Dynamic Programming Algorithms in Semiring and Hypergraph Frameworks

Liang Huang

Clunch talk and WPE II exam

November 28th, 2006
Dynamic Programming

- Dynamic Programming is everywhere in NLP
  - Viterbi Algorithm for Hidden Markov Models
  - CKY Algorithm for Context-free Parsing
  - Forward-Backward and Inside-Outside Algorithms
  - A* Parsing

- Two dimensional survey
  - two frameworks: semirings and hypergraphs
  - two algorithms: Viterbi and Dijkstra

- Focus on Optimization Problems
## Two Dimensional Survey

<table>
<thead>
<tr>
<th>search space</th>
<th>traversing order</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>graphs with semirings</strong> (e.g., FSMs)</td>
<td></td>
<td>topological (acyclic)</td>
<td>Viterbi</td>
</tr>
<tr>
<td><strong>hypergraphs with weight functions</strong> (e.g., CFGs)</td>
<td></td>
<td>best-first (superior)</td>
<td>Dijkstra</td>
</tr>
<tr>
<td><strong>Generalized Viterbi</strong></td>
<td></td>
<td></td>
<td>Knuth</td>
</tr>
</tbody>
</table>

Liang Huang
Motivations for Semirings

- in a weighted graph, we need two operators:
  - extension (multiplicative) and summary (additive)
  - the weight of a path is the product of edge weights
  - the weight of a vertex is the summary of path weights

\[
d(\pi_1) = \bigotimes_{e_i \in \pi_1} w(e_i) = w(e_1) \otimes w(e_2) \otimes w(e_3)
\]

\[
d(t) = \bigoplus_{\pi_i} w(\pi_i) = w(p_1) \oplus w(p_2) \oplus \cdots
\]
A **monoid** is a triple \((A, \otimes, I)\) where

1. \(\otimes\) is a closed **associative binary operator** on the set \(A\),
2. \(I\) is the **identity element** for \(\otimes\), i.e., for all \(a \in A\), \(a \otimes I = I \otimes a = a\).

A monoid is **commutative** if \(\otimes\) is commutative.

A **semiring** is a 5-tuple \(R = (A, \oplus, \otimes, 0, 1)\) such that

1. \((A, \oplus, 0)\) is a commutative monoid.
2. \((A, \otimes, I)\) is a monoid.
3. \(\otimes\) distributes over \(\oplus\): for all \(a, b, c\) in \(A\),

\[
(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c),
\]

\[
c \otimes (a \oplus b) = (c \otimes a) \oplus (c \otimes b).
\]

4. \(0\) is an **annihilator** for \(\otimes\): for all \(a\) in \(A\), \(0 \otimes a = a \otimes 0 = 0\).
## Examples

<table>
<thead>
<tr>
<th>Semiring</th>
<th>Set</th>
<th>$\oplus$</th>
<th>$\otimes$</th>
<th>$\bar{0}$</th>
<th>$\bar{1}$</th>
<th>intuition/application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>${0, 1}$</td>
<td>$\lor$</td>
<td>$\land$</td>
<td>0</td>
<td>1</td>
<td>logical deduction, recognition</td>
</tr>
<tr>
<td>Viterbi</td>
<td>$[0, 1]$</td>
<td>$\max$</td>
<td>$\times$</td>
<td>0</td>
<td>1</td>
<td>prob. of the best derivation</td>
</tr>
<tr>
<td>Inside</td>
<td>$\mathbb{R}^+ \cup {+\infty}$</td>
<td>$+$</td>
<td>$\times$</td>
<td>0</td>
<td>1</td>
<td>prob. of a string</td>
</tr>
<tr>
<td>Real</td>
<td>$\mathbb{R} \cup {+\infty}$</td>
<td>$\min$</td>
<td>$+$</td>
<td>$+\infty$</td>
<td>0</td>
<td>shortest-distance</td>
</tr>
<tr>
<td>Tropical</td>
<td>$\mathbb{R}^+ \cup {+\infty}$</td>
<td>$\min$</td>
<td>$+$</td>
<td>$+\infty$</td>
<td>0</td>
<td>with non-negative weights</td>
</tr>
<tr>
<td>Counting</td>
<td>$\mathbb{N}$</td>
<td>$+$</td>
<td>$\times$</td>
<td>0</td>
<td>1</td>
<td>number of paths</td>
</tr>
</tbody>
</table>

Liang Huang
Ordering

- **idempotent**
  A semiring \((A, ⊕, ⊗, 0, 1)\) is **idempotent** if for all \(a\) in \(A\), \(a ⊕ a = a\).

