Learnability and Language Acquisition

Computational Learning of Syntax

Alexander Clark

Department of Philosophy
King's College, London

LSA Summer Institute 2015, Chicago

What is the course about?

- How to learn grammars from strings.

What is the course about?

- How to learn grammars from strings.
- The relation between these methods and
    - Language acquisition
    - Linguistic theory in general

Classic model

$$\begin{tikzcd} \text{PLD} \arrow[r] & \boxed{\text{LAD}} \arrow[r] & \hat{G} \end{tikzcd}$$

Classic model

$$\begin{tikzcd} \text{PLD} \arrow[r] & \boxed{\text{LAD}} \arrow[r] & \hat{G} \end{tikzcd}$$

Three questions

1. What are the inputs? (the PLD)
2. What are the outputs? (the $\hat{G}$)
3. What conditions does the LAD have to satisfy?

Course Outline

Mon	A first learning algorithm. Basic principles of learnability.
Thur	Distributional analysis. Learning Context-free grammars;
Mon	Mildly context sensitive grammars and beyond. Copying.
Thu	Strong learning; language acquisition and linguistic theory.

Topics for today

- The role of learnability in linguistics.
- A first learning algorithm.
- Basic principles of learning:
    - Inputs and Outputs
    - Convergence
    - Computational complexity

Central problem of linguistics

Chomsky's questions

1. What constitutes knowledge of a language?
2. How is this knowledge acquired by its speakers?

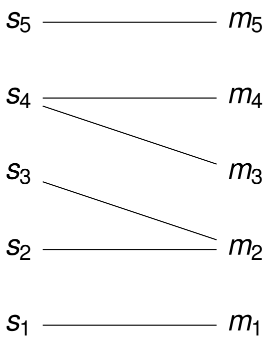
Jackendoff 2011

1. An account of speakers’ ability to create and understand an unlimited number of sentences of their language(s). (knowledge of a language or competence)
2. An account of how speakers acquire knowledge of a language. (acquisition)
3. An account of how the human species acquired the ability to acquire the knowledge of a language. (evolution)

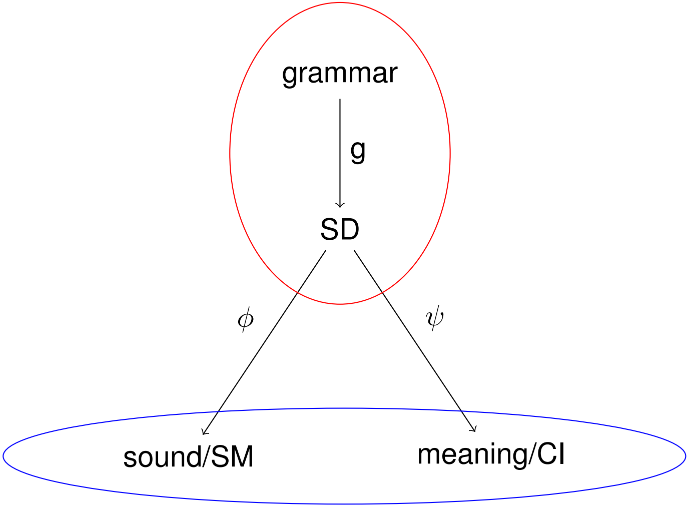
Knowledge of language

- Grammars (I-languages): finite generative devices
- Languages (E-languages): infinite sets of
  - sound/meaning pairs
  - sequences of acoustic categories/words/. . .

Sound/meaning pairs



Architecture



Inputs to the learning algorithm

Three classes of objects: strings, meanings, and trees.
Accessibility for the child:

- Strings – complete
- Meanings – partial
- Trees – no information at all

The scientific question

How do children acquire language?

The scientific question

How do children acquire language?

- It’s complicated.

The scientific question

How do children acquire language?

- It's complicated.
- Isolate the possible contribution of one information source: *surface level distributional features*.
- We are going to take an idealized mathematical perspective.

A great quote from Partha Niyogi

Another aspect of the book is its focus on mathematical models where the relationship between various objects may be formally (provably) studied. A complementary approach is to consider the larger class of computational models where one resorts to simulations. Mathematical models with their equations and proofs, and computational models with their equations and simulations provide different and important windows of insight into the phenomena at hand.
In the first, one constructs idealized and simplified models but one can now reason precisely about the behavior of such models and therefore be very sure of one's conclusions. In the second, one constructs more realistic models but because of the complexity, one will need to resort to heuristic arguments and simulations. In summary, for mathematical models the assumptions are more questionable but the conclusions are more reliable - for computational models, the assumptions are more believable but the conclusions more suspect.

Two strategies

Mathematical model

- Define a specific learning algorithm.
- Prove that the algorithm is correct for some class of grammars/languages.

Computational models

- Natural language corpora (perhaps CHILDES)
- Run computer program.
- Evaluate the output in some way.

Weak learning

- We just consider a language as a set of strings L.
- We want to converge by learning a grammar that generates the same set of strings as L.
- We will just use positive data: strings of examples.

Weak learning

- We just consider a language as a set of strings *L*.
- We want to converge by learning a grammar that generates the same set of strings as *L*.
- We will just use positive data: strings of examples.

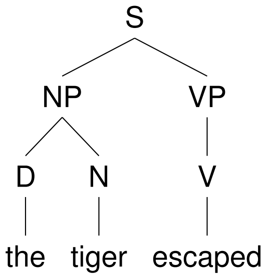
Important questions

- What are these strings of? what are the terminal symbols?
- What does the set of strings represent?
- Do we want to use a set rather than a distribution?

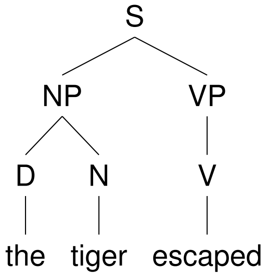
Notation

- Finite alphabet
$$\Sigma$$
- Set of all finite strings over this:
$$\Sigma^*$$
- Set of all nonempty finite strings over this:
$$\Sigma^+$$
- Empty string λ
- A formal language:
$$L \subseteq \Sigma^*$$
(Normally $L \subset \Sigma^+$)

Context free grammars



Context free grammars



Good questions

- What does the symbol NP mean?
- What does a production NP $\rightarrow$ D N mean?

Context free grammars

Notation

A context free grammar G is a collection of finite sets:

- Σ a set of terminal symbols
- V a set of nonterminal symbols
- $S \in V$ a start symbol
- P a set of productions $N \to \alpha$ where
    - $N \in V$
    - α is a finite string of symbols from Σ and V

Context free grammars

Notation

A context free grammar G is a collection of finite sets:

- Σ a set of terminal symbols
- V a set of nonterminal symbols
- $S \in V$ a start symbol
- P a set of productions $N \to \alpha$ where
    - $N \in V$
    - α is a finite string of symbols from Σ and V

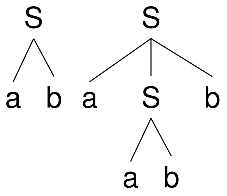
We write the derivation process using $N \overset{*}{\Rightarrow}_G w$, and

$$\mathcal{L}(G, N) = \{w \mid N \overset{*}{\Rightarrow}_G w\}$$

$$\mathcal{L}(G) = \mathcal{L}(G, S)$$

Trivial example

- Terminal symbols a, b
- One nonterminal S
- $S \rightarrow ab$ and $S \rightarrow aSb$



$$\mathcal{L}(G) = \{ab, aabb, aaabbb, \dots\}$$

Propositional logic

$A, (\neg B), (A \vee (\neg B)), \ldots$

Alphabet

rain, snow, hot, cold, danger	$A_1, A_2, \ldots$
and, or, implies, iff	$\wedge, \vee, \rightarrow, \leftrightarrow$
not	$\neg$
open, close	$(,)$

Propositional logic

$A, (\neg B), (A \vee (\neg B)), \ldots$

Alphabet

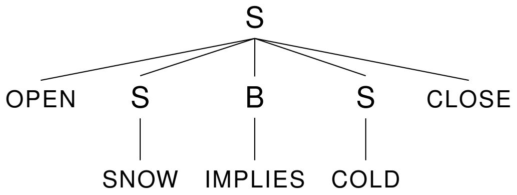
rain, snow, hot, cold, danger	$A_1, A_2, \ldots$
and, or, implies, iff	$\wedge, \vee, \rightarrow, \leftrightarrow$
not	$\neg$
open, close	$(,)$

- rain
- open snow implies cold close
- open snow implies open not hot close close

Grammar

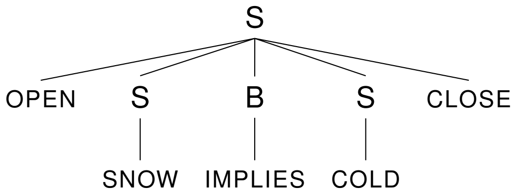
- $\Sigma = \{\text{RAIN}, \text{SNOW}, \ldots, \text{OPEN}, \ldots \text{NOT}\}$
- $V = \{S, B\}$
- $P = \{S \rightarrow \text{RAIN}, S \rightarrow \text{SNOW}, S \rightarrow \text{OPEN } SBS \text{ CLOSE}, S \rightarrow \text{OPEN NOT}S \text{ CLOSE}, B \rightarrow \text{AND}, B \rightarrow \text{OR}, B \rightarrow \text{IMPLIES} \ldots\}$

Derivations



$$S \overset{*}{\Rightarrow}_G \text{OPEN SNOW IMPLIES COLD CLOSE}$$

Derivations



$$S \overset{*}{\Rightarrow}_G \text{OPEN SNOW IMPLIES COLD CLOSE}$$

Simple example, but still an infinite, non-regular language, which is hierarchically structured.

Classic model

$$\text{PLD} \xrightarrow{\qquad\qquad} \boxed{\text{LAD}} \xrightarrow{\qquad\qquad} \hat{G}$$

Three questions

1. What are the inputs?: Strings
2. What are the outputs?: Context-free grammars
3. What conditions does the LAD have to satisfy?:
    3.1 Correctness
    3.2 Efficiency

Correctness of Weak Learners

- The learner must produce a grammar that is correct, if the PLD is big enough.
  - If the input is drawn from L_1 then it should produce a grammar for L_1
  - If the input is drawn from L_2 then it should produce a grammar for L_2
  - …
- For every language L in some class of languages $\mathcal{L}$ the learner produces the right answer.
- This is called the *learnable class* for a learner.

