$

for $\alpha, \beta, \gamma \in L$ then, it is metric.

If smoothness term also has these properties then we can use two algorithms called as move making algorithm, also known as alpha-beta swap, and second algorithm called as alpha-expansion, only when V is metric.

11.2.8 Move Algorithm

Let us understand how labelling is associated with partition of pixels. Say we have set of pixels or set of nodes in a graph and we need to label each of them. Any labelling f can be uniquely represented by partition of image pixel $P = \{\mathcal{P}_l : l \in L\}$ where, $\mathcal{P}_l$ is the subset of pixels assigned label l.

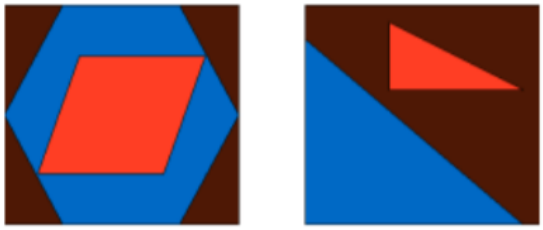

Figure 11.10: Partitions

Above shown are two different labelling where in left image the middle nodes are coloured red.

11.2.9 $\alpha - \beta$ swap

Definition Given a pair of labels α, β a move from partition P to partition P' is called an $\alpha - \beta$ swap if $P_l = P'_l$ for any label $l \neq \alpha, \beta$.

Here, at first we randomly initialise some pixels, then choose two of the labels. Let say we choose sky and house. Then in $\alpha - \beta$ swap some of the sky's become house and some of

the house's becomes sky, whereas others remains same. We do graph cut for every pair of labels.

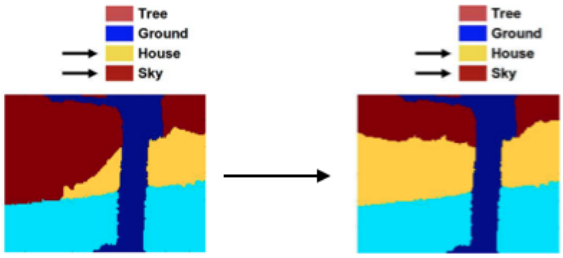

Figure 11.11: $\alpha - \beta$ swap

In the above shown Figure 11.11, $\alpha - \beta$ swap has been described for house and sky.

11.2.10 α − expansion

Definition Given a label α a move from partition P to partition P' is called an α − expansion if $P_\alpha \subset P'_\alpha$ and $P_l \subset P'_l$ for any label $l \neq \alpha$.

Here, the high level concept is one versus all. Let say we begin we tree. Then first we assume that everything is tree and we do tree Vs non tree, then ground Vs non ground, then house Vs non house and then sky Vs non sky. In this way we are expanding.

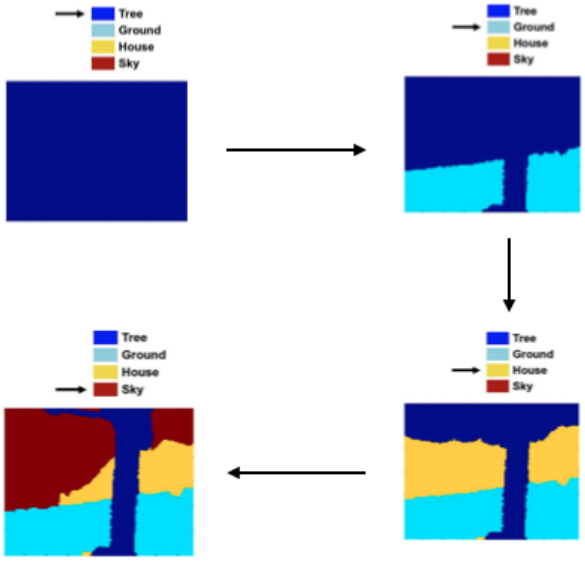

Figure 11.12: α − expansion

11.2.11 $\alpha - \beta$ swap basic algorithm

Basic algorithm for $\alpha - \beta$ swap is as follows :

1. Start with an arbitrary labelling f
2. Set success $\leftarrow 0$
3. For each pair of label $\{\alpha, \beta\} \in L$
   (a) Find $\hat{f} = argminE(f')$ within one $\alpha - \beta$ swap of f
   (b) If $E(\hat{f}) < E(f)$ then,
   Set $f \leftarrow \hat{f}$
   Set success $\leftarrow 1$
4. If success != 1 then go to step 2 else return f

11.2.12 $\alpha -$ expansion basic algorithm

Basic algorithm for $\alpha -$ expansion is as follows :

1. Start with an arbitrary labelling f
2. Set success $\leftarrow 0$
3. For every label $\alpha \in L$
   (a) Find $\hat{f} = argminE(f')$ within one $\alpha -$ expansion of f
   (b) If $E(\hat{f}) < E(f)$ then,
   Set $f \leftarrow \hat{f}$
   Set success $\leftarrow 1$
4. If success != 1 then go to step 2 else return f