MAP Inference with MILP
Reminders

• Homework 2: BP for Syntax Trees
  – Out: Sat, Sep. 28
  – Due: Sat, Oct. 12 at 11:59pm

• Last chance to switch between 10-418 / 10-618 is October 7th (drop deadline)

• Today’s after-clas office hours are un-cancelled (i.e. I am having them)
MBR DECODING
Minimum Bayes Risk Decoding

• Suppose we given a loss function \( l(y', y) \) and are asked for a single tagging

• How should we choose just one from our probability distribution \( p(y|x) \)?

• A minimum Bayes risk (MBR) decoder \( h(x) \) returns the variable assignment with minimum expected loss under the model’s distribution

\[
h_{\theta}(x) = \arg\min_{\hat{y}} \mathbb{E}_{y \sim p_{\theta}(.|x)} [l(\hat{y}, y)]
\]

\[
= \arg\min_{\hat{y}} \sum_{y} p_{\theta}(y| x) l(\hat{y}, y)
\]
Minimum Bayes Risk Decoding

\[
h_\theta(x) = \arg\min_{\hat{y}} \mathbb{E}_{y \sim p_\theta(\cdot | x)}[\ell(\hat{y}, y)]
\]

Consider some example loss functions:

The **Hamming loss** corresponds to accuracy and returns the number of incorrect variable assignments:

\[
\ell(\hat{y}, y) = \sum_{i=1}^{V} (1 - \mathbb{I}(\hat{y}_i, y_i))
\]

The MBR decoder is:

\[
\hat{y}_i = h_\theta(x)_i = \arg\max_{\hat{y}_i} p_\theta(\hat{y}_i | x)
\]

This decomposes across variables and requires the variable marginals.
Minimum Bayes Risk Decoding

\[ h_\theta(x) = \arg\min_\hat{y} \mathbb{E}_{y \sim p_\theta(\cdot | x)}[\ell(\hat{y}, y)] \]

Consider some example loss functions:

The **0-1 loss function** returns 1 only if the two assignments are identical and 0 otherwise:

\[ \ell(\hat{y}, y) = 1 - \mathbb{I}(\hat{y}, y) \]

The MBR decoder is:

\[ h_\theta(x) = \arg\min_\hat{y} \sum_y p_\theta(y | x)(1 - \mathbb{I}(\hat{y}, y)) \]

\[ = \arg\max_\hat{y} p_\theta(\hat{y} | x) \]

which is exactly the MAP inference problem!
Linear Programming

Whiteboard

- Example of Linear Program in 2D
- LP Standard Form
- Converting an LP to Standard Form
- LP and its Polytope
- Simplex algorithm (tableau method)
- Interior points algorithm(s)
Integer Linear Programming

Whiteboard

– Example of an ILP in 2D
– Example of an MILP in 2D
Background: Nonconvex Global Optimization

Goal: optimize over the blue surface.
Background: Nonconvex Global Optimization

Goal: optimize over the blue surface.
Background: Nonconvex Global Optimization

**Relaxation**: provides an upper bound on the surface.
Background: Nonconvex Global Optimization

**Branching:** partitions the search space into subspaces, and enables tighter relaxations.
Background: Nonconvex Global Optimization

**Branching:** partitions the search space into subspaces, and enables tighter relaxations.
Background: Nonconvex Global Optimization

**Branching:** partitions the search space into subspaces, and enables tighter relaxations.
Background: Nonconvex Global Optimization

The max of all relaxed solutions for each of the partitions is a global upper bound.
Background: Nonconvex Global Optimization

We can **project** a relaxed solution onto the feasible region.
Background: Nonconvex Global Optimization

The **incumbent** is **$\varepsilon$-optimal** if the relative difference between the **global upper bound** and the **incumbent score** is less than $\varepsilon$. 
How much should we subdivide?
How much should we subdivide?

**BRANCH-AND-BOUND**

- Method for *recursively subdividing* the search space
- *Subspace order* can be determined heuristically (e.g. best-first search with depth-first plunging)
- *Prunes* subspaces that can’t yield better solutions
Background: Nonconvex Global Optimization

If the **subspace upper bound** is worse than the **current incumbent**, we can **prune** that subspace.
Background: Nonconvex Global Optimization

If the \textit{subspace upper bound} is worse than the \textit{current incumbent}, we can \textit{prune} that subspace.
Limitations:

Branch-and-Bound for the Viterbi Objective

• The Viterbi Objective
  – Nonconvex
  – NP Hard to solve (Cohen & Smith, 2010)

• Branch-and-bound
  – Kind of tricky to get it right...
  – Curse of dimensionality kicks in quickly
    • Nonconvex quadratic optimization by LP-based branch-and-bound usually fails with more than 80 variables (Burer and Vandenbussche, 2009)
    • Our smallest (toy) problems have hundreds of variables

• Preview of Experiments
  – We solve 5 sentences, but on 200 sentences, we couldn’t run to completion
  – Our (hybrid) global search framework incorporates local search
  – This hybrid approach sometimes finds higher likelihood (and higher accuracy) solutions than pure local search
BRANCH-AND-BOUND INGREDIENTS

Mathematical Program
Relaxation
Projection
(Branch-and-Bound Search Heuristics)
Background: Nonconvex Global Optimization

We solve the relaxation using the Simplex algorithm.
Background: Nonconvex Global Optimization

We can **project** a relaxed solution onto the feasible region.
Integer Linear Programming

Whiteboard

– Branch and bound for an ILP in 2D
Algorithm 2.1 Branch-and-bound

**Input:** Minimization problem instance $R$.

**Output:** Optimal solution $x^*$ with value $c^*$, or conclusion that $R$ has no solution, indicated by $c^* = \infty$.

