Chapter 10

Probabilistic Grammars

10.1 Outline

- HMMs
- PCFGs
- pTSGs and pTAGS
- Highlight: Zuidema et al., 2008, CogSci; Cohn & Goldwater & co., NAACL 2009

10.2 Weighted Finite-State Transducers

A WEIGHTED FINITE-STATE TRANSDUCER (wFSA) T has the following components:

- A finite INPUT ALPHABET Σ, which includes the empty string ϵ;
- A finite OUTPUT ALPHABET Δ, which includes the empty string ϵ;
- A finite set of STATES q, with distinguished members as follows:
  - One element of q is the INITIAL STATE q_0;
  - Some subset of q are FINAL STATES F, and each final state q_i has an associated weight w_i;
- A set of TRANSITIONS E, with each transition consisting of:
  - A source state q and a destination state q';
  - A mapping of an input symbol $\sigma \in \Sigma$ to an output symbol $\delta \in \Delta$;
  - An WEIGHT w associated with the transition.

10.3 MDL

a	00001
b	00010
c	00011
d	00100
e	00101
f	00110
g	00111
h	01000
i	01001
j	01010
k	01011
l	01100
m	01101
n	01110
o	01111
p	10000
q	10001
r	10010
s	10011
t	10100
u	10101
v	10110
w	10111
x	11000
y	11001
z	11010
END	11011

1	1
2	010
3	011
4	00100
5	00101
6	00110
7	00111
8	0001000
9	0001001
10	0001010
11	0001011
12	0001100
13	0001101
14	0001110
15	0001111
16	000010000

1	**S**	→NP VP	1	Det	→the
1	NP	→Det N′	0.5	N	→dog
0.8	N′	→N	0.5	N	→cat
0.2	N′	→N′ PP	1	P	→near
1	PP	→P NP	1	V	→growled
1	VP	→V			

$N = \{\text{S}, \text{NP}, \text{Det}, \text{N}', \text{N}, \text{VP}, \text{V}, \text{PP}, \text{P}\}$
$V = \{\text{the}, \text{dog}, \text{cat}, \text{near}, \text{growled}\}$
$S = \text{S}$

Figure 10.1: A simple probabilistic context-free grammar, with the start symbol in bold.

10.4 Probabilistic context-free grammars

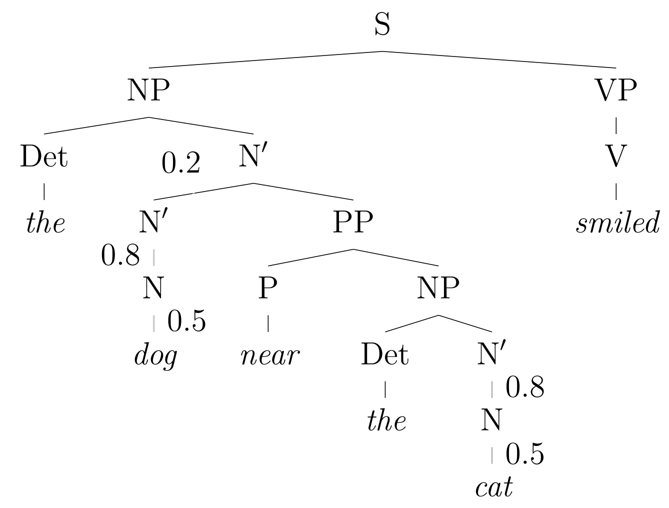
A probabilistic context-free grammar consists of a tuple (N, V, S, R, P) such that:

- N is a finite set of non-terminal symbols;
- V is a finite set of terminal symbols;
- S is the start symbol;
- R is a finite set of rules of the form $X \to \alpha$ where $X \in N$ and α is a sequence of symbols drawn from $V \cup N$;
- P is a mapping from R into probabilities, such that for each $X \in N$,

$$\sum_{[X\to\alpha]\in R} P(X \to \alpha) = 1$$

Figure 10.1 shows a simple PCFG. Figure 10.2 shows one of the trees that this PCFG generates. Note, however, that there are infinitely many trees that can be generated by even this simple grammar. Because the category PP can contain the category N′, the rule N′ $\to$ N′ PP effectively turns one N′into two. This rule can be applied arbitrarily many times. [consider a different example that involves center-embedding?]

