

Part II

Technical background

Fausto Carcassi

Plan

- Formal grammars (30m)
- Semantics for formal grammars (30m)
- Bayesian inference (30m)

Part I	Introduction: On the very idea of an LoT
Part II	Technical background
Part III	Bayesian program induction (LOTlib3)
Part IV	Case studies
Part V	Summary & Future prospects

Formal grammars

What's a *grammar*

Start with a *language* (Typically, a natural language)

A notion of **well-formedness** independent of meaning

- E.g., '3 is more curious than the past table'
- We call this *grammatical* well-formedness.

We can build an **abstract device** to encode grammaticality

- Two types of such devices are popular: automata and formal grammars.
- There is a correspondence between automata and grammars!
- In the rest of the course we'll just use grammars.

Formal grammars

Grammars: Infinite use of finite means

Four ingredients:

1. A finite set N of *nonterminal symbols*
2. A finite set Σ of *terminal symbols*
3. A finite set P of *production rules*
4. A symbol S in N : the *start symbol*

Formal grammars

- | | |
|--------------------------------|--|
| 1. N (nonterminal symbols) | 1. $\{S, x, y\}$ |
| 2. Σ (terminal symbols) | 2. $\{a, b\}$ |
| 3. P (production rules) | 3. 1. $S \rightarrow x$
2. $x \rightarrow xy$
3. $y \rightarrow a$
4. $x \rightarrow b$ |
| 4. S (start symbol) | |

Let's derive a sentence in this grammar!

Formal grammars

- | | |
|--------------------------------|--|
| 1. N (nonterminal symbols) | 1. $\{S, x, y\}$ |
| 2. Σ (terminal symbols) | 2. $\{a, b\}$ |
| 3. P (production rules) | 3. 1. $S \rightarrow x$
2. $x \rightarrow xy$
3. $by \rightarrow ba$
4. $x \rightarrow b$ |
| 4. S (start symbol) | |

Let's derive a sentence in this grammar!

Formal grammars

1. N (nonterminal symbols)
2. Σ (terminal symbols)
3. P (production rules)

4. S (start symbol)

1. $\{S, x, y\}$
2. $\{a, b\}$
3.
 1. $S \rightarrow x$
 2. $x \rightarrow xy$
 3. $by \rightarrow ba$
 4. $x \rightarrow b$
 5. $b \rightarrow a$ **WRONG!** Why?

Let's derive a sentence in this grammar!

CFG - Context Free Grammars

Context-free grammar (CFGs) are grammars with rules of the form:

$$A \rightarrow \alpha$$

Where:

- A : *single* nonterminal symbol
- α : (possibly empty) string of terminals and/or nonterminals

PCFG - Probabilistic CFG

$$G = (N, \Sigma, P, S, \Pi)$$

1. N : nonterminal symbols
2. Σ : terminal symbols
3. P : production rules
4. S : start symbol
5. Π : probabilities on production rules

New: a function $\Pi: P \rightarrow \mathbb{R}$

- *Conditional* probability of applying rule $\alpha \rightarrow \beta$
- *Conditional* on the left side being α .

Every derivation has a probability of being derived

- The product of the probabilities of the applied rules.
- Higher probability to smaller trees

PCFG - Probabilistic CFG

1. N (nonterminal symbols)
2. Σ (terminal symbols)
3. P (production rules)
4. S (start symbol)
5. Π (probabilities on production rules)

- | | | |
|----|-----------------------------|-------|
| 1. | $\{S\}$ | |
| 2. | $\{a, b\}$ | Π |
| 3. | 1. $S \rightarrow aSa$ | 0.3 |
| | 2. $S \rightarrow bSb$ | 0.3 |
| | 3. $S \rightarrow \epsilon$ | 0.2 |
| | 4. $S \rightarrow a$ | 0.1 |
| | 5. $S \rightarrow b$ | 0.1 |

Interpreting a grammar

Writing down functions

- 1, 2, 3, ..., +, %, 'every', 'some', ... **VS** x, y, z, ...
 - Unsaturated vs saturated 'x + 1' vs '1 + 1'
- Unsaturated to function
 - $f(x) = x+1$
 - $f(x)$ notation is *inconvenient*: forces us to name
- Solution: *lambda* expressions
 - Start with expression $x + 2$
 - Make λ expression w/ variable $\lambda x. x + 2$
 - *Function* from bound variable to the evaluated expression

Multiple λ s and languages

- We can nest lambda expressions!
 - For instance: $\lambda y. \lambda x. x + y$
- Any expression can go inside
 - E.g., English: $\lambda x. x$ is a bird
- We'll mostly use variations of predicate logic.

