Grammatical inference: learning models

Module X3IT040, Colin de la Higuera, Nantes & Le Mans, 2020

cc BY

UNIVERSITÉ DE NANTES

Statistical and symbolic language modeling

ATAL
LS2N LABORATOIRE DES SCIENCES DU NUMÉRIQUE DE NANTES

Outline

1 Why have models?

2 Identification in the limit.

3 Probabilistic settings.

Note about these slides:
a number of these were drafted by Jose Oncina

1. Why have models?

1.1 Some convergence criteria

- What would we like to say?
- That in the near future, given some string, we can predict if this string belongs to the language or not
- To predict, given the first *n* symbols, the next symbol in the string
- It would be nice to be able to **bet** **€1000** on this

1.2 (if not) What would we like to say?

- That if the solution we have returned is not good, then that is because the learning data was bad (insufficient, biased)

- Key idea:

blame the data, not the algorithm

1.3 Suppose we cannot say anything of the sort?

- Then that means that we may be terribly wrong even in a favourable setting

- Thus there is a hidden bias

- Hidden bias: the learning algorithm is supposed to be able to learn anything inside class $\mathcal{A}$, but can really only learn things inside class $\mathcal{B}$, with $\mathcal{B} \subset \mathcal{A}$

1.4 The “trick”

- Replace the learning problem with the identification problem
- Instead of discussing “With data set S, if I learn G, is G a good answer?”
- Discuss: “Given a target G_T and data somehow generated from/for G_T, my learner returns G_H. How good is G_H?”

cc BY

1.5 The price to pay

- In practice, who tells us that there is a target grammar?
- In practice, we don’t have the target (even if there is one)
- Identification will often lead to overfitting

2. Identification in the limit

CC BY

2.1 Non probabilistic setting

- Identification in the limit
- Resource bounded identification in the limit
- Active learning (query learning)

CC BY

2.2 Identification in the limit

- E. M. Gold. Language identification in the limit. *Information and Control*, 10(5):447–474, 1967
  (http://web.mit.edu/6.863/www/spring2010/readings/gold67limit.pdf)

- E. M. Gold. Complexity of automaton identification from given data. *Information and Control*, 37:302–320, 1978
  (http://www-personal.umich.edu/~yinw/papers/Gold78.pdf)

2.3 The general idea

- Information is presented to the learner who updates its hypothesis after each piece of data
- At some point, always, the learner will have found the correct concept and not change from it

2.4 Example

2	{2}
3	{2, 3}
5	Fibonacci numbers
7	
11	Prime numbers
103	
23	
31	

2.5 A game: beating the box

1. A black box generates numbers from a sequence.

2. We have to guess the next number.

3. The black box indicates yes or no depending on if we have guessed the next element of the sequence (and gives us this next element).

2.5 Some questions

- Can we always beat the box?
  - Not if the box can change its rule on the fly after seeing your guess.
  - Not if the function is not computable.

- When do we stop?
  - When after a certain point we do not change our mind... But then we don’t know for sure we are correct!

2.6 Polynomials

- The rules are polynomials with integer coefficients ($3n^4+7n^2-15$)

- Can we beat the box?

- 2 techniques
  - by enumeration
  - by interpolation

2.6 Enumeration

- Obtain an enumeration of all polynomials

- After each example return the smallest (the first in the list) polynomial consistent with all seen examples

2.6 Enumeration: example

$$\cancel{0} \; ; \; \cancel{3} \; ; \; \cancel{n+3} \; ; \; \cancel{n-5} \; ; \; n^2-n+3 \; ; \; \ldots$$

$p(0) = 3$ $\quad$ $p(1) = 3$ $\quad$ $p(2) = 5$

Current hypothesis 3

Current hypothesis 3

Current hypothesis n^2-n+3

cdlh 2020, Statistical and symbolic language modeling X3IT040

2.6 Enumeration: enumerate the polynomials

- Enumerate for d = 0,1,2,… all polynomials of degree at most d and that use coefficients in the range [-d;d]

