Optimization for decoding

Speaker: Issam Laradji
Problem Statement

- Consider the problem of decoding for Markov random fields (MRF)
  - Determine the assignment of \( n \) random variables \( X_1, \ldots, X_n \)
  
  \[ \chi_i = \{0, 1, \ldots, k - 1\} \]
Problem Statement

- Consider the problem of decoding for markov random fields (MRF)
  - Determine the assignment of $n$ random variables $X_1, \ldots, X_n$
    $$\chi_i = \{0, 1, \ldots, k - 1\}$$

- The maximum a posteriori (MAP) - or the most likely labeling
  $$\arg\max_x p(x) = \arg\max_x w^T F(x)$$
Problem Statement

● Consider the problem of decoding for Markov random fields (MRF)
  ○ Determine the assignment of $n$ random variables $X_1, \ldots X_n$
    \[ \chi_i = \{0, 1, \ldots, k - 1\} \]

● The maximum a posteriori (MAP) - or the most likely labeling
  \[ \arg\max_x p_w(x) = \arg\max_x w^T F(x) \]

● No computation of entropy required
Problem Statement

- Consider the problem of decoding for Markov random fields (MRF)
  - Determine the assignment of $n$ random variables $X_1, \ldots, X_n$
    \[ \chi_i = \{0, 1, \ldots, k - 1\} \]

- The maximum a posteriori (MAP) - or the most likely labeling
  \[ \arg\max_x p_w(x) = \arg\max_x w^T F(x) \]

- No computation of entropy required
  \[ \sup_{\mu \in \mathcal{M}} \{w^T \mu + H(p_\mu)\} \]
Problem Statement

- Consider the problem of decoding for Markov random fields (MRF)
  - Determine the assignment of $n$ random variables $X_1, \ldots, X_n$
  
  $\chi_i = \{0, 1, \ldots, k - 1\}$

- The maximum a posteriori (MAP) - or the most likely labeling

\[
\arg\max_x p_w(x) = \arg\max_x w^T F(x)
\]

- No computation of entropy required

\[
\sup_{\mu \in \mathcal{M}} \{w^T \mu + H(p_\mu)\}
\]
Problem Statement

- For pairwise UGM the decoding problem is integer-quadratic programming,
  \[
  \arg \max_{x \in \{0,1\}^k} p_w(x) = \arg \max_{x \in \{0,1\}^k} w^T F(x)
  \]

- Because
  \[
  w^T F(x) = \sum_{s \in V} w_s x_s + \sum_{(s,t) \in E} w_{st} x_s x_t
  \]

- Integer Programming is NP-hard
- Cast (relax) it to have linear constraints
Problem Statement

- Quadratic unconstrained binary optimization (QUBO)
  \[
  \arg \max_{x \in \{0,1\}^k} p_w(x) = \arg \max_{x \in \{0,1\}^k} w^T F(x)
  \]

- Quadratic programming,
  \[
  \arg \max_{x \in \{0,1\}^k} p_w(x) = \arg \max_{x \in \{0,1\}^k} w^T F(x)
  \]
Quadratic programming to linear programming

- Quadratic programming,

\[
\arg \max_{x \in \{0,1\}^k} p_w(x) = \arg \max_{x \in \{0,1\}^k} w^T F(x)
\]

- Achieve a tighter bound by converting it to a linear program (best paper award)

\[
\arg \max_{x \in \{0,1\}^k} w^T F(x) = \arg \max_{\mu \in M(G)} w^T \mu
\]

\[
M(G) = \{\mu \in \mathbb{R}^d \mid \exists w \in \mathbb{R}^d \text{ s.t. } \mu = \mathbb{E}_{P(x,w)}[F(x)]\}
\]
Proof for equivalence

\[
\max_p \sum_x p(x) w^T F(x) = \max_x w^T F(x)
\]

Original formulation

\[
\max_p \sum_x p(x) w^T F(x) = \max_p w^T \sum_x p(x) F(x)
\]

\[
= \max_p w^T \mathbb{E}[F(x)]
\]

\[
= \max_p w^T \mu_p
\]

\[
= \max_{\mu \in \mathcal{M}(G)} w^T \mu
\]

Desired formulation
Marginal Polytope: the set of mean vectors that can arise from some joint distribution $\mu$

