Generalized Views of Dynamic Programming

Algorithms for NLP

October 28, 2014
Midterms

- They’ve been graded
  
  Mean 52.08 ($\sigma = 16.74$)
  25th percentile 43.5
  Median 49
  75th percentile 66.5

- You’ll get an email soon (to your andrew ID)

- We will discuss answers at the recitation

- You can see your exams at office hours
  - You cannot keep them
Homework

• Homework #3 due today by 5
• Homework #4 has been posted
  – Divided into 4A and 4B
  – 4A is due on Nov 11 (two weeks)
  – 4B is due on Nov 25 (two weeks after that)
Where We Are

• Finite-state formalisms and algorithms
• Context-free formalisms and algorithms
• Today we're going to try to unify some of the basic ideas that are common to most of the algorithms we've seen.
  – Logical deduction framework for dynamic programming, including parsing algorithms.
Dynamic Programming

• Break a problem into slightly smaller problems with optimal substructure.
  – For example, the best path to v depends on best paths to all u such that (u,v) ∈ E.
• Overlapping subproblems: each subproblem gets used repeatedly, and there aren't too many of them.
• Things to remember in general:
  – The graph may too big to represent explicitly; exhaustive calculation may be too expensive.
  – The graph may or may not have properties that make “clever” orderings possible.
Today

• A declarative view of dynamic programming algorithms
• Many algorithms in one: semirings
• Going beyond graphs to hypergraphs
• Recovering a parse
Kinds of Programming

• **Declarative**: express the logic of the program, not its control flow. Says *what* to do.
  – Regular expressions and logic programming are two examples; also LaTeX.
  – Actual execution might be efficient or horribly inefficient – much more is left to the “compiler.”

• **Imperative**: state the sequence of steps to be performed; closer to hardware. Says *how* to proceed.
Graph Reachability, in Prolog

- reachable(X) :- initial(X).
- reachable(Y) :- reachable(X), edge(X, Y).
Graph Reachability, in Prolog

- reachable(X) :- initial(X).
- reachable(Y) :- reachable(X), edge(X, Y).

**Problem encoding:**
- initial(a).
- edge(a, b). edge(b, d).
  - edge(b, e). edge(c, b).
  - edge(c, d). edge(d, d).
  - edge(d, e).
Graph Reachability, in Prolog

- reachable(X) :- initial(X).
- reachable(Y) :- reachable(X), edge(X, Y).
- goal :- reachable(e).
Graph Reachability, in Prolog

- reachable(X) :- initial(X).
- reachable(Y) :- reachable(X), edge(X, Y).
- goal :- reachable(e).

**Proof:**

initial(a) \(\rightarrow\) reachable(a)
reachable(a), edge(a, b) \(\rightarrow\) reachable(b)
reachable(b), edge(b, e) \(\rightarrow\) reachable(e)
reachable(e) \(\rightarrow\) goal
Another View

initial(a)

reachable(a) edge(a, b)

reachable(b) edge(b, e)

reachable(e)

goal
Another Proof

initial(a)

----------
reachable(a) edge(a, b)

----------
reachable(b) edge(b, d)

----------
reachable(d) edge(d, e)

----------
reachable(e)

----------
goal
Another View (Database)

<table>
<thead>
<tr>
<th>initial</th>
<th>edge</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>“from”</th>
<th>“to”</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>b</td>
<td>d</td>
</tr>
<tr>
<td>b</td>
<td>e</td>
</tr>
<tr>
<td>c</td>
<td>b</td>
</tr>
<tr>
<td>c</td>
<td>d</td>
</tr>
<tr>
<td>d</td>
<td>d</td>
</tr>
<tr>
<td>d</td>
<td>e</td>
</tr>
</tbody>
</table>

| reachable |      |
A Different Problem: Finite-State Recognition

• reachable(Q, 0) :- startstate(Q).

• reachable(Q, I) :- reachable(R, I - 1),
    transition(R, W, Q),
    symbol(W, I).