Distributional learning

Several reasons to take distributional learning seriously:

- Cognitively plausible (Saffran et al. 1996, Mintz, 2002)

Distributional learning

Several reasons to take distributional learning seriously:

- Cognitively plausible (Saffran et al. 1996, Mintz, 2002)
- It works in practice: large scale lexical induction (Curran, J. 2003)

Distributional learning

Several reasons to take distributional learning seriously:

- Cognitively plausible (Saffran et al. 1996, Mintz, 2002)
- It works in practice: large scale lexical induction (Curran, J. 2003)
- Linguists use it as a constituent structure test (Carnie, A, 2008)

Distributional learning

Several reasons to take distributional learning seriously:

- Cognitively plausible (Saffran et al. 1996, Mintz, 2002)
- It works in practice: large scale lexical induction (Curran, J. 2003)
- Linguists use it as a constituent structure test (Carnie, A, 2008)
- Historically, PSGs were intended to be the output from distributional learning algorithms.

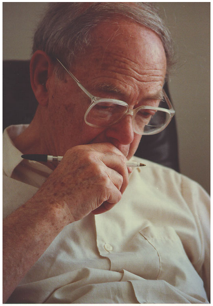
Chomsky (1968/2006)

“The concept of "phrase structure grammar" was explicitly designed to express the richest system that could reasonable be expected to result from the application of Harris-type procedures to a corpus.”

Distributional Learning

Zellig Harris (1949, 1951)

Here as throughout these procedures X and Y are substitutable if for every utterance which includes X we can find (or gain native acceptance for) an utterance which is identical except for having Y in the place of X



Example

Is ’cat’ substitutable for ’dog’?

- The cat is over there.
- I want a dog for Christmas.
- I want a Siamese cat for Christmas.
- Put a cat-flap in the door to the kitchen.
- An Alsatian is a breed of dog.
- He continues to dog my footsteps.
- I would rather have a dog than a cat as a pet.

Example

Is ’cat’ substitutable for ’dog’?

- The dog is over there.
- I want a cat for Christmas.
- I want a Siamese dog for Christmas.
- Put a dog-flap in the door to the kitchen.
- An Alsatian is a breed of cat.
- He continues to cat my footsteps.
- I would rather have a dog than a dog as a pet.
- I would rather have a cat than a dog as a pet.

Example

Distribution of “cat” in English

Infinite set of full contexts that ’cat’ can appear in :
“the □ is over there”
“I want a □ for Christmas”
...

- We can observe the distribution simply by looking at positive examples.
- We can see a similarity in distribution of “cat” and “dog”.
- Distributional learning is based on this idea.

Distributional Learning

[Harris, 1964]

- Look at the dog
- Look at the cat

Distributional Learning

[Harris, 1964]

- Look at the dog
- Look at the cat
- That cat is crazy

Distributional Learning

[Harris, 1964]

- Look at the dog
- Look at the cat
- That cat is crazy
- That dog is crazy

English counterexample

- I can swim
- I may swim
- I want a can of beer

English counterexample

- I can swim
- I may swim
- I want a can of beer
- *I want a may of beer

English counterexample

- She is Italian
- She is a philosopher
- She is an Italian philosopher

English counterexample

- She is Italian
- She is a philosopher
- She is an Italian philosopher
- *She is an a philosopher philosopher

Logic example

Propositional logic is *substitutable*:

- open rain and cold close
- open rain implies cold close
- open snow implies open not hot close

Logic example

Propositional logic is *substitutable*:

- open rain and cold close
- open rain implies cold close
- open snow implies open not hot close
- open snow and open not hot close

Substitutable Languages

- $lur \in L$
- $lvr \in L$
- $l'ur' \in L$
- $\Rightarrow l'vr' \in L$

Formally

Definition

A language L is substitutable if for all strings $l, r, l', r' \in \Sigma^*$ and for all strings $u, v \in \Sigma^+$,

$$lur, lvr, l'ur' \in L \Rightarrow l'vr' \in L$$

Not all substitutable languages are context-free

Formally

The Syntactic Congruence

Two nonempty strings u, v are congruent ($u \equiv_L v$) if for all $l, r \in \Sigma^*$
$lur \in L \Leftrightarrow lvr \in L$

Formally

The Syntactic Congruence

Two nonempty strings u, v are congruent ($u \equiv_L v$) if for all $l, r \in \Sigma^*$
$lur \in L \Leftrightarrow lvr \in L$

Complete mutual substitutability!

Congruences

A congruence is a relation which is well-behaved with respect to the structure of the space.

- The syntactic congruence is a congruence because if $u \equiv v$ then $u \cdot w \equiv v \cdot w$.
- It is a congruence with respect to the concatenation of strings.

For any strings u, v

$$[uv] \supseteq [u][v]$$

For any strings u, v

$$[uv] \supseteq [u][v]$$

Proof

Suppose $u' \in [u], v' \in [v]$

- $uv \in L$
- $u'v \in L$
- $u'v' \in L$

Alternative Definition

L is substitutable if
$lur \in L, lvr \in L \Rightarrow u \equiv_L v$

- If we see two strings that share one context in common, then we assume they share all contexts.

A very restrictive property

Substitutable language

$\{a^n c b^n \mid n > 0\}$

Not a substitutable language

$\{a, aa\}$
(Since *a* and *aa* both occur in the context $\Box$ but are not congruent)

Not a substitutable language

$\{a^n b^n \mid n > 0\}$
Since *a* and *aab* both occur in the context $\Box b$.

Algorithm

- There is a very simple algorithm for learning all substitutable languages which are also context-free languages.
- We will explain it with an illustrative example.

Example

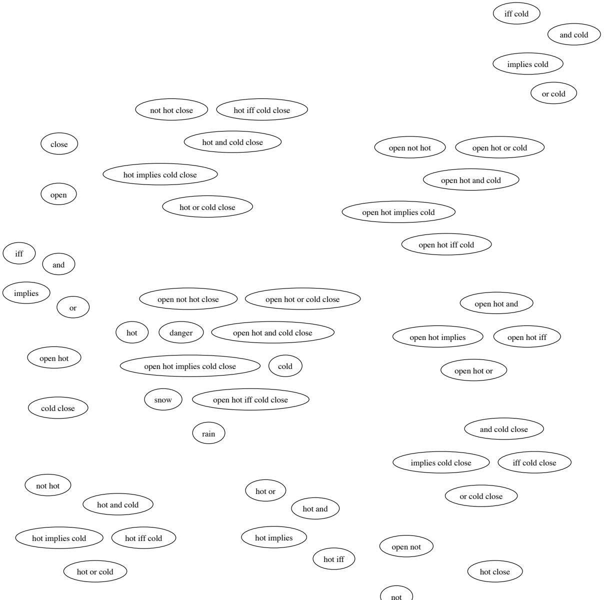
Input data $D \subseteq L$

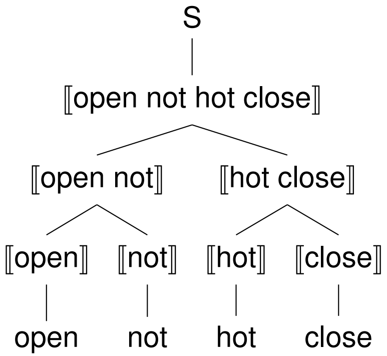
- hot
- cold
- open hot or cold close
- open not hot close
- open hot and cold close
- open hot implies cold close
- open hot iff cold close
- danger
- rain
- snow

One production for each example

- $S \rightarrow$ hot
- $S \rightarrow$ cold
- $S \rightarrow$ open hot or cold close
- $S \rightarrow$ open not hot close
- $S \rightarrow$ open hot and cold close
- $S \rightarrow$ open hot implies cold close
- $S \rightarrow$ open hot iff cold close
- $S \rightarrow$ danger
- $S \rightarrow$ rain
- $S \rightarrow$ snow

Nonterminal for each substring




S
⟦open not hot close⟧
⟦open not⟧
⟦hot close⟧
⟦open⟧
⟦not⟧
⟦hot⟧
⟦close⟧
open
not
hot
close

A trivial grammar

Input data D

$D = \{w_1, w_2, \dots, w_n\}$ are nonempty strings.

Binarise this every way

One nonterminal $[\![w]\!]$ for every substring w.

- $[\![a]\!] \rightarrow a$
- $S \rightarrow \{w\}$, $w \in D$
- $[\![w]\!] \rightarrow [\![u]\!][\![v]\!]$ when $w = u \cdot v$

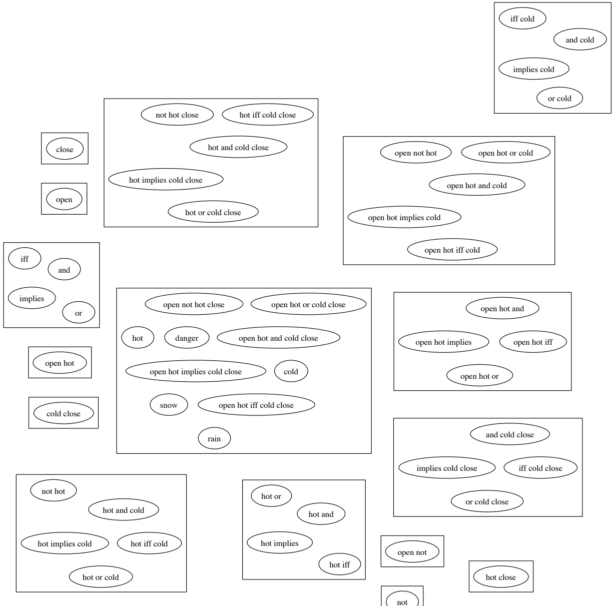
This grammar does not generalize at all! It just generates D.

Nonterminals

Nonterminals correspond to congruence classes:

- If u is a substring then we have a nonterminal $[\![u]\!]$
- $[\![u]\!]$ should generate u and all strings congruent to u: we want: $\mathcal{L}(G, [\![u]\!]) = [u]$
- So if we observe *lur* and *lvr* then we know that
    - $u \equiv v$,
    - $[u] = [v]$
    - So we want $\mathcal{L}(G, [\![u]\!]) = \mathcal{L}(G, [\![v]\!])$
    - So we either merge $[\![u]\!]$ and $[\![v]\!]$ or add productions $[\![u]\!] \to [\![v]\!]$ and $[\![v]\!] \to [\![u]\!]$.

Nonterminal for each cluster



Productions

Observation

If $w = u \cdot v$ then $[w] \supseteq [u] \cdot [v]$

Productions

Observation

If $w = u \cdot v$ then $[w] \supseteq [u] \cdot [v]$

Add production

$[\![w]\!] \rightarrow [\![u]\!][\![v]\!]$

Productions

Observation

If $w = u \cdot v$ then $[w] \supseteq [u] \cdot [v]$

Add production

$[\![w]\!] \rightarrow [\![u]\!][\![v]\!]$

Consequence

If L is substitutable, then

$$\mathcal{L}(G, [\![w]\!]) \subseteq [w]$$

$$\mathcal{L}(G) \subseteq L$$

Convergence

- As *D* increases the grammar will increase, and the language defined may increase (cannot decrease).
- When *D* is sufficiently big, the grammar will generate the whole language.