0

$-n-1$, $-n$, $-n+1$, -1, 0, 1, $n-1$, n, $n+1$

$-2n^2-2n-2$, $-2n^2-2n-1$, $-2n^2-2n$, $-2n^2-2n+1$, $-2n^2-2n+2$, $-2n^2-n-2$, $-2n^2-n-1$, $-2n^2-n$, $-2n^2-n+1$, $-2n^2-n+2$, $-2n^2-2$, $-2n^2-1$, $-2n^2$, $-2n^2+1$, $-2n^2+2$, $-2n^2+n-2$, $-2n^2+n-1$, $-2n^2+n$, $-2n^2+n+1$, $-2n^2+n+2$, $-2n^2+2n-2$, $-2n^2+2n-1$, $-2n^2+2n$, $-2n^2+2n+1$, $-2n^2+2n+2$, $-n^2-2n-2$

$-3n^3-3n^2-3n-3$, ...

…

cdlh 2020, Statistical and symbolic language modeling X3IT040

2.6 Enumeration: why does it work?

- Suppose the target is some polynomial of degree k and of maximal absolute coefficient j

- Let $d = \max\{j,k\}$

- Then after a finite number of steps this polynomial must appear because all polynomials smaller will have disappeared

2.6 Enumeration: properties of the algorithm

- It works
- It is very time consuming (more than d^d polynomials have to be checked)
- It can be adapted to any recursively enumerable class

2.6 Enumeration: identification

- Suppose we have a procedure to enumerate descriptions of all the rules

- The enumeration method goes over the enumeration until it finds the first description compatible with the examples

2.7 Interpolation

- At each step k build the polynomial p such that:

$$p(0) = f(0),$$
$$p(1) = f(1),$$
$$p(2) = f(2),$$
$$\ldots,$$
$$p(k) = f(k)$$

- This can identify any polynomial and requires much less computation time

cdlh 2020, Statistical and symbolic language modeling X3IT040

2.7 Interpolation: Lagrange

- Polynomial in X passing through $(a_0,b_0),\ldots,(a_n,b_n)$:

$$P(X) = \sum_{k=0} b_k L_k(X)$$

$$L_k(X) = \prod_{i \neq k}$$

CC BY

2.7 Interpolation: note

- In neither of the methods do we know if we have reached identification

- Suppose the values have been 0,0,0,0,0

- How do we know the polynomial is 0 and not something like $n(n-1)(n-2)(n-3)(n-4)$?

cdlh 2020, Statistical and symbolic language modeling X3IT040

2.8 Presentation

A presentation is :

- a function $\varphi : \mathbb{N} \rightarrow X$
- where X is some set,
- and such that φ is associated to a language L through a function *yields* : $yields(\varphi) = L$
- if $\varphi(\mathbb{N}) = \psi(\mathbb{N})$ then $yields(\varphi) = yields(\psi)$
- I.e. if two presentations differ only by order, they correspond to the same language

2.8 Presentation: some types (1)

- An *informed* presentation (or an informant) of $L \subseteq \Sigma^*$ is a function $\varphi : \mathbb{N} \rightarrow \Sigma^* \times \{-,+\}$ such that $\varphi(\mathbb{N}) = (L,+) \cup (\text{L},-)$

- φ is an infinite succession of all the elements of Σ^* labelled to indicate if they belong or not to L

2.8 Presentation: some types (2)

- A *text* presentation of a language $L \subseteq \Sigma^*$ is a function $\varphi : \mathbb{N} \to \Sigma^*$ such that $\varphi(\mathbb{N}) = L$

- φ is an infinite succession of all the elements of L

(note: there can be repetitions; small technical difficulty with $\emptyset$)

2.8 Presentation: example

for $\{a^n b^n : n \in \mathbb{N}\}$

- Legal presentation from text: λ, a^2b^2, a^7b^7,…

- Illegal presentation from text: *ab*, *ab*, *ab*,…

- Legal presentation from informant : (λ,+), (*abab*,-), (a^2b^2,+), (a^7b^7,…,+), (*aab*,-),…