• goal :- reachable(Q, N), finalstate(Q), length(N).
A Few Comments

• There are multiple paths to the goal; each corresponds to a different proof.

• We have said nothing about an algorithm for finding proofs.
  – This is the “declarative mindset”: we define the problem first, then later talk about general ways of solving problems in that class.
  – Intuitively, efficiency means getting a proof fast, minimizing extra work.

• This kind of proof is sometimes called bottom-up deduction - start with grounded axioms and work up to the theorem you want to prove (goal).
Generalizations

• Boolean values → arbitrary semirings
• “Branching” proofs
Semiring \((K, \oplus, \otimes, 0, 1)\)

- **K**: a set of values
- **⊕**: an associative, commutative operator with an identity value \(0\)
  - \((x \oplus y) \oplus z = x \oplus (y \oplus z)\)
  - \(x \oplus y = y \oplus x\)
  - \(x \oplus 0 = x\)
- **⊗**: an associative operator that distributes over **⊕**, with an identity value \(1\)
  - \((x \otimes y) \otimes z = x \otimes (y \otimes z)\)
  - \(x \otimes (y \oplus z) = (x \otimes y) \oplus (x \otimes z)\)
  - \(x \otimes 1 = x\)
Boolean Semiring (\{F, T\}, \lor, \land, F, T)

- \{T, F\}: a set of values
- \lor: an associative, commutative operator with an identity value F
  - \((x \lor y) \lor z = x \lor (y \lor z)\)
  - \(x \lor y = y \lor x\)
  - \(x \lor F = x\)
- \land: an associative operator that distributes over \lor, with an identity value T
  - \((x \land y) \land z = x \land (y \land z)\)
  - \(x \land (y \lor z) = (x \land y) \lor (x \land z)\)
  - \(x \land T = x\)
“Cost” Semiring \((\mathbb{R}_+, \min, +, \infty, 0)\)

- \(\mathbb{R}_+ \cup \{\infty\}\): a set of values
- \(\min\): an associative, commutative operator with an identity value \(\infty\)
  - \(\min(\min(x, y), z) = \min(x, \min(y, z))\)
  - \(\min(x, y) = \min(y, x)\)
  - \(\min(x, \infty) = x\)
- \(+\): an associative operator that distributes over \(\min\), with an identity value \(0\)
  - \((x + y) + z = x + (y + z)\)
  - \(x + \min(y, z) = \min(x + y, x + z)\)
  - \(x + 0 = x\)
Graph Reachability: Cost Version

- \texttt{reachable}(X) min = \texttt{initial}(X).
- \texttt{reachable}(Y) min = \texttt{reachable}(X) + \texttt{edge}(X, Y).

- \textit{Problem encoding:}
  \texttt{initial}(a) := 0.
  \texttt{edge}(a, b) := 3.
  \texttt{edge}(b, d) := 5.
  ... etc.
### Database View

<table>
<thead>
<tr>
<th>initial</th>
<th>a</th>
<th>0</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>edge</th>
<th>edge</th>
<th>reachable</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>from</strong></td>
<td><strong>to</strong></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td>3</td>
</tr>
<tr>
<td>b</td>
<td>d</td>
<td>5</td>
</tr>
<tr>
<td>b</td>
<td>e</td>
<td>8</td>
</tr>
<tr>
<td>c</td>
<td>b</td>
<td>2</td>
</tr>
<tr>
<td>c</td>
<td>d</td>
<td>2</td>
</tr>
<tr>
<td>d</td>
<td>d</td>
<td>1</td>
</tr>
<tr>
<td>d</td>
<td>e</td>
<td>2</td>
</tr>
</tbody>
</table>
Another Cost Program

• \( a(I, J) \ \text{min} = a(I - 1, J - 1) + x(X, I) + y(X, J) \).
• \( a(I, J) \ \text{min} = a(I - 1, J - 1) + s + x(X, I) + y(Y, J) \).
• \( a(I, J) \ \text{min} = a(I, J - 1) + i + y(Y, J) \).
• \( a(I, J) \ \text{min} = a(I - 1, J) + d + x(X, I) \).
• \( \text{goal min} = a(M, N) + x(I)(M) + y(I)(N) \).