- Use of a given PCFG
  - A PCFG gives probabilities of *trees*
  - The *yield* of a tree is (deterministically) a string
  - Thus we can use a given PCFG for all sorts of Bayesian inferences:
    - $P(w)$
    - $P(T|w)$
    - $P(T|I)$ for some input I
    - $P(w|I)$ for some input I



$$
\begin{aligned}
\mathrm{P(T)} &= 1 \times 1 \times 1 \times 0.2 \times 0.8 \times 0.5 \times 1 \times 1 \times 1 \times 1 \times 0.8 \times 0.5 \times 1 \times 1 \\
&= 0.032
\end{aligned}
$$

Figure 10.2: A simple tree

    * $P(w_i|w_{1\dots i-1})$ – Prediction (conditional word probabilities)

  - Unifying these notions as probabilistic intersection
  - This can be done efficiently when the evidence is formalizable as a weighted finite-state automaton/transducer

- Estimation of PCFGs
  - Supervised
  - Unsupervised: EM and Bayesian

10.4.1 Probabilities from wCFGs

[discuss wCFGs]

The complete structure generated by a CFG or CFG derivation is a *tree*. We have already discussed the case of PCFGs; now let us turn to the probabilistic interpretation of weighted CFGs (wCFGs). Define a tree T's WEIGHT $w(T)$ assigned by the wCFG as the product of the weights of each of the rules that make up the tree (with a rule having its weight counted as many times as it is used). Now consider the infinite set $\mathcal{T}$ of all trees that can be generated by the PCFG. The normalizing constant Z (see Section 2.8) is just the sum of the weights of all these trees:

$$Z \equiv \sum_{T \in \mathcal{T}} w(T)$$

The probability assigned to tree T is then just its weight divided by this normalizing constant:[1]

$$P(T) = \frac{w(T)}{Z}$$

Following the definitions given back in Chapter **??**, if the normalizing constant Z is equal to 1, then the rule weights alone give us a PROPER probability distribution over the trees, with a tree's probability equal to the product of its rule weights.

Notice, though, that nothing we have said thus far entails anything about bounds on the weights of individual rules in wCFGs. For example, the following wCFG (with start symbol NP) is proper by the definitions given above, even though two of the rule weights exceed 1:

1/4	NP $\rightarrow$N	2	N $\rightarrow$cats
		2	N $\rightarrow$dogs

One might then wonder whether wCFGs and PCFGs are equivalent formalisms for putting probability distributions over tree structures. Fortunately, the answer is yes: for every wCFG with finite normalization constant there is a PCFG that puts the same probability distribution on possible trees (Abney et al., 1999; Chi, 1999).[2] Furthermore, there are constructive methods for finding this PCFG (Chi, 1999). This close equivalence between wCFGs and PCFGs will come in very convenient in the later sections of this chapter.

10.5 The intersection between probabilistic CFGs and FSAs

One of the remarkable properties of finite-state machines and more expressive classes of grammars, including context-free grammars, is that of CLOSURE under intersection. That is, for any context-free grammar G and finite-state automaton A defined over the same terminal vocabulary Σ, there is some context-free grammar $G' = G \cap A$ that accepts all and only those strings accepted by both G and A. This was shown by Bar-Hillel et al. (1964) in a CONSTRUCTIVE proof—that is, Bar Hillel gave a procedure for determining the grammar $G \cap A$.

[1]Provided the normalizing constant Z is finite. If Z is not finite, then the probability of a tree is undefined.

[2]It has additionally been shown that the conditional distribution on trees given strings—$P(T|w)$—determined by a given wCFG can also be expressed by some PCFG, even when the overall partition function of the wCFG is non-finite (Smith and Johnson, 2007).

