Algorithms and complexity of analyzing unrestricted stochastic context-free grammars

Kousha Etessami

U. of Edinburgh

Based on joint works with:

Alistair Stewart, U. of Edinburgh

Mihalis Yannakakis, Columbia University

FSMNLP

St. Andrews, July 17th, 2013

Stochastic Context-Free Grammars

$$R \xrightarrow{1/3} aBBcGdR$$

$$R \xrightarrow{1/2} bcBbR$$

$$R \xrightarrow{1/6} \epsilon$$

$$B \xrightarrow{1/4} eeRRf$$

$$B \xrightarrow{3/4} g$$

$$G \xrightarrow{1} aBcRRb$$

Stochastic Context-Free Grammars

$$R \xrightarrow{1/3} aBBcGdR$$

$$R \xrightarrow{1/2} bcBbR$$

$$R \xrightarrow{1/6} \epsilon$$

$$B \xrightarrow{1/4} eeRRf$$

$$B \xrightarrow{3/4} g$$

$$G \xrightarrow{1} aBcRRb$$

Leftmost derivation

$$\underline{R}$$

Stochastic Context-Free Grammars

$$R \xrightarrow{1/3} aBBcGdR$$

$$R \xrightarrow{1/2} bcBbR$$

$$R \xrightarrow{1/6} \epsilon$$

$$B \xrightarrow{1/4} eeRRf$$

$$B \xrightarrow{3/4} g$$

$$G \xrightarrow{1} aBcRRb$$

Leftmost derivation

$$\underline{R}$$

$$\xrightarrow{1/2} \quad bc\underline{B}bR$$

Stochastic Context-Free Grammars

$$R \xrightarrow{1/3} aBBcGdR$$

$$R \xrightarrow{1/2} bcBbR$$

$$R \xrightarrow{1/6} \epsilon$$

$$B \xrightarrow{1/4} eeRRf$$

$$B \xrightarrow{3/4} g$$

$$G \xrightarrow{1} aBcRRb$$

Leftmost derivation

$$\underline{R}$$

$$\xrightarrow{1/2} \quad bc\underline{B}bR$$

$$\xrightarrow{1/4} \quad bcee\underline{R}RfbR$$

Stochastic Context-Free Grammars

$$R \xrightarrow{1/3} aBBcGdR$$

$$R \xrightarrow{1/2} bcBbR$$

$$R \xrightarrow{1/6} \epsilon$$

$$B \xrightarrow{1/4} eeRRf$$

$$B \xrightarrow{3/4} g$$

$$G \xrightarrow{1} aBcRRb$$

Leftmost derivation

$$\underline{R}$$

$$\xrightarrow{1/2} \quad bc\underline{B}bR$$

$$\xrightarrow{1/4} \quad bcee\underline{R}RfbR$$

$$\xrightarrow{1/6} \quad bcee\ \underline{R}fbR$$

Stochastic Context-Free Grammars

$$R \xrightarrow{1/3} aBBcGdR$$

$$R \xrightarrow{1/2} bcBbR$$

$$R \xrightarrow{1/6} \epsilon$$

$$B \xrightarrow{1/4} eeRRf$$

$$B \xrightarrow{3/4} g$$

$$G \xrightarrow{1} aBcRRb$$

Leftmost derivation

$$\begin{array}{ll} & \underline{R} \\ \xrightarrow{1/2} & bc\underline{B}bR \\ \xrightarrow{1/4} & bcee\underline{R}RfbR \\ \xrightarrow{1/6} & bcee\ \underline{R}fbR \\ \xrightarrow{1/6} & bcee\ \ fb\underline{R} \end{array}$$

Stochastic Context-Free Grammars

$$R \xrightarrow{1/3} aBBcGdR$$

$$R \xrightarrow{1/2} bcBbR$$

$$R \xrightarrow{1/6} \epsilon$$

$$B \xrightarrow{1/4} eeRRf$$

$$B \xrightarrow{3/4} g$$

$$G \xrightarrow{1} aBcRRb$$

Leftmost derivation

$$\begin{array}{ll}
 & \underline{R} \\
\xrightarrow{1/2} & bc\underline{B}bR \\
\xrightarrow{1/4} & bcee\underline{R}RfbR \\
\xrightarrow{1/6} & bcee\ \underline{R}fbR \\
\xrightarrow{1/6} & bcee\ fb\underline{R} \\
\xrightarrow{1/6} & bceefb
\end{array}$$

probability of this derivation: $\frac{1}{2} \cdot \frac{1}{4} \cdot \frac{1}{6}^3$

Stochastic Context-Free Grammars

$$R \xrightarrow{1/3} aBBcGdR$$

$$R \xrightarrow{1/2} bcBbR$$

$$R \xrightarrow{1/6} \epsilon$$

$$B \xrightarrow{1/4} eeRRf$$

$$B \xrightarrow{3/4} g$$

$$G \xrightarrow{1} aBcRRb$$

Leftmost derivation

$$\begin{array}{ll} & \underline{R} \\ \xrightarrow{1/2} & bc\underline{B}bR \\ \xrightarrow{1/4} & bcee\underline{R}RfbR \\ \xrightarrow{1/6} & bcee\ \underline{R}fbR \\ \xrightarrow{1/6} & bcee\ fb\underline{R} \\ \xrightarrow{1/6} & bceefb \end{array}$$

probability of this derivation: $\frac{1}{2} \cdot \frac{1}{4} \cdot \frac{1}{6}^3$

Total "inside" probability of generating string *bceefb* is the sum of the probabilities of all its (left-most) derivations.

Some basic computational questions for SCFGs

Given an unrestricted SCFG, G, what is the complexity of these tasks?

1. Compute the probability, p_G, that G generates a finite parse tree, i.e., the probability that a random derivation of G eventually terminates.

Some basic computational questions for SCFGs

Given an unrestricted SCFG, G, what is the complexity of these tasks?

1. Compute the probability, p_G, that G generates a finite parse tree, i.e., the probability that a random derivation of G eventually terminates.
2. Given also a string, w, compute the "inside" probability, $p_{G,w}$, that a derivation of G generates the finite string w.

Some basic computational questions for SCFGs

Given an unrestricted SCFG, G, what is the complexity of these tasks?

1. Compute the probability, p_G, that G generates a finite parse tree, i.e., the probability that a random derivation of G eventually terminates.
2. Given also a string, w, compute the "inside" probability, $p_{G,w}$, that a derivation of G generates the finite string w.
3. Given also a string, w, compute the maximum parse tree/probability, $p^{\max}_{G,w}$ of w, i.e., the maximum probability (left-most) derivation of w by G. And, decide if $p^{\max}_{G,w} \geq q$, for a given rational probability q.

Some basic computational questions for SCFGs

Given an unrestricted SCFG, G, what is the complexity of these tasks?

1. Compute the probability, p_G, that G generates a finite parse tree, i.e., the probability that a random derivation of G eventually terminates.
2. Given also a string, w, compute the "inside" probability, $p_{G,w}$, that a derivation of G generates the finite string w.
3. Given also a string, w, compute the maximum parse tree/probability, $p^{\max}_{G,w}$ of w, i.e., the maximum probability (left-most) derivation of w by G. And, decide if $p^{\max}_{G,w} \geq q$, for a given rational probability q.
4. Given also a DFA, D, compute the probability, $p_{G,D}$ that G generates a string in the regular language, $L(D)$.

Some basic computational questions for SCFGs

Given an unrestricted SCFG, G, what is the complexity of these tasks?

1. Compute the probability, p_G, that G generates a finite parse tree, i.e., the probability that a random derivation of G eventually terminates.
2. Given also a string, w, compute the "inside" probability, $p_{G,w}$, that a derivation of G generates the finite string w.
3. Given also a string, w, compute the maximum parse tree/probability, $p^{\max}_{G,w}$ of w, i.e., the maximum probability (left-most) derivation of w by G. And, decide if $p^{\max}_{G,w} \geq q$, for a given rational probability q.
4. Given also a DFA, D, compute the probability, $p_{G,D}$ that G generates a string in the regular language, $L(D)$.
5. Convert G to normal form (e.g., CNF), G', such that G and G' are suitably "equivalent" (also in terms of probabilities of strings).

Some basic computational questions for SCFGs

Given an unrestricted SCFG, G, what is the complexity of these tasks?