• \( a(0, 0) := 0 \).
More Readable

- \( \text{align}(I, J) \text{ min} = \text{align}(I - 1, J - 1) + x(X, I) + y(X, J). \)
- \( \text{align}(I, J) \text{ min} = \text{align}(I - 1, J - 1) + \text{subcost} + x(X, I) + y(Y, J). \)
- \( \text{align}(I, J) \text{ min} = \text{align}(I, J - 1) + \text{inscost} + y(Y, J). \)
- \( \text{align}(I, J) \text{ min} = \text{align}(I - 1, J) + \text{delcost} + x(X, I). \)
- \( \text{goal} \text{ min} = \text{align}(M, N) + x\text{len}(M) + y\text{len}(N). \)
- \( \text{align}(0, 0) := 0. \)
Another Notation

- \( \text{align}(I, J) = \min \left( \min_X [ \text{align}(I-1, J-1) + x(X, I) + y(X, J) ], \right. \)
  \( \min_X, Y [ \text{align}(I-1, J-1) + \text{subcost} + x(X, I) + y(Y, J) ], \)
  \( \min_Y [ \text{align}(I, J-1) + \text{inscost} + y(Y, J) ], \)
  \( \min_X [ \text{align}(I-1, J) + \text{delcost} + x(X, I) ] \) \)

- \( \text{goal} = \min_{M, N} \text{align}(M, N) + \text{xlen}(M) + \text{ylen}(N). \)

- \( \text{align}(0, 0) = 0. \)
Another Cost Program

• \( a_{i,j} = \min(a_{i-1,j-1} \text{ if } x_i = y_j, \)

\[
\begin{align*}
& a_{i-1,j-1} + \text{subcost}, \\
& a_{i,j-1} + \text{inscost}, \\
& a_{i-1,j} + \text{delcost})
\end{align*}
\]

• \( \text{goal} = a_{\text{length}(x), \text{length}(y)} \)
• \( a_{0,0} = 0 \)
Written Another Way

• $a(I, J) \min = a(I - 1, J - 1) + x(X, I) + y(X, J)$.

• $a(I, J) \min = a(I - 1, J - 1) + \underline{\text{subcost}} + x(X, I) + y(Y, J)$.

• $a(I, J) \min = a(I, J - 1) + \underline{\text{inscost}} + y(Y, J)$.

• $a(I, J) \min = a(I - 1, J) + \underline{\text{delcost}} + x(X, I)$.

• goal $\min = a(M, N) + xlength(M) + ylength(N)$.

• $a(0, 0) := 0$. 
# Classic View

<table>
<thead>
<tr>
<th></th>
<th>x_1</th>
<th>x_2</th>
<th>x_3</th>
<th>x_4</th>
<th>x_5</th>
<th>x_6</th>
<th>x_7</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y_1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y_2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y_3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y_4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Classic View

<table>
<thead>
<tr>
<th></th>
<th>x₁</th>
<th>x₂</th>
<th>x₃</th>
<th>x₄</th>
<th>x₅</th>
<th>x₆</th>
<th>x₇</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>y₁</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y₂</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y₃</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>y₄</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Database View

<table>
<thead>
<tr>
<th></th>
<th>inscost</th>
<th>delcost</th>
<th>subcost</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>“pos”</td>
<td>“sym”</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>b</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>a</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>n</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>a</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>n</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>a</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>a</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>y</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>“pos”</td>
<td>“sym”</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>b</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>o</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>n</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>a</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>n</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>z</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>a</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>a</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>“xpos”</td>
<td>“ypos”</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
A Few Comments

• Building up the whole database or the whole graph may not be the most efficient way to solve the problem ... but you should see that these are equivalent to the original problem.

• Recall that when calculating Levenshtein distance, we sometimes want to know the actual alignment ... how do we do this?