The notation of λ -calculus

Notation for applying an argument to a function

- $\lambda x. P(x)$
- Apply argument N to function $\lambda x. P(x)$: $(\lambda x. P(x))N$
- β -reduction:
 - $(\lambda x. P(x))N$
 - Replace N in every occurrence of x and remove the lambda
 - $P(N)$
- We can also rename variables w/ alpha conversion (Let's ignore this)

Example of β reduction

- $\left(\left((\lambda x. \lambda y. x(y)) \lambda z. P(z) \right) a \right) b$

Substitute $\lambda z. P(z)$ for x into $\lambda y. x(y)$:

- $\left((\lambda y. \lambda z. P(z)(y)) a \right) b$

Substitute a for y in $\lambda z. P(z)(y)$

- $(\lambda z. P(z)(a)) b$

Substitute b for z in $P(z)(a)$:

- $P(b)(a)$

Compositional interpretation

Let's build an *interpretation function* for a grammar!

- Associate each sentence with a meaning!

E.g., propositional logic

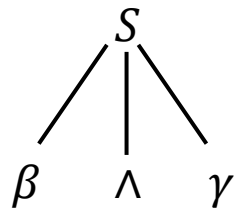
- $S \rightarrow p \mid q \mid (S \wedge S) \mid (S \vee S) \mid \neg S$

Interpretation of basic symbols:

- $I_{@}(p) = \text{True}$
- $I_{@}(q) = \text{False}$
- $I_{@}(\wedge) = \lambda x. \lambda y. x = 1 \text{ and } y = 0$
- $I_{@}(\vee) = \lambda x. \lambda y. x = 1 \text{ or } y = 0$
- $I_{@}(\neg) = \lambda x. x = 0$

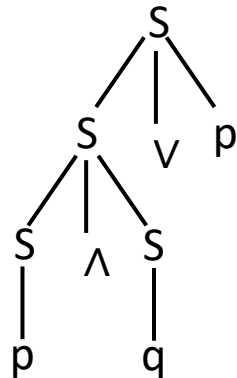
Compositional interpretation

Meaning of complex sentences:

If α has the form  then $I(\alpha) = (I(\wedge)(I(\beta)) I(\gamma))$

Can you tell what the rules are for the other entries?

Example:



Bayesian inference

Probability & conditional probability

What is probability?

- Feature of system in the world
 - Dice behaviour over many rolls
- Degree of support btw propositions
 - “Given that it’s raining, we’ll *probably* get wet”
- Strength or degree of a belief, or *credence*
 - “I think George is *probably* at the party”

We write

- $P(A)$ for “credence of A”
- $P(A \mid B)$ for “credence of A given B”

In general, $P(A \mid B) \neq P(B \mid A)$

- $P(X \text{ flies} \mid X \text{ is a Kakapo})$ is **high**, $P(X \text{ is a Kakapo} \mid X \text{ flies})$ is **low**



A motivating example

Question: “Is my new haircut better than the old one?”

- You are completely unsure: 50/50
 - Mother of a friend says “Yes” → new belief?
 - Your worst enemy says “Yes” → new belief?
- You are pretty sure it’s worse than it was: 90/10
 - Mother of a friend says “Yes” → new belief?
 - Your worst enemy says “Yes” → new belief?

What is at play?

- Prior
- Likelihood

A motivating example

$$P(\text{Improvement} \mid \text{'Yes' from mum}) = \frac{P(\text{'Yes' from mum} \mid \text{Improvement})P(\text{Improvement})}{P(D)}$$

The components of Bayes theorem

$$\overbrace{P(H \mid D)}^{\text{Posterior}} = \frac{\overbrace{P(D \mid H)}^{\text{Likelihood}} \overbrace{P(H)}^{\text{Prior}}}{\underbrace{P(D)}_{\text{Evidence}}}$$

Four ingredients in Bayes theorem:

- | | |
|----------------------|--|
| 1. Posterior | Probability of hypothesis given data |
| 2. Likelihood | Probability of the data <i>given</i> the hypothesis |
| 3. Prior | Probability of the hypothesis, NOT conditioned on data |
| 4. Evidence | Probability of the data, NOT conditioned on H |