CC BY

2.8 Presentation: naming function ($\mathbb{L}$)

- Given a presentation φ, φ_n is the set of the first n elements in φ

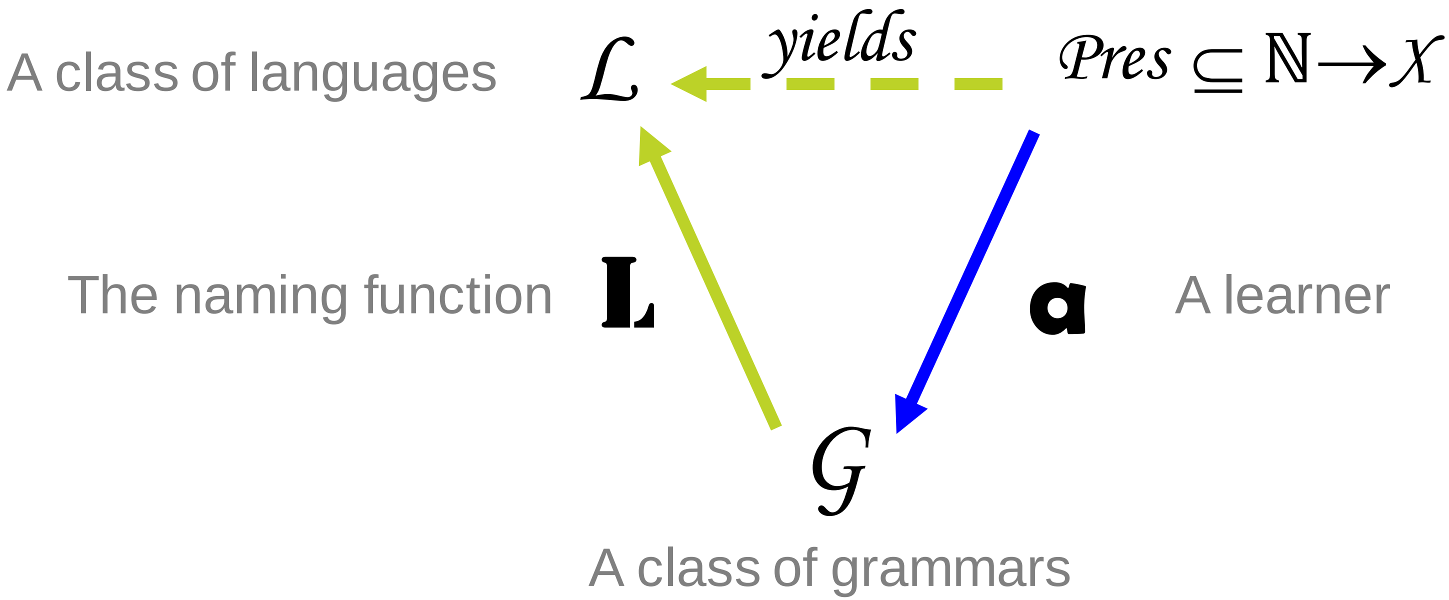
- A learning algorithm $\boldsymbol{a}$ is a function that takes as input a set φ_n and returns a representation of a language

- Given a grammar G, $\mathbb{L}(G)$ is the language generated/recognised/represented by G

2.8 Presentation: convergence to a hypothesis

- Let L be a language from a class $\mathcal{L}$
- let φ be a presentation of L
- let φ_n be the first n elements in φ
- **a** *converges* to G with φ if
  - $\forall n \in \mathbb{N}$: $\mathbf{a}(\varphi_n)$ halts and gives an answer
  - $\exists n_0 \in \mathbb{N}$: $n \geq n_0 \Rightarrow \mathbf{a}(\varphi_n) = G$