1. Initialize $\mathcal{L} := \{R\}$, $\hat{c} := \infty$. \hspace{1cm} [init]
2. If $\mathcal{L} = \emptyset$, stop and return $x^* = \hat{x}$ and $c^* = \hat{c}$. \hspace{1cm} [abort]
3. Choose $Q \in \mathcal{L}$, and set $\mathcal{L} := \mathcal{L} \setminus \{Q\}$. \hspace{1cm} [select]
4. Solve a relaxation $Q_{\text{relax}}$ of $Q$. If $Q_{\text{relax}}$ is empty, set $\tilde{c} := \infty$. Otherwise, let $\tilde{x}$ be an optimal solution of $Q_{\text{relax}}$ and $\tilde{c}$ its objective value. \hspace{1cm} [solve]
5. If $\tilde{c} \geq \hat{c}$, goto Step 2. \hspace{1cm} [bound]
6. If $\tilde{x}$ is feasible for $R$, set $\hat{x} := \tilde{x}$, $\hat{c} := \tilde{c}$, and goto Step 2. \hspace{1cm} [check]
7. Split $Q$ into subproblems $Q = Q_1 \cup \ldots \cup Q_k$, set $\mathcal{L} := \mathcal{L} \cup \{Q_1, \ldots, Q_k\}$, and goto Step 2. \hspace{1cm} [branch]
Branch and Bound

Algorithm 2.1

**Input**: Minimization problem instance \( R \).

**Output**: Optimal solution \( x^* \) with value \( c^* \), or conclude that \( R \) has no solution, indicated by \( c^* = \infty \).

1. Initialize \( L := \{ R \} \), \( \hat{c} := \infty \).

2. If \( L = \emptyset \), stop and return \( x^* = \hat{x} \) and \( c^* = \hat{c} \).

3. Choose \( Q \in L \), and set \( L := L \setminus \{ Q \} \).

4. Solve a relaxation \( Q_{\text{relax}} \) of \( Q \). If \( Q_{\text{relax}} \) is empty, set \( \hat{c} := \infty \). Otherwise, let \( \hat{x} \) be an optimal solution of \( Q_{\text{relax}} \) and \( \hat{c} \) its objective value.

5. If \( \hat{c} \geq \hat{c} \), go to Step 2.

6. If \( \hat{x} \) is feasible for \( R \), set \( \hat{x} := \hat{x} \), \( \hat{c} := \hat{c} \), and go to Step 2.

7. Split \( Q \) into subproblems \( Q = Q_1 \cup \ldots \cup Q_k \), set \( L := L \cup \{ Q_1, \ldots, Q_k \} \), and go to Step 2.

The intention of the bounding in Step 5 is to avoid a complete enumeration of all potential solutions of \( R \), which usually exponentially many. In order for bounding to be effective, good lower (dual) bounds \( \hat{c} \) and upper (primal) bounds \( \hat{c} \) must be available. Lower bounds are calculated with the help of a relaxation \( Q_{\text{relax}} \) which should be easy to solve. Upper bounds can be found during the branch-and-bound algorithm in Step 6, but they can also be generated by primal heuristics.

The node selection in Step 3 and the branching scheme in Step 7 determine important decisions of a branch-and-bound algorithm that should be tailored to the given problem class. Both of them have a major impact on how early good primal solutions can be found in Step 6 and how fast the lower bounds of the open \( R \) are solved.

Figure 2.1. Branch-and-bound search tree.
2.1. Branch and Bound

In mixed integer programming, the most widely used relaxation is the LP relaxation (see Definition 1.5), which proved to be very successful in practice. Currently, almost all efficient commercial and academic MIP solvers are LP relax-based branch-and-bound algorithms. This includes the solvers mentioned in Section 1.3.

Besides supplying a dual bound that can be exploited for the bounding in Step 5, the LP relaxation can also be used to guide the branching decisions of Step 7. The most popular branching strategy in MIP solving is to split the domain of an integer variable $x_j$, $j \in I$, with fractional LP value $\tilde{x}_j \not\in \mathbb{Z}$ into two parts, thus creating the two subproblems $Q_1 = Q \cap \{x_j \leq \lfloor \tilde{x}_j \rfloor\}$ and $Q_2 = Q \cap \{x_j \geq \lceil \tilde{x}_j \rceil\}$ (see Figure 2.2). Methods to select a fractional variable as branching variable are discussed in Chapter 5.

Figure 2.2. LP based branching on a single fractional variable.