Lagrangian Relaxation for MAP Inference

SPFLODD
October 8, 2013
Outline

• An elegant example of a relaxation to TSP
• A common problem in NLP: finding consensus
• Basic Lagrangian relaxation
• Solving the problem with subgradient
• AD$^3$: an alternative approach to decomposition and optimization using the augmented Lagrangian
Traveling Salesman Problem

• Given: a graph \((V, E)\) with edge weight function \(\theta\)

• Tour: a subset of \(E\) corresponding to a path that starts and ends in the same place, and visits every other node exactly once.

• TSP: Find the maximum-scoring tour.
  – NP-hard

\[
\max_{y \in \mathcal{Y}_{\text{tour}}} \sum_{e \in E} y_e \theta_e
\]
Another Problem

• 1-tree: a tree on edges for \{2, ..., |V|\}, plus two edges from \(E\) that link the tree to vertex 1.
  – All tours are 1-trees.
  – All 1-trees where every vertex has degree 2 are tours.
  – Easy to solve.
Held and Karp (1971)

\[ \mathcal{V}_{\text{tour}} = \left\{ y : y \in \mathcal{V}_{1\text{-tree}} \land \forall i \in \{1, \ldots, |V|\}, \sum_{e : i \in e} y_e = 2 \right\} \]

\[ \max_{y \in \mathcal{V}_{\text{tour}}} \sum_{e \in E} y_e \theta_e \quad \text{transforming the constraints} \]

\[ \max_{y \in \mathcal{V}_{1\text{-tree}}} \sum_{e \in E} y_e \theta_e \quad \text{s.t.} \quad \forall i, \sum_{e : i \in e} y_e = 2 \]

\[ L(u) = \max_{y \in \mathcal{V}_{1\text{-tree}}} \sum_{e \in E} y_e \theta_e + \sum_{i=1}^{|V|} u_i \left( \sum_{e : i \in e} y_e - 2 \right) \]
LR Algorithm for TSP

1. Initialize $u^{(0)} = 0$
2. Repeat for $k = 1, 2, ...$

$$y^{(k)} \leftarrow \arg \max_{y \in \mathcal{Y}_{1\text{-tree}}} \sum_{e \in E} y_e \theta_e + \sum_{i=1}^{\left| \mathcal{V} \right|} u_i^{(k-1)} \left( \sum_{e: i \in e} y_e - 2 \right)$$

$$\forall i, u_i^{(k)} \leftarrow u_i^{(k-1)} - \delta_k \left( \sum_{e: i \in e} y_e - 2 \right)$$

If this converges to a solution that satisfies the constraints, it is a solution to the TSP.
Lagrangian Relaxation, More Generally

• Assume a linear scoring function that is “hard” to maximize.

\[
\max_{y \in \mathcal{Y}} \theta^\top y
\]

• Rewrite the problem as something easier, with linear constraints (relaxation):

\[
\mathcal{Y} = \{ y \in \mathcal{Y}' : Ay = b \}
\]

\[
\max_{y \in \mathcal{Y}'} \theta^\top y \quad \text{s.t.} \quad Ay = b
\]

• Tackle the dual problem:

\[
\min_u \max_{y \in \mathcal{Y}'} \theta^\top y + u^\top (Ay - b)
\]
Theory

• The dual function (of \( u \)) upper bounds the MAP problem.
• A subgradient algorithm can be applied to minimize the dual; it will converge in the limit.
• If the solution to the dual problem satisfies the constraints, it is also a solution to the primal (relaxed) problem (\( \mathcal{Y}' \)).
  – If the relaxation is \( \text{tight} \), we also have a solution to the original primal problem (\( \mathcal{Y} \)).
Dual Decomposition (A Special Case of LR)

• Assume the objective decomposes into two parts, coupled only through the linear constraints:

\[
\begin{align*}
\max_{y \in \mathcal{Y}, z \in \mathcal{Z}} & \quad \theta^\top y + \psi^\top z \\
\text{s.t.} & \quad A y + C z = b
\end{align*}
\]

• The relaxation:

\[
\max_{y \in \mathcal{Y}, z \in \mathcal{Z}} \quad \theta^\top y + \psi^\top z \equiv \begin{pmatrix} \max_{y \in \mathcal{Y}} \theta^\top y, \max_{z \in \mathcal{Z}} \psi^\top z \end{pmatrix}
\]
Dual Decomposition

\[
\begin{align*}
\min_{\mathbf{u}} \max_{\mathbf{y} \in \mathcal{Y}, \mathbf{z} \in \mathcal{Z}} \theta^\top \mathbf{y} + \psi^\top \mathbf{z} + \mathbf{u}^\top (A\mathbf{y} + C\mathbf{z} - \mathbf{b})
\end{align*}
\]

