Optimization for Machine Learning

Exercise sheet 2

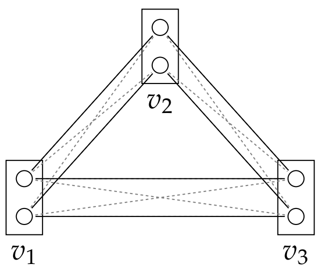
Dr. Bogdan Savchynskyy <bogdan.savchynskyy@iwr.uni-heidelberg.de>
Sebastian Stricker <sebastian.stricker@stud.uni-heidelberg.de>

Exercise 2.1 Implement the transformation of graphical models into an integer linear program (ILP). Optimize the ILP with an off-the-shelf solver and verify the solution.

Note: The Python-LP library (pulp[1]*) allows to build and solve LP/ILPs conveniently.*

Exercise 2.2 Construct the LP relaxation for a given input graphical model programmatically. Optimize the LP with an off-the-shelf solver and verify the solution. Implement the *naïve rounding* scheme to obtain an integer labeling. How do the LP optimum, the rounded solution and the ILP optimum compare in general?

Exercise 2.3 Optimize the LP relaxation of the following graphical model. What is special about the solution of the model? Round the solution to obtain an integer labeling and optimize the same model with an ILP solver. What do you observe? Explain why.



$$\theta_1 = [0.5, 0.5],$$
$$\theta_2 = [0.0, 0.0],$$
$$\theta_3 = [0.2, 0.2]$$

$$\forall \{u, v\} \in \mathcal{E} : \theta_{uv} = \begin{cases} 1 & \text{if } x_u = x_v, \\ 0 & \text{otherwise.} \end{cases}$$

Exercise 2.4 On our website we provide multiple random acyclic and cyclic graphical models. Compute the LP and ILP optima. What do you observe? Explain why.

acyclic model	LP	ILP
model 1	...	...
...	...	...

cyclic model	LP	ILP
model 1	...	...
...	...	...

[1] https://pypi.org/project/PuLP/

Exercise 2.5 One problem in computer vision is the reconstruction of depth information from two stereo image pairs. The basic idea is to rectifiy the stereo image pair and compute the disparity for each pixel. The task of this exercise is to reason about the practicability of using generic LP/ILP solvers for solving these (relatively large-scale) problems.

On our website we provide a graphical model formulation of the *tsukuba* scene which is part of the Middlebury stereo benchmark. Additionally, we provide downsampled variants.

filename	image dimensions	#vars	#contrs	LP	ILP
tsu_down_01.uai	384x288	...	...	...	...
tsu_down_02.uai	192x144	...	...	...	...
tsu_down_04.uai	96x72	...	...	...	...
tsu_down_08.uai	48x36	...	...	...	...
tsu_down_16.uai	24x18	...	...	...	...
tsu_down_32.uai	12x9	...	...	...	...

How big (in terms of variables/constraints) are the resulting LP/ILP problems? How much time does the inference need? Can you solve all instances in reasonable time? How practical is it to solve these kind of vision problems with general LP/ILP-solvers?

Exercise 2.6 Selecting a good set of segmentations out of a pool of possible candidates (over-segmentation) is a combinatorial optimization problem that can also be formulated as an ILP. On our website we provide an archive containing 500 binary segmentation masks. Write a Python program that reads all the segmentation candidates and outputs a set of non-overlapping segmentations such that the total covered image area (number of pixels) is maximized by solving the ILP formulation of the problem. Visualize your results!