Viterbi Training for PCFGs: Hardness Results and Competitiveness of Uniform Initialization

Shay Cohen    Noah Smith

Carnegie Mellon University

July 14, 2010
Hardness results for unsupervised learning of PCFGs

- Background and problem definition
- Main hardness result
- Extensions
- Open problems
- Conclusion
Let $p(x, z | \theta)$ be some parametrized statistical model.

Viterbi EM identifies $\theta$ and $z$ given $x$. 
Let $p(x, z \mid \theta)$ be some parametrized statistical model

Viterbi EM identifies $\theta$ and $z$ given $x$

Let $x_1, \ldots, x_n$ be the observed data

Algorithm (Viterbi EM)

1. \textit{start with some $\theta$}

2. set $z_i \leftarrow \arg\max_{z_i} p(x_i, z_i \mid \theta) \iff \text{“E-step”}$

3. set $\theta \leftarrow \arg\max_{\theta} \prod_{i=1}^{n} p(x_i, z_i \mid \theta) \iff \text{“M-step”}$

4. go to step 2 unless converged
Simple and useful algorithm. Recent examples include:

Machine translation (Brown et al., 2003)

Language acquisition (Goldwater and Johnson, 2005)

Coreference resolution (Choi and Cardie, 2007)

Question answering (Wang et al., 2007)

Grammar induction (Spitkovsky et al., 2010)

We focus on Viterbi EM for PCFGs

- $z_i$ - parse tree, $x_i$ - sentence, $\theta$ - rule probabilities
Viterbi EM is coordinate ascent, and it greedily tries to find:

\[
\langle \theta, z_1, ..., z_n \rangle = \argmax_{\theta, z_1, ..., z_n} \prod_{i=1}^{n} p(x_i, z_i | \theta)
\]

We call this maximization problem “Viterbi training”

Viterbi EM finds local maximum for Viterbi training
Viterbi training

Viterbi EM is coordinate ascent, and it greedily tries to find:

\[
\langle \theta, z_1, \ldots, z_n \rangle = \arg\max_{\theta, z_1, \ldots, z_n} \prod_{i=1}^{n} p(x_i, z_i | \theta)
\]

We call this maximization problem “Viterbi training”

Viterbi EM finds local maximum for Viterbi training

- **Main question**: can we hope to optimize this objective function and find the global maximum?

- ... computational complexity answers this kind of question
We usually show that a problem $A$ is hard by showing that another hard problem $B$ can be solved if we could solve $A$

The type of problem we usually do this for is “decision problems” (answer is 0 or 1)

“Hardness” in this paper refers to being able to solve all problems in the NP class (“NP hardness”)

We convert every input $x$ of $B$ to an input $x'$ of $A$ such that

$$B(x) = 1 \iff A(x') = 1$$
Viterbi training optimizes an objective function. To convert to a decision problem we define:

**Problem (Viterbi Train)**

**Input:** $G$ context-free grammar, $x_1, \ldots, x_n$ sentences, $\alpha \in [0, 1]$

**Output:** 1 if there are $\theta$ and $z_1, \ldots, z_n$ derivation trees such that

$$\prod_{i=1}^{n} p(x_i, z_i \mid \theta) \geq \alpha$$

and 0 otherwise.

Note that knowing how to optimize the likelihood means we can solve this decision problem.

Viterbi Train is in NP (witness: parse trees and parameters)
We show that Viterbi Train is NP hard by showing that there is a reduction from 3-SAT (an NP hard problem) to Viterbi Train.

Problem (3-SAT)

**Input:** A formula \( \phi = \bigwedge_{i=1}^{m} (a_i \lor b_i \lor c_i) \) in conjunctive normal form, such that each clause has 3 literals.

**Output:** 1 if there is a satisfying assignment for \( \phi \) and 0 otherwise.

For example, if we have the formula

\[
\phi = (a \lor b \lor c) \land (\neg a \lor b \lor c)
\]

then a satisfying assignment is \( a = 0, b = 0, c = 1 \)
We map every instance of 3-SAT (a formula $\phi$) to a grammar $G$ and a string $x$ such that

$$\max_{z,\theta} p(x, z | \theta) = 1$$

if and only if there is a satisfying assignment for the formula

The maximizing $z$ and $\theta$ will contain a description of the assignment

Since 3-SAT is NP hard, Viterbi Train is NP hard
The reduction (an example)