1. Compute the probability, p_G, that G generates a finite parse tree, i.e., the probability that a random derivation of G eventually terminates.
2. Given also a string, w, compute the "inside" probability, $p_{G,w}$, that a derivation of G generates the finite string w.
3. Given also a string, w, compute the maximum parse tree/probability, $p^{\max}_{G,w}$ of w, i.e., the maximum probability (left-most) derivation of w by G. And, decide if $p^{\max}_{G,w} \geq q$, for a given rational probability q.
4. Given also a DFA, D, compute the probability, $p_{G,D}$ that G generates a string in the regular language, $L(D)$.
5. Convert G to normal form (e.g., CNF), G', such that G and G' are suitably "equivalent" (also in terms of probabilities of strings).

It may surprise you to know: until recently, for arbitrary (unrestricted) SCFGs, the complexity of all of these problems was open.

Stochastic Context-Free G

$$R \xrightarrow{1/3} aBBcGaabR$$

$$R \xrightarrow{1/2} bcBbR$$

$$R \xrightarrow{1/6} \epsilon$$

$$B \xrightarrow{1/4} bbRRc$$

$$B \xrightarrow{3/4} a$$

$$G \xrightarrow{1} aBcRRb$$

Question

What is the probability of termination, i.e., eventually generating a finite string, starting with one non-terminal, R ?

Stochastic Context-Free G

$$R \xrightarrow{1/3} aBBcGaabR$$

$$R \xrightarrow{1/2} bcBbR$$

$$R \xrightarrow{1/6} \epsilon$$

$$B \xrightarrow{1/4} bbRRc$$

$$B \xrightarrow{3/4} a$$

$$G \xrightarrow{1} aBcRRb$$

Question

What is the probability of termination, i.e., eventually generating a finite string, starting with one non-terminal, R ?

$$x_R \;=$$

$$R \xrightarrow{1/3} aBBcGaabR$$

$$R \xrightarrow{1/2} bcBbR$$

$$R \xrightarrow{1/6} \epsilon$$

$$B \xrightarrow{1/4} bbRRc$$

$$B \xrightarrow{3/4} a$$

$$G \xrightarrow{1} aBcRRb$$

Question

What is the probability of termination, i.e., eventually generating a finite string, starting with one non-terminal, R ?

$$x_R = \frac{1}{3}x_B^2 x_G x_R + \frac{1}{2}x_B x_R + \frac{1}{6}$$

$$R \xrightarrow{1/3} aBBcGaabR$$
$$R \xrightarrow{1/2} bcBbR$$
$$R \xrightarrow{1/6} \epsilon$$

$$B \xrightarrow{1/4} bbRRc$$
$$B \xrightarrow{3/4} a$$

$$G \xrightarrow{1} aBcRRb$$

Question

What is the probability of termination, i.e., eventually generating a finite string, starting with one non-terminal, R ?

$$
\begin{aligned}
x_R &= \frac{1}{3}x_B^2 x_G x_R + \frac{1}{2}x_B x_R + \frac{1}{6} \\
x_B &= \frac{1}{4}x_R^2 + \frac{3}{4} \\
x_G &= x_B x_R^2
\end{aligned}
$$

Stochastic Context-Free G

$$R \xrightarrow{1/3} aBBcGaabR$$

$$R \xrightarrow{1/2} bcBbR$$

$$R \xrightarrow{1/6} \epsilon$$

$$B \xrightarrow{1/4} bbRRc$$

$$B \xrightarrow{3/4} a$$

$$G \xrightarrow{1} aBcRRb$$

Question

What is the probability of termination, i.e., eventually generating a finite string, starting with one non-terminal, R ?

$$
\begin{aligned}
x_R &= \frac{1}{3}x_B^2 x_G x_R + \frac{1}{2}x_B x_R + \frac{1}{6} \\
x_B &= \frac{1}{4}x_R^2 + \frac{3}{4} \\
x_G &= x_B x_R^2
\end{aligned}
$$

We get nonlinear fixed point equations,

$$\bar{\mathbf{x}} = P(\bar{\mathbf{x}}).$$

Stochastic Context-Free G

$$R \xrightarrow{1/3} aBBcGaabR$$

$$R \xrightarrow{1/2} bcBbR$$

$$R \xrightarrow{1/6} \epsilon$$

$$B \xrightarrow{1/4} bbRRc$$

$$B \xrightarrow{3/4} a$$

$$G \xrightarrow{1} aBcRRb$$

Question

What is the probability of termination, i.e., eventually generating a finite string, starting with one non-terminal, R ?

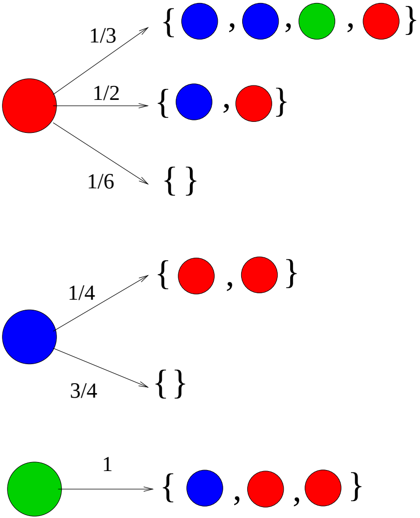
$$\begin{aligned} x_R &= \frac{1}{3}x_B^2 x_G x_R + \frac{1}{2}x_B x_R + \frac{1}{6} \\ x_B &= \frac{1}{4}x_R^2 + \frac{3}{4} \\ x_G &= x_B x_R^2 \end{aligned}$$

We get nonlinear fixed point equations,

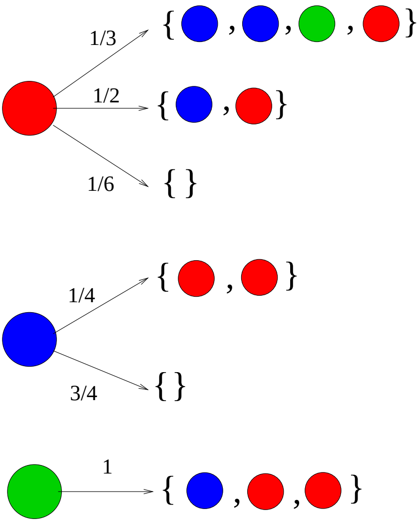
$$\bar{\mathbf{x}} = P(\bar{\mathbf{x}}).$$

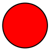
Fact: Termination probabilities (also called the partition function of the SCFG) are the least fixed point, $\mathbf{q}^* \in [0,1]^3$, of $\bar{\mathbf{x}} = P(\bar{\mathbf{x}})$.

Multi-type Branching Processes (Kolmogorov,1940s)

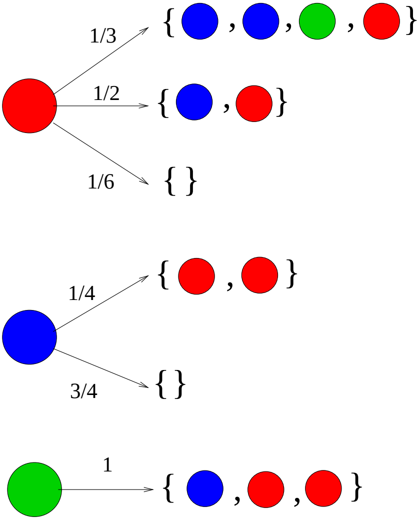


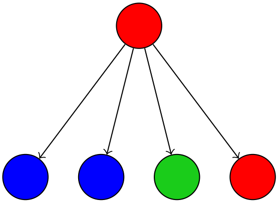
Multi-type Branching Processes (Kolmogorov,1940s)



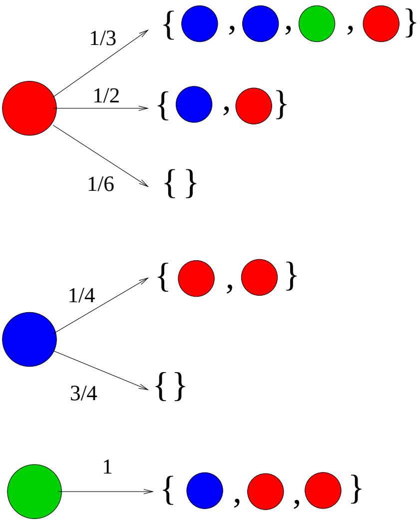


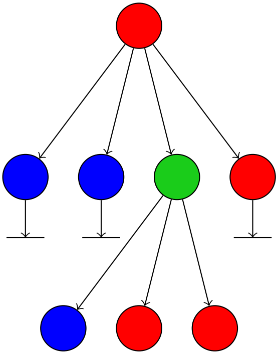
Multi-type Branching Processes (Kolmogorov,1940s)



