Overview

- Primal LP
- Dual LP
- Coordinate-Ascent Algorithm

Note: These slides are rather brief and not self-contained. For more information please consult the paper referenced at the end.

The Primal LP (Part 1)

minimise $\sum_{i \in I} \lambda_i x_i$

subject to $x \in \text{conv}(C_j) \ (j \in J)$.

Definitions (Part 1)

Here we used the following codes, variables and vectors. The code

- $A_i \subseteq \{0, 1\}^{[0] \cup J_i}, \ (i \in I)$, is the set containing the all-zeros vector and the all-ones vector of length $|J_i| + 1$
- $B_j \subseteq \{0, 1\}^{I_j}, \ j \in J$, is the code $C_j$ punctured at the positions $I \setminus J_j$. (For the codes $C$ under consideration this means that $B_j$ contains all vectors of length $|I_j|$ of even parity.)
- For $i \in I$ we will also use the vectors $u_i$ where the entries are indexed by $\{0\} \cup J_i$ and denoted by $u_{i,j} \triangleq [u_i]_j$, and for $j \in J$ we will use the vectors $v_j$ where the entries are indexed by $I_j$ and denoted by $v_{j,i} \triangleq [v_j]_i$.
- We will use a similar notation for the entries of $a_i$ and $b_j$, i.e. we will use $a_{i,j} \triangleq [a_i]_j$ and $b_{j,i} \triangleq [b_j]_i$, respectively.
The Primal LP (Part 2)

\[ \begin{align*}
\text{min.} \quad & \sum_{i \in I} \lambda_i x_i \\
\text{subj. to} \quad & x_i = u_{i,0} \quad (i \in I), \\
& u_{i,j} = v_{j,i} \quad ((i, j) \in \mathcal{E}), \\
& \sum_{a_i \in A_i} \alpha_{i,a_i} = u_i \quad (i \in I), \\
& \sum_{b_j \in B_j} \beta_{j,b_j} = v_j \quad (j \in J), \\
& \alpha_{i,a_i} \geq 0 \quad (i \in I, \ a_i \in A_i), \\
& \beta_{j,b_j} \geq 0 \quad (j \in J, \ b_j \in B_j), \\
& \sum_{a_i \in A_i} \alpha_{i,a_i} = 1 \quad (i \in I), \\
& \sum_{b_j \in B_j} \beta_{j,b_j} = 1 \quad (j \in J).
\end{align*} \]

The Primal LP (Part 3)

With suitable cost functions, the above optimization problem can also be formulated as an unconstrained optimization problem. Minimize

\[ \sum_{i \in I} \lambda_i x_i + \sum_{i \in I} \left[ x_i = u_{i,0} \right] + \sum_{(i,j) \in \mathcal{E}} \left[ u_{i,j} = v_{j,i} \right] + \sum_{i \in I} A_i(u_i) + \sum_{j \in J} B_j(v_j), \]

where for all \( i \in I \) and all \( j \in J \), respectively, we introduced

\[ A_i(u_i) \triangleq \left[ \sum_{a_i \in A_i} \alpha_{i,a_i} = u_i \right] + \left[ \sum_{a_i \in A_i} \alpha_{i,a_i} \geq 0 \right] + \left[ \sum_{a_i \in A_i} \alpha_{i,a_i} = 1 \right], \]

\[ B_j(v_j) \triangleq \left[ \sum_{b_j \in B_j} \beta_{j,b_j} = v_j \right] + \left[ \sum_{b_j \in B_j} \beta_{j,b_j} \geq 0 \right] + \left[ \sum_{b_j \in B_j} \beta_{j,b_j} = 1 \right]. \]

Definitions (Part 1)

If \( A \) is a statement, then

\[ [A] \triangleq \begin{cases} 
1 & \text{if } A \text{ is true} \\
0 & \text{if } A \text{ is false}
\end{cases} \]

\[ [A] \triangleq -\log[A] = \begin{cases} 
0 & \text{if } A \text{ is true} \\
\infty & \text{if } A \text{ is false}
\end{cases} \]

FFG representing the Primal LP
FFG representing the Dual LP

Comparison of FFGs Representing the Primal and Dual LP

Dual LP

\[
\begin{align*}
\max & \quad \sum_{i \in I} \phi'_i + \sum_{j \in J} \theta'_j \\
\text{subj to} & \quad \phi'_i \leq \min_{a_i \in A_i} \langle -u'_i, a_i \rangle \quad (i \in I), \\
& \quad \theta'_j \leq \min_{b_j \in B_j} \langle -v'_j, b_j \rangle \quad (j \in J), \\
& \quad u'_{i,j} = -v'_{j,i} \quad ((i, j) \in E), \\
& \quad u'_{i,0} = -x'_i \quad (i \in I), \\
& \quad x'_i = \lambda_i \quad (i \in I).
\end{align*}
\]

Coordinate-Ascent Algorithm (Part 1)

We propose a coordinate-ascent-type algorithm for solving the dual LP. The main idea is to select edges \((i, j) \in E\) according to some update schedule: for each selected edge \((i, j) \in E\) we then replace the old values of \(u'_{i,j}, \phi'_i, \) and \(\theta'_j\) by new values such that the dual cost function is increased (or at least not decreased). Practically, this means that we have to find a \(\pi'_{i,j}\) such that \(h'(\pi'_{i,j}) \geq h'(u'_{i,j})\), where

\[
h'(u'_{i,j}) \triangleq \min_{a_i \in A_i} \langle -u'_i, a_i \rangle + \min_{b_j \in B_j} \langle -v'_j, b_j \rangle.
\]

A simple way to achieve this is by setting

\[
\pi'_{i,j} \triangleq \arg\max_{u'_{i,j}} h'(u'_{i,j}).
\]

The variables \(\phi'_i\) and \(\theta'_j\) are then updated accordingly so that we obtain a new (dual) feasible point.
Coordinate-Ascent Algorithm (Part 2)

**Lemma**

The function $H'(u'_{i,j})$ is maximised by any value $u'_{i,j}$ that lies in the closed interval between

$$(S'_{i,0} - S'_{i,1}) \quad \text{and} \quad -(T'_{j,0} - T'_{j,1}),$$

where

$$S'_{i,0} \triangleq - \min_{a_i \in A_i} \langle -\hat{u}_i, \tilde{a}_i \rangle, \quad T'_{j,0} \triangleq - \min_{b_j \in B_j} \langle -\hat{v}_j, \tilde{b}_j \rangle,$$

$$S'_{i,1} \triangleq - \min_{a_i \in A_i} \langle -\hat{u}_i, \tilde{a}_i \rangle, \quad T'_{j,1} \triangleq - \min_{b_j \in B_j} \langle -\hat{v}_j, \tilde{b}_j \rangle.$$

Note that the differences $S'_{i,0} - S'_{i,1}$ and $T'_{j,0} - T'_{j,1}$, which are required for computing $u'_{i,j}$, can be obtained very efficiently by using the min-sum algorithm.

Coordinate-Ascent Algorithm (Part 3)

Problematic Contour for Coordinate-Ascent Algorithm

References

For more info, see e.g.
