Advanced Algorithms Course. Lecture Notes. Part 6

A Machine Learning Problem: Image Segmentation

This problem asks to label each pixel of a digital image as foreground (part of an object) or background. The picture is represented as an undirected graph $G = (V, E)$ where nodes are pixels and edges exist between any two neighbored pixels. For every pixel i we are also given two numbers a_i and b_i expressing the strength of belief that pixel i is foreground or background, respectively. We do not discuss here in depth how these values are obtained (criteria could be, for example, the colors and positions of pixels), we just consider them as input to our problem. A further assumption is that the picture does not comprise too many switches between foreground and background, that is, it shows a few large and connected objects. Therefore we introduce penalties for label switches: For each pair of neighbored pixels i, j we charge a penalty p_{ij} if i and j have different labels. Altogether this gives rise to the following optimization problem: Split V into sets A and B (foreground and background) so as to maximize

$$q(A,B) := \sum_{i\in A} a_i + \sum_{j\in B} b_j - \sum_{(i,j)\in E, i\in A, j\in B} p_{ij}.$$

That is, the segmentation should respect the classification criteria for the single pixels (strength of belief) but it should not need too many switches.

We can reduce this problem to Minimum Cut as follows. First observe that it is equivalent to minimizing

$$q'(A,B) := \sum_{i\in A} b_i + \sum_{j\in B} a_j + \sum_{(i,j)\in E, i\in A, j\in B} p_{ij}.$$

That is, we want to minimize the penalties for both false labels and switches. As we need a directed graph, we replace every edge (i, j) with two opposite directed edges with capacity p_{ij}. We insert a source s and sink t, and for every pixel k we insert edges (s,k) with capacity a_k, and (k,t) with capacity b_k. Now any $s-t$ cut (A,B) has capacity $q'(A,B)$. Thus, an optimal segmentation corresponds to a minimum cut.

Project Selection

Let P be a set of possible projects to choose from. Project i has revenue p_i. Value p_i can also be negative, in this case the project is an investment for other projects: Some projects depend on others. These dependencies are given as a directed graph $G=(P,E)$ where an edge (i,j) means: if i shall be done, then j must be done, too (before i can even start). Clearly, G must be acyclic, since projects in a directed cycle of dependencies can never be done. We call a set of projects $A \subset P$ feasible if A respects these precedence constraints. The problem is to select a feasible set A that maximizes $\sum_{i\in A} p_i$. This is also known as the Open-Pit Mining problem. (One can easily imagine the reason.)

We reduce Project Selection (Open-Pit Mining) to Minimum Cut. We insert a source s and a sink t. Edges are (s,i) with capacity p_i, if $p_i > 0$, and (i,t) with capacity $-p_i$, if $p_i < 0$. Edges in G (for the precedence constraints) get a huge capacity. Hence none of these edges can go from A to B in a minimum cut $(A\cup\{s\}, B\cup\{t\})$. It follows that A is feasible whenever $(A\cup\{s\}, B\cup\{t\})$ is a minimum cut. Now we can solve the Minimum Cut problem and need not worry about the feasibilty of A.

It remains to show that minimizing the cut capacity is in fact equivalent to maximizing the revenue. This is proved in a few lines:

$$c(A\cup\{s\}, B\cup\{t\}) = \sum_{p_i>0, i\in B} p_i - \sum_{p_i<0, i\in A} p_i$$

holds by the definition of capacity. We artificially add zero:

$$c(A\cup\{s\}, B\cup\{t\}) = \sum_{p_i>0, i\in B} p_i - \sum_{p_i<0, i\in A} p_i - \sum_{p_i>0, i\in A} p_i + \sum_{p_i>0, i\in A} p_i.$$

Now we can group the terms in a different way:

$$c(A\cup\{s\}, B\cup\{t\}) = \sum_{p_i>0} p_i - \sum_{i\in A} p_i.$$

Note that the first term is constant and the second term is the revenue.

Baseball Elimination

Do you like to see a fun application after all these serious ones? A set S of baseball teams are playing in a league. For every team x we know the number w_x of wins so far, and for each pair of teams x, y we know the number g_{xy} of games x against y that are left. You are concerned about your favourite team: Can z still become champion? You only want to answer this yes/no question.

First think a while and realize what the difficulty of the problem is. Next, it may be hard to believe that this has anything to do with flows. It has, but this time the idea is more tricky.

For answering the “can z still become champion?” question it is safe to assume that z will win all its remaining games (all other cases would be worse). Now let m be the total number of wins of z. For convenience define $S' := S \setminus \{z\}$ and $g^* := \sum_{x,y \in S'} g_{xy}$ (the total number of remaining games). Teams and games will be represented as nodes of a directed graph. We insert a source s and a sink t. Roughly speaking, the role of s is to “generate wins” that “pass through” the game nodes and team nodes and are eventually “absorbed” by t.

We create a node v_x for each $x \in S'$, and a games node u_{xy} when $g_{xy} > 0$. Edges (s, u_{xy}) with capacity g_{xy} will “forward wins” to the game nodes. (Each game has a winner.) Edges (u_{xy}, v_x) and (u_{xy}, v_y) with huge capacity will pass one unit of flow to the winner of each game between x and y. Edges (v_x, t) with capacity $m - w_x$ “collect the wins” from the teams. The capacities ensure that no team x can reach more than m wins in total. That means: z has a chance to become champion if and only if there exists a flow with value g^*.

But, in the case that z is no longer a candidate for championship, how can we convince an enthusiast of team z of this sad truth? (Most likely he will not understand any word about flows, and probably he will even doubt our computed result.) Clearly, z cannot become champion anymore if, in a subset T of teams, the average number of wins from past games and games inside T exceeds m. This quantity does not depend on the results of the outstanding games inside T. Formally, this condition is: $\sum_{x \in T} w_x + \sum_{x,y \in T} g_{xy} > m|T|$. Note that such a set T provides (hopefully) an easily understandable proof that z is out. Interestingly, such T does always exist and can be computed via a minimum cut in the graph constructed above. But we do not go into further details.