1. Initialize \( \mathbf{u}^{(0)} = 0 \)
2. Repeat for \( k = 1, 2, \ldots \):

\[
\begin{align*}
\mathbf{y}^{(k)} &\leftarrow \max_{\mathbf{y} \in \mathcal{Y}} \theta^\top \mathbf{y} + \mathbf{u}^{(k-1)}^\top A\mathbf{y} \\
\mathbf{z}^{(k)} &\leftarrow \max_{\mathbf{z} \in \mathcal{Z}} \psi^\top \mathbf{z} + \mathbf{u}^{(k-1)}^\top C\mathbf{z} \\
\mathbf{u}^{(k)} &\leftarrow \mathbf{u}^{(k-1)} - \delta_k \left( A\mathbf{y}^{(k)} + C\mathbf{z}^{(k)} - \mathbf{b} \right)
\end{align*}
\]
Consensus Problems in NLP

• Key example:
  – Find the jointly-best parse (under a WCFG) and sequence labeling (under an HMM); see Rush et al. (2010)

• Other examples:
  – Finding a lexicalized phrase structure parse that is jointly-best under a WCFG and a dependency model (Rush et al., 2010)
Example Run \((k = 1)\)

\[\forall i \in \{1, \ldots, n\}, \forall N \in \mathcal{N}, y[N, i, i] = z[N, i]\]

\[
u[N, i]^{(1)} = u[N, i]^{(0)} - \delta_k \left( y[N, i, i]^{(1)} - z[N, i]^{(1)} \right)\]

\[
u[A, 1] = -1\]
\[
u[N, 2] = -1\]
\[
u[V, 5] = -1\]
\[
u[N, 1] = 1\]
\[
u[V, 2] = 1\]
\[
u[N, 5] = 1\]
Example Run (k = 2)

\(\forall i \in \{1, \ldots, n\}, \forall N \in \mathcal{N}, y[N, i, i] = z[N, i]\)

\[
u[N, i]^{(2)} = u[N, i]^{(1)} - \delta_k \left( y[N, i, i]^{(2)} - z[N, i]^{(2)} \right)
\]

\[\downarrow u[N, 1]\]
\[\downarrow u[V, 1]\]
\[\uparrow u[A, 1]\]
\[\uparrow u[N, 1]\]
Example Run \((k = 3)\)

\[
\forall i \in \{1, \ldots, n\}, \forall N \in \mathcal{N}, y[N, i, i] = z[N, i]
\]
“Certificate”

• Proof that we have solved the original problem: constraints hold.
  – This is easy to check given $y$ and $z$.

• In published NLP papers so far, this happens most of the time (better than 98%).
What can go wrong?

• It can take many iterations to converge.
• Oscillation between different solutions; failure to agree.
  – Suggested solution: add more variables for “bigger parts” and enforce agreement among them with more constraints.
What does this have to do with ILP?

• The linear constraints are expressed in terms of an integer-vector representation of the output space.
  – Just like when we treated decoding as an ILP.
• The subproblems *could* be expressed as ILPs, though we’d prefer to use poly-time combinatorial algorithms to solve them if we can.
Consensus Problems, Revisited

• What if we just have a hard combinatorial optimization problem?
  – There isn’t always a straightforward decomposition.

• Martins et al. (2011): shatter a decoding problem into many “small” subproblems (instead of two “big” ones).
  – Instead of dynamic programming as a subroutine, LP relaxations of “small” subproblems.
  – Extra LP relaxation step.
Martins’ Alternative Formulation

• Original problem:
  \[
  \max_{y_1 \in \mathcal{Y}_1, \ldots, y_S \in \mathcal{Y}_S, w \in \mathbb{R}^D} \sum_{s=1}^{S} \theta_s^T y_s \\
  \text{s.t. } \forall s, A_s w = y_s
  \]

• Convex relaxation:
  \[
  \max_{y_1 \in \text{conv} \mathcal{Y}_1, \ldots, y_S \in \text{conv} \mathcal{Y}_S, w \in \mathbb{R}^D} \sum_{s=1}^{S} \theta_s^T y_s \\
  \text{s.t. } \forall s, A_s w = y_s
  \]