The dimension of $\mu$ is $2|V| + 2^2|E|$
Linear programming global constraint

- Global constraint: The edge marginals in $\mu$ must arise from a common joint distribution

$$\max_{\mu \in M(G)} w^T \mu$$

- Number of constraints is exponential in the number of edges

- Therefore we relax the linear program to achieve a polynomial time approximation
First-order relaxation

- Pairwise relaxation (Assume dependency between pairs only)

\[
\text{LOCAL}(G) = \left\{ \mu \in \mathbb{R}^d \left| \begin{array}{l}
\sum_{x_j} \mu_{ij}(x_i, x_j) = \mu_i(x_i) \quad \forall ij \in E, x_i \\
\sum_{x_i} \mu_{ij}(x_i, x_j) = \mu_j(x_j) \quad \forall ij \in E, x_j \\
\sum_{x_i} \mu_i(x_i) = 1 \quad \forall i \in V \\
\mu_i(x_i) \geq 0, \quad \mu_{ij}(x_i, x_j) \geq 0
\end{array} \right. \right\}
\]

\[\max_{\mu \in M(G)} w^T \mu \approx \max_{\mu \in \text{LOCAL}(G)} w^T \mu\]
First-order relaxation

\( \text{LOCAL}(G) = \left\{ \mathbf{\mu} \in \mathbb{R}^d \left| \begin{array}{l}
\sum_{x_j} \mu_{ij}(x_i, x_j) = \mu_i(x_i) \quad \forall ij \in E, x_i \\
\sum_{x_i} \mu_{ij}(x_i, x_j) = \mu_j(x_j) \quad \forall ij \in E, x_j \\
\sum_{x_i} \mu_i(x_i) = 1 \\
\mu_i(x_i) \geq 0, \quad \mu_{ij}(x_i, x_j) \geq 0 \\
\end{array} \right. \right\} \)
Higher order relaxations

- Edge marginals are consistent on larger subsets

\[
\text{TRI}(G) = \left\{ \mu \geq 0 \mid \exists \tau \geq 0, \begin{align*}
\sum_{x_j} \mu_{ij}(x_i, x_j) &= \mu_i(x_i) \quad \forall j \in E, x_i \\
\sum_{x_i} \mu_{ij}(x_i, x_j) &= \mu_j(x_j) \quad \forall ij \in E, x_i \\
\sum_{x_i} \mu_i(x_i) &= 1 \quad \forall i \in V \\
\tau_{ij}(x_i, x_j) &= \mu_{ij}(x_i, x_j) \quad \forall ij \in E, x_i, x_j \\
\sum_{x_k} \tau_{ijk}(x_i, x_j, x_k) &= \tau_{ij}(x_i, x_j), \quad \forall i, j, k
\end{align*} \right\}.
\]

- Constraints grow exponentially in the size of the clusters considered
Cutting-plane algorithm

- Few carefully chosen constraints would suffice
  - An integer solution is a MAP assignment
- Solve pairwise LP then find valid constraints to add to the relaxation
Cutting-plane algorithm
Cutting-plane algorithm

- Add valid cycle constraints - cycle of the graph should be consistent with some joint distribution

\[
\text{CYCLE}(C) = \left\{ \mu \in \mathbb{R}^d \mid \exists \tau_C \geq 0, \quad \sum_{x_C \setminus i,j} \tau_C(x_C) = \mu_{ij}(x_i, x_j) \quad \forall i,j \in C, x_i, x_j \right\}.
\]

\(^1\)We use \( C \) to refer to both a set of edges (e.g., with notation \( ij \in C \)), and the variables involved in these edges. The notation \( x_C \) refers to an assignment to all of the variables in the cycle \( C \), and \( C \setminus \{i,j\} \) refers to the set of variables in \( C \) except for \( i \) or \( j \). Also, \( \sum_{x_C \setminus i,j} \tau_C(x_C) \) means the sum over all assignments \( x_{C \setminus i,j} \) to the variables in \( C \setminus \{i,j\} \) of \( \tau_C(x_{C \setminus i,j}, x_i, x_j) \), where \( x_i \) and \( x_j \) are instantiated outside of the sum.
Cutting-plane algorithm