Theorem [Clark and Eyraud, 2007]

- If the language is a substitutable context-free language, then the hypothesis grammar will converge to a correct grammar.
- Efficient; provably correct

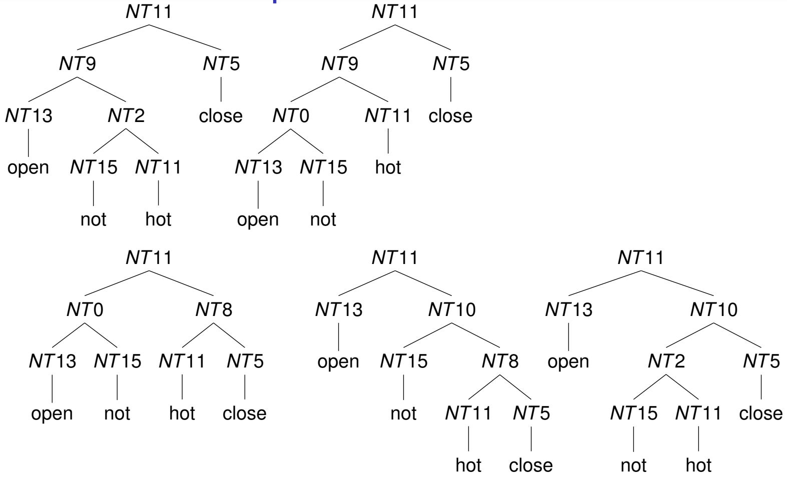
Efficiency in two senses

- Computational efficiency: the amount of computation time grows slowly as a function of the size of the input data.
- Sample efficiency: the amount of data needed to produce a correct hypothesis grows slowly as a function of the size of the target grammar.

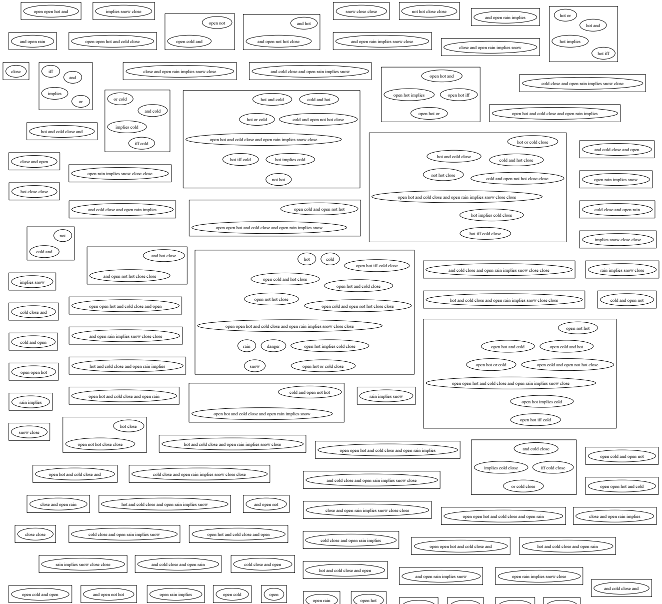
Critique

- Natural languages aren’t substitutable
- Congruence classes are inappropriate
- Natural languages aren’t context-free
- Only weak learning – doesn’t learn right structures

open not hot close

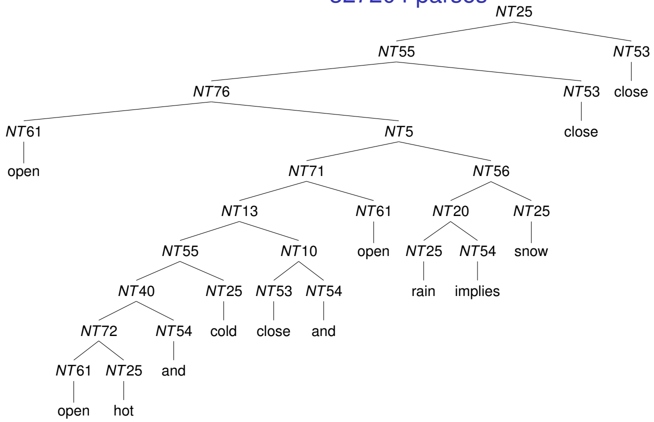


Larger data set: 92 nonterminals, 435 Productions


open open hot and
implies snow close
open not
open cold and
and hot
and open not hot close
snow close close
not hot close close
and open rain implies
hot or
hot and
hot implies
hot iff
and open rain
open open hot and cold close
and open rain implies snow close
close and open rain implies snow
close
iff
and
implies
or
close and open rain implies snow close
and cold close and open rain implies snow
open hot and
open hot implies
open hot iff
open hot or
cold close and open rain implies snow close
or cold
and cold
implies cold
iff cold
hot and cold
cold and hot
hot or cold
cold and open not hot close
open hot and cold close and open rain implies snow close
hot iff cold
hot implies cold
not hot
open hot and cold close and open rain implies
hot and cold close and
close and open
open rain implies snow close close
hot close close
hot or cold close
hot and cold close
cold and hot close
not hot close
cold and open not hot close close
open hot and cold close and open rain implies snow close close
hot implies cold close
hot iff cold close
and cold close and open
open rain implies snow
cold close and open rain
implies snow close close
and cold close and open rain implies
open cold and open not hot
open open hot and cold close and open rain implies snow
not
cold and
and hot close
and open not hot close close
implies snow
hot
cold
open hot iff cold close
open cold and hot close
open hot and cold close
open not hot close
open cold and open not hot close close
open hot and cold close and open rain implies snow close close
rain
danger
open hot implies cold close
snow
open hot or cold close
and cold close and open rain implies snow close close
rain implies snow close
hot and cold close and open rain implies snow close close
cold and open not
open not hot
open hot and cold
open cold and hot
open hot or cold
open cold and open not hot close
open open hot and cold close and open rain implies snow close
open hot implies cold
open hot iff cold
cold close and
open open hot and cold close and open
cold and open
and open rain implies snow close close
open open hot
hot and cold close and open rain implies
rain implies
open hot and cold close and open rain
cold and open not hot
open hot and cold close and open rain implies snow
rain implies snow
snow close
hot close
open not hot close close
hot and cold close and open rain implies snow close
open open hot and cold close and open rain implies
and cold close
implies cold close
iff cold close
or cold close
open cold and open not
open open hot and cold
open hot and cold close and
cold close and open rain implies snow close close
and cold close and open rain implies snow close
close and open rain
hot and cold close and open rain implies snow
and open not
close and open rain implies snow close close
open open hot and cold close and open rain
close and open rain implies
close close
cold close and open rain implies snow
open hot and cold close and open
cold close and open rain implies
open open hot and cold close and
hot and cold close and open rain
rain implies snow close close
and cold close and open rain
cold close and open
hot and cold close and open
and open rain implies snow
open rain implies snow close
and cold close and
open cold and open
and open not hot
open rain implies
open cold
open
open rain
open hot

open open hot and cold close and open rain implies snow close close

327204 parses



Analysis

- Computationally efficient.
- Infinite number of languages infinitely many of which are not regular (infinite and “hierarchically structured”)
- Uses only positive data
- Learns classes of words as well
- Simple proofs of correctness

Formal definition of algorithm

```
**Data**: A sequence of strings $w_1, w_2, \ldots$
**Data**: $\Sigma$
**Result**: A sequence of CFGs $G_1, G_2, \ldots$
let $G := \langle \Sigma, \{S\}, S, \varnothing \rangle$;
for $n = 1, 2, \ldots$ do
    if $G$ does not generate $w_n$ then
        $G = \hat{G}(w_1, \ldots, w_n)$;
    end if
    output $G$;
end for
```

Formal definition of grammar construction procedure

```
**Data**: A finite set of strings $D = w_1, w_2, \ldots, w_n$
**Data**: $\Sigma$
**Result**: A CFG $G$
Let $V = \{S\} \cup \{[\![u]\!] \mid u \in \mathrm{Sub}(D)\}$
Let $P_L = \{[\![a]\!] \to a \mid a \in \mathrm{Sub}(D)\}$
Let $P_I = \{S \to [\![w]\!] \mid w \in D\}$
Let $P_B = \{[\![uv]\!] \to [\![u]\!][\![v]\!]\}$
Let $P_U = \{[\![u]\!] \to [\![v]\!] \mid lur \in D, lvr \in D\}$
Let $P = P_L \cup P_I \cup P_B \cup P_U$
output $\langle \Sigma, V, S, P \rangle$
```

Why does this work? (I)

Safe generalization

The substitutability property guarantees that we won't overgeneralise.

Why does this work? (I)

Safe generalization

The substitutability property guarantees that we won't overgeneralise.

- So if the language isn't substitutable, we need a better way of testing whether strings are congruent (or whatever).

Why does this work? (II)

Representational adequacy

The nonterminals can (must!) be congruence classes.

Why does this work? (II)

Representational adequacy

The nonterminals can (must!) be congruence classes.

- Suppose $N \overset{*}{\Rightarrow} u$ and $N \overset{*}{\Rightarrow} v$
- Assume that $S \overset{*}{\Rightarrow} lNr$
- Then $lur \in L$ and $lvr \in L$
- So $u \equiv v$
- Moreover, we only need one nonterminal for each congruence class.
- But if the language is not substitutable this no longer holds.

Why does this work? (III)

Efficient enumeration of substructures.

We can decompose the strings efficiently.

Why does this work? (III)

Efficient enumeration of substructures.

We can decompose the strings efficiently.

- If w is of length n
- then there are only $n(n+1)/2$ ways to decompose it into $l \cdot u \cdot r$

Why does this work? (IV)

Efficient parsing

We can test whether $w \in L(G)$ efficiently.

Why does this work? (IV)

Efficient parsing

We can test whether $w \in L(G)$ efficiently.

- If we can't parse efficiently we (probably/almost certainly) can't learn efficiently.

Classes of grammars

- The representation class is the class of all CFG.
- The learnable class is the class of substitutable CF languages.
- The hypothesis class is rather hard to define: the algorithm sometimes outputs grammars that define languages that are not substitutable.
- There are languages which the learner will correctly learn some of the time, but not all of the time.
- Moreover whether a grammar defines a substitutable language is not a decidable property, so it may be *impossible* to restrict the learner.

Two “universals”?

(Both of these seem to be false.)

- Natural languages are representable by CFGs.
- Natural languages are substitutable.

Question?

What happens (i.e. how does it generalize) when the learner receives:

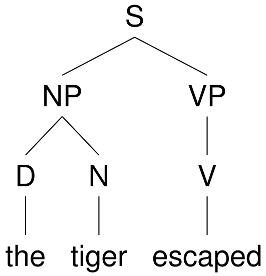
- the cat
- the big cat

Question?

