Delayed Column Generation (aka variable generation)

Optimization - 10725
Carlos Guestrin
Carnegie Mellon University

February 13th, 2008

Why constraint generation converges

x_{OPT}

- LP with many many constraints: $\min_x c'x$, $a_i x \geq b_i \ \forall i \in I$
- Solve with subset of constraints: $\min_x c'x \leftarrow x_\Omega$ optimal for relaxed problem; $a_i x \geq b_i \ \forall i \in \Omega$
  - (also called "cutting planes") $\Omega \subseteq I$
- Relaxed problem, bound on objective: (*) $c'x_\Omega \leq c'x_{OPT}$
- If solution x_Ω is feasible wrt all constraints: I $\Rightarrow$ (**) $c'x_{OPT} \leq c'x_\Omega$ (optimal, feasible)

  x_Ω is optimal $\Leftarrow$ $c'x_\Omega \leq c'x_{OPT} \leq c'x_\Omega$ $\leftarrow$
- If solution x_Ω is infeasible wrt all constraints: I

  $\exists$ a violated constraint, algorithm will add one, and continue (if $|I| < \infty$)

Constraint generation and duality

- Primal problem with many constraints:

$$\max_x \sum_j b_j x_j$$
$$s.t. \sum_j a_{ij} x_j \le c_i, \quad \forall i \in \mathcal{I}$$

$\max_x b'x$

$Ax \le c$

- Constraint generation: find most important constraints
- What's the dual equivalent? most variables are zero

incrementally build set of non-zero variables

- Dual:

$\min_y \sum_i c_i y_i$

$\sum_i a_{ij} y_i = b_j \quad \forall j \in 1, \ldots, n$

$y_i \ge 0 \quad \forall i \in I$

at solution to dual y_{opt}

$|I| - n$ variables must be zero

n variables may or may not be zero

©2008 Carlos Guestrin 3

Column generation (aka variable generation)

- Dual problem:

$$\min_y \sum_{i \in \mathcal{I}} c_i y_i$$
$$s.t. \sum_{i \in \mathcal{I}} a_{ij} y_i = b_j, \quad \forall j \in 1, \ldots, m$$
$$y_i \ge 0, \ \forall i \in \mathcal{I}$$

- Many many variables!!
- At optimal basic feasible solution
  - Most variables are zero $|I| - n$ must be zero
- Idea:
  - Set most variables to zero $I - \Omega$ must be zero
  - Solve problem with other variables: $\min_{y_i : i \in \Omega} \sum_{i \in \Omega} c_i y_i$

    $\sum_{i \in \Omega} a_{ij} y_i = b_j \ \forall j$

    $y_i \ge 0 \ \forall i \in \Omega$

    set Ω of vars that may be non zero
  - Incrementally increase sets of non-zero variables

©2008 Carlos Guestrin 4

Solving problem with subset of variables

- Solve problem with subset of variables

$$\min_y \quad \sum_{i\in\Omega} c_i y_i$$
$$s.t. \quad \sum_{i\in\Omega} a_{ij} y_i = b_j, \quad \forall j \in 1,\ldots,m$$
$$y_i \geq 0, \ \forall i \in \Omega$$

- Rest of variables set to zero
- Questions:
  - How do we decide what variables to use?
  - How do we decide when we are done?

also min $|\varepsilon_j|$

What variables should we add?

if you are infeasible start by adding slack variables ε_j

$$\min_y \quad \sum_{i\in\Omega} c_i y_i$$
$$s.t. \quad \sum_{i\in\Omega} a_{ij} y_i = b_j \ {+\varepsilon_j}, \quad \forall j \in 1,\ldots,m$$
$$y_i \geq 0, \ \forall i \in \Omega$$

- Same as simplex
  - you did this in your HW
- Solve problem with variables Ω
  - At optimal basic feasible solution, set of basic variables B ← LP solver returns B
- Find submatrix corresponding to basic variables A_B
  - Cost of these variables c_B

$$c_B = \begin{pmatrix} c_{B_1} \\ \vdots \\ c_{B_m} \end{pmatrix} \qquad A_B = \begin{pmatrix} a_{1B_1} & \cdots & a_{mB_1} \\ a_{1B_2} & & \vdots \\ \vdots & & \vdots \\ a_{1B_m} & \cdots & a_{mB_m} \end{pmatrix}$$

- Reduced cost for each potential new variable x_i (y_i), for i ∈ *I* :
  $$\bar{c}_i = c_i - c_B' A_B^{-1} A_i$$
  - If all are positive? if $\forall i \ \bar{c}_i \geq 0 \Rightarrow$ solution is optimal, can ignore rest of variables
  - Otherwise:
    add some variable y_i such that $\bar{c}_i < 0$ ("best" is one smallest $\bar{c}_i$)

A_i the ith column of A

- Guaranteed to converge to optimal solution

Column generation summary

- Dual of constraint generation
- Also useful for problems with infinitely many variables

infeasibility in dual or unboundedness in primal:

constraint generation look for constraint

solver returns 1. unbounded 2. direction or a ray of unboundness

- Some problems
  - Have efficient separation oracles
    - In these, constraint generation is useful
  - Have efficient variable generation oracles
    - In these, column generation is useful

- Both methods can be useful in polynomially large problems
  - E.g., when constraint matrix is too large to fit in memory
    - By incrementally solving the problem, bound amount of memory needed at each iteration

ray

constraint

compare vectors to stop ray

- If you have many many variables and constraints
  - Can use a combination of constraint and column generation