• Dual:
  \[
  \min_{u_1, \ldots, u_S} \max_{y_1 \in \text{conv} \mathcal{Y}_1, \ldots, y_S \in \text{conv} \mathcal{Y}_S, w \in \mathbb{R}^D} \sum_{s=1}^{S} \theta_s^T y_s + \sum_{s} u_s^T (y_s - A_s w)
  \]
Augmented Lagrangian
(Hestenes, 1969; Powell, 1969)

\[
\min_{u_1, \ldots, u_S} \max_{y_1 \in \text{conv}(\mathcal{Y}_1), \ldots, y_S \in \text{conv}(\mathcal{Y}_S), w \in \mathbb{R}^D} \sum_{s=1}^{S} \theta_s^T y_s + \sum_{s} u_s^T (y_s - A_s w) + \frac{\rho}{2} \sum_{s} \|y_s - A_s w\|^2
\]
Alternating Directions Method of Multipliers

(Gabay and Mercier, 1976; Glowinski and Marroco, 1975)

Dual Decomposition (AD$^3$)

• Alternate between updating $\mathbf{y}$ and $\mathbf{w}$:

$$\forall s, \mathbf{y}_s \leftarrow \arg \max_{\mathbf{y}_s \in \text{conv}(Y_s)} \quad \theta_s^\top \mathbf{y}_s + \mathbf{u}_s^\top \mathbf{y}_s + \frac{\rho}{2} \| \mathbf{y}_s - \mathbf{A}_s \mathbf{w} \|^2_2$$

$$\mathbf{w} \leftarrow \arg \max_{\mathbf{w}} \sum_s \mathbf{u}_s^\top \mathbf{A}_s \mathbf{w} + \frac{\rho}{2} \sum_s \| \mathbf{y}_s - \mathbf{A}_s \mathbf{w} \|^2_2$$

• Subgradient step for dual variables $\mathbf{u}$ is similar to before:

$$\forall s, \mathbf{u}_s^{(k)} \leftarrow \mathbf{u}_s^{(k-1)} - \delta_k (\mathbf{y}_s - \mathbf{A}_s \mathbf{w})$$
Massive Decomposition

• Most extreme: every factor (MN) or “part” is a separate subproblem.

\[ \forall s, \mathbf{y}_s \leftarrow \arg \max_{\mathbf{y}_s \in \text{conv}(\mathcal{Y}_s)} \mathbf{\theta}_s^\top \mathbf{y}_s + \mathbf{u}_s^\top \mathbf{y}_s + \frac{\rho}{2} \| \mathbf{y}_s - \mathbf{A}_s \mathbf{w} \|_2^2 \]

• Some kinds of MN factors can be solved very efficiently ...
XOR, OR, OR-with-Output
Solvable in $O(K \log K)$
AD$^3$ and “Big” Subproblems?

• Return to Rush and Collins’ example.
  – One subproblem is “WCFG” and one is “HMM tagger.”

\[
\forall s, y_s \leftarrow \arg \max_{y_s \in \text{conv}(\mathcal{Y}_s)} \theta_s^\top y_s + u_s^\top y_s + \frac{\rho}{2} ||y_s - A_s w||_2^2
\]

  – In dependency parsing, “max arborescence” might be a subproblem.

  – Why can’t we use AD$^3$?
Pros and Cons

• Con: Subproblems are now *quadratic*.
  – Linear decoders as subroutines?
• Con: Fractional solutions.
• Pro: Better stopping criteria: residuals.
  – Primal residuals measure amount by which primal constraints are violated.
  – Dual residuals measure amount by which dual optimality is violated.
• Pro: Certificates as before (for each $s$, $A_s w = y_s$)
Convergence of $AD^3$ vs. Subgradient

Dependency parsing:
• ADMM = $AD^3$
• Sec Ord = Second order model for which subgradient optimization is possible
• Full = second order model with all-siblings, directed paths, and non-projective arcs
Take-Home Messages

• Dual decomposition is useful for consensus problems.
  – Subgradient DD when there are a few subproblems with good specialized solvers.
  – AD$^3$ when you’ve got a big problem with lots of hard and soft constraints. (There is a library.)

• Attractive guarantees (cf. beam search).

• Only MAP inference.
References


• “Alternating directions dual decomposition” by A. Martins et al., arXiv 1212.6550.