Let $\phi = (a \lor \neg b \lor c) \land (\neg a \lor b \lor c) \land (d \lor \neg c \lor a)$

$C_1 \land C_2 \land C_3$

We create the following context-free grammar:
$\Sigma = \{0, 1\}$ \quad \text{Terminal symbols}

For the variables, $a, b, c, d$ we create the rules:

\[
\begin{align*}
  V_a &\rightarrow 0 & V_a &\rightarrow 1 & V_{\neg a} &\rightarrow 0 & V_{\neg a} &\rightarrow 1 \\
  V_b &\rightarrow 0 & V_b &\rightarrow 1 & V_{\neg b} &\rightarrow 0 & V_{\neg b} &\rightarrow 1 \\
  V_c &\rightarrow 0 & V_c &\rightarrow 1 & V_{\neg c} &\rightarrow 0 & V_{\neg c} &\rightarrow 1 \\
  V_d &\rightarrow 0 & V_d &\rightarrow 1 & V_{\neg d} &\rightarrow 0 & V_{\neg d} &\rightarrow 1
\end{align*}
\]

\quad \text{Assignment rules}
The reduction (an example)

\[ \phi = (a \lor \neg b \lor c) \land (\neg a \lor b \lor c) \land (d \lor \neg c \lor a) \]

C_1 \quad C_2 \quad C_3

We have so far: \( V_\bullet \to 0|1 \) and \( V_{\neg \bullet} \to 0|1 \) (assignment rules)

For the variables, \( a, b, c, d \) we create the rules:

\[
\begin{align*}
U_{a,1} & \rightarrow V_a V_{\neg a} & U_{a,0} & \rightarrow V_{\neg a} V_a \\
U_{b,1} & \rightarrow V_b V_{\neg b} & U_{b,0} & \rightarrow V_{\neg b} V_b \\
U_{c,1} & \rightarrow V_c V_{\neg c} & U_{c,0} & \rightarrow V_{\neg c} V_c \\
U_{d,1} & \rightarrow V_d V_{\neg d} & U_{d,0} & \rightarrow V_{\neg d} V_d
\end{align*}
\]

\[\text{\leftarrow Consistency rules}\]
The reduction (an example)

\[ \phi = \underbrace{(a \lor \neg b \lor c)}_{C_1} \land \underbrace{(-a \lor b \lor c)}_{C_2} \land \underbrace{(d \lor \neg c \lor a)}_{C_3} \]

We have so far: assignment rules and \( U_{\bullet,1} \rightarrow V_{\bullet}V_{\neg\bullet} \) and \( U_{\bullet,0} \rightarrow V_{\neg\bullet}V_{\bullet} \) (consistency rules)

For the clauses \( C_1 \), \( C_2 \) and \( C_3 \) we create the rules:

\[
\begin{align*}
S_1 & \rightarrow C_1 \\
S_2 & \rightarrow S_1 C_2 \\
S_3 & \rightarrow S_2 C_3 \\
S & \rightarrow S_3
\end{align*}
\]

\( S \) is the start symbol of the grammar
The reduction (an example)

\[ \phi = \underbrace{(a \lor \neg b \lor c)}_{C_1} \land \underbrace{(-a \lor b \lor c)}_{C_2} \land \underbrace{(d \lor \neg c \lor a)}_{C_3} \]

We have so far: assignment rules, consistency rules and clause rules

For the clause \( C_1 \), for example, we create the rules:

\[
\begin{align*}
C_1 & \rightarrow U_{a,1} \quad U_{b,1} \quad U_{c,1} \\
C_1 & \rightarrow U_{a,0} \quad U_{b,1} \quad U_{c,1} \\
C_1 & \rightarrow U_{a,1} \quad U_{b,0} \quad U_{c,1} \\
C_1 & \rightarrow U_{a,1} \quad U_{b,1} \quad U_{c,0} \\
C_1 & \rightarrow U_{a,0} \quad U_{b,1} \quad U_{c,0} \\
C_1 & \rightarrow U_{a,1} \quad U_{b,0} \quad U_{c,0} \\
C_1 & \rightarrow U_{a,0} \quad U_{b,0} \quad U_{c,0}
\end{align*}
\]
The reduction (an example)

\[ \phi = (a \lor \neg b \lor c) \land (\neg a \lor b \lor c) \land (d \lor \neg c \lor a) \]

We have so far: assignment rules, consistency rules, clause rules and satisfaction rules – that’s the complete grammar!