2.8 Presentation: identification in the limit



$$\boxed{\begin{array}{c} \exists m \ \forall n \geq m \\ \mathbf{L}(\mathbf{a}(\varphi_n)) \ = \ yields(\varphi) \\ \mathbf{a}(\varphi_n) \ = \ \mathbf{a}(\varphi_m) \end{array}}$$

$$\boxed{\varphi(\mathbb{N}) = \psi(\mathbb{N}) \Rightarrow yields(\varphi) = yields(\psi)}$$

CC BY

2.9 Consistency and conservatism

- We say that the learning function $\mathbf{a}$ is *consistent* if $\mathbf{a}(\varphi_n)$ is consistent with φ_n $\forall n$

- A consistent learner is always consistent with the past

- We say that the learning function $\mathbf{a}$ is *conservative* if whenever $\mathbf{a}(\varphi_n)$ is consistent with $\varphi(n+1)$, we have $\mathbf{a}(\varphi_n) = \mathbf{a}(\varphi_{n+1})$

- A conservative learner doesn't change his mind needlessly

CC BY

2.9 What about efficiency?

- We can try to bound:
  - global time
  - update time
  - errors before converging (IPE)
  - mind changes (MC)
  - queries
  - good examples needed

2.9 Resource bounded identification in the limit

- Definitions of IPE, CS, MC, update time, etc.

- What should we try to measure?
  - The size of G ?
  - The size of L ?
  - The size of φ ?
  - The size of φ_n ?

2.9 The size of G : $\|G\|$

- The size of a grammar is the number of bits needed to encode the grammar

- Better some value polynomial in the desired quantity

- Example:
  – DFA: # of states
  – CFG: # of rules * length of rules
  – …

cdlh 2020, Statistical and symbolic language modeling X3IT040

2.9 The size of *L*

- If no grammar system is given, meaningless

- If $\mathcal{G}$ is the class of grammars then $\|L\| = \min\{\|G\| : G \in \mathcal{G} \wedge \mathbf{L}(G) = L\}$

- Example:
  - the size of a regular language when considering DFA is the number of states of the minimal DFA that recognizes it

CC BY

2.9 Is a grammar representation reasonable?

- Difficult question: typical arguments are that NFA are better than DFA because you can encode more languages with less bits.

- Yet redundancy is necessary!

2.9 Proposal

- A grammar class is reasonable if it encodes sufficient different languages

- *I.e.* with n bits you have 2^{n+1} encodings so optimally you should have 2^{n+1} different languages

CC BY

2.9 But

- We should allow for redundancy and for some strings that do not encode grammars

- Therefore a grammar representation is reasonable if there exists a polynomial $p()$ and for any n the number of different languages encoded by grammars of size n is in $\theta(2^n)$

cc BY

2.10 Gold's key result (1967)-1

- Any recursively enumerable class of languages is identifiable in the limit from an informant

- Why?
- Because the enumeration algorithm will work:
- Let the target be represented by grammar G_k (the k^{th} grammar in the enumeration, with k as small as possible)
- For every grammar G_i, with $i<k$, there exists a string w belonging to $L(G_i)\backslash L(G_k) \cup L(G_k)\backslash L(G_i)$. When this string w appears with its label, G_i ceases to be an acceptable candidate. And since this string has to appear sooner or later, we are done.

cdlh 2020, Statistical and symbolic language modeling X3IT040

Gold's key result (1967)-2

- No class of languages containing all finite languages and at least one infinite language is identifiable in the limit from text

- Why?
- Proof is more complicated. Intuitively, after having seen a sample *S* (all containing strings in the infinite language) it is impossible to decide if we are learning *S* or the infinite language.

As a consequence

- Regular languages are identifiable in the limit from an informant
- Regular languages are not identifiable in the limit from text

- The same applies to context-free languages and everything else (of interest)

- Or does it?