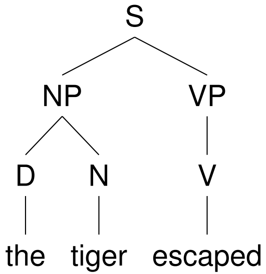
What happens (i.e. how does it generalize) when the learner receives:

- the cat
- the big cat
- the very big cat

Context free grammars



Context free grammars



Good questions

- What does the symbol NP mean?
- What does a production NP $\rightarrow$ D N mean?

A partial answer

- A nonterminal is/represents a congruence class

$$[u]$$

- A production represents an inclusion relation between these classes:

$$[u] \supseteq [v][w]$$

Machine Learning

Machine Learning definition; Tom Mitchell

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

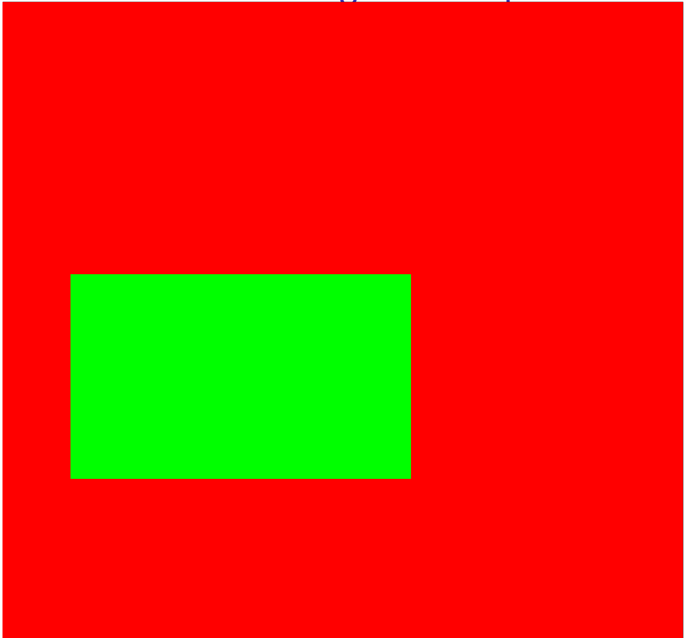
Theoretical analysis

- Computational learning theory
- Algorithmic learning theory
- Inductive inference

Diagrams

- A point on the plane is a string of words.
- The plane consists of every possible string of words.
- A language is an infinite set of sentences: a region in the plane. e.g. a rectangle.
- Points inside the rectangle represent grammatical sentences. Points outside the rectangle represent ungrammatical sentences.

Target concept



Terminology

- The instance space X: the plane
- Instances x are points in the plane
- A concept C is a rectangle which divides the points into positive and negative examples (infinitely many of each)
- The concept space $\mathcal{C}$ is in this case the set of all axis aligned rectangles (infinitely many of these)
- The finite representation is just 4 numbers.

For simplicity we just assume rational numbers rather than reals.

Learning

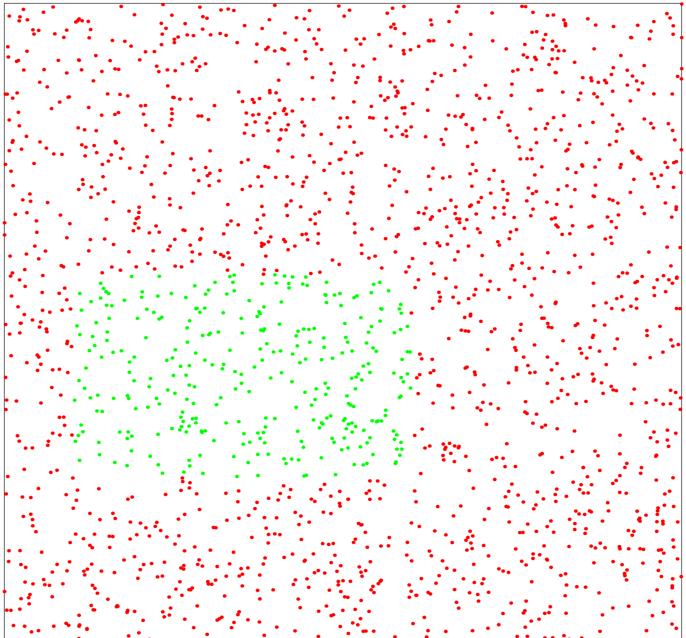
- The learner is a function from the finite input examples to a representation (the hypothesis).
- The hypothesis typically generates many more examples than the input data.
- Typically it generates an infinite number of examples, whereas we only have a small finite set of examples.
- Generalization is key!
- (Imitation is not the answer! cf. Kristen)

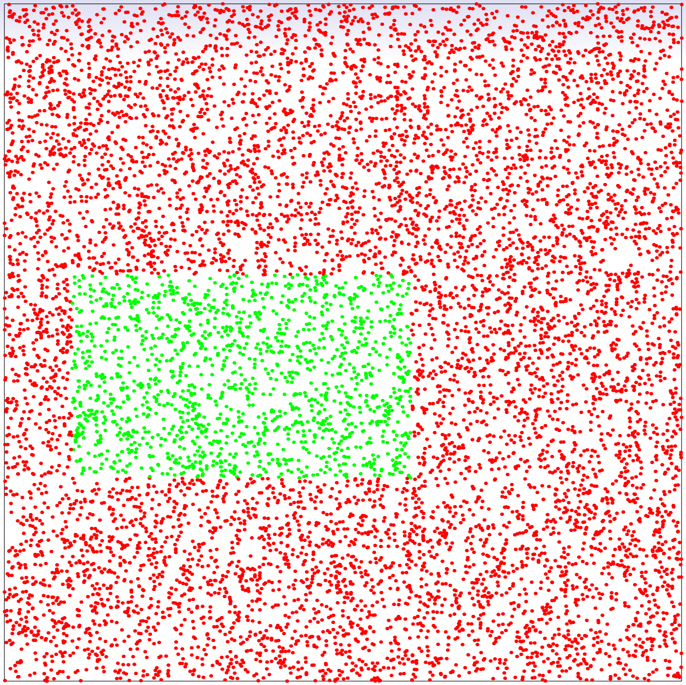
Labeled data

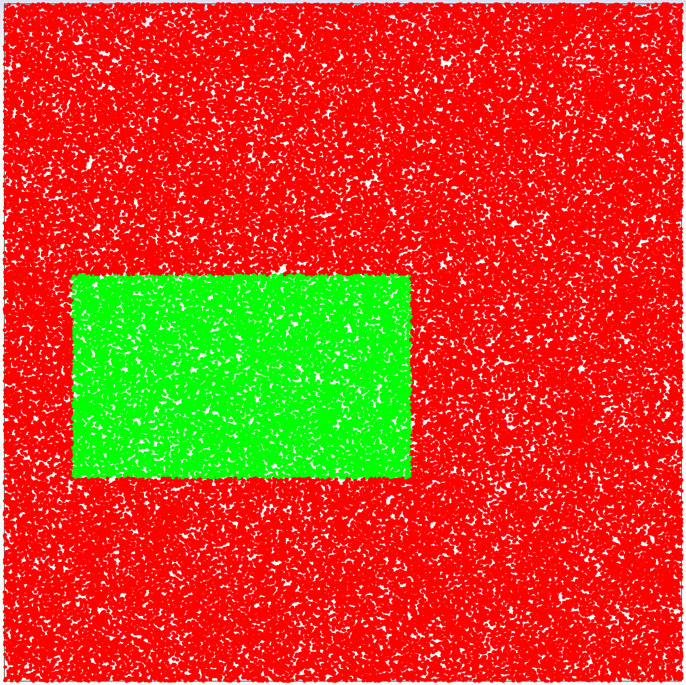
Supervised learning

each example is labeled:
green means it is in the language – grammatical
red means it is not in the language – ungrammatical

Finite data



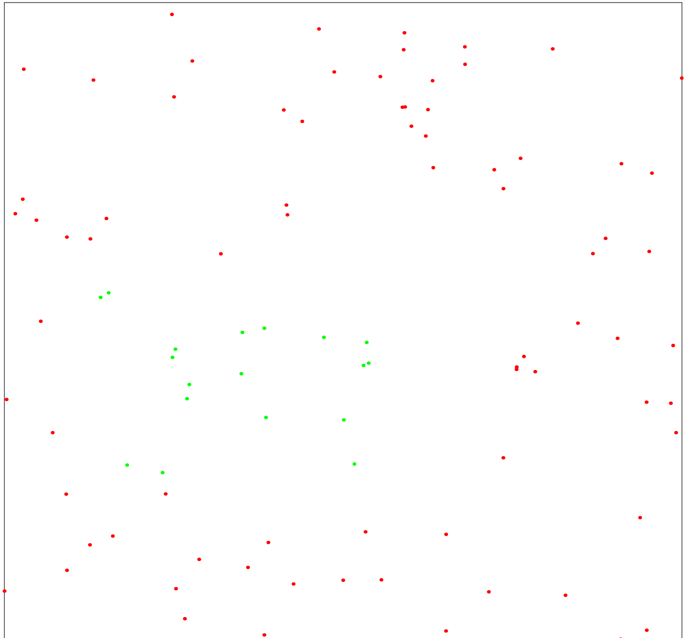




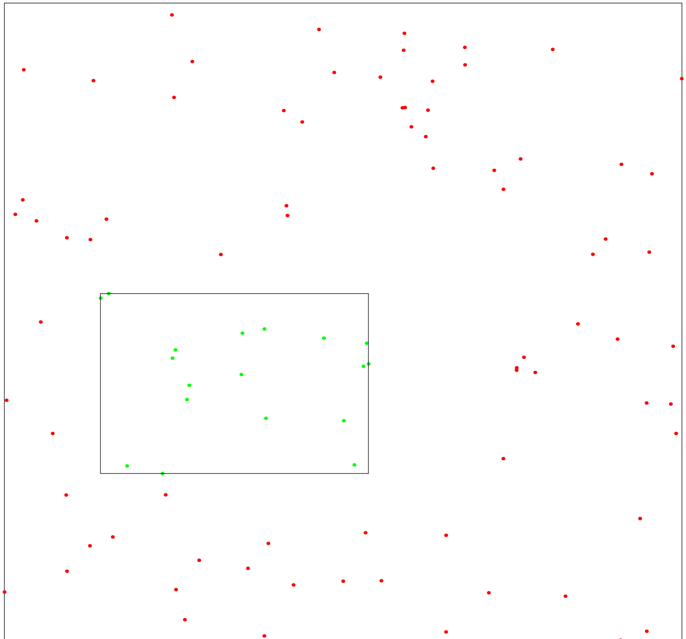
Obvious comment

- More data means a more accurate hypothesis.
- You might never get an exactly correct hypothesis.