• Each alignment corresponds to a single proof
“Cost”/Derivation Semiring

• $K = \{ (x, p) : x \in \mathbb{R}^+ \cup \{\infty\}, \text{ and } p \text{ is a set of derivation strings } \}$

• $(x_1, p_1) \oplus (x_2, p_2) = (\min(x_1, x_2), x_1 < x_2 ? p_1 : x_1 = x_2 ? p_1 \cup p_2 : p_2)$

• $(x_1, p_1) \otimes (x_2, p_2) = (x_1 + x_2, p_1 \cdot p_2)$ (not commutative!)

• $0 = (\infty, \{\})$

• $1 = (0, \{\varepsilon\})$

• Question: what are the $x$, $y$, $\text{inscost}$, $\text{delcost}$, $\text{subcost}$, $\text{xlength}$, $\text{ylength}$ axiom values?
Derivations as Strings

\[ x(b, 1) := (0, \text{“b”}). \]
\[ x(a, 2) := (0, \text{“a”}). \]
\[ y(b, 1) := (0, \text{“b”}). \]
\[ y(o, 2) := (0, \text{“o”}). \]

... b b sub a o n n a a n n ins z a a

... b b del a ins o ...

... b b ins o del a ...

del b ins b ...

inscost := (1, “ins”).

delcost := (1, “del”).

subcost := (1, “sub”).
Generalizing Edit Distance

• $\text{align}(I, J) \min = \text{align}(I - 1, J - 1) + x(X, I) + y(X, J)$.

• $\text{align}(I, J) \min = \text{align}(I - 1, J - 1) + \text{subcost}(X, Y) + x(X, I) + y(Y, J)$.

• $\text{align}(I, J) \min = \text{align}(I, J - 1) + \text{inscost}(Y) + y(Y, J)$.

• $\text{align}(I, J) \min = \text{align}(I - 1, J) + \text{delcost}(X) + x(X, I)$.

• $\text{goal} \min = \text{align}(M, N) + x\text{len}(M) + y\text{len}(N)$.

• $\text{align}(0, 0) := 0$. 
Generalizing Edit Distance

- \( \text{align}(I, J) \min = \text{align}(I - 1, J - 1) + x(X, I) + y(X, J). \)
- \( \text{align}(I, J) \min = \text{align}(I - 1, J - 1) + \text{subcost} + x(X, I) + y(Y, J). \)
- \( \text{align}(I, J) \min = \text{align}(I, J - 1) + \text{inscost} + y(Y, J). \)
- \( \text{align}(I, J) \min = \text{align}(I - 1, J) + \text{delcost} + x(X, I). \)
- \( \text{align}(I, J) \min = \text{align}(I - 2, J - 2) + \text{transcost} + x(A, I - 1) + x(B, I) + y(B, J - 1) + y(A, J). \)
- \( \text{goal} \min = \text{align}(M, N) + \text{xlen}(M) + \text{ylen}(N). \)
- \( \text{align}(0, 0) := 0. \)
Exercise

• Can you write down the WFSA best path algorithm as a semiring-weighted logic program?
  – What is the structure?
  – What semiring is appropriate?
WFSA Best Path

- reachable(Q, 0) max = π(Q).
- reachable(Q, l) max = reachable(R, l – 1) × δ(R, W, Q) × x(W, l).
- goal max = reachable(Q, N) × ξ(Q) × xlength(N).
Viterbi

• reachable(Q, 0) max = $\pi(Q)$.
• reachable(Q, $I$) max = reachable(R, $I - 1$) × $n(R, W)$ × $y(R, Q)$ × $x(W, I)$.
• goal max = reachable(Q, N) × $\xi(Q)$ × xlength(N).
Another Algorithm

- \texttt{constit}(X, l - 1, l) :- \texttt{word}(W, l), \texttt{unary}(X, W).
- \texttt{constit}(X, I, K) :- \texttt{constit}(Y, I, J), \texttt{constit}(Z, J, K), \texttt{binary}(X, Y, Z).
- \texttt{goal} :- \texttt{constit}("S", 0, N), \texttt{length}(N).