Exercises

- Write an algorithm which can text-identify in the limit the following classes
- Or prove that the class is not text-identifiable in the limit
- And is your algorithm polynomial, conservative, consistent?
- $\Sigma=\{a,b\}$
- FIN(Σ) is the set of all finite languages over Σ

Example

- Let us consider FIN(Σ)
- We are presented with an infinite sequence of elements of some target language T: $w_0, w_1, \ldots. w_n, \ldots$ This presentation is complete
- At step n our algorithm returns $H_n = \{w_0, w_1, \ldots. w_n\}$

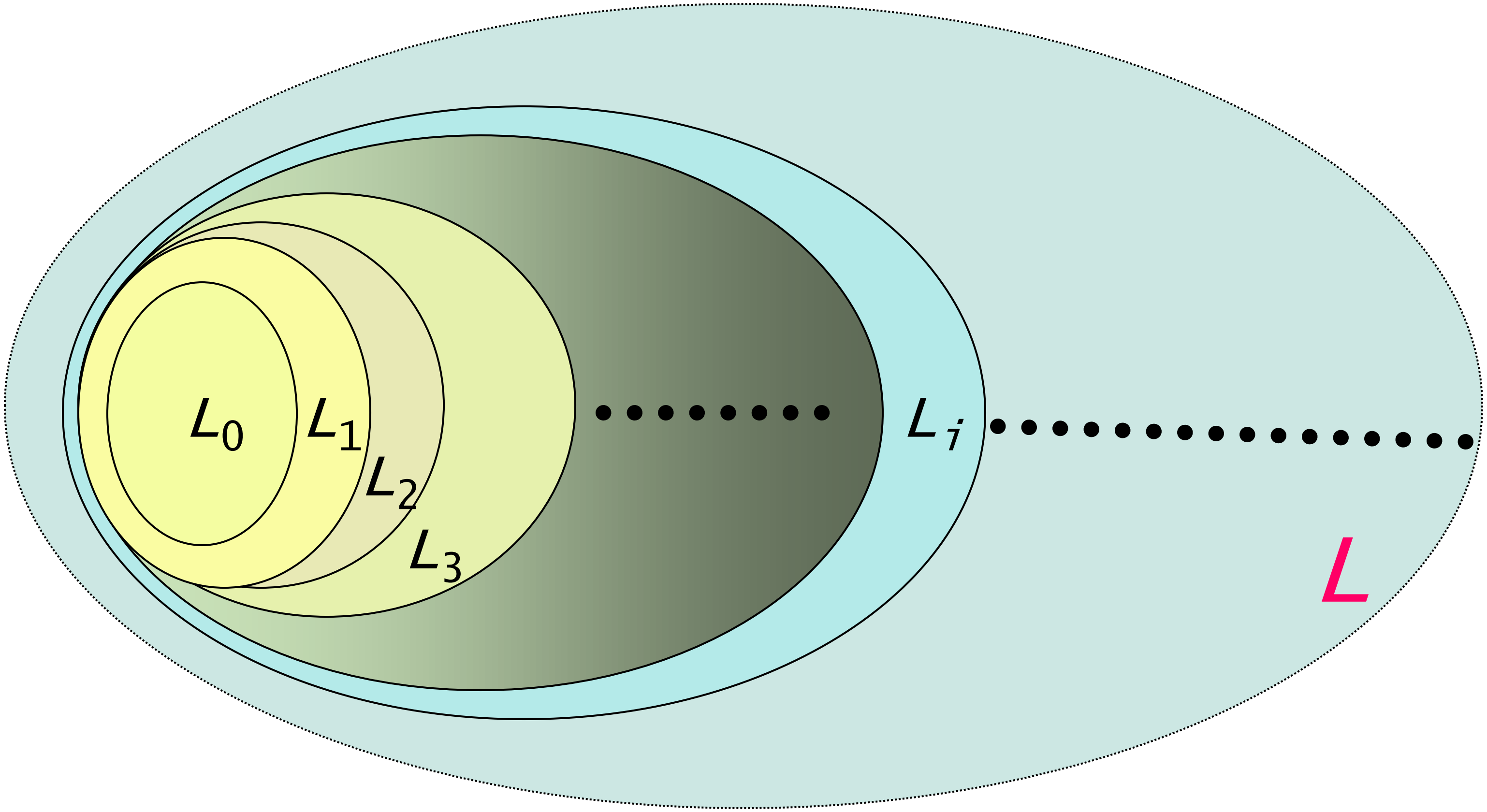
1. Our algorithm identifies H_n in the limit. Given any target and any complete presentation, the set of first ranks at which the elements of T appear in the presentation is finite. So the max of this set exists. At that rank we will have $H_n = T$.
2. The algorithm only needs polynomial update time
3. The algorithm is consistent as the update of the solution leads to a consistent solution
4. The algorithm is conservative as $H_{n+1} \neq H_n$.only when w_{n+1} *is a new sunseen string.*