We need to decide on the string to parse, \( x \)

Set \( x = 101010 101010 101010 \)
The reduction (an example)

\[ \phi = (a \lor \neg b \lor c) \land (\neg a \lor b \lor c) \land (d \lor \neg c \lor a) \]

\[
\begin{array}{c}
C_1 \\
101010
\end{array}
\begin{array}{c}
C_2 \\
101010
\end{array}
\begin{array}{c}
C_3 \\
101010
\end{array}
\]

We can use a parse for \( x \) to extract an assignment for the variables
Extracting an assignment

\[ \phi = (a \lor \neg b \lor c) \land (\neg a \lor b \lor c) \land (d \lor \neg c \lor a) \]

If we use the rule \( V_a \rightarrow 0 \) set the variable \( a \) to 0
If we use the rule \( V_a \rightarrow 1 \) set the variable \( a \) to 1
Same for other variables
Note that we use \( V_a \rightarrow \bullet \) and \( V_{\neg a} \rightarrow \bullet \) together
Consistent assignments

\[ \phi = (a \lor \neg b \lor c) \land (\neg a \lor b \lor c) \land (d \lor \neg c \lor a) \]

\( C_1 \) \quad \( C_2 \) \quad \( C_3 \)

But! What if we use both \( V_a \rightarrow 0 \) and \( V_a \rightarrow 1 \)?
Consistent assignments

\[ \phi = (a \lor \neg b \lor c) \land (\neg a \lor b \lor c) \land (d \lor \neg c \lor a) \]

But! What if we use both \( V_a \rightarrow 0 \) and \( V_a \rightarrow 1 \)?

Lemma

Let \( \theta \) be weights for the grammar we constructed. If the (multiplicative) weight of the Viterbi parse of 101010 101010 101010 is 1, then the assignment extracted from the parse tree is consistent
Finding a satisfying assignment

\[ \phi = (a \vee \neg b \vee c) \land (\neg a \vee b \vee c) \land (d \vee \neg c \vee a) \]

Lemma

There exists \( \theta \) such that the Viterbi parse of \( 101010 \overline{101010} \overline{101010} \) is 1 if and only if \( \phi \) is satisfiable. The satisfying assignment is the one extracted from the parse tree with weight 1
Problem (Viterbi Train)

**Input:** $G$ context-free grammar, $x_1, \ldots, x_n$ sentences, $\alpha \in [0, 1]$

**Output:** 1 if there are $\theta$ and $z_1, \ldots, z_n$ derivation trees such that

$$\prod_{i=1}^{n} p(x_i, z_i \mid \theta) \geq \alpha$$

and 0 otherwise.

Corollary

*Viterbi Train is NP hard*

In fact, we have NP completeness (Viterbi Train is in NP)
Reminder, Viterbi Train tries to maximize:

$$\max_{\theta, z_1, \ldots, z_n} \prod_{i=1}^{n} p(x_i, z_i \mid \theta)$$

We know it is hard to find the exact maximum. Can we hope to approximate the maximal solution?
The question we ask is: "is there a $\rho \in (0, 1]$ such that there is an efficient algorithm which returns $z_1', \ldots, z_n'$ and $\theta'$ such that

$$\prod_{i=1}^{n} p(x_i, z_i' | \theta') \geq \rho \left( \max_{\theta, z_1, \ldots, z_n} \prod_{i=1}^{n} p(x_i, z_i | \theta) \right)$$

for any input sentences $x_1, \ldots, x_n$ and a grammar $G$?"
The question we ask is: “is there a \( \rho \in (0, 1] \) such that there is an efficient algorithm which returns \( z'_1, \ldots, z'_n \) and \( \theta' \) such that

\[
\prod_{i=1}^{n} p(x_i, z'_i \mid \theta') \geq \rho \left( \max_{\theta, z_1, \ldots, z_n} \prod_{i=1}^{n} p(x_i, z_i \mid \theta) \right)
\]

for any input sentences \( x_1, \ldots, x_n \) and a grammar \( G \)? ”

Under the \( \text{P} \neq \text{NP} \) assumption, the answer is negative for any \( \rho \in (\frac{1}{2}, 1] \).
The main argument for this negative result relies on:

\[
\max_{\theta, z_1, \ldots, z_n} \prod_{i=1}^{n} p(x_i, z_i \mid \theta) < 1 \quad \iff \quad \max_{\theta, z_1, \ldots, z_n} \prod_{i=1}^{n} p(x_i, z_i \mid \theta) \leq \frac{1}{2}
\]
Approximate solutions