- How is this different from before?
CKY

- \texttt{constit}(X, I - 1, I) :- \texttt{word}(W, I), \texttt{unary}(X, W).
- \texttt{constit}(X, I, K) :- \texttt{constit}(Y, I, J), \texttt{constit}(Z, J, K), \texttt{binary}(X, Y, Z).
- \texttt{goal} :- \texttt{constit}(“S”, 0, N), \texttt{length}(N).

- How is this different from before?

- The appropriate connections among \texttt{constit} items aren't representable as a graph; a single step of the proof uses \texttt{two} \texttt{constits} as antecedents.
Hypergraph View

constit(X, I, K)

constit(Y, I, J)

constit(Z, J+1, K)

rewrite(X, Y, Z)
Hypergraph View

\[
\text{consMt}(X, I, K) \\
\text{rewrite}(X, Y, Z) \\
\text{consMt}(Y, I, J) \\
\text{consMt}(Z, J+1, K) \\
\text{rewrite}(X, Y, W) \\
\text{rewrite}(X, Y, Z) \\
\text{consMt}(W, J+1, K) \\
\text{constit}(Y, I, J) \\
\text{constit}(Z, J+1, K) \\
\text{constit}(Y, I, J+1) \\
\text{constit}(Z, J+2, K)
\]
Hypergraph View

consMt(X, I, K)

consMt(Y, I, J)

consMt(Z, J+1, K)

rewrite(X, Y, Z)

consMt(Y, I, J+1)

consMt(Z, J+2, K)

rewrite(X, Y, W)

consMt(W, J+1, K)
Classic (Chart) View

```
<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```

Each cell holds a map from nonterminals to values, plus backpointers.
Classic (Chart) View

“to”

“from”

each cell holds a map from nonterminals to values, plus backpointers
Some “Semiring” Variations on CKY

1. Store the derivations or backpointers
2. Augment production rules with weights
3. Best parse
Weighted CKY

• \(\text{constit}(X, I - 1, I) \max = \text{word}(W, I) \times \text{unary}(X, W)\).
• \(\text{constit}(X, I, K) \max = \text{constit}(Y, I, J) \times \text{constit}(Z, J, K) \times \text{binary}(X, Y, Z)\).
• \(\text{goal} \ max = \text{constit}(“S”, 0, N) \times \text{length}(N)\).

• Set values to be rule weights.
• “Max-times” semiring.
• This is variation #2; with backpointers or storing the best derivation, it is #3.
Weighted CKY with Backpointers, Procedurally

**input:** x, G (in CNF)
**output:** true iff x is in L(G)
initialize all cells of C to 0
initialize all cells of B to ∅
for i = 1 ... n
  for each A ∈ N
    \[ C[A, i-1, i] = \omega(A \rightarrow x_i) \]
for ℓ = 2 ... n
  for i = 0 ... n - ℓ
    k = i + ℓ
    for j = i + 1 ... k - 1
      for each X → YZ
        \[ \alpha = C[Y, i, j] \times C[Z, j, k] \times \omega(X \rightarrow YZ) \]
      if \( \alpha > C[X, i, k] \):
        B[X, i, k] = (&B[Y, i, j], &B[Z, j, k])
        \[ C[X, i, k] = \alpha \]
return backtrace(B)
Probabilistic CKY With Log Trick

• \(\text{constit}(X, I - 1, I) \max = \text{word}(W, I) + \text{unary}(X, W).\)
• \(\text{constit}(X, I, K) \max = \text{constit}(Y, I, J) + \text{constit}(Z, J, K) + \text{binary}(X, Y, Z).\)
• \(\text{goal} \max = \text{constit}(“S”, 0, N) + \text{length}(N).\)

• Set values to be log-probabilities.
• “Max-plus” semiring.
A Few Comments

• All of the following are different representations of the same thing:
  – deductive proof using the logic program
  – (hyper)path to goal vertex
  – derivation in the grammar

• The logic program tells us how to build up the graph or hypergraph, as well as how values for every theorem/vertex are to be calculated.