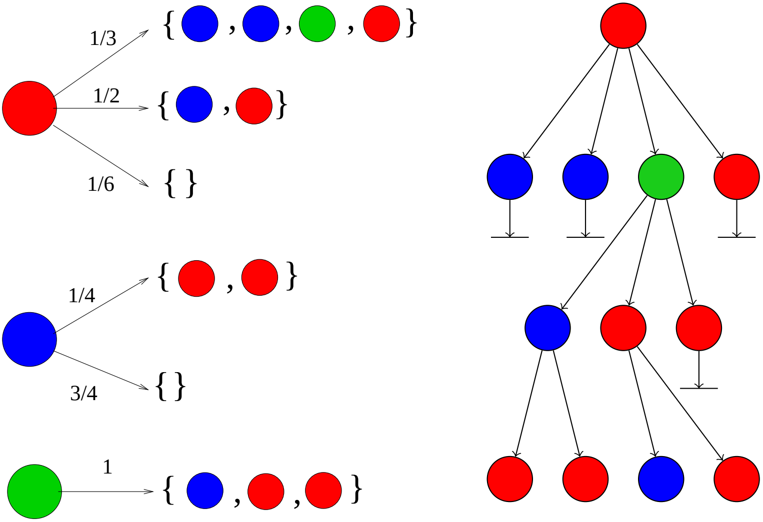


Multi-type Branching Processes (Kolmogorov,1940s)

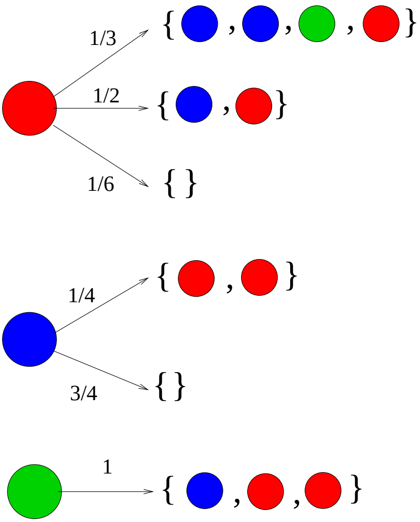


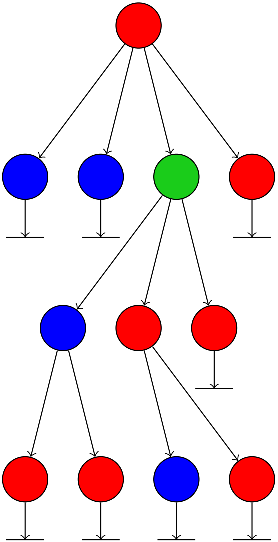


Multi-type Branching Processes (Kolmogorov,1940s)

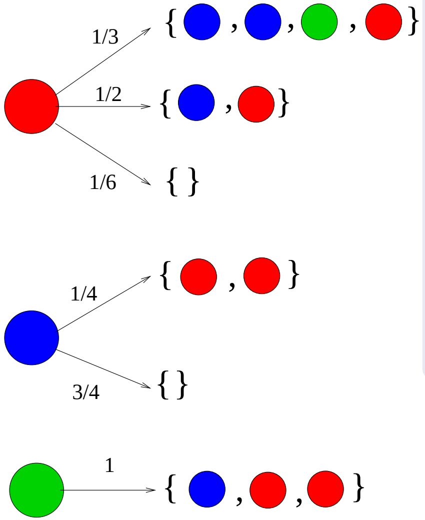


Multi-type Branching Processes (Kolmogorov,1940s)

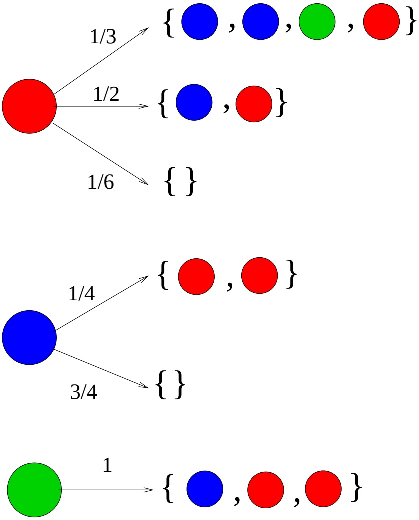




Multi-type Branching Processes (Kolmogorov 1940s)



Question: What is the probability of eventual extinction, starting with one  ?



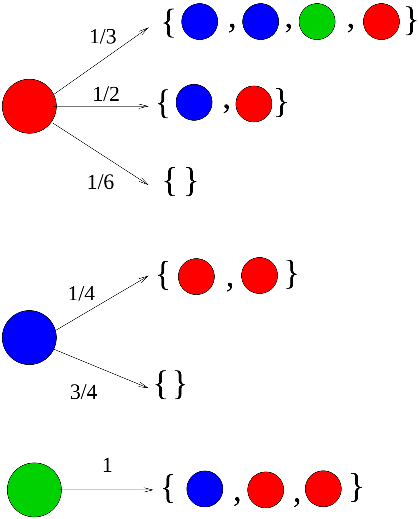
Question: What is the probability of eventual extinction, starting with one ?

$$
\begin{aligned}
x_R &= \frac{1}{3}x_B^2 x_G x_R + \frac{1}{2}x_B x_R + \frac{1}{6} \\
x_B &= \frac{1}{4}x_R^2 + \frac{3}{4} \\
x_G &= x_B x_R^2
\end{aligned}
$$

We get the same fixed point equations:

$$\bar{\mathbf{x}} = P(\bar{\mathbf{x}}).$$

Multi-type Branching Processes (Kolmogorov 1940s)



Question: What is the probability of eventual extinction, starting with one  ?

$$x_R = \frac{1}{3}x_B^2 x_G x_R + \frac{1}{2}x_B x_R + \frac{1}{6}$$

$$x_B = \frac{1}{4}x_R^2 + \frac{3}{4}$$

$$x_G = x_B x_R^2$$

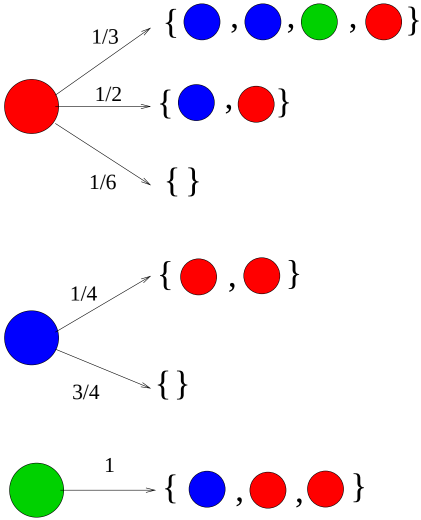
We get the same fixed point equations:

$$\bar{\mathbf{x}} = P(\bar{\mathbf{x}}).$$

Fact

The extinction probabilities are the least fixed point, $\mathbf{q}^* \in [0,1]^3$, of $\bar{\mathbf{x}} = P(\bar{\mathbf{x}})$.

Multi-type Branching Processes (Kolmogorov 1940s)



Question: What is the probability of eventual extinction, starting with one  ?

$$x_R = \frac{1}{3} x_B^2 x_G x_R + \frac{1}{2} x_B x_R + \frac{1}{6}$$

$$x_B = \frac{1}{4} x_R^2 + \frac{3}{4}$$

$$x_G = x_B x_R^2$$

We get the same fixed point equations:

$$\bar{\mathbf{x}} = P(\bar{\mathbf{x}}).$$

Fact

The extinction probabilities are the least fixed point, $\mathbf{q}^* \in [0,1]^3$, of $\bar{\mathbf{x}} = P(\bar{\mathbf{x}})$. $q_R^* = 0.276$; $q_B^* = 0.769$; $q_G^* = 0.059$.