This result is extended in the probabilistic regimen: for any probabilistic context-free grammar G and probabilistic finite-state automaton A over vocabulary Σ, there is a probabilistic context-free grammar $G' = G \cap A$ that assigns probabilities to strings in exactly the appropriate way, namely that for all strings w,

$$\begin{aligned} P_{G'}(w) &\propto P_G(w)P_A(w) \\ &= \frac{1}{Z} P_G(w)P_A(w) \end{aligned}$$

where $Z = \sum_{w \in \Sigma^*} P_G(w)P_A(w)$. Furthermore, Bar-Hillel's procedure for constructing the intersection PCFG can be extended due to results of Chi (1999) and Nederhof and Satta (2003). First, one constructs a *weighted* CFG—one that is neither state-wise nor globally proper—as follows. Let R be the set of rules and S the start symbol in G, and q and E respectively be the set of states and transitions in A. First, we construct a *weighted* intersection grammar G^* whose nonterminal symbols all have the form ${}_qX_{q'}$ where X is a symbol (either terminal or non-terminal) in G and q, q' are states in A; the terminal vocabulary of G^* is simply Σ. Specifically, G^* has start symbol S and the following set of rules and probabilities:

- For every rule $X \to Y^1Y^2 \dots Y^n$ with probability p in G, and for every sequence of states $q_0, q_1, \dots, q_n$ in A, include a rule ${}_{q_0}X_{q_n} \to {}_{q_0}Y^1_{q_1}{}_{q_1}Y^2_{q_2} \dots {}_{q_{n-1}}Y^n_{q_n}$ with weight p;

- For every transition $q \overset{x}{\to} q'$ with probability p in A, include a rule ${}_qx_{q'} \to x$ with weight p;

- Recall that S is the start symbol of G; if q^* is the start state of A, then for every final state q in A with probability p, include a rule $S \to {}_{q^*}S_q$ with weight p.

Now, the resulting grammar G^* will not in general be either globally or state-wise proper: that is, the total weight assigned to all trees licensed by G^* will sum to less than 1, and for some symbols in G^* the total weight of all rules beginning with that right-hand side will not equal 1, either. However, Chi (1999) and Abney et al. (1999) showed that there is always some other PCFG G' that *is* statewise proper and is still strongly equivalent to G^*—that is, that G^* and G' assign the same weight to every tree (see also Smith and Johnson, 2007). We take this G' as the intersection grammar $G \cap A$.

10.6 Exercises

Exercise 10.1: Converting any weighted FSA into one with a unique final state of cost zero.

Exercise 10.2

Estimating a small PCFG

Consider the so-called "NP/S" garden path in online language comprehension:

(1) a. Sammy heard the noise bothered Pat.
    b. Sammy heard that the noise bothered Pat.

Write a small context-free grammar with enough rules to cover this data, including the local ambiguity in which *the noise* is misparsed as the direct object of *heard*. Estimate rule probabilities as follows: for each rule, write a tree-search expression corresponding using a `tgrep2`, `Tregex`, or `TIGERSearch` that obtains a frequency count from a parsed corpus such as the Penn Treebank. Use relative frequency estimation or some other estimation technique to obtain rule probabilities from these counts. Then, based on your resulting PCFG, compute the

- posterior probabilities of the direct-object and sentential-complement analyses of the sentences for each sentence prefix that contains at least the input tokens through *the*.
- Conditional word negative log-probabilities (i.e. surprisal values) of each word in each sentence beginning with *the*.

Finally, change something in your grammar (up to you). How does the change you make affect the above probabilities? Discuss.

Appendix A

Mathematics notation and review

This appendix gives brief coverage of the mathematical notation and concepts that you'll encounter in this book. In the space of a few pages it is of course impossible to do justice to topics such as integration and matrix algebra. Readers interested in strengthening their fundamentals in these areas are encouraged to consult XXX [calculus] and Healy (2000).

A.1 Sets ($\{\}, \cup, \cap, \emptyset$)

The notation $\{a, b, c\}$ should be read as "the set containing the elements a, b, and c". With sets, it's sometimes a convention that lower-case letters are used as names for elements, and upper-case letters as names for sets, though this is a weak convention (after all, sets can contain anything—even other sets!).

$A \cup B$ is read as "the union of A and B", and its value is the set containing exactly those elements that are present in A, in B, or in both.

$A \cap B$ is read as "the intersection of A and B", and its value is the set containing only those elements present in both A and B.

$\emptyset$, or equivalently $\{\}$, denotes the empty set—the set containing nothing. Note that $\{\emptyset\}$ isn't the empty set—it's the set containing only the empty set, and since it contains something, it isn't empty!