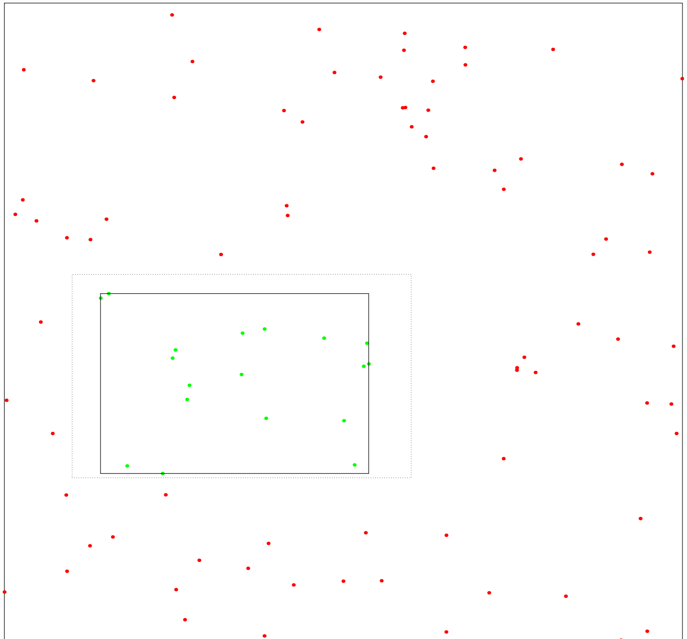
Error



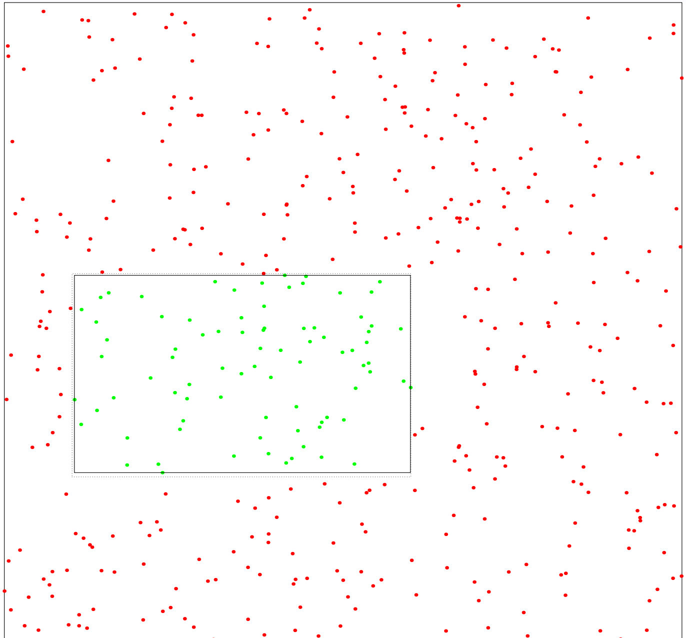
Error



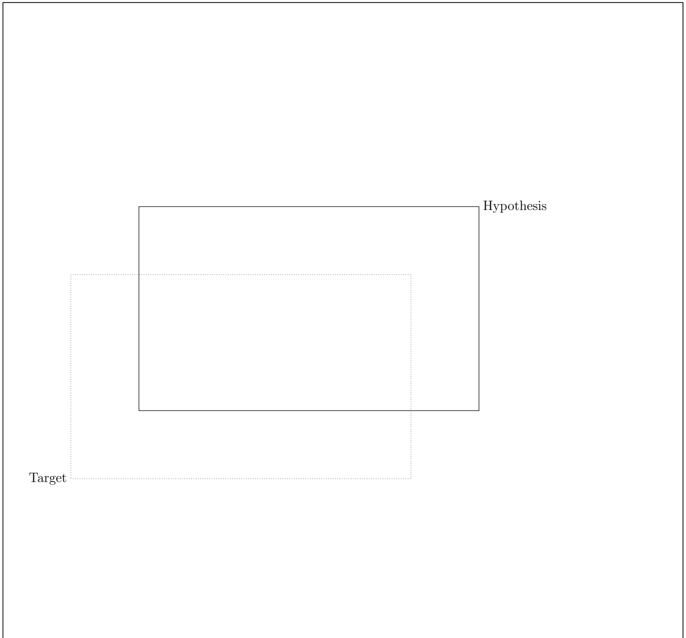
Error



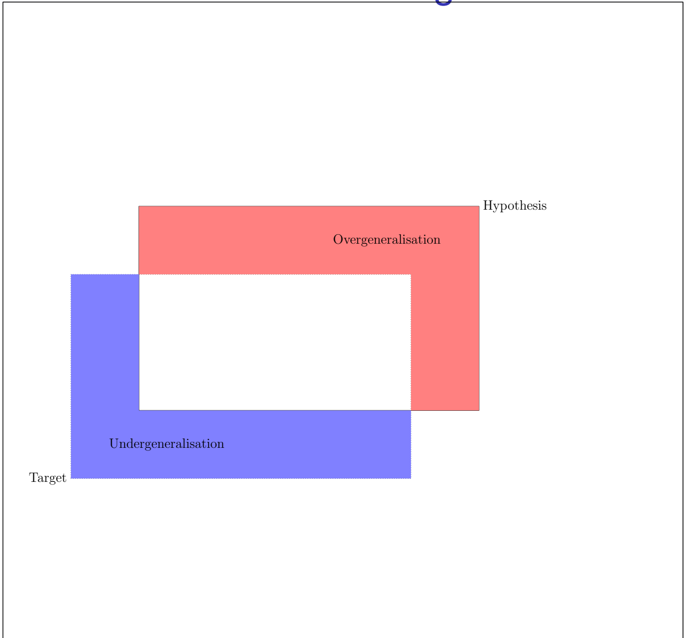
Error



Error



Measuring error



Measuring error

Why is this right? We cannot look in someones head and see if they have the right hypothesis.

Observation

- Do they object to sentences that we think are acceptable?
- Do they produce sentences that we think are unaccetaple?

The most we can say is that these events are rare.

Hypothesis class versus concept class

Hypothesis class

The set of hypotheses that the learner will generate under various inputs.

Concept class

The set of target concepts that the learner may encounter.

Hypothesis class versus concept class

Hypothesis class

The set of hypotheses that the learner will generate under various inputs.

Concept class

The set of target concepts that the learner may encounter.

These can be different:

- The learner might encounter some triangles even though it always generate rectangles
- or the learner might not know that it is only going to receive rectangles and conjecture pentagons sometimes.

Hypothesis class versus concept class

“Proper” learning

In the example, the learner has prior knowledge of the concept class.
Hypothesis class equals concept class.

The hypothesis class might be too small

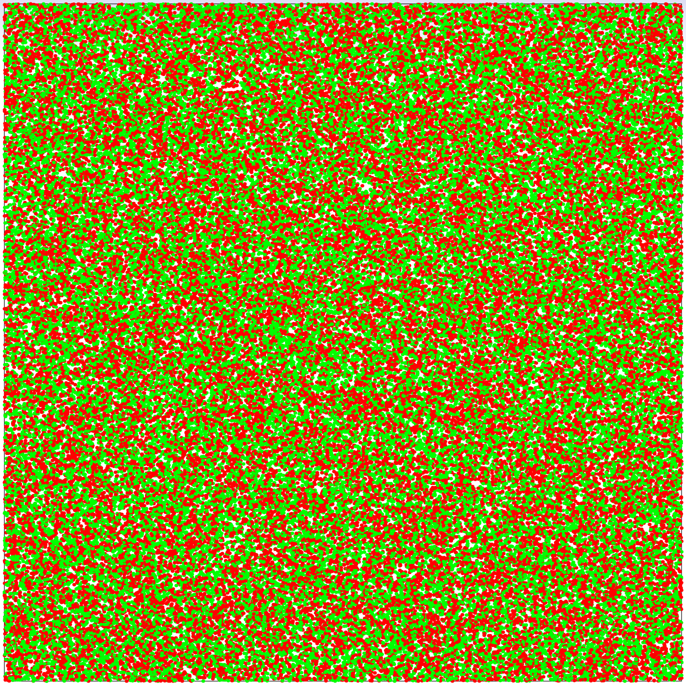
In this case the learner will sometimes fail

The concept class might be much smaller than the hypothesis class

For example, the hypothesis class might be all convex polygons.

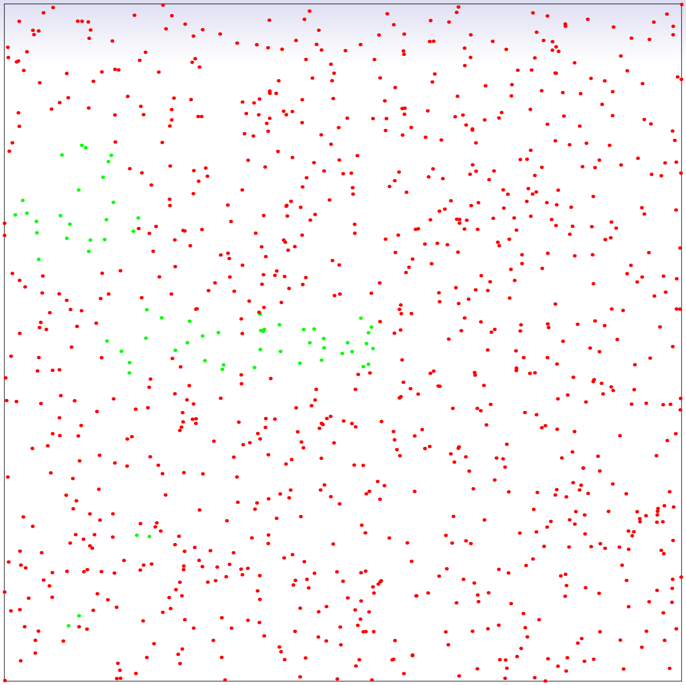
Concept class must be bounded

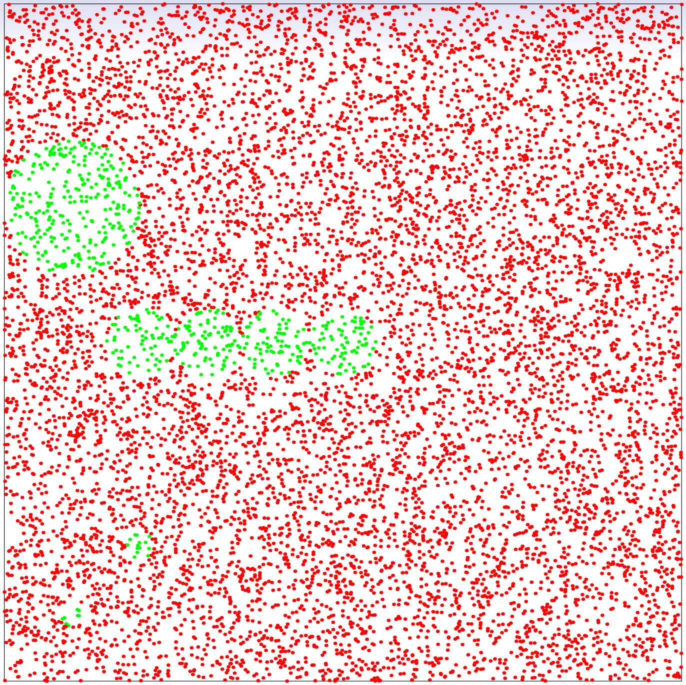
- We can’t learn arbitrary sets.
- There must be some restrictions on the concepts: e.g. connected regions.