1. Solve the LP relaxation (in Iteration 1, use the pairwise relaxation).
2. Construct the projection graph $G_\pi$ and pseudomarginals $\mu_\pi$ using Eqs. 3.9-3.11.
3. Run $\text{SeperateCycles}(G_\pi, \mu_\pi)$ to see if there are any violated cycle inequalities.
4. Add all cycle inequalities returned by Step 3 to the LP relaxation.
5. Return to Step 1, but now solve using the tighter relaxation.
2. Construct the projection graph $G_\pi$ and pseudomarginals $\mu_\pi$ using Eqs. 3.9-3.11.

$$V_\pi = \bigcup_{i \in V} \pi_i,$$

$$E_\pi \subseteq \{(\pi_i^q, \pi_j^r) \mid (i, j) \in E, q \leq |\pi_i|, r \leq |\pi_j|\}. \quad \pi_i^q : \chi_i \to \{0, 1\}$$

$$\mu_\pi^m(x_m) = \sum_{s_i \in \chi_i : \pi_i^q(s_i) = x_m} \mu_i(s_i) \quad \forall m = \pi_i^q \in V_\pi$$

$$\mu_\pi^{mn}(x_m, x_n) = \sum_{s_i \in \chi_i : \pi_i^q(s_i) = x_m, s_j \in \chi_j : \pi_j^r(s_j) = x_n} \mu_{ij}(s_i, s_j) \quad \forall mn = (\pi_i^q, \pi_j^r) \in E_\pi.$$
3. Run \text{SeparateCycles}(G_{\pi}, \mu_{\pi}) to see if there are any violated cycle inequalities.

Algorithm \text{SeparateCycles}(G_{\pi}, \mu_{\pi})

1. // Initialize the auxiliary graph used in the shortest path computation.
2. let $G' = (V', E')$, where
3. \quad $V' = \cup_{i \in V_{\pi}} \{i_1, i_2\}$
4. \quad $E' = \cup_{(i,j) \in E_{\pi}} \{(i_1, i_2), (i_1, j_2), (i_2, j_1), (i_2, j_2)\}$
5. // Setup the edge weights.
6. for each edge $ij \in E_{\pi}$
7. \quad $w(i_1, j_2) = \mu_{ij}^\pi(0, 0) + \mu_{ij}^\pi(1, 1)$ // Cut
8. \quad $w(i_2, j_1) = \mu_{ij}^\pi(0, 0) + \mu_{ij}^\pi(1, 1)$ // Cut
9. \quad $w(i_1, j_1) = \mu_{ij}^\pi(0, 1) + \mu_{ij}^\pi(1, 0)$ // Not cut
10. \quad $w(i_2, j_2) = \mu_{ij}^\pi(0, 1) + \mu_{ij}^\pi(1, 0)$ // Not cut
11. // Run the shortest path algorithm, once for each node.
12. for each node $i \in V_{\pi}$
13. \quad // Find shortest path $P_i$ from $i_1$ to $i_2$ on graph $G'$ with weights $w$
14. \quad $P_i = \text{ShortestPath}(i_1, i_2, G', w)$
15. \quad // Make sure that this is a simple cycle in $G_{\pi}$.
16. \quad if $\exists j \neq i$ such that $j_1, j_2 \in P_i$
17. \quad \quad Discard $P_i$.
18. return $\{P_i : w(P_i) < 1\}$

Figure 3-4: Given the projection graph $G_{\pi} = (V_{\pi}, E_{\pi})$ and edge pseudomarginals $\mu_{\pi}$, find the most violated cycle inequality.
3. Run \textbf{SeparateCycles}(G_{\pi}, \mu_{\pi}) to see if there are any violated cycle inequalities.