- **comparison**
  \((a ≤ b) ⇔ (a ⊕ b = a)\) defines a partial ordering.

- **examples: boolean, viterbi, tropical, real, ...**
  
  \((\{0, 1\}, ∨, ∧, 0, 1)\)  \((\mathbb{R}^+ \cup \{+∞\}, \text{min}, +, +∞, 0)\)

  \(([0, 1], \text{max}, ⊗, 0, 1)\)  \((\mathbb{R} \cup \{+∞\}, \text{min}, +, +∞, 0)\)

- **total-order for optimization problems**
  A semiring is **totally-ordered** if ⊕ defines a total ordering.

- **examples: all of the above**
Monotonicity

- **monotonicity**
  
  Let $K = (A, \oplus, \otimes, 0, 1)$ be a semiring, and $\leq$ a partial ordering over $A$. We say $K$ is **monotonic** if for all $a, b, c \in A$

  \[(a \leq b) \Rightarrow (a \otimes c \leq b \otimes c) \quad (a \leq b) \Rightarrow (c \otimes a \leq c \otimes b)\]

- **optimal substructure in dynamic programming**

- **[lemma]** idempotent $\Rightarrow$ monotone

  - our focus, totally-ordered semirings, are monotone

\[\text{free lunch!}\]
## Two Dimensional Survey

<table>
<thead>
<tr>
<th>Search Space</th>
<th>Traversing Order</th>
<th>Traversing Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>graphs with semirings (e.g., FSMs)</td>
<td>Viterbi</td>
<td>Dijkstra</td>
</tr>
<tr>
<td>hypergraphs with weight functions (e.g., CFGs)</td>
<td>Generalized Viterbi</td>
<td>Knuth</td>
</tr>
</tbody>
</table>

**Topological (acyclic)**

**Best-first (superior)**
DP on Graphs

- Optimization problems on graphs => generic shortest-path problem
- Weighted directed graph $G=(V, E)$ with a function $w$ that assigns each edge a weight from a semiring
- Compute the best weight of the target vertex $t$
- Generic update along edge $(u, v)$
  \[
  d(v) \oplus = d(u) \otimes w(u, v)
  \]
- How to avoid cyclic updates?
  - Only update when $d(u)$ is fixed
  \[
  d(v) \leftarrow d(v) \oplus (d(u) \otimes w(u, v))
  \]
Viterbi Algorithm for DAGs

1. topological sort

2. visit each vertex v in sorted order and do updates
   • for each incoming edge (u, v) in E
   • use d(u) to update d(v): \( d(v) \oplus = d(u) \otimes w(u, v) \)
   • key observation: d(u) is fixed to optimal at this time

\[
\begin{align*}
& u \\
& \text{w}(u, v) \\
& v \\
& \text{w}(u, v)
\end{align*}
\]

• time complexity: \( O(V + E) \)
Variant: forward-update

1. topological sort

2. visit each vertex v in sorted order and do updates
   • for each outgoing edge (v, u) in E
   • use d(v) to update d(u): \( d(u) \oplus = d(v) \otimes w(v, u) \)
   • key observation: d(v) is fixed to optimal at this time

   \[ d(u) \oplus = d(v) \otimes w(v, u) \]

   • time complexity: \( O(V + E) \)
Examples

• [Number of Paths in a DAG]
  • just use the counting semiring \((\mathbb{N}, +, \times, 0, 1)\)
  • note: this is not an optimization problem!

• [Longest Path in a DAG]
  • just use the semiring \((\mathbb{R} \cup \{-\infty\}, \max, +, -\infty, 0)\)

• [Part-of-Speech Tagging with a Hidden Markov Model]
Dijkstra Algorithm

- Dijkstra does not require acyclicity
- instead of topological order, we use best-first order
- but this requires *superiority* of the semiring

Let $K = (A, \oplus, \otimes, \overline{0}, \overline{1})$ be a semiring, and $\leq$ a partial ordering over $A$. We say $K$ is superior if for all $a, b \in A$

$$a \leq a \otimes b, \quad b \leq a \otimes b.$$ 

- basically, combination always gets worse
- or, no negative edge in a graph

\[
\begin{align*}
d(u) \otimes w(e) & \quad \rightarrow \quad d(u) \otimes w(e)
\end{align*}
\]
Dijkstra Algorithm

- keep a cut \((S : V - S)\) where \(S\) vertices are fixed
- maintain a priority queue \(Q\) of \(V - S\) vertices
- each iteration choose the best vertex \(v\) from \(Q\)
- move \(v\) to \(S\), and use \(d(v)\) to forward-update others

\[
d(u) \oplus = d(v) \otimes w(v, u)
\]

Time complexity:
- \(O((V+E) \lg V)\) (binary heap)
- \(O(V \lg V + E)\) (fib. heap)
Viterbi vs. Dijkstra

- structural vs. algebraic constraints
- Dijkstra only applicable to optimization problems

monotonic optimization problems

acyclic: Viterbi

many NLP problems

superior: Dijkstra

cyclic FSMs/grammars

forward-backward (Inside semiring)

max-margin models
What if both fail?