The "inside" probability of a string.

$$S \xrightarrow{1/3} \text{"Hello"}\ R$$

$$R \xrightarrow{1/3} BBGR$$

$$R \xrightarrow{1/2} BR$$

$$R \xrightarrow{1/6} \epsilon$$

$$B \xrightarrow{1/4} RR$$

$$B \xrightarrow{3/4} \epsilon$$

$$G \xrightarrow{1} BRR$$

The "inside" probability of a string.

$$S \xrightarrow{1/3} \text{"Hello"}\ R$$

$$R \xrightarrow{1/3} B B G R$$

$$R \xrightarrow{1/2} B R$$

$$R \xrightarrow{1/6} \epsilon$$

$$B \xrightarrow{1/4} R R$$

$$B \xrightarrow{3/4} \epsilon$$

$$G \xrightarrow{1} B R R$$

Question

What is the inside probability of generating the string "Hello", starting at S?

The "inside" probability of a string.

$S \xrightarrow{1/3}$ "Hello" R

$R \xrightarrow{1/3} BBGR$

$R \xrightarrow{1/2} BR$

$R \xrightarrow{1/6} \epsilon$

$B \xrightarrow{1/4} RR$

$B \xrightarrow{3/4} \epsilon$

$G \xrightarrow{1} BRR$

Question

What is the inside probability of generating the string "Hello", starting at S?

Again, it is the same termination probability, q^*_R, as before.

The "inside" probability of a string.

$S \xrightarrow{1/3}$ "Hello" R

$R \xrightarrow{1/3} BBGR$

$R \xrightarrow{1/2} BR$

$R \xrightarrow{1/6} \epsilon$

$B \xrightarrow{1/4} RR$

$B \xrightarrow{3/4} \epsilon$

$G \xrightarrow{1} BRR$

Question

What is the inside probability of generating the string "Hello", starting at S?

Again, it is the same termination probability, q^*_R, as before.

In general, computing/approximating inside probabilities is more involved than just computing termination probabilities. (But we will show it can be reduced in P-time to computing termination probabilities.)

The "inside" probability of a string.

$$S \xrightarrow{1/3} \text{“Hello”}\ R$$

$$R \xrightarrow{1/3} B\,B\,G\,R$$

$$R \xrightarrow{1/2} B\,R$$

$$R \xrightarrow{1/6} \epsilon$$

$$B \xrightarrow{1/4} R\,R$$

$$B \xrightarrow{3/4} \epsilon$$

$$G \xrightarrow{1} B\,R\,R$$

Question

What is the inside probability of generating the string "Hello", starting at S?

Again, it is the same termination probability, q^*_R, as before.

In general, computing/approximating inside probabilities is more involved than just computing termination probabilities. (But we will show it can be reduced in P-time to computing termination probabilities.)

In NLP terminination probabilities are also called the partition function of the SCFG. They have lots of applications (see, e.g., [Nederhof-Satta,2008]).

Probabilistic Polynomial Systems of Equations and SCFGs

$$\frac{1}{3}x_B^2 x_G x_R + \frac{1}{2}x_B x_R + \frac{1}{6}$$

is a Probabilistic Polynomial: the coefficients are positive and sum to 1.

A Probabilistic Polynomial System (PPS), is a system of n equations

$$\mathbf{x} = P(\mathbf{x})$$

in n variables where each $P_i(x)$ is a probabilistic polynomial.

Every multi-type Branching Process (BP) with n types, and every SCFG with n nonterminals, corresponds to a PPS, and vice-versa.

Weighted Context-Free Grammars (WCFGs)

$$R \xrightarrow{3} a\,B\,B\,c\,G\,d\,R$$

$$R \xrightarrow{1/2} b\,c\,B\,b\,R$$

$$R \xrightarrow{5} \epsilon$$

$$B \xrightarrow{4} e\,e\,R\,R\,f$$

$$B \xrightarrow{3/4} g$$

$$G \xrightarrow{2} a\,B\,c\,R\,R\,b$$

Weighted Context-Free Grammars (WCFGs)

$$R \xrightarrow{3} aBBcGdR$$

$$R \xrightarrow{1/2} bcBbR$$

$$R \xrightarrow{5} \epsilon$$

$$B \xrightarrow{4} eeRRf$$

$$B \xrightarrow{3/4} g$$

$$G \xrightarrow{2} aBcRRb$$

Leftmost derivation

$\underline{R}$

Weighted Context-Free Grammars (WCFGs)

$$R \xrightarrow{3} aBBcGdR$$

$$R \xrightarrow{1/2} bcBbR$$

$$R \xrightarrow{5} \epsilon$$

$$B \xrightarrow{4} eeRRf$$

$$B \xrightarrow{3/4} g$$

$$G \xrightarrow{2} aBcRRb$$

Leftmost derivation

$$\underline{R}$$

$$\xrightarrow{1/2} \quad bc\underline{B}bR$$

Weighted Context-Free Grammars (WCFGs)

$$R \xrightarrow{3} a\,B\,B\,c\,G\,d\,R$$

$$R \xrightarrow{1/2} bc\,B\,b\,R$$

$$R \xrightarrow{5} \epsilon$$

$$B \xrightarrow{4} ee\,R\,R\,f$$

$$B \xrightarrow{3/4} g$$

$$G \xrightarrow{2} a\,B\,c\,R\,R\,b$$

Leftmost derivation

$$\underline{R}$$

$$\xrightarrow{1/2} \quad bc\,\underline{B}\,b\,R$$

$$\xrightarrow{4} \quad bcee\,\underline{R}\,R\,fb\,R$$

Weighted Context-Free Grammars (WCFGs)

$$R \xrightarrow{3} aBBcGdR$$

$$R \xrightarrow{1/2} bcBbR$$

$$R \xrightarrow{5} \epsilon$$

$$B \xrightarrow{4} eeRRf$$

$$B \xrightarrow{3/4} g$$

$$G \xrightarrow{2} aBcRRb$$

Leftmost derivation

$$\underline{R}$$

$$\xrightarrow{1/2} bc\underline{B}bR$$

$$\xrightarrow{4} bcee\underline{R}RfbR$$

$$\xrightarrow{5} bcee\ \underline{R}fbR$$

Weighted Context-Free Grammars (WCFGs)

$$R \xrightarrow{3} aBBcGdR$$

$$R \xrightarrow{1/2} bcBbR$$

$$R \xrightarrow{5} \epsilon$$

$$B \xrightarrow{4} eeRRf$$

$$B \xrightarrow{3/4} g$$

$$G \xrightarrow{2} aBcRRb$$

Leftmost derivation

$$
\begin{array}{ll}
 & \underline{R} \\
\xrightarrow{1/2} & bc\underline{B}bR \\
\xrightarrow{4} & bcee\underline{R}RfbR \\
\xrightarrow{5} & bcee\ \underline{R}fbR \\
\xrightarrow{5} & bcee\ \ fb\underline{R}
\end{array}
$$

Weighted Context-Free Grammars (WCFGs)

$$R \xrightarrow{3} aBBcGdR$$

$$R \xrightarrow{1/2} bcBbR$$

$$R \xrightarrow{5} \epsilon$$

$$B \xrightarrow{4} eeRRf$$

$$B \xrightarrow{3/4} g$$

$$G \xrightarrow{2} aBcRRb$$

Leftmost derivation

$$\begin{aligned} & \underline{R} \\ \xrightarrow{1/2} \quad & bc\underline{B}bR \\ \xrightarrow{4} \quad & bcee\underline{R}RfbR \\ \xrightarrow{5} \quad & bcee\ \underline{R}fbR \\ \xrightarrow{5} \quad & bcee\ fb\underline{R} \\ \xrightarrow{5} \quad & bceefb \end{aligned}$$

weight of this derivation: $\frac{1}{2} \cdot 4 \cdot 5^3$

Weighted Context-Free Grammars (WCFGs)

$$R \xrightarrow{3} aBBcGdR$$

$$R \xrightarrow{1/2} bcBbR$$

$$R \xrightarrow{5} \epsilon$$

$$B \xrightarrow{4} eeRRf$$

$$B \xrightarrow{3/4} g$$

$$G \xrightarrow{2} aBcRRb$$

Leftmost derivation

$$\begin{array}{ll} & \underline{R} \\ \xrightarrow{1/2} & bc\underline{B}bR \\ \xrightarrow{4} & bcee\underline{R}RfbR \\ \xrightarrow{5} & bcee\ \underline{R}fbR \\ \xrightarrow{5} & bcee\ \ fb\underline{R} \\ \xrightarrow{5} & bceefb \end{array}$$

weight of this derivation: $\frac{1}{2} \cdot 4 \cdot 5^3$

Total weight of string *bceefb* is the sum of the weights of all its (left-most) derivations. (This may in general be ∞.)

