Bid Optimization Using Non-Concave Forecasting Models

AJ Minich, Master’s Student in Electrical Engineering, Stanford University

EE364b: Convex Optimization II Class Project, Spring 2011

Problem

• Online search engines (Google, Bing) auction advertising space (Fig. 1)
• advertisers bid on each relevant search term ('keyword') to maximize website traffic and product sales (conversions)

• Online search engines (Google, Bing) auction advertising space (Fig. 1)
• advertisers bid on each relevant search term ('keyword') to maximize website traffic and product sales (conversions)

• goal: develop a method that determines how to place bids to maximize conversions within a given budget

Model

• \( x \in \mathbb{R}_+^n \) are our bids (in dollars) for \( n \) distinct keywords
• all bids must satisfy \( 0 \leq x_i \leq b_{\text{max}} \)
• sum of all bids must not exceed a total budget \( B_{\text{max}} \)
• a set of forecasting functions \( f_i \) predict the expected revenue (in dollars) from bidding on keywords \( i = 1, \ldots, n \)
• our optimization problem to maximize expected revenue subject to budget constraints is

\[
\begin{align*}
\text{maximize} & \quad \sum_{i=1}^{n} f_i(x_i) \\
\text{subject to} & \quad \sum_{i=1}^{n} x_i \leq B_{\text{max}} \quad \text{(bid range)} \\
& \quad 0 \leq x_i \leq b_{\text{max}} \quad \text{for all } i
\end{align*}
\]

• forecasting functions \( \{f_i\} \) are guaranteed to be monotone nondecreasing and differentiable, but not necessarily concave
• for example, the shape of a given \( f_i \) may be the classic auction S-curve:


\[
\text{Fig. 2: An example S-curve, showing the expected revenue (in dollars) generated from bidding on a single keyword.}
\]

Dual decomposition

• dual of the problem separates revenue maximization into \( n \) subproblems connected by a dual variable \( \lambda \):

\[
\begin{align*}
\text{minimize}_x & \quad \max_{\lambda} \left( \sum_{i=1}^{n} f_i(x_i) - \lambda x_i + b_{\text{max}} \lambda \right) \\
\text{subject to} & \quad 0 \leq x_i \leq b_{\text{max}} \\
& \quad \lambda \geq 0
\end{align*}
\]

• since revenue is a nondecreasing function, and \( \lambda \) is a scalar, we can perform bisection on \( \lambda \) to find \( x^* \):

\[
\begin{align*}
\lambda^{(k)} = \frac{\lambda^{\text{upper}} + \lambda^{\text{lower}}}{2} \\
x^{(k+1)} = \arg \max_x \left( \sum_{i=1}^{n} f_i(x_i) - \lambda^{(k)} x_i \right) \\
\lambda^{\text{upper}} = \lambda^{(k)} \text{ if } \sum_{i=1}^{n} x_i > B_{\text{max}} \\
\lambda^{\text{lower}} = \lambda^{(k)} \text{ otherwise}
\end{align*}
\]

• iterate until \( \lambda^{\text{upper}} - \lambda^{\text{lower}} < \epsilon \)

Non-concave maximization

• finding \( x^{(k+1)} \) requires finding \( \arg \max_x \left( \sum_{i=1}^{n} (f_i(x_i) - \lambda^{(k)} x_i) \right) \) given \( \lambda^{(k)} \)

\[
\text{price-weighted bid function } f_i(x_i, \lambda) = f_i(x_i) - \lambda x_i \text{ is non-concave:}
\]


\[
\text{Fig. 3: A price-weighted bid function for the forecasting function } f \text{, shown in Fig. 2 with three values of } \lambda. \text{ The maximum values are shown as large disks on each plot. With } \lambda = 0, \text{ we max out the keyword bid; with } \lambda = 1, \text{ we partially bid on the keyword; with } \lambda = 1.5, \text{ the keyword is not worth bidding on.}
\]

• the maximum value \( f_i^{\text{opt}} \) can only occur at three types of points:
  1. \( x^*_i = 0 \) (no-bid condition)
  2. \( x^*_i = b_{\text{max}} \) (max-bid condition)
  3. \( x^*_i = \frac{1}{\lambda} \left( \frac{1}{f_i(x_i, \lambda)} = 0 \right) \) (competitive-bid condition, indicating a local maximum or minimum).

• can maximize \( f_i(x_i, \lambda) \) by evaluating the function at only these values of \( x_i \)
• gives fast and robust method for iterative non-concave maximization

Results

• best bidding strategy found for a random scenario with 10 bids and a budget limit of $15:

\[
\text{Fig. 4: The best bid strategy found for maximizing revenue. Each line represents the bid-revenue tradeoff for a different keyword, with the best bid shown as a ball on each line.}
\]

• entire budget spent, resulting in $17.83 in revenue
• chosen bids indicate that algorithm maximized the highest-return bids, and divided the remaining budget between keywords with lower returns
• three of the ten keywords received no bid, due to their low return rates
• duality gap is less than or equal to $0.23 (0.01% of the budget)
• weak duality of the problem restricts us from saying that the results are optimal, but small duality gap demonstrates that the achieved revenue is only slightly suboptimal
• results for several random problem instances solved on a 1.3 GHz processor with 3 GB of memory:

<table>
<thead>
<tr>
<th>Keywords</th>
<th>( B_{\text{max}} )</th>
<th>Best Revenue</th>
<th>Duality Gap</th>
<th>Iterations</th>
<th>Runtime</th>
</tr>
</thead>
<tbody>
<tr>
<td>10000</td>
<td>$1500</td>
<td>$2069.10</td>
<td>$1.46 (0.00%)</td>
<td>30</td>
<td>0.12 secs</td>
</tr>
<tr>
<td>10000</td>
<td>$2000</td>
<td>$2402.57</td>
<td>$8.49 (0.00%)</td>
<td>18</td>
<td>0.11 secs</td>
</tr>
<tr>
<td>10000</td>
<td>$2500</td>
<td>$2647.00</td>
<td>$9.66 (0.00%)</td>
<td>32</td>
<td>0.23 secs</td>
</tr>
<tr>
<td>10000</td>
<td>$15000</td>
<td>$20580.09</td>
<td>$96.45 (0.00%)</td>
<td>59</td>
<td>0.89 secs</td>
</tr>
<tr>
<td>10000</td>
<td>$20000</td>
<td>$23844.75</td>
<td>$199.83 (0.01%)</td>
<td>93</td>
<td>1.84 secs</td>
</tr>
<tr>
<td>10000</td>
<td>$25000</td>
<td>$26339.95</td>
<td>$185.07 (0.01%)</td>
<td>94</td>
<td>1.86 secs</td>
</tr>
</tbody>
</table>

• duality gap indicates that in all of these cases, the best bidding strategy found is very nearly optimal

Conclusions

• Although bid revenue maximization typically involves non-concave objectives and constraints, dual decomposition solves the problem for individual keywords, resulting in a simple algorithm that generates nearly optimal bidding strategies.
• The runtime scales linearly with the number of bids to consider, and could be significantly decreased through the use of parallel processing, with one processor assigned to each keyword and a central server tracking and updating the dual variable \( \lambda \).