Figure 3-5: Illustration of graph used in shortest path algorithm for finding the most violated cycle inequality. The dashed edges denote cut edges (i.e., if used in the shortest path, they are assigned to $F$), while the solid edges denote edges that are not cut. The algorithm is as follows: To find the most violated cycle inequality on a cycle involving node $j$, find the shortest path from $j_1$ to $j_2$ in the graph (edge weights are discussed in Section 3.4.2). To find the most violated cycle inequality overall, considering all cycles, repeat this for every node (e.g., also look for the shortest path from $k_1$ to $k_2$). The red and blue paths demonstrate two different cycle inequalities. The red path, from $k_1$ to $k_2$, denotes the cycle inequality $C = \{ki, ij, jk\}$, $F = \{jk\}$. The blue path, from $j_1$ to $j_2$, denotes the cycle inequality $C = \{ji, ik, kj\}$, $F = C$, i.e. all three edges are cut. Since the paths begin in the top component and end in the bottom component, each path must have an odd number of cut edges. Thus, $|F|$ is always odd, as required to obtain a valid inequality.
Cutting-plane algorithm

1. Solve the LP relaxation (in Iteration 1, use the pairwise relaxation).
2. Construct the projection graph $G_\pi$ and pseudomarginals $\mu_\pi$ using Eqs. 3.9-3.11.
3. Run `SeperateCycles(G_\pi, \mu_\pi)` to see if there are any violated cycle inequalities.
4. Add all cycle inequalities returned by Step 3 to the LP relaxation.
5. Return to Step 1, but now solve using the tighter relaxation.
Semidefinite programming

- Some NP-hard combinatorial optimization problems have convex relaxations that are semidefinite programs.
- SDP relaxation is very tight in practice

\[
SDP: \quad \text{minimize} \quad C \bullet X \\
\text{s.t.} \quad A_i \bullet X = b_i, \ i = 1, \ldots, m, \\
X \succeq 0,
\]
Semidefinite programming

\[ SDP : \text{ minimize } \quad C \cdot X \]

\[ \text{s.t.} \quad A_i \cdot X = b_i, \quad i = 1, \ldots, m, \]

\[ X \succeq 0, \]

\[ LP : \text{ minimize } \quad c \cdot x \]

\[ \text{s.t.} \quad a_i \cdot x = b_i, \quad i = 1, \ldots, m \]

\[ x \in \mathbb{R}^n_+. \]

\[ A_i = \begin{pmatrix} a_{i1} & 0 & \ldots & 0 \\ 0 & a_{i2} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & a_{in} \end{pmatrix}, \quad i = 1, \ldots, m, \quad \text{and} \quad C = \begin{pmatrix} c_1 & 0 & \ldots & 0 \\ 0 & c_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & c_n \end{pmatrix}. \]
Maxcut problem

- Determine a subset $S$ of the nodes $N$ for which the sum of the weights of the edges that cross from $S$ to its complement $S^\complement$ is maximized.

\[
\text{MAXCUT} : \quad \text{maximize}_{x} \quad \frac{1}{4} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} (1 - x_{i}x_{j})
\]

s.t. \quad x_{j} \in \{-1, 1\}, \quad j = 1, \ldots, n.

\[
\text{MAXCUT} : \quad \text{maximize}_{x,Y} \quad \frac{1}{4} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} - W \bullet Y
\]

s.t. \quad Y_{jj} = 1, \quad j = 1, \ldots, n

\quad Y = xx^T.
Maxcut problem

- Remove Rank-1 restriction

\[ Y = xx^T \]

\[ \text{MAXCUT} : \quad \text{maximize}_{y,x} \quad \frac{1}{4} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} - W \bullet Y \]

\[ \text{s.t.} \quad Y_{jj} = 1, \quad j = 1, \ldots, n \]

\[ Y = xx^T. \]

To SDP formulation

\[ \text{RELAX} : \quad \text{maximize}_y \quad \frac{1}{4} \sum_{i=1}^{n} \sum_{j=1}^{n} w_{ij} - W \bullet Y \]

\[ \text{s.t.} \quad Y_{jj} = 1, \quad j = 1, \ldots, n \]

\[ Y \succeq 0. \]
Summary

- Classic decoding using Quadratic programming with integer constraints
  - Can be relaxed to Quadratic programming with linear constraints (polynomial time for pairwise)
- Quadratic programming to linear programming (tighter)
  - Relaxation based on local consistency of mean vectors
  - Higher-order relaxation leads to an NP-hard optimization problem
- Cutting-plane algorithm for tight solutions that is computationally tractable
  - Find and add violated constraints to the optimization
- NP-hard optimization problems like MAXCUT have convex relaxations that are semidefinite programs, which are very tight in practice.
Thank you!