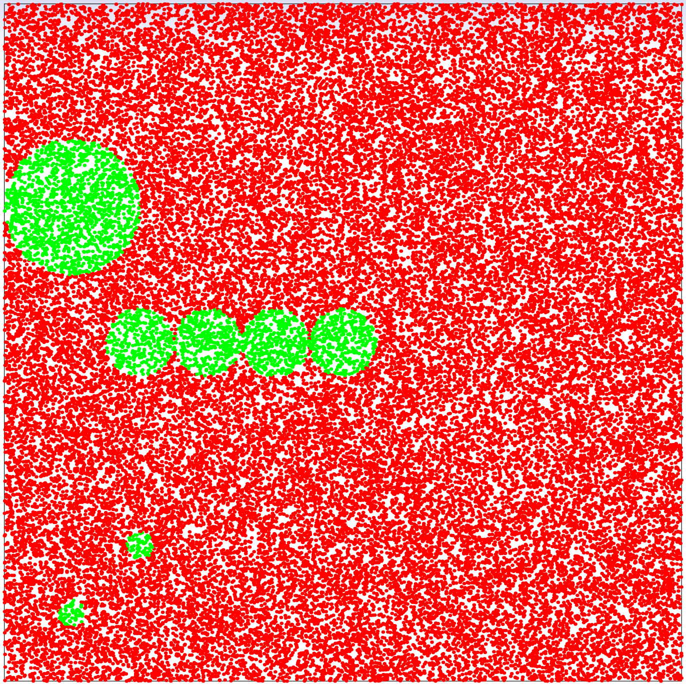


Complex hypotheses

If the shape is more complex then we will need more data to learn.





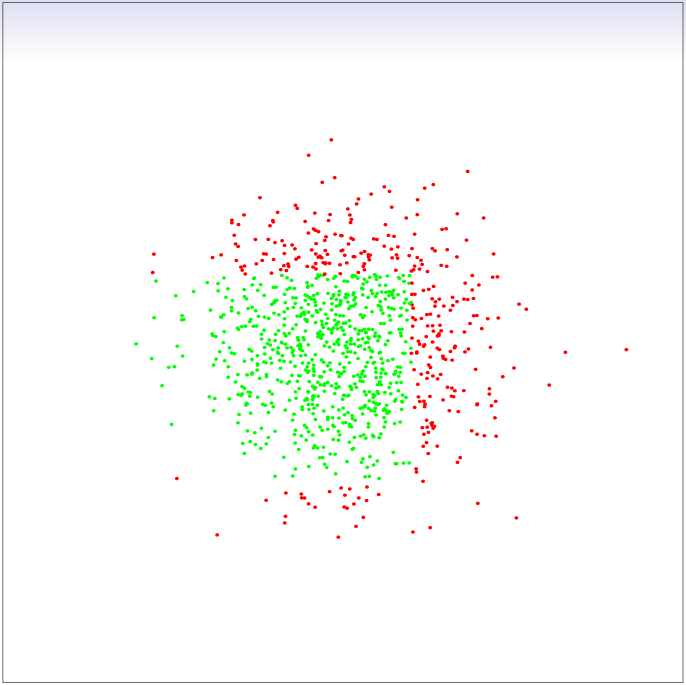


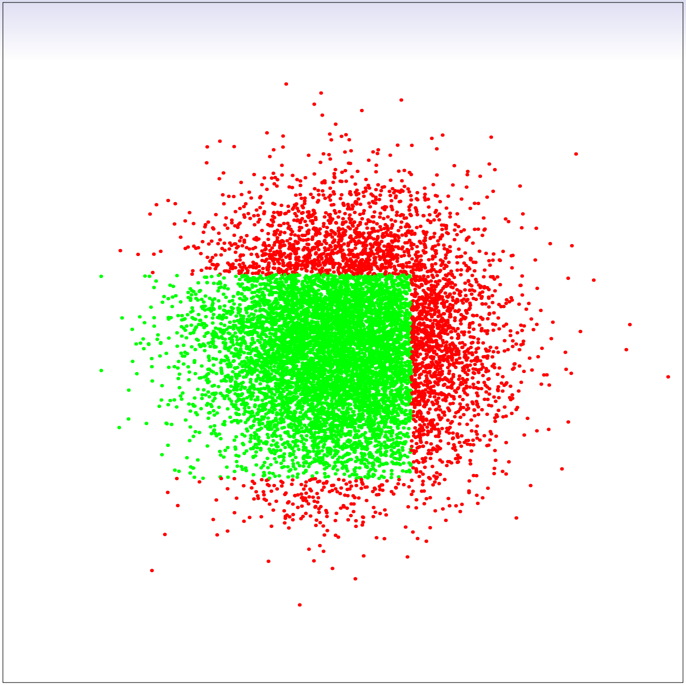
Complex hypotheses

- If we only have 1000 data points, we can't hope to learn a complex hypothesis class of 10000 tiny balls.
- But if we had a lot of data (much more than 10000) then we could still learn it
- So amount of data we need to learn depends on the accuracy and the complexity of the target.

Distribution of examples

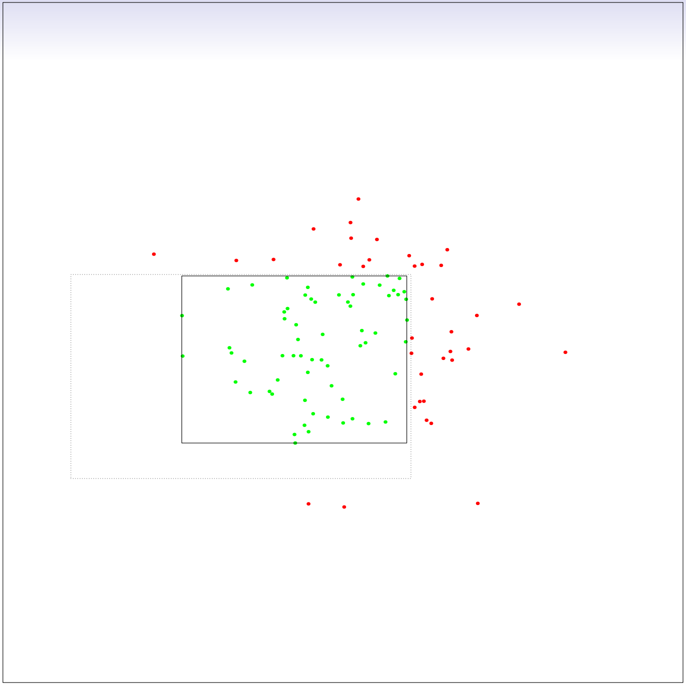
- So far all of these points are distributed evenly, uniformly over the plane.
- The area of a region is proportional to the probability
- But it might be that points are more likely to fall in one area of the plane than another.

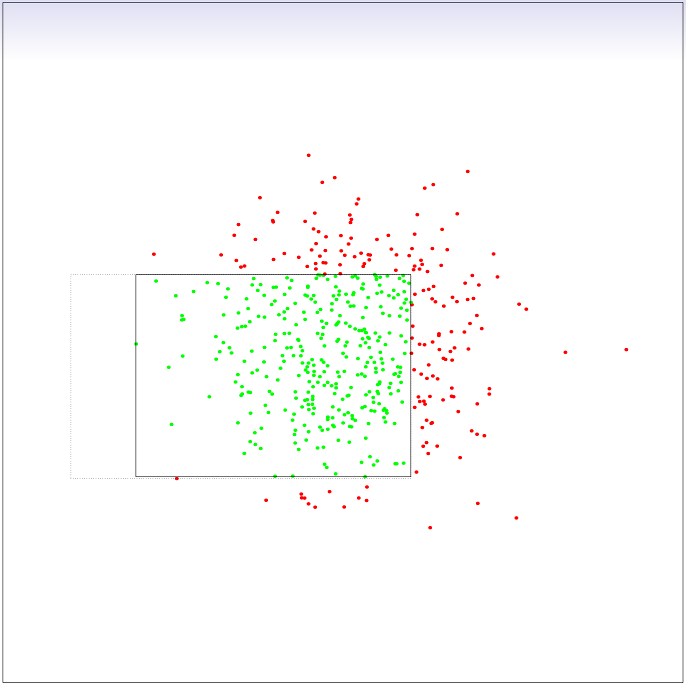




Error

- In this case the error must be “fair”.
- We can’t expect the learner to get the left part right, but it doesn’t matter as they are unimportant.
- We will measure the error by the probability of the area not the area





Rectangles

Rectangles can be learned by the simple algorithm: pick the smallest rectangle that includes the target concept.

Prior knowledge

That the concepts are rectangles.

Is this necessary?

Can we conclude that the learner must know that the concepts are rectangles?

Rectangles

Rectangles can be learned by the simple algorithm: pick the smallest rectangle that includes the target concept.

Prior knowledge

That the concepts are rectangles.

Is this necessary?

Can we conclude that the learner must know that the concepts are rectangles?

No

There are other algorithms that **don't** make this assumption that can also learn rectangles and many other shapes as well.

Difference in goals

Engineering versus Cognitive modeling

Machine learning

Solving problems
Proving that classes can be learned

Language acquisition

Understanding properties of the learner
What can we infer about the learner from its performance?