Limit point (to prove that a class is not identifiable

- A class $\mathcal{L}$ of languages has a limit point *if* there exists an infinite sequence L_n ($n \in \mathbb{N}$) of languages in $\mathcal{L}$ such that $L_0 \subset L_1 \subset \dots L_k \subset \dots$, and there exists another language $L \in \mathcal{L}$ such that $L = \cup_{n \in \mathbb{N}} L_n$
- L is called a limit point of $\mathcal{L}$

CC BY

L is a limit point



cc BY

Theorem

If $\mathcal{L}$ admits a limit point, then $\mathcal{L}$ is not learnable from text

Proof: Let s^i be a presentation in length-lex order for L_i, and s be a presentation in length-lex order for L. Then $\forall n \in \mathbb{N}\ \exists i : \forall k \leq n$ $s^i{}_k = s_k$

Note: having a limit point is a sufficient condition for non learnability; not a necessary condition

More classes (exercise)

- Co-FIN(Σ) is the set of all languages over Σ whose complement is finite
- SEG(Σ) is the set of all $S_{j,k}$ languages given by j and k in $\mathbb{N}$ with $j \leq k$. Each $S_{j,k}=\{w \in \Sigma^*: j \leq |w| \leq k\}$
- TSS(Σ) is the set of all k-TSS languages (for any k)

- Again, $\Sigma=\{a,b\}$

Simple pattern languages

- A pattern is a string over $\Sigma=\{a,b\}$ and an unbounded set of variables $\{x_0,x_1,x_2,\ldots\}$
- For example $\pi=aax_1x_0bax_1$ is a pattern
- An instanciation of a pattern is a string obtained by substituting each variable by a non empty string in $\Sigma=\{a,b\}$
- For example, given the pattern π above, *abbababb* is a possible instanciation
- $L(\pi)$ is the set of all instanciations of π.

- PATTERNS(Σ) is the set of all pattern languages over alphabet Σ.

- Can you identify in the limit PATTERNS(Σ)?

CC BY

3. Probabilistic settings

CC BY

3.1 Probabilistic settings

- PAC learning
- Identification with probability 1
- PAC learning distributions

3.2 Learning a language from sampling

- We have a distribution over Σ^*

- We sample twice:
  - once to learn
  - once to see how well we have learned

- The PAC setting

cc BY

3.3 PAC-learning

- $\mathcal{L}$ a class of languages
- $\mathcal{G}$ a class of grammars
- $\varepsilon > 0$ and $\delta > 0$
- m a maximal length over the strings
- n a maximal size of machines

Leslie Valiant

Turing Award 2011

http://people.seas.harvard.edu/~valiant/

L. Pitt. Inductive inference, *Dfa's, and computational complexity.*
In Analogical and Inductive Inference, number 397 in LNAI, pages 18–44. Springer-Verlag, 1989.

L. G. Valiant. *A theory of the learnable.*
Communications of the Association for Computing Machinery, 27(11):1134–1142, 1984.