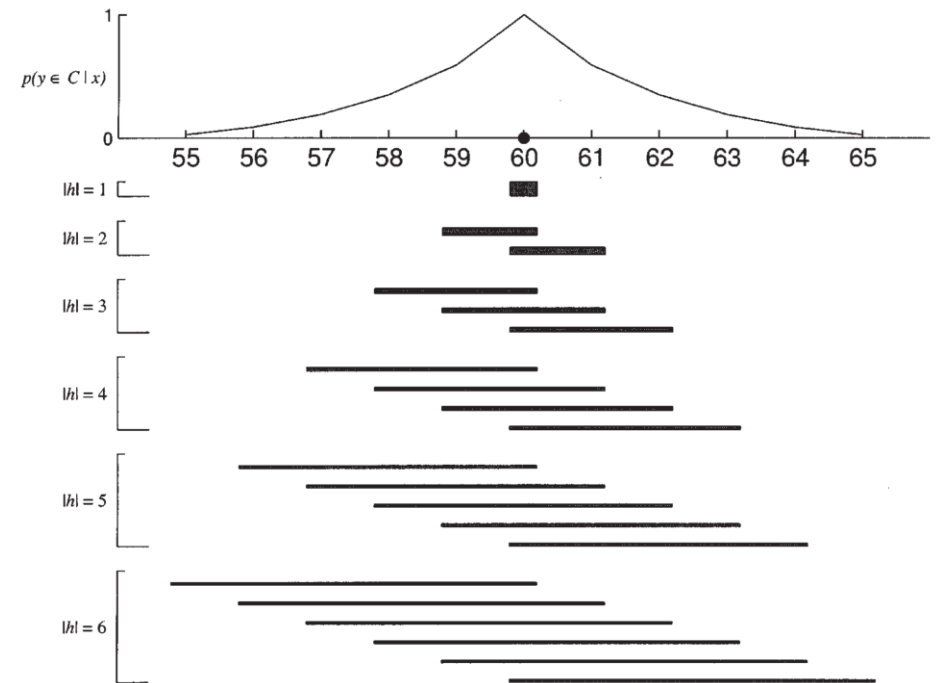
Case study: Simple category learning

Task: learn a category from examples.

- The space is simply the integers from 1 to 50
- The examples are numbers from the category
- The category is a *convex* region
- We get examples from inside the category

Bayesian category learning

- What's the space of hypotheses?
- What's the posterior, likelihood, and prior?
- What happens if we get more observations?