- monotonic optimization problems
- acyclic: Viterbi
- many NLP problems
- superior: Dijkstra

Generalized Bellman-Ford
(CLR, 1990; Mohri, 2002)

Or, first do strongly-connected components (SCC)
which gives a DAG; use Viterbi globally on this SCC-DAG;
use Bellman-Ford locally within each SCC
What if both work?

monotonic optimization problems

acyclic: Viterbi

many NLP problems

superior: Dijkstra

full Dijkstra is slower than Viterbi

\[ O((V + E) \log V) \quad \text{vs.} \quad O(V + E) \]

but it can finish as early as the target vertex is popped

\[ a \ (V + E) \log V \quad \text{vs.} \quad V + E \]

Q: how to (magically) reduce \( a \)?
A* Search

- $d(v)$: the distance from source $s$ to $v$
- $h(v)$: the distance from $v$ to target $t$
- $\hat{h}(v)$: an optimistic estimate of $h(v)$
- now, prioritize the queue by $d(v) \otimes \hat{h}(v)$
- Dijkstra is a special case where $\hat{h}(v) = 1$
- also requires $d(v) \otimes \hat{h}(v)$ never improves (superior)
- hope: $d(t) \otimes \hat{h}(t) = d(t)$ can be popped sooner
## Two Dimensional Survey

<table>
<thead>
<tr>
<th>Search Space</th>
<th>Topological (Acyclic)</th>
<th>Best-First (Superior)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Graphs with Semirings (e.g., FSMs)</strong></td>
<td>Viterbi</td>
<td>Dijkstra</td>
</tr>
<tr>
<td><strong>Hypergraphs with Weight Functions (e.g., CFGs)</strong></td>
<td>Generalized Viterbi</td>
<td>Knuth</td>
</tr>
</tbody>
</table>
(Directed) Hypergraphs

- a generalization of graphs
  - edge => hyperedge: several vertices to one vertex
  - \( e = (T(e), h(e), f_e) \). arity \(|e| = |T(e)|\)
- a totally-ordered weight set \( R \)
  - we borrow the \( \oplus \) operator to be the comparison
- weight function \( f_e : R^{|e|} \) to \( R \)
  - generalizes the \( \otimes \) operator in semirings

\[
d(v) \oplus = f_e(d(u_1), d(u_2))
\]
Hypergraphs and Deduction

Nederhof, 2003: a

\[ \text{fe} (a, b) \]

\[(B, i, k) \]
\[\text{u}_1 : a \]
\[\text{u}_2 : b \]
\[\text{fe} \]
\[\text{v} : \text{fe} (a, b) \]

\[(C, k, j) \]
\[\text{u}_1 : a \]
\[\text{u}_2 : b \]
\[\text{fe} \]
\[\text{v} : \text{fe} (a, b) \]

\[(A, i, j) \]
\[\text{u}_1 : a \]
\[\text{u}_2 : b \]
\[\text{fe} \]
\[\text{v} : a \times b \times \text{Pr}(A \rightarrow B C) \]

\[(A, i, j) \]
\[(B, i, k) \]
\[(C, k, j) \]
\[A \rightarrow B C \]

\[ \text{Pr}(A \rightarrow B C) \]

Liang Huang

Dynamic Programming

22
Weight Functions and Semirings

\[ d(u) \rightarrow_{w(e)} d(u) \otimes w(e) \]

\[ d(u) \rightarrow_{f_e} f_e(d(u)) \]

\[ f_e(a) = a \otimes w(e) \]

\[ f_e(a_1, ..., a_k) = a_1 \otimes ... \otimes a_k \otimes w(e) \]

also extend monotonicity and superiority to weight functions
## Two Dimensional Survey

<table>
<thead>
<tr>
<th>Search Space</th>
<th>Traversing Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>graphs with semirings (e.g., FSMs)</td>
<td>topological (acyclic)</td>
</tr>
<tr>
<td>hypergraphs with weight functions (e.g., CFGs)</td>
<td>best-first (superior)</td>
</tr>
</tbody>
</table>