Unlabeled examples

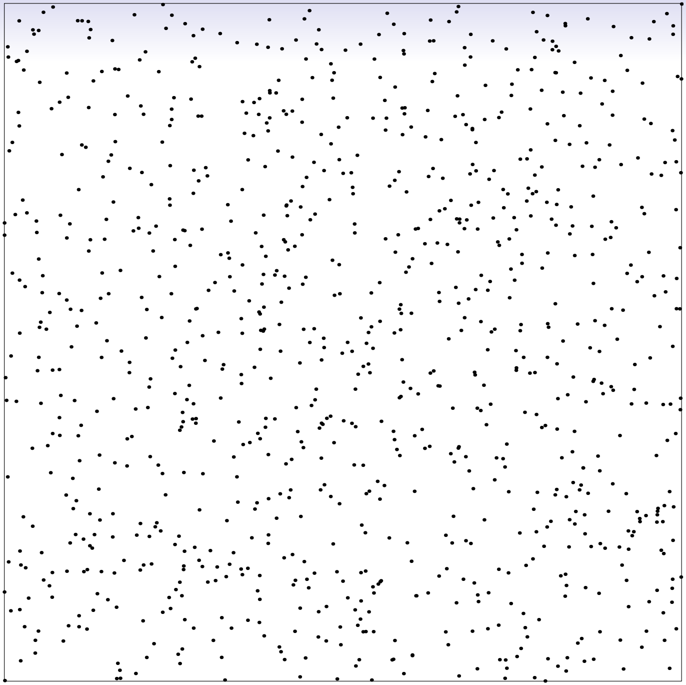
Labeled examples

Each point comes with a color that tells you whether it is in the concept or not.

Unlabeled examples

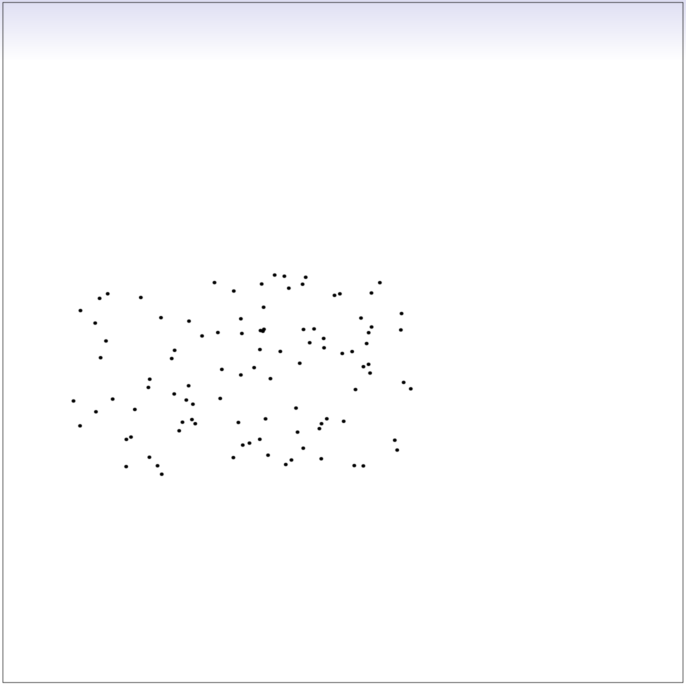
Each point is black

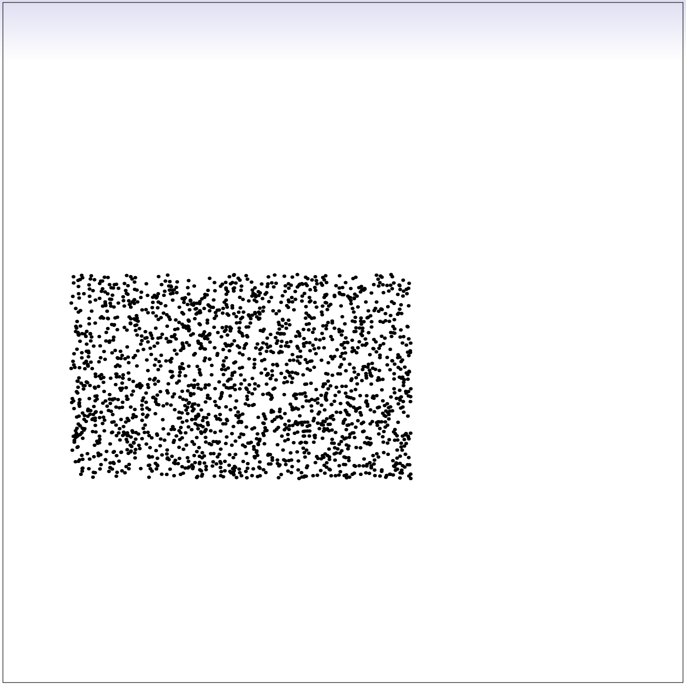
If we just have unlabeled examples then the distribution of the examples must be restricted in some way or we can't learn.



Positive only examples

- If they are unlabeled but restricted to positive examples then we can still learn.
- No negative data





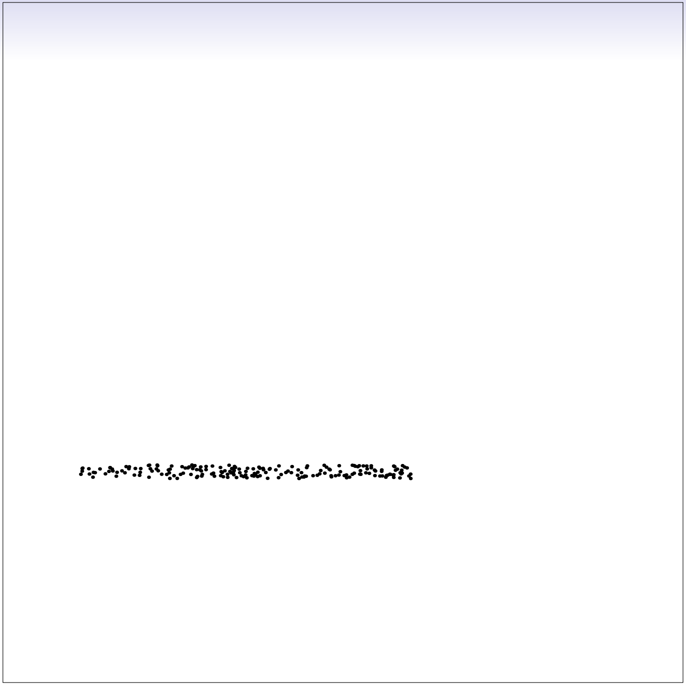
Asymmetry

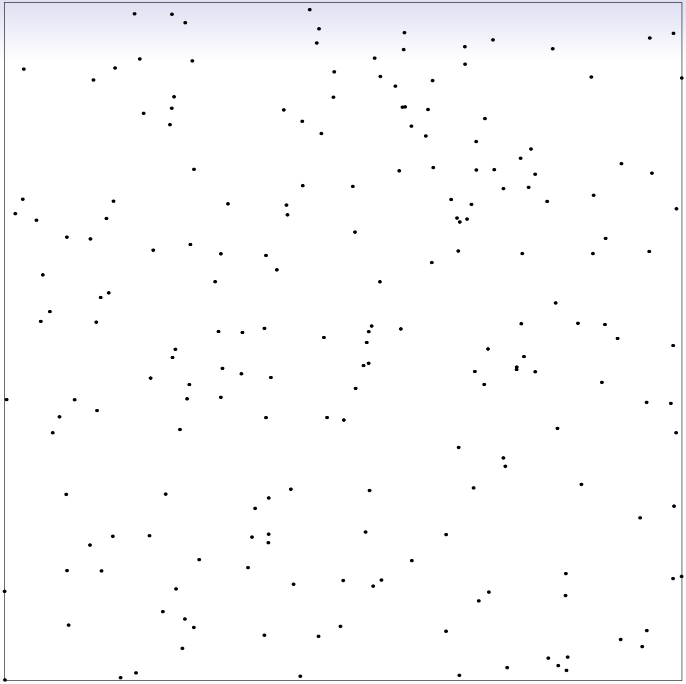
Small concept

If the concept is very small, then positive examples are much more useful than negative examples.

Large concept

If the concept is very large, then negative examples are much more useful than positive ones.

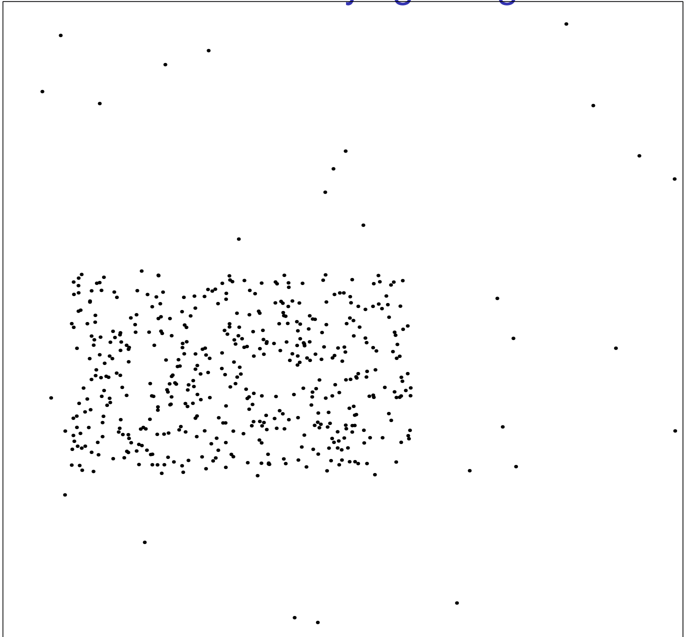




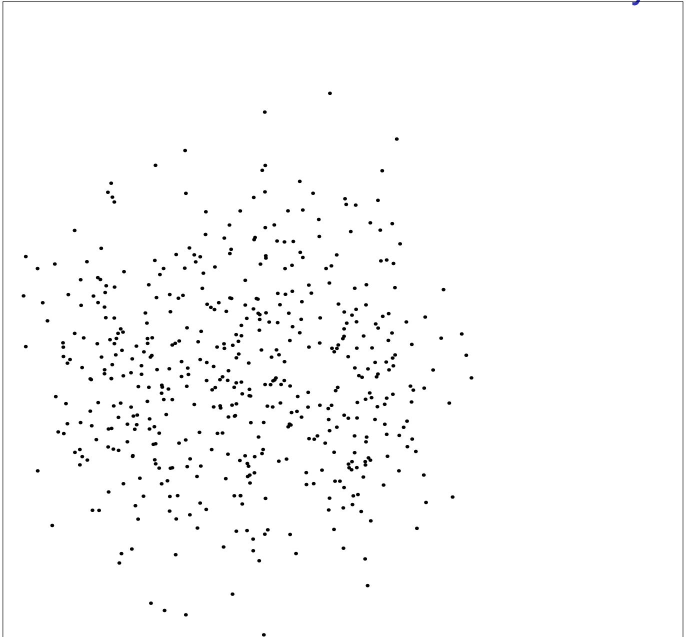
Noise

- In the real world, there is “noise”.
- A general problem in learning and perception for all domains.
- We are assuming that all of the examples are in the concept – i.e. are grammatical
- In reality, some will be misheard or corrupted in some way.