Lemma

\[
\max_{\theta, z_1, \ldots, z_n} \prod_{i=1}^{n} p(x_i, z_i \mid \theta) < 1 \quad \Rightarrow \quad \max_{\theta, z_1, \ldots, z_n} \prod_{i=1}^{n} p(x_i, z_i \mid \theta) \leq \frac{1}{2}
\]

- Maximal value is less than 1 \(\Rightarrow\) we have a nonterminal which is used with more than one rule in the derivations
Approximate solutions

Lemma

\[
\max_{\theta,z_1,\ldots,z_n} \prod_{i=1}^{n} p(x_i, z_i \mid \theta) < 1 \implies \max_{\theta,z_1,\ldots,z_n} \prod_{i=1}^{n} p(x_i, z_i \mid \theta) \leq \frac{1}{2}
\]

- Maximal value is less than 1 \(\implies\) we have a nonterminal which is used with more than one rule in the derivations
- Let \(A \rightarrow \alpha\) be one of these rules
Approximate solutions

Lemma

\[
\max_{\theta, z_1, \ldots, z_n} \prod_{i=1}^{n} p(x_i, z_i \mid \theta) < 1 \implies \max_{\theta, z_1, \ldots, z_n} \prod_{i=1}^{n} p(x_i, z_i \mid \theta) \leq \frac{1}{2}
\]

- Maximal value is less than 1 \(\Rightarrow\) we have a nonterminal which is used with more than one rule in the derivations
- Let \(A \to \alpha\) be one of these rules
- Say \(A \to \alpha\) appears \(k\) times and \(A\) appears \(r\) times in \(z_1, \ldots, z_n\)
Approximate solutions

Lemma

\[
\max_{\theta, z_1, \ldots, z_n} \prod_{i=1}^{n} p(x_i, z_i | \theta) < 1 \implies \max_{\theta, z_1, \ldots, z_n} \prod_{i=1}^{n} p(x_i, z_i | \theta) \leq \frac{1}{2}
\]

- Maximal value is less than 1 \(\implies\) we have a nonterminal which is used with more than one rule in the derivations
- Let \(A \rightarrow \alpha\) be one of these rules
- Say \(A \rightarrow \alpha\) appears \(k\) times and \(A\) appears \(r\) times in \(Z_1, \ldots, Z_n\)
- We know \(r \geq k + 1\)
Approximate solutions

Lemma

\[
\max_{\theta, z_1, \ldots, z_n} \prod_{i=1}^{n} p(x_i, z_i \mid \theta) < 1 \iff \max_{\theta, z_1, \ldots, z_n} \prod_{i=1}^{n} p(x_i, z_i \mid \theta) \leq \frac{1}{2}
\]

- Maximal value is less than 1 \(\Rightarrow\) we have a nonterminal which is used with more than one rule in the derivations
- Let \(A \rightarrow \alpha\) be one of these rules
- Say \(A \rightarrow \alpha\) appears \(k\) times and \(A\) appears \(r\) times in \(z_1, \ldots, z_n\)
- We know \(r \geq k + 1\)
- MLE term in the objective for \(A \rightarrow \alpha\):

\[
\left( \frac{k}{r} \right)^k
\]
Approximate solutions

Lemma

\[
\max_{\theta, z_1, \ldots, z_n} \prod_{i=1}^{n} p(x_i, z_i | \theta) < 1 \implies \max_{\theta, z_1, \ldots, z_n} \prod_{i=1}^{n} p(x_i, z_i | \theta) \leq \frac{1}{2}
\]

- Maximal value is less than 1 \(\implies\) we have a nonterminal which is used with more than one rule in the derivations
- Let \(A \rightarrow \alpha\) be one of these rules
- Say \(A \rightarrow \alpha\) appears \(k\) times and \(A\) appears \(r\) times in \(z_1, \ldots, z_n\)
- We know \(r \geq k + 1\)
- MLE term in the objective for \(A \rightarrow \alpha\):

\[
\left( \frac{k}{r} \right)^k \leq \left( \frac{k}{k+1} \right)^k \leq \frac{1}{2}
\]
Approximate solutions