- **Viterbi** for graphs with semirings (e.g., FSMs)
- **Dijkstra** for hypergraphs with weight functions (e.g., CFGs)
- **Generalized Viterbi** for families of graphs with semirings
- **Knuth** for hypergraphs with weight functions

Liang Huang
Viterbi Algorithm for DAGs

1. topological sort

2. visit each vertex \( v \) in sorted order and do updates
   - for each incoming edge \((u, v)\) in \( E \)
   - use \( d(u) \) to update \( d(v) \):
     - key observation: \( d(u) \) is fixed to optimal at this time
     - time complexity: \( O(V + E) \)

\[
d(v) \oplus = d(u) \otimes w(u, v)
\]
Viterbi Algorithm for DAHs

1. topological sort

2. visit each vertex $v$ in sorted order and do updates
   - for each incoming hyperedge $e = ((u_1, \ldots, u_{|e|}), v, f_e)$
   - use $d(u_i)$’s to update $d(v)$
   - key observation: $d(u_i)$’s are fixed to optimal at this time

\[
d(v) \oplus = f_e(d(u_1), \ldots, d(u_{|e|}))
\]

- time complexity: $O(V + E)$ (assuming constant arity)

Liang Huang

Dynamic Programming 26
Example: CKY Parsing

- parsing with CFGs in Chomsky Normal Form (CNF)
- typical instance of the generalized Viterbi for DAHs
- many variants of CKY ~ various topological ordering

\[ O(n^3 |P|) \]
Forward Variant for DAHs

1. topological sort

2. visit each vertex \( v \) in sorted order and do updates
   
   - for each outgoing hyperedge \( e = ((u_1, ..., u_{|e|}), h(e), f_e) \)
   
   - if \( d(u_i) \)'s have all been fixed to optimal
     
     - use \( d(u_i) \)'s to update \( d(h(e)) \)

- time complexity: \( O(V + E) \)

Q: how to avoid repeated checking?

maintain a counter \( r[e] \) for each \( e \):
how many tails yet to be fixed?
fire this hyperedge only if \( r[e] = 0 \)
Example: Treebank Parsers

- State-of-the-art statistical parsers
  - (Collins, 1999; Charniak, 2000)
  - no fixed grammar (every production is possible)
  - can’t do backward updates
    - don’t know how to decompose a big item
  - forward update from vertex \((X, i, j)\)
    - check all vertices like \((Y, j, k)\) or \((Y, k, i)\) in the chart (fixed)
    - try combine them to form bigger item \((Z, i, k)\) or \((Z, k, j)\)
Dijkstra Algorithm

- keep a cut \((S : V - S)\) where \(S\) vertices are fixed
- maintain a priority queue \(Q\) of \(V - S\) vertices
- each iteration choose the best vertex \(v\) from \(Q\)
- move \(v\) to \(S\), and use \(d(v)\) to forward-update others

\[ d(u) \oplus = d(v) \otimes w(v, u) \]

Time complexity:
- \(O((V+E) \log V)\) (binary heap)
- \(O(V \log V + E)\) (fib. heap)
Knuth (1977) Algorithm

- keep a cut \((S : V - S)\) where \(S\) vertices are fixed
- maintain a priority queue \(Q\) of \(V - S\) vertices
- each iteration choose the best vertex \(v\) from \(Q\)
- move \(v\) to \(S\), and use \(d(v)\) to forward-update others

Time complexity:
- \(O((V+E) \lg V)\) (binary heap)
- \(O(V \lg V + E)\) (fib. heap)
Example: A* Parsing

- Use A* search on top of the Knuth Algorithm
- Showed significant speed up with carefully designed heuristic functions (Klein and Manning, 2003)

[open problem] can you still define heuristic function if weight functions are not semiring-composed?

Liang Huang
monotonic optimization problems

acyclic: Viterbi

max-margin parsing

PCFG parsing with CNF

cyclic grammars

generalized Bellman-Ford (open)

many NLP problems

superior: Knuth

Inside-Outside Alg. (Inside semiring)

Liang Huang

Dynamic Programming
Conclusion

- surveyed two general frameworks and two important algorithms in DP
  - adapted the theory of semirings to weight functions
  - demonstrated advantages of modularity
- focused on optimization problems
- covered typical NLP applications
  - a better understanding of these theories helps NLP
  - a generic DP language for NLP: Dyna (Eisner et al.)
- suggested some open problems