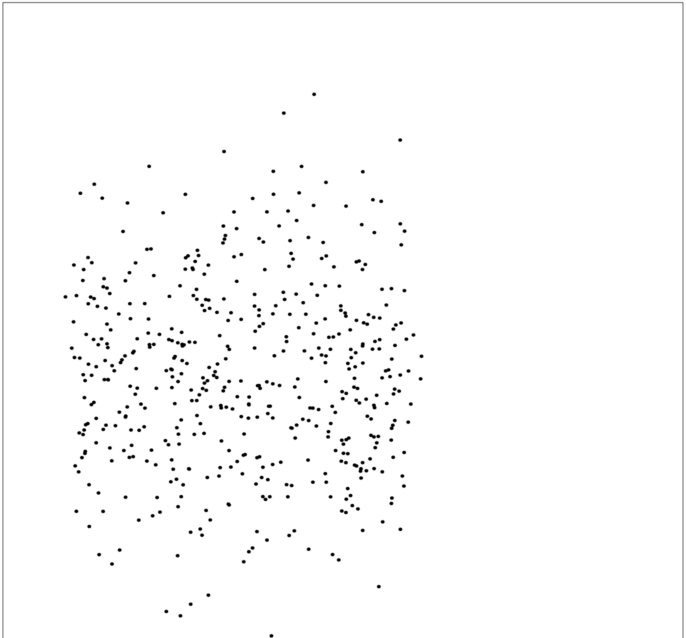
Noise with underlying categorial distinction



Noise with blurred boundary



Mixture

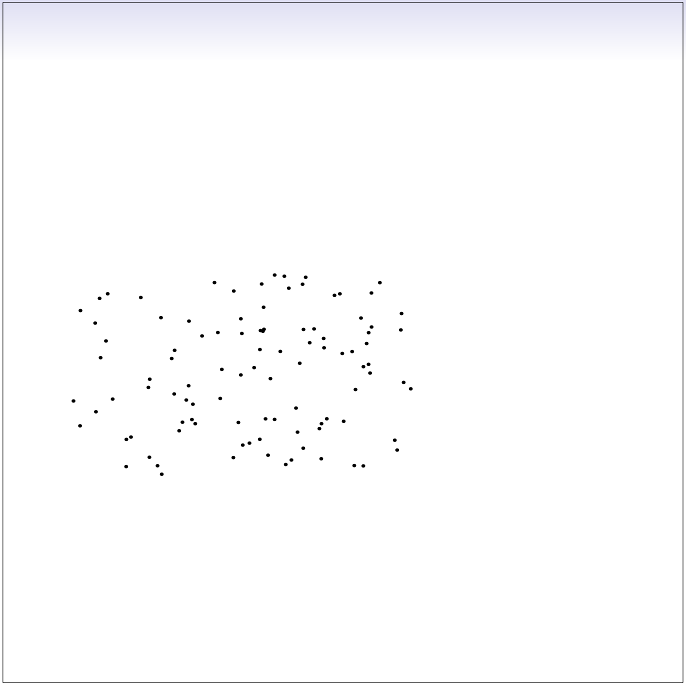


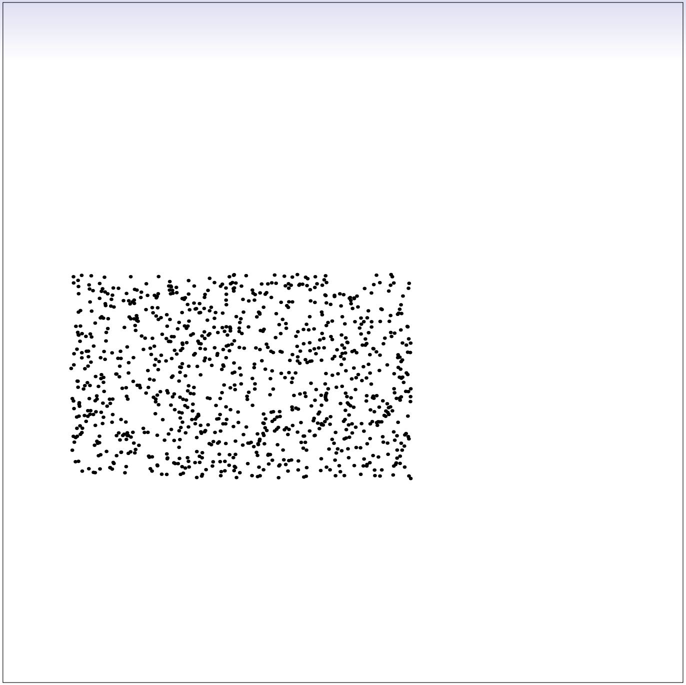
Negative evidence

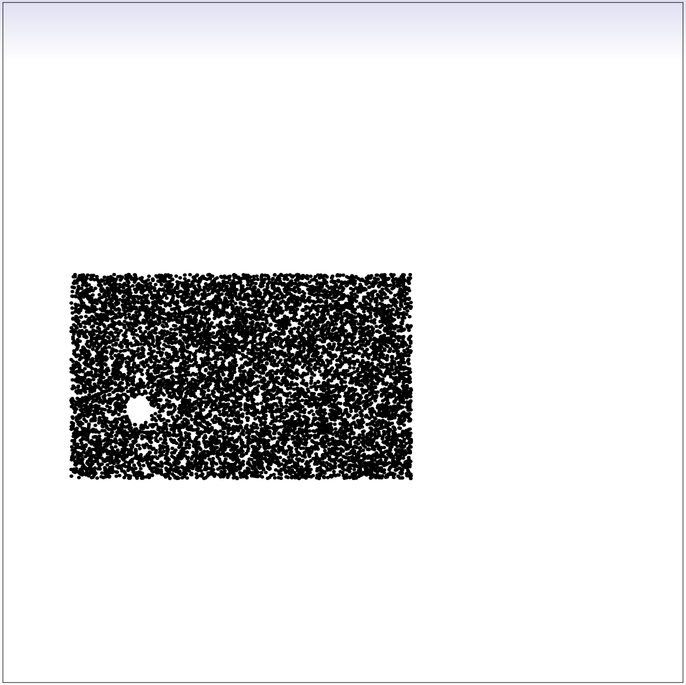
- Negative evidence is not considered a problem in NLP or in unsupervised learning: either in theory or in practice.
- Direct negative evidence is completely useless in practice: almost all long strings of English words are ungrammatical.
    - Jim address tasting array umpet tag ever zoo minibeasts dodo
    - party Victoria claps wrecking weakness spanked grips apricots lunchbox bell
    - surgery gymnast taxi washable ropes cleaner measurer Scotsman rummage gracious

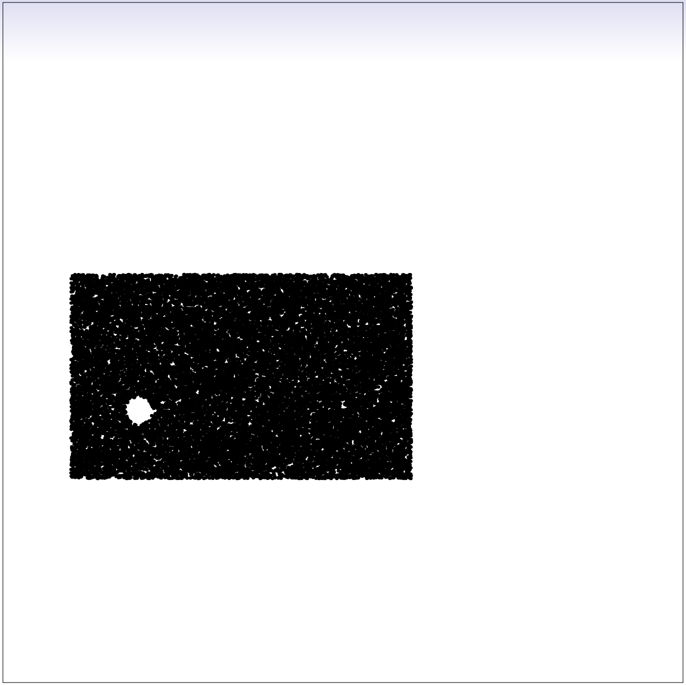
Overgeneralisation

- the claim is sometimes made that recovering from overgeneralisation is impossible with only positive data
- this example is too simple to exhibit this.









Shrinking the hypothesis

Why do we switch from the larger incorrect hypothesis that is too general to a smaller one, even though we have not seen any labeled negative examples?

Shrinking the hypothesis

Why do we switch from the larger incorrect hypothesis that is too general to a smaller one, even though we have not seen any labeled negative examples?

Indirect negative evidence

Because there is a space with no examples in, where we would expect there to be examples.

Problem is hard

Two classes of problems: information theoretic and complexity theoretic.

Information theory

Is there in principle enough information in the input?

Computational Complexity

Can we use this information to construct our hypothesis efficiently?

Problem is hard

Two classes of problems: information theoretic and complexity theoretic.

Information theory

Is there in principle enough information in the input?

Computational Complexity

Can we use this information to construct our hypothesis efficiently?

Tractable Cognition Thesis (van Rooij, 2008)

Human cognitive capacities are constrained by the fact that humans are finite systems with limited resources for computation.

Why can we learn rectangles/polygons/convex regions?

- Computationally it is easy to compute the smallest rectangle that includes all the positive examples.
- Information theoretically: convex regions satisfy a simple closure property.

$$\vec{x} \in L, \vec{y} \in L \Rightarrow \alpha\vec{x} + (1 - \alpha)\vec{y} \in L$$

Why can we learn rectangles/polygons/convex regions?

- Computationally it is easy to compute the smallest rectangle that includes all the positive examples.
- Information theoretically: convex regions satisfy a simple closure property.

$$\vec{x} \in L, \vec{y} \in L \Rightarrow \alpha \vec{x} + (1 - \alpha)\vec{y} \in L$$

What are the *rectangles* and *convex regions* when we are learning generative grammars?

Putnam (1967)

In the absence of any knowledge of what general multipurpose learning strategies might even look like, the assertion that such strategies (which absolutely must exist and be employed by all humans) cannot account for this or that learning process, that the answer or an answer schema must be ’innate’, is utterly unfounded.
Until we understand the strategies which make general learning possible - and vague talk of ’classes of hypotheses’ - and ’weighting functions’ is utterly useless here - no discussion of the limits of learning can even begin.

Formal models of learning

Formalising the problem

- Gold's model of identification in the limit.
- Probably approximately correct (PAC) learning. (Valiant)

Elements of the learning model

- We have a class of representations (e.g. CFGs)
- These define sets of instances (e.g. strings)
- A learner is a function:
    - From finite sequences of instances
    - To representations
- To be successful, the learner has to converge:
    - To a (the?) right answer
    - Under some conditions (amount of data, amount of computation)
    - for every grammar in some class

E Mark Gold

Language Identification in the Limit, 1967

Seminal paper and worth reading in the original.

- Almost the first formal