**Lemma**

\[
\max_{\theta,z_1,\ldots,z_n} \prod_{i=1}^{n} p(x_i, z_i \mid \theta) < 1 \implies \max_{\theta,z_1,\ldots,z_n} \prod_{i=1}^{n} p(x_i, z_i \mid \theta) \leq \frac{1}{2}
\]

- Maximal value is less than 1 ⇒ we have a nonterminal which is used with more than one rule in the derivations
- Let \( A \rightarrow \alpha \) be one of these rules
- Say \( A \rightarrow \alpha \) appears \( k \) times and \( A \) appears \( r \) times in \( z_1, \ldots, z_n \)
- We know \( r \geq k + 1 \)
- MLE term in the objective for \( A \rightarrow \alpha \):

\[
\left( \frac{k}{r} \right)^k \leq \left( \frac{k}{k+1} \right)^k \leq \frac{1}{2}
\]

- Therefore, the whole objective, which multiplies in \( \left( \frac{k}{r} \right)^k \), must be smaller than 1/2
There is experimental evidence that Viterbi EM converges fast.
There is experimental evidence that Viterbi EM converges fast

True or false? Viterbi EM converges in a polynomial number of iterations
There is experimental evidence that Viterbi EM converges fast.

**True or false?** Viterbi EM converges in a polynomial number of iterations.

If it is true, we cannot hope for Viterbi EM to even get us approximately close to the maximum likelihood (in the general case).

However, Viterbi EM can do quite well! (see Spitkovsky et al. at CoNLL later this week)
There is experimental evidence that Viterbi EM converges fast

**True or false?** Viterbi EM converges in a polynomial number of iterations

If it is true, we cannot hope for Viterbi EM to even get us approximately close to the maximum likelihood (in the general case)

However, Viterbi EM can do quite well! (see Spitkovsky et al. at CoNLL later this week)
Other results

A variant of Viterbi EM, called conditional Viterbi EM, maximizes the conditional likelihood $p(z \mid x, \theta)$ in the M-step.

Theorem

*The decision problem of conditional Viterbi EM (for PCFGs) is NP hard.*
Other results

A variant of Viterbi EM, called **conditional Viterbi EM**, maximizes the conditional likelihood $p(z \mid x, \theta)$ in the M-step.

**Theorem**

*The decision problem of conditional Viterbi EM (for PCFGs) is NP hard*

***

What about just EM (marginalized likelihood)?

**Theorem**

*The decision problem of EM (for PCFGs) is NP hard*

Complements well-known results *(Abe and Warmuth, 1992)*

See paper!
Open problems

Note that our grammar is not recursive – the results can be strengthened to HMMs

The grammar grows linearly with the size of the formula

Does the problem become more tractable if we limit the size of the grammar?
  - Constant size - *maybe polynomial in length of input*?
Note that our grammar is not recursive – the results can be strengthened to HMMs.

The grammar grows linearly with the size of the formula.

Does the problem become more tractable if we limit the size of the grammar?

- Constant size - maybe polynomial in length of input?
- Constant number of rules not rewriting to terminals (use recursive power) - maybe
Open problems

Note that our grammar is not recursive – the results can be strengthened to HMMs

The grammar grows linearly with the size of the formula

Does the problem become more tractable if we limit the size of the grammar?

- Constant size - maybe polynomial in length of input?
- Constant number of rules not rewriting to terminals (use recursive power) - maybe
- Universal grammar for all formulas? yes, size depends on number of variables

See paper for relationship to $k$-means clustering
We described hardness results for Viterbi training.

We described evidence that Viterbi EM is not an approximation algorithm in the traditional sense.

This does not mean that Viterbi EM cannot get good performance (likelihood vs. evaluation metric).

Read paper for more: some motivation for using uniform initialization with Viterbi EM.
Questions?
Global maximization vs. initialization bias

- Initialization gives bias, could be better than global optimization
- Global optimization can lead to degenerate solutions
- Problem should disappear if we have more data
- The same way we want to maximize marginalized likelihood globally (but use EM instead), we want to maximize the likelihood with respect to the elements as well
We could imagine switching to (negative) log-likelihood – the core hardness result stays the same, we would just change the range of $\alpha$ to $[0, \infty)$.

The multiplicative approximation result for the log-likelihood becomes an additive approximation result for the negated log-likelihood.

A multiplicative approximation result for the log-likelihood becomes rather vacuous (but should still hold) – because our reduction makes sure that the minimal negated log-likelihood is going to be 0 if there is a satisfying formula.