Monotone Polynomial Systems of Equations and WCFGs

$$3\, x_B^2 x_G x_R + \frac{2}{3} x_B x_R + 5$$

is a Monotone Polynomial: the coefficients are positive (but they don't necessarily sum to one).

A Monotone Polynomial System (MPS), is a system of n equations

$$\mathbf{x} = P(\mathbf{x})$$

in n variables where each $P_i(x)$ is a monotone polynomial.

Every Weighted-CFG (WCFG) with n nonterminals, corresponds to a MPS, and vice-versa.

Basic properties of PPSs, $\mathbf{x} = P(\mathbf{x})$ for SCFGs

For every PPS, $P : [0,1]^n \rightarrow [0,1]^n$ defines a monotone map on $[0,1]^n$.

Proposition

- *A PPS, $x = P(x)$ has a least fixed point, $q^* \in [0,1]^n$.*
  (q^* can be irrational.)
- $q^* = \lim_{k\to\infty} P^k(\mathbf{0})$.
- *q^* is vector of extinction/termination probabilities (the partition function) for the BP (SCFG).*

Question

Can we compute the probabilities q^* efficiently (in P-time)?

First considered by Kolmogorov & Sevastyanov (1940s).

Basic properties of MPSs, $\mathbf{x} = P(\mathbf{x})$, for WCFGs

For every MPS, $P : [0,\infty]^n \rightarrow [0,\infty]^n$ defines a monotone map on $[0,\infty]^n$.

Proposition

- *A MPS, $x = P(x)$ has a least fixed point, $q^* \in [0,+\infty]^n$. (q^* can be irrational.)*
- $q^* = \lim_{k\rightarrow\infty} P^k(\mathbf{0})$.
- *q^* is the (generalized) "partition function" for the WCFG.*

Question

Can we compute q^* efficiently (in P-time) for MPSs and WCFGs?

Value iteration can require exponentially many iterations, already for simple PPSs and SCFGs

Why not just do value iteration?
I.e., start with $x^0 := \mathbf{0}$, and let $x^{i+1} := P(x^i) = P^i(0)$, $i = 1, 2, 3, \ldots$

Question

How many iterations, m, is required for $x^m := P^m(0)$ to be within i bits of precision (i.e., to within additive error $1/2^i$) of the solution q^*?

Answer

In the worst case, at least exponentially many iterations in i, even for a fixed univariate PPS:

Univariate PPS:

$$x = (1/2)x^2 + 1/2$$

Fact ([E.-Yannakakis'05]) **:** $q^* = 1$, but for all $m \leq 2^i$,

$$|1 - P^m(0)| \geq 1/2^i$$

two "hard" problems

Sqrt-Sum: the square-root sum problem is the following decision problem:

Given $(d_1, \ldots, d_n) \in \mathbb{N}^n$ and $k \in \mathbb{N}$, decide whether $\sum_{i=1}^{n} \sqrt{d_i} \leq k$.

Solvable in PSPACE.

Open problem ([GareyGrahamJohnson'76]) whether it is in NP (or even the polynomial time hierarchy).

PosSLP: Given an arithmetic circuit (Straight Line Program) with gates $\{+, *, -\}$ with integer inputs, decide whether the output is > 0.

PosSLP captures all of polynomial time in the unit-cost arithmetic RAM model of computation.

[Allender, Bürgisser, Kjeldal-Petersen, Miltersen,2006] Gave a (Turing) reduction from Sqrt-Sum to PosSLP and showed both can be decided in the Counting Hierarchy: $P^{PP^{PP^{PP}}}$. Nothing better is known.

some bad news

Theorem ([E.-Yannakakis'05,'07])

Both Sqrt-Sum and PosSLP are P-time reducible to both of the following problems:

1. *Given a PPS, $x = P(x)$, decide whether $q_1^* \geq 1/2$.*
   (Or, decide whether $q_1^* \geq p$ for any given rational $p \in (0,1)$.)

2. *Given a MPS, $x = P(x)$, even one that has an LFP, $q^* \in [0,1]^n$, compute any non-trivial approximation of q^*. More precisely:*
   - *For any fixed $\epsilon > 0$, given a MPS with the promise that either (a) $q_1^* = 1$, or (b) $q_1^* \leq \epsilon$; decide which of (a) or (b) is the case.*

toward some good news

Newton's method

Seeking a solution to $F(\mathbf{x}) = 0$, we start at initial guess vector $\mathbf{x}^{(0)}$, and compute the sequence, $x^{(k)}$, $k \to \infty$, where:

$$\mathbf{x}^{(k+1)} := \mathbf{x}^{(k)} - (F'(\mathbf{x}^{(k)}))^{-1} F(\mathbf{x}^{(k)})$$

Here $F'(\mathbf{x})$, is the *Jacobian matrix*, of partial derivatives, given by

$$F'(\mathbf{x}) = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} \cdots \frac{\partial F_1}{\partial x_n} \\ \vdots \vdots \vdots \\ \frac{\partial F_n}{\partial x_1} \cdots \frac{\partial F_n}{\partial x_n} \end{bmatrix}$$

For PPSs, Newton iteration looks like:

$$\mathbf{x}^{(k+1)} := \mathbf{x}^{(k)} + (I - P'(\mathbf{x}^{(k)}))^{-1}(P(\mathbf{x}^{(k)}) - \mathbf{x}^{(k)})$$

where $P'(x)$ is the Jacobian matrix of $P(x)$. For $z \in [0,1]^n$, let:

$$\mathcal{N}_P(z) := z + (I - P'(z))^{-1}(P(z) - z)$$

define the Newton operator on $x = P(x)$.

Newton on PPSs and MPSs

Let $F(\mathbf{x}) = P(\mathbf{x}) - \mathbf{x}$.
We can decompose $x = P(x)$ into its strongly connected components (SCCs), based on variable dependencies, and eliminate "0" variables.

Theorem (Decomposed Newton's method for MPSs [E.-Yannakakis'05])

Starting at $x_0 := \mathbf{0}$, and working "bottom-up" on the SCCs of the decomposition DAG of $x = P(x)$, Newton's method "monotonically converges" to the LFP $q^* \in [0, \infty)^n$, i.e., $\lim_{k\to\infty} \mathbf{x}_k \uparrow q^*$.

Implemented in PReMo (`http://groups.inf.ed.ac.uk/premo/`), by D. Wojtczak [Wojtczak-E.,'07]. Also implemented by [Nederhof-Satta,'08]. Experiments show some good performance.

What is Newton's worst case behavior for PPSs and MPSs?

[Esparza,Kiefer,Luttenberger,'10] subsequently studied the convergence of Newton's method on PPSs and MPSs in greater detail.

- They gave simple examples of PPSs, $x = P(x)$, requiring exponentially many iterations (as a function of the encoding size $|P|$ of the equations) to converge to within additive error $< 1/2$. Example ([SEY'13]): $x_i = \frac{1}{2}x_i^2 + \frac{1}{2}x_{i-1}$, for $i = 1, \ldots, n$; $x_0 = 1$;
- For strongly-connected equation systems they gave an exponential upper bound, as a function of the size of the system, and linear in the number of bits of precision required.
- They gave no upper bounds on the number of iterations, *as a function of the system size*, for arbitrary PPSs (or MPSs).

Recently [Stewart-E.-Yannakakis'2013], we have given worst-case upper bounds for Newton on arbitrary PPSs and MPSs, as a function of both $|P|$ and $\log(1/\epsilon)$, to converge to within error $\epsilon > 0$.
Our bounds are essentially optional in several parameters.

P-time approximation for PPSs

Theorem ([E.-Stewart-Yannakakis,STOC'12])

Given a PPS, $\mathbf{x} = P(\mathbf{x})$, with LFP $\mathbf{q}^* \in [0,1]^n$, we can compute a rational vector $\mathbf{v} \in [0,1]^n$ such that

$$\|\mathbf{v} - \mathbf{q}^*\|_\infty \leq 2^{-j}$$

in time polynomial in both the encoding size $|P|$ of the equations and in j (the number of "bits of precision").

We use Newton's method..... but how?

Qualitative problems for PPSs are in P-time

Proposition

For a PPS or MPS, $x = P(x)$, deciding whether $q^*_i = 0$ is in P-time.

Proof: Easy AND-OR graph reachability. $\square$

Theorem ([E.-Yannakakis'05])

For PPSs, $x = P(x)$, deciding whether $q^*_i = 1$ is in P-time.

Proof: combines eigenvalue methods and graph-theoretic methods. After "decomposition", key problem can be reduced to deciding whether certain moment matrices (Jacobian of $P(x)$ evaluated at the all 1 vector) have spectral radius > 1. ([Kolmogorov-Sevastyanov,'47,Harris'63]).

This is closely related to old work on checking consistency of SCFGs by [Booth-Thompson,1973].
However, warning: [Booth-Thompson,'73] make some mis-statements.

Algorithm for deciding if a SCFG is consistent ([EY'05])

An SCFG, G, is called consistent if the termination probabilitity starting from the start nonterminal, S, is 1.

Input: An SCFG, G, with start non-terminal S.
Output: YES if G is consistent, NO if it is not.

1. Remove all nonterminals unreachable from S.
2. If there are any useless nonterminals left (i.e., nonterminals that do not derive any terminal string), return NO.
3. Otherwise, for the remaining SCFG, let $x = P(x)$ be the associated PPS, and let $\lambda = \varrho(P'(\mathbf{1}))$ be the spectral radius of the moment matrix $P'(\mathbf{1})$ (Jacobian of $P(x)$, evaluated at the all 1-vector). If $\lambda > 1$ then return NO; otherwise (i.e., if $\lambda \leq 1$) return YES.

For a non-negative matrix M, checking whether $\varrho(M) \leq 1$ can be done easily using linear programming ([E.-Yannakakis'05]), and it can even be done by solving a linear system of equations [Esparza-Gaiser-Kiefer,2010].

Algorithm for approximating the LFP q^* for PPSs

1. Find and remove all variables x_i such that $q_i^* = 0$ or $q_i^* = 1$.
2. On the resulting system of equations, run Newton's method starting from $\mathbf{0}$.

Algorithm for approximating the LFP q^* for PPSs

1. Find and remove all variables x_i such that $q_i^* = 0$ or $q_i^* = 1$.
2. On the resulting system of equations, run Newton's method starting from $\mathbf{0}$.

Theorem ([ESY'12])

Given a PPS $\mathbf{x} = P(\mathbf{x})$ *with LFP* $\mathbf{0} < \mathbf{q}^* < \mathbf{1}$, *if we apply Newton starting at* $\mathbf{x}^{(0)} = \mathbf{0}$, *then*

$$\|\mathbf{q}^* - \mathbf{x}^{(4|P|+j)}\|_\infty \leq 2^{-j}$$

Algorithm for approximating the LFP q^* for PPSs

1. Find and remove all variables x_i such that $q_i^* = 0$ or $q_i^* = 1$.
2. On the resulting system of equations, run Newton's method starting from $\mathbf{0}$.

Theorem ([ESY'12])

Given a PPS $\mathbf{x} = P(\mathbf{x})$ *with LFP* $\mathbf{0} < \mathbf{q}^* < \mathbf{1}$, *if we apply Newton starting at* $\mathbf{x}^{(0)} = \mathbf{0}$, *then*

$$\|\mathbf{q}^* - \mathbf{x}^{(4|P|+j)}\|_\infty \leq 2^{-j}$$

Theorem ([ESY'12])

Given a PPS $\mathbf{x} = P(\mathbf{x})$ *with LFP* $\mathbf{0} < \mathbf{q}^* < \mathbf{1}$, *if we apply Newton starting at* $\mathbf{x}^{(0)} = \mathbf{0}$, *then*

$$\|\mathbf{q}^* - \mathbf{x}^{(32|P|+2j+2)}\|_\infty \leq 2^{-2^j}$$

Algorithm with rounding

1. Find and remove all variables x_i such that $q_i^* = 0$ or $q_i^* = 1$.
2. On the resulting system of equations, run Newton's method starting from $\mathbf{0}$.
3. After each iteration, round down to a multiple of 2^{-h}

Theorem ([ESY'12])

If, after each Newton iteration, we round down to a multiple of 2^{-h} where $h := 4|P| + j + 2$, then after h iterations $\|\mathbf{q}^* - \mathbf{x}^{(h)}\|_\infty \leq 2^{-j}$.

Thus, we obtain a P-time algorithm (in the standard Turing model) for approximating q^*.

High level picture of proof

- For a PPS, $x = P(x)$, with LFP $\mathbf{0} < \mathbf{q}^* < \mathbf{1}$, $P'(q^*)$ is a non-negative square matrix, and (we show)

$$\text{(spectral radius of } P'(q^*)\text{)} \equiv \varrho(P'(q^*)) < 1$$

- So, $(I - P'(q^*))$ is non-singular, and $(I - P'(q^*))^{-1} = \sum_{i=0}^{\infty}(P'(q^*))^i$.
- We can show the # of Newton iterations needed to get within $\epsilon > 0$ is

$$\approx\approx \log \|(I - P'(q^*))^{-1}\|_\infty + \log \frac{1}{\epsilon}$$

- $\|(I - P'(q^*))^{-1}\|_\infty$ is tied to the distance $|1 - \varrho(P'(q^*))|$, which in turn is related to $\min_i(1 - q_i^*)$, which we can lower bound.
- Uses lots of Perron-Frobenius theory.

Proof outline: some key lemmas

$(\mathbf{1} - \mathbf{q}^*)$ is the vector of survival probabilities.

Lemma

If $\mathbf{q}^* - \mathbf{x}^{(k)} \leq \lambda(\mathbf{1} - \mathbf{q}^*)$ *for some* $\lambda > 0$, *then* $\mathbf{q}^* - \mathbf{x}^{(k+1)} \leq \frac{\lambda}{2}(\mathbf{1} - \mathbf{q}^*)$.

Lemma

For any PPS with LFP $\mathbf{q}^*$, *such that* $\mathbf{0} < \mathbf{q}^* < \mathbf{1}$, *for any* i,

$$q_i^* \leq 1 - 2^{-4|P|}.$$

Complexity of quantitative decision problems for SCFGs

Proposition

Given a PPS, $x = P(x)$, and a probability p, deciding whether $q_i^* \leq p$ is in PSPACE.

Proof.

$$\exists \mathbf{x}(\mathbf{x} = P(\mathbf{x}) \wedge x_i \leq p)$$

is expressible in the existential theory of reals. There are PSPACE decision procedures for $\exists\mathbb{R}$ ([Canny'89,Renegar'92]). □

Recall:

Theorem ([E.-Yannakakis,'05,'07])

Given a PPS, $x = P(x)$, deciding whether $q_i^* \leq 1/2$ (or $q_i^* \leq p$ for any $p \in (0, 1)$), is both Sqrt-Sum-hard and PosSLP-hard.

The quantitative decision problem for SCFG termination probability is PosSLP-equivalent

Theorem ([E.-Stewart-Yannakakis'12])

Given a PPS, $x = P(x)$, and a probability p, deciding whether $q_i^* < p$ is P-time (many-one) reducible to PosSLP. (And thus PosSLP-equivalent.)

- Thus the quanitative decision problem for the partition function of SCFGs captures the full power of polynomial time in the unit-cost arithmetic RAM model of computation.

  By [Allender, et. al.'06], it is solvable in the Counting Hierarchy, and that is the best complexity we know in the standard (Turing) model of computation.

Applications: inside probabilities and CNF

Theorem

There is a P-time algorithm that, given as input a SCFG G, a string w and a rational $\delta > 0$ in binary, approximates the inside probability $p_{G,w}$ within δ, i.e., computes a rational v such that $|v - p_{G,w}| < \delta$.

To prove this, we first show, using approximated termination probabilities, that any SCFG, G, can be tranformed in P-time to an approximately equivalent SCFG, G', in Chomsky Normal Form.

Note: There may not exist any CNF form SCFG, with rational rule probabilities, that is exactly equivalent to G, i.e., that generates the same probability distribution on strings.

Applications: inside probabilities and CNF

Theorem

There is a P-time algorithm that, given as input a SCFG G, a string w and a rational $\delta > 0$ in binary, approximates the inside probability $p_{G,w}$ within δ, i.e., computes a rational v such that $|v - p_{G,w}| < \delta$.

To prove this, we first show, using approximated termination probabilities, that any SCFG, G, can be tranformed in P-time to an approximately equivalent SCFG, G', in Chomsky Normal Form.

Note: There may not exist any CNF form SCFG, with rational rule probabilities, that is exactly equivalent to G, i.e., that generates the same probability distribution on strings.

Theorem

There is a P-time algorithm that, given a SCFG G, a natural number N in unary, and a rational $\delta > 0$ in binary, computes a new SCFG G in CNF such that $|p_{G,w} - p_{G,w}| < \delta$ for all strings w of length at most N.

Tranforming a SCFG to CNF: "conditioned" SCFG

For a nonterminal A, let $E(A)$ be the probability that A generates the empty string. Let $NE(A) = 1 - E(A)$. We can compute $E(A)$ and $NE(A)$ in P-time: $E(A)$ is the termination probability of A after we remove all rules with terminals on the RHS.

For removing ϵ-rules: transform each rule r of the form $A \xrightarrow{p} BC$, to the following three rules:

$$r(1): \; A \xrightarrow{\frac{p * NE(B) * NE(C)}{NE(A)}} BC$$

$$r(2): \; A \xrightarrow{\frac{p * NE(B) * E(C)}{NE(A)}} B$$

$$r(3): \; A \xrightarrow{\frac{p * E(B) * NE(C)}{NE(A)}} C$$

NOTE: we can't compute the new rule probabilities exactly: they can be irrational. But we can approximate them "sufficiently well".

Rest of the construction of approximate CNF SCFG

- After removing ϵ-rules, one has to remove all remaining linear rules, $A \xrightarrow{p} B$.
- This can be done by solving certain linear systems of equations.
- However, the rule probabilities p are approximated, and thus so are the coefficients of the linear system of equations.
- We have to prove that these linear systems are well-conditioned enough so that solving the approximate linear system yields a good enough approximation of the (unique) solution to the actual system of equations (which has irrational coefficients).
  We prove this.

Using approximate CNF SCFG to approximate inside probabilities

- Once we have the approximate CNF SCFG, G', we apply the standard CKY dynamic programming algorithm to compute the inside probability of w on G'.
- To prove this provides a P-time approximation of the inside probability $p_{G,w}$ for the original SCFG, G, we have to show that the approximation errors do not blow up.

SCFGs and regular languages

Given an SCFG, G, and a deterministic finite automaton, D, our aim is to compute (approximate), the probability that G generates a string accepted by D, i.e., in $L(D)$.

This problem has many applications in NLP. Special cases (which have been studied extensively in NLP) include:

- prefix probability
- infix probability

Theorem

- *Given an SCFG G and a DFA D, we can compute an approximation to the probability $Pr_G(L(D))$ that G generates a string accepted by D, to within additive error 2^{-j} in time polynomial in j, $|G|$ and $|D|$, as long as G is non-critical.*
  (In fact, it suffices if G has bounded critical depth.)
- *Every SCFG, G, generated by the EM (inside-outside) algorithm (i.e., learned by EM from a corpus of strings) is non-critical. Thus, on such SCFGs, we can always approximate $Pr_G(L(D))$ in P-time.*

Towards a proof: Product (Intersection) of SCFG and DFA

- The classic "product/intersection" construction of CFGs and regular languages [Bar-Hillel, Perles, Shamir, 1964], generalized to SCFGs (see, e.g., [Nederhof-Satta'03]):
  given a SCFG, G, and a DFA, D, construct a product WCFG, $G \otimes D$, whose nonterminals are of the form (sAt) where A is a nonterminal of G and s, t are states of D.
- The rules of $G \otimes D$ inherit their probabilities from the rules of G.
- If s_0 is the start state of D and f is the (unique) final state, if S is the start nonterminal of G, then $Pr_G(L(D))$ is the termination probability starting at nonterminal $(s_0 S f)$ of $G \otimes D$.
- Unfortunately, the rule weights for a nonterminal (sAt) in the resulting WCFG, $G \otimes D$ no longer add up to 1 (so it is a WCFG, not a SCFG).

Question: Can we nevertheless use Newton's method on the MPS for $G \otimes D$, to get a P-time algorithm for approximating $Pr_G(L(D))$?

Towards a proof: Product (Intersection) of SCFG and DFA

- The classic "product/intersection" construction of CFGs and regular languages [Bar-Hillel, Perles, Shamir, 1964], generalized to SCFGs (see, e.g., [Nederhof-Satta'03]):
  given a SCFG, G, and a DFA, D, construct a product WCFG, $G \otimes D$, whose nonterminals are of the form (sAt) where A is a nonterminal of G and s, t are states of D.
- The rules of $G \otimes D$ inherit their probabilities from the rules of G.
- If s_0 is the start state of D and f is the (unique) final state, if S is the start nonterminal of G, then $Pr_G(L(D))$ is the termination probability starting at nonterminal (s_0Sf) of $G \otimes D$.
- Unfortunately, the rule weights for a nonterminal (sAt) in the resulting WCFG, $G \otimes D$ no longer add up to 1 (so it is a WCFG, not a SCFG).

Question: Can we nevertheless use Newton's method on the MPS for $G \otimes D$, to get a P-time algorithm for approximating $Pr_G(L(D))$? Yes.

The MPS for the product $G \otimes D$ is special.

- Consider the PPS, $x = P_G(x)$ associated with SCFG G,
- Consider also the MPS, $y = P_{G\otimes D}(y)$ associated with the WCFG $G \otimes D$.

Lemma

If we perform Newton's method on both these systems starting at $x^{(0)} = 0$ *and* $y^{(0)} = 0$, *then*

$$\forall A \forall s \qquad x_A^{(k)} = \sum_t y_{(sAt)}^{(k)}$$

$$\forall A \forall s \qquad (q_G^* - x^{(k)})_A = \sum_t (q_{G\otimes D}^* - y^{(k)})_{(sAt)}$$

$$\|q_G^* - x^{(k)}\|_\infty \geq \|q_{G\otimes D}^* - y^{(k)}\|_\infty$$

In other words, Newton converges "at the same rate" on $x = P_G(x)$ and $y = P_{G\otimes D}(y)$ to their respective LFPs, q_G^* and $q_{G\otimes D}^*$.

Critical SCFGs, and critical depth of SCFGs

A SCFG G is called critical if the associated PPS, $x = P_G(x)$ has $\varrho(P'_G(q^*_G)) = 1$.

Example

$$S \xrightarrow{\frac{1}{2}} SS$$

$$S \xrightarrow{\frac{1}{2}} a$$

Fact: We can detect whether or not $x = P_G(x)$ is critical in P-time.

Critical Depth of an SCFG

The critical depth of an SCFG G is the maximum number of critical strongly-connected components of $x = P_G(x)$ in any path through the dependency graph of variables of $x = P_G(x)$.

We can compute the critical depth of a given SCFG in P-time.

Note that a non-critical SCFG has critical depth 0.

Newton on the product MPS, $y = P_{G\otimes D}(y)$

Algorithm:

- Find and remove all variables y_z such that $q^*_z = 0$.
- On the resulting system, apply Newton's method starting from $\mathbf{0}$.

Theorem

Given a non-critical SCFG, G, and a DFA, D, with product MPS $y = P_{G\otimes D}(y)$, with LFP $0 < q^* \leq 1$, if we apply Newton staring at $y^{(0)} = 0$, then after $k \geq 14|G| + j + \log d + 3$ iterations, $\|q^*_{G\otimes D} - y^{(k)}\|_\infty \leq 2^j$.

And, we can do this with suitable rounding, to approximate $q^*_{G\otimes D}$ in P-time. In fact, more generally:

Theorem

Given an SCFG, G, and DFA, D, we can (by applying Newton) compute a rational vector $\mathbf{v}$ such that $\|q^*_{G\otimes D} - v\|_\infty \leq 2^{-j}$, in time polynomial in $|G|$, $|D|$, j, and 2^{c_G}, where c_G is the critical-depth of G.

EM always produces non-critical SCFGs

It is a well-known fact that the EM algorithm always produces consistent SCFGs, i.e., with termination probability $= 1$. (See, e.g., [Chi-Geman'98], [Sanchez-Benedi,'97], [Nederhof-Satta,'06].)
In fact:

Theorem

Any SCFG learned by standard supervised or unsupervised (i.e., EM, inside-outside) maximum likelihood estimation methods from a corpus of parse trees or strings, respectively, is non-critical.

Thus, on SCFGs learned via EM, we can compute $Pr_G(L(D))$ in P-time.

Maximum probability parsing

Consider the following SCFG:

$$A_i \xrightarrow{1} A_{i-1}A_{i-1} \quad ; \quad i = 1, \ldots, n$$

$$A_0 \xrightarrow{\frac{1}{2}} b \quad ; \quad A_0 \xrightarrow{\frac{1}{2}} \epsilon$$

Question: What is the maximum probability of a parse tree for string "b"?

Maximum probability parsing

Consider the following SCFG:

$$A_i \xrightarrow{1} A_{i-1}A_{i-1} \quad ; \quad i = 1, \ldots, n$$
$$A_0 \xrightarrow{\frac{1}{2}} b \quad ; \quad A_0 \xrightarrow{\frac{1}{2}} \epsilon$$

Question: What is the maximum probability of a parse tree for string "b"?

Answer(easy): $p^{\max}_{G,b} = 1/2^{2^n}$, and any parse tree for "b" has 2^n nodes.

Maximum probability parsing

Consider the following SCFG:

$$A_i \xrightarrow{1} A_{i-1}A_{i-1} \quad ; \quad i = 1, \ldots, n$$
$$A_0 \xrightarrow{\frac{1}{2}} b \quad ; \quad A_0 \xrightarrow{\frac{1}{2}} \epsilon$$

Question: What is the maximum probability of a parse tree for string "b"?

Answer(easy): $p^{\max}_{G,b} = 1/2^{2^n}$, and any parse tree for "b" has 2^n nodes.

Question

Can we nevertheless compute the (exact) maximum parse probability of string w, and a maximum probability parse tree for w (if it exists), in polynomial time, given any SCFG, G, and string w?

Maximum probability parsing

Consider the following SCFG:

$$A_i \xrightarrow{1} A_{i-1}A_{i-1} \quad ; \quad i = 1, \ldots, n$$

$$A_0 \xrightarrow{\frac{1}{2}} b \quad ; \quad A_0 \xrightarrow{\frac{1}{2}} \epsilon$$

Question: What is the maximum probability of a parse tree for string "b"?

Answer(easy): $p^{\max}_{G,b} = 1/2^{2^n}$, and any parse tree for "b" has 2^n nodes.

Question

Can we nevertheless compute the (exact) maximum parse probability of string w, and a maximum probability parse tree for w (if it exists), in polynomial time, given any SCFG, G, and string w?

Answer: Yes, if we assume deep conjectures in number theory, and unconditionally if a bounded number of distinct probabilities label the rules of G.

Succinct representation of small/large numbers

Numbers can be respresented succinctly in Product of Exponentials (PoE) notation, by giving a list of rational numbers (in binary):

$$\langle a_1, \ldots, a_n \rangle$$

and another list of integers (in binary):

$$\langle b_1, \ldots, b_n \rangle$$

such that together the two lists denote the number:

$$a_1^{b_1} a_2^{b_2} \ldots a_n^{b_n}$$

Note: such numbers can be very small or very large, e.g., in $O(n)$ bits we can denote the number 2^{2^n}.

Question

Can we nevertheless compare two numbers given in PoE, and decide whether one is $\geq$ another in P-time?

Succinct representation of small/large numbers

Numbers can be respresented succinctly in Product of Exponentials (PoE) notation, by giving a list of rational numbers (in binary):

$$\langle a_1, \ldots, a_n \rangle$$

and another list of integers (in binary):

$$\langle b_1, \ldots, b_n \rangle$$

such that together the two lists denote the number:

$$a_1^{b_1} a_2^{b_2} \ldots a_n^{b_n}$$

Note: such numbers can be very small or very large, e.g., in $O(n)$ bits we can denote the number 2^{2^n}.

Question

Can we nevertheless compare two numbers given in PoE, and decide whether one is $\geq$ another in P-time? Yes!...

maximum probability parsing and the ABC conjecture

Theorem

Given any SCFG, G, and any terminal string $w \in \Sigma^*$:

A. *If either the Lang-Waldschmidt Conjecture holds or Baker's refinement of the ABC conjecture holds,*
B. *or else, if the number of distinct probabilities labeling the rules of G is bounded by a fixed constant, c,*

then the following all hold:

1. *There is a P-time algorithm (in the standard Turing model) for computing the exact probability $p^{\max}_{G,w}$ in* succinct product of exponentials notation *(PoE), and for computing (if $p^{\max}_{G,w} > 0$) a maximum probability parse tree $t^{\max}_w$ for w where $t^{\max}_w$ is represented succinctly as a DAG (straight-line program).*
2. *Given another string $w' \in \Sigma^*$, there is a P-time algorithm (in the Turing model), to decide whether $p^{\max}_{G,w} \geq p^{\max}_{G,w'}$.*

Key to proof

- We can use variations of methods (based on Knuth's extension of Dijkstra's shortest path algorithm to WCFGs), in order to compute $p^{\max}_{G,w}$ in P-time in the unit-cost arithmetic RAM model of computation, where the only arithemtic operations used are $\{*, /\}$.
- We then see that P-time in this model of computation can be simulated in P-time in the standard Turing model, precisely if we can compare numbers given in PoE in P-time.
- It turns out that deep number theoretic conjectures about linear forms in logarithms, like Lang-Waldschmidt and Baker's refinement of the ABC conjecture, imply that we can compare PoE numbers in P-time.
- Furthermore, if the number of bases of the two PoE numbers is bounded by a constant, a deep Theorem [Baker-Wüstholz'93], implies we can compare such PoE numbers in P-time.

Conclusion

- The partition function (termination probabilities) for arbitrary SCFGs can be computed in P-time, using Newton's method (and some spectral preprocessing).

Conclusion

- The partition function (termination probabilities) for arbitrary SCFGs can be computed in P-time, using Newton's method (and some spectral preprocessing).
- Building on this, many of the familiar computational problems on SCFGs that are needed regularly in NLP have more sophisticated but efficient (P-time) algorithms for arbitrary SCFGs, not only for SCFGs in a normal form like CNF, and not only for SCFGs that lack ϵ-rules.

Conclusion

- The partition function (termination probabilities) for arbitrary SCFGs can be computed in P-time, using Newton's method (and some spectral preprocessing).
- Building on this, many of the familiar computational problems on SCFGs that are needed regularly in NLP have more sophisticated but efficient (P-time) algorithms for arbitrary SCFGs, not only for SCFGs in a normal form like CNF, and not only for SCFGs that lack ϵ-rules.
- Furthermore, computing the probability that an SCFG generates a string in a regular language can be done efficiently for arbitrary regular languages, not just for special cases like prefix probabilities. (As long as the SCFG is learned by maximum likelihood estimation, e.g., by EM or by supervised learning, or as long as it is not (deeply) critical.)

- K. Etessami and M. Yannakakis.
  Recursive Markov Chains, Stochastic Grammars, and Monotone Systems of Nonlinear Equations.
  Journal of the ACM, 56(1), 2009.
- K. Etessami, A. Stewart, and M. Yannakakis.
  Polynomial time algorithms for multi-type branching processes and stochastic context-free grammars. STOC 2012: 579-588.
  Proceedings of *STOC*, pp. 579-588, 2012. Full version: arXiv:1201.2374
- A. Stewart, K. Etessami, and M. Yannakakis.
  Upper Bounds for Newton's Method on Monotone Polynomial Systems, and P-Time Model Checking of Probabilistic One-Counter Automata.
  Proceedings of *CAV*, pp. 495-510, 2013. Full version: arXiv:1302.3741
- K. Etessami, A. Stewart, and M. Yannakakis.
  Stochastic Context-Free Grammars, Regular Languages, and Newton's Method.
  Proceedings of *ICALP*, pp. 199-211, 2013. Full version: arXiv:1302.6411
- K. Etessami, A. Stewart, and M. Yannakakis.
  A note on the complexity of comparing succinctly represented integers, with an application to maximum probability parsing.
  Unpublished manuscript. Preprint available at arXiv:1304.5429, 2013.