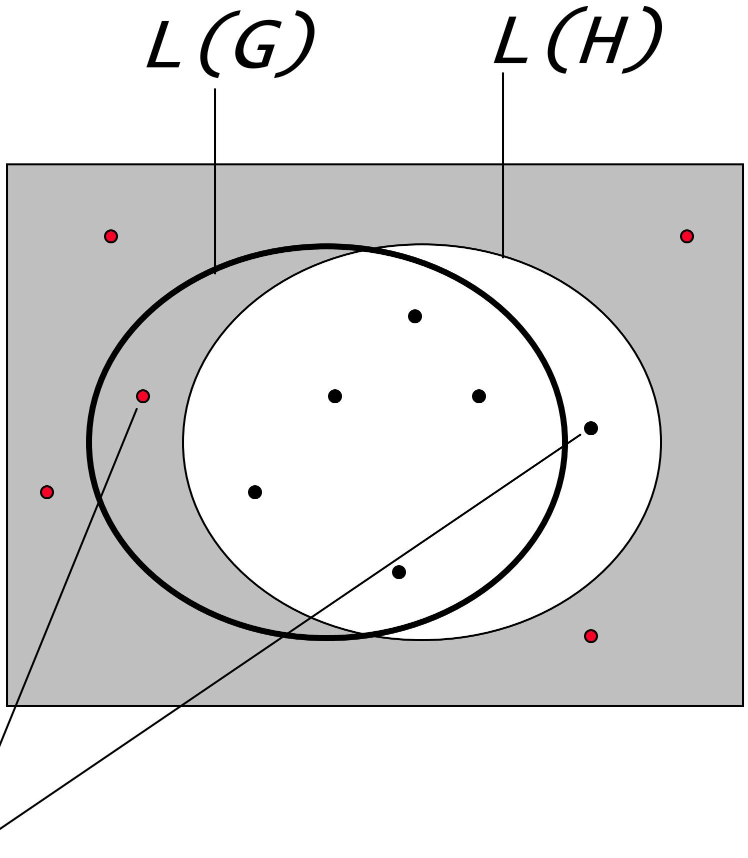
3.3 PAC-learning

H is ε - AC (approximately correct)*

if

$$\Pr_D[H(x) \neq G(x)] < \varepsilon$$

3.3 PAC-learning



Errors: we want this < ε

3.3 (French radio)

Unless there is a surprise there should be no surprise

(after the elections, on 3rd of June 2008)

3.3 Results

- Using cryptographic assumptions, we cannot PAC-learn DFA

- Cannot PAC-learn NFA, CFGs with membership queries either

CC BY

3.3 Alternatively

- Instead of learning classifiers in a probabilistic world, learn directly the distributions!

- Learn probabilistic finite automata (deterministic or not).

3.3 No error

- This calls for identification in the limit with probability 1

- Means that the probability of not converging is 0

- The learner is given an infinite sequence of examples drawn following the target distribution

- At some point, the learner stabilises itself on a grammar which is correct
  - Correct structure
  - Correct weights (exactly!)

3.3 A simple (?) example

- A coin is tossed following a distribution
  - Heads: p
  - Tails: 1-p

- How do you identify p?

- A key reference for this:

Dana Angluin

http://hcil2.cs.umd.edu/photos/ConferencePhotos/C466%20-%20DANA%20ANGLUIN/C466002.jpg

3.3 Results

- If probabilities are computable, we can learn with probability 1 finite state automata

- But not with bounded (polynomial) resources

- Or it becomes very tricky (with added information)

3.3 With error

- PAC definition
- But error should be measured by a distance between the target distribution and the hypothesis
- L_1, L_2, L_∞ ?

CC BY

3.3 Results

- Too easy with L_∞
- Too hard with L_1
- Nice algorithms for biased classes of distributions

3.3 Conclusion

- A number of paradigms to study identification of learning algorithms
- Some to learn classifiers
- Some to learn distributions

Except where otherwise noted, this work is licensed under
a Creative Commons Attribution 4.0 International License.

http://creativecommons.org/licenses/by/4.0/