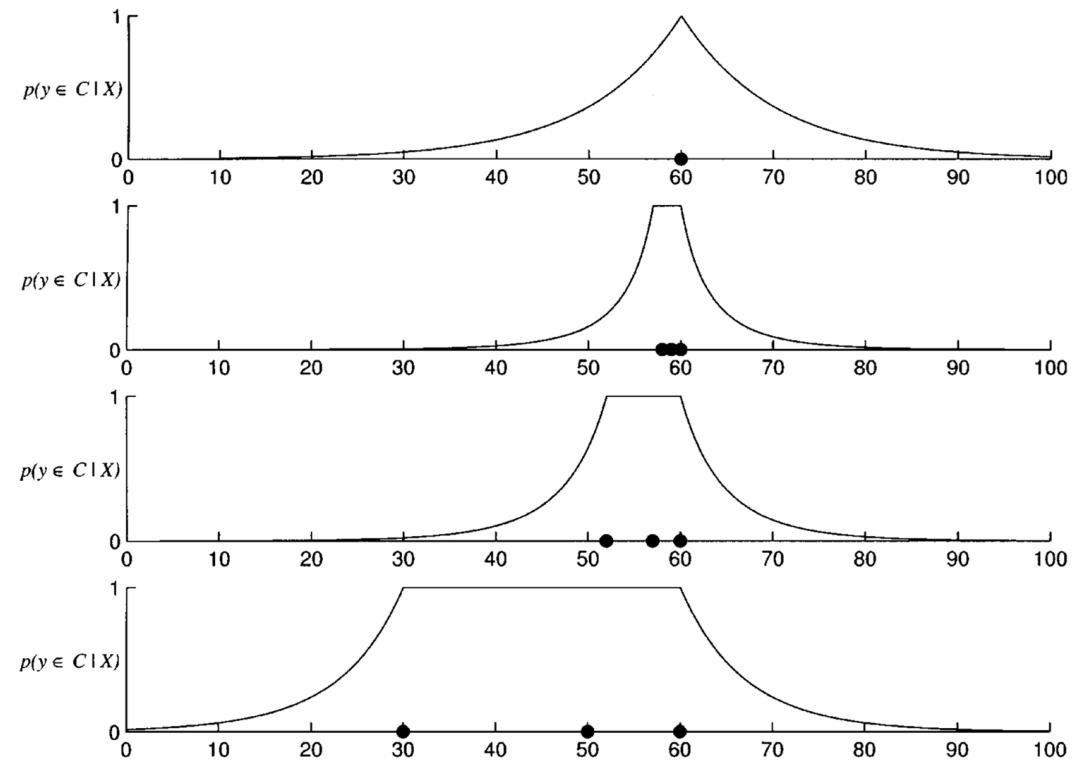


Tenenbaum & Griffiths (2001)

Case study: Simple category learning

Strong sampling \rightarrow *size effect*

- Can you see why intuitively?
- Can you see why formally?



Tenenbaum & Griffiths (2001)

The grand plan of the pLoT

Interpreted PCFG

Defines **hypotheses** H

Defines **prior** over H

<- Models the Language of Thought

<- Sentences in the LoT

<- Prior probability of LoT sentences

Observations O

Generated by unknown H

<- Data for learning

<- To be represented in the LoT

Likelihood $P(O \mid H)$

<- From a world model

Bayesian inference

Gets $P(H \mid O)$

<- Gives us a distribution over LoT sentences

Our best guess for the representation content!

Conclusions

We've learned about

- PCFGs
- their interpretation
- Bayesian category learning

We can use this as a model of the pLoT

- Can you see how?

Next session:

- Combine into computational model!

Part I	Introduction: On the very idea of an LoT
Part II	Technical background
Part III	Bayesian program induction (LOTlib3)
Part IV	Case studies
Part V	Summary & Future prospects

If there's time left...

Type theory

(This part might be confusing!)

Type theory

- For reasons that will become clear soon, we associate each of our expressions with a *type*.
- Let's define the set of types:
 - e and t are types
 - If σ and τ are types, then $\langle \sigma, \tau \rangle$ is a type
 - Nothing else is a type
- And how to interpret them:
 - e refers to the set of individuals
 - t refers to the set of truth values
 - $\langle \sigma, \tau \rangle$ refers to the set of functions from objects of type σ to objects of type τ

Type theory

Let's consider some expressions and what type they are:

- $\lambda x. P(x)$, where P is a predicate and x an individual.
 - $\langle e, t \rangle$
- $\lambda x. \lambda y. Q(x, y)$, where Q is a predicate with two arguments and x and y individuals
 - $\langle e, \langle e, t \rangle \rangle$
- $\lambda X. X(a)$, where a is an individual and X is a predicate
 - $\langle \langle e, t \rangle, t \rangle$
- $\lambda X. \lambda Y. X(a) \wedge Y(a)$, where a is an individual and X and Y predicates
 - $\langle \langle e, t \rangle, \langle \langle e, t \rangle, t \rangle \rangle$
- Basically, it can get as complicated as you want!

Type theory

In order to keep things tidy, we can put domain restrictions after a colon. Therefore, we can write the type of each argument of a lambda function as follows:

- $\lambda x: x \in e. P(x)$, where P is a predicate
- $\lambda x: x \in e. \lambda y: y \in e. Q(x, y)$, where Q is a predicate with two arguments
- $\lambda X: X \in \langle e, t \rangle. X(a)$, where a is an individual
- $\lambda X: X \in \langle e, t \rangle. \lambda Y: Y \in \langle e, t \rangle. X(a) \wedge Y(a)$

Sometimes, you'll also see the type written as a suffix:

- $\lambda x e. P(x)$

Language to grammar

- We've seen grammar \rightarrow language
- We can also go language \rightarrow grammar
- Let's try to write a grammar that produces all the palindromes in $\{a, b\}^*$

1. N (nonterminal symbols)
2. Σ (terminal symbols)
3. P (production rules)
4. S (start symbol)

1. $\{S\}$
2. $\{a, b\}$
3.
 1. $S \rightarrow aSa$
 2. $S \rightarrow bSb$
 3. $S \rightarrow \epsilon$
 4. $S \rightarrow a$
 5. $S \rightarrow b$

Language to grammar

- All strings with the form: $a^n b^n$

1. N (nonterminal symbols)
2. Σ (terminal symbols)
3. P (production rules)
4. S (start symbol)

1. $\{S\}$
2. $\{a, b\}$
3.
 1. $S \rightarrow aSb$
 2. $S \rightarrow e$