[introduce set complementation if necessary]

A.1.1 Countability of sets

[briefly describe]

A.2 Summation ($\sum$)

Many times we'll want to express a complex sum of systematically related parts, such as $1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5}$ or $x_1 + x_2 + x_3 + x_4 + x_5$, more compactly. We use SUMMATION notation for this:

$$\sum_{i=1}^{5}\frac{1}{i} = 1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\frac{1}{5} \qquad \sum_{i=1}^{5}x_i = x_1+x_2+x_3+x_4+x_5$$

In these cases, i is sometimes called an INDEX VARIABLE, linking the RANGE of the sum (1 to 5 in both of these cases) to its contents. Sums can be nested:

$$\sum_{i=1}^{2}\sum_{j=1}^{2}x_{ij} = x_{11}+x_{12}+x_{21}+x_{22} \qquad \sum_{i=1}^{3}\sum_{j=1}^{i}x_{ij} = x_{11}+x_{21}+x_{22}+x_{31}+x_{32}+x_{33}$$

Sums can also be infinite:

$$\sum_{i=1}^{\infty}\frac{1}{i^2} = \frac{1}{1^2}+\frac{1}{2^2}+\frac{1}{3^2}+\dots$$

Frequently, the range of the sum can be understood from context, and will be left out; or we want to be vague about the precise range of the sum. For example, suppose that there are n variables, x_1 through x_n. In order to say that the sum of all n variables is equal to 1, we might simply write

$$\sum_i x_i = 1$$

A.3 Product of a sequence ($\prod$)

Just as we often want to express a complex sum of systematically related parts, we often want to express a product of systematically related parts as well. We use PRODUCT notation to do this:

$$\prod_{i=1}^{5}\frac{1}{i} = 1\times\frac{1}{2}\times\frac{1}{3}\times\frac{1}{4}\times\frac{1}{5} \qquad \prod_{i=1}^{5}x_i = x_1x_2x_3x_4x_5$$

Usage of product notation is completely analogous to summation notation as described in Section A.2.

A.4 "Cases" notation ($\{$)

Some types of equations, especially those describing probability functions, are often best expressed in the form of one or more conditional statements. As an example, consider a six-sided die that is weighted such that when it is rolled, 50% of the time the outcome is a six, with the other five outcomes all being equally likely (i.e. 10% each). If we define a discrete random variable X representing the outcome of a roll of this die, then the clearest way of specifying the probability mass function for X is by splitting up the real numbers into three groups, such that all numbers in a given group are equally probable: (a) 6 has probability 0.5; (b) 1, 2, 3, 4, and 5 each have probability 0.1; (c) all other numbers have probability zero. Groupings of this type are often expressed using "cases" notation in an equation, with each of the cases expressed on a different row:

$$P(X = x) = \begin{cases} 0.5 & x = 6 \\ 0.1 & x \in \{1, 2, 3, 4, 5\} \\ 0 & \text{otherwise} \end{cases}$$

A.5 Logarithms and exponents

The log in base b of a number x is expressed as $log_b x$; when no base is given, as in $logx$, the base should be assumed to be the mathematical constant e. The expression $exp[x]$ is equivalent to the expression e^x. Among other things, logarithms are useful in probability theory because they allow one to translate between sums and products: $\sum_i \log x_i = \log \prod_i x_i$. Derivatives of logarithmic and exponential functions are as follows:

$$\frac{d}{dx} \log_b x = \frac{1}{x \log b}$$
$$\frac{d}{dx} y^x = y^x \log y$$

A.6 Integration ($\int$)

Sums are always over countable (finite or countably infinite) sets. The analogue over a continuum is INTEGRATION. Correspondingly, you need to know a bit about integration in order to understand continuous random variables. In particular, a basic grasp of integration is essential to understanding how Bayesian statistical inference works.

One simple view of integration is as computing "area under the curve". In the case of integrating a function f over some range $[a, b]$ of a one-dimensional variable x in which $f(x) > 0$, this view is literally correct. Imagine plotting the curve $f(x)$ against x, extending straight lines from points a and b on the x-axis up to the curve, and then laying the plot down on a table. The area on the table enclosed on four sides by the curve, the x-axis, and the two additional straight lines is the integral

$$\int_a^b f(x) \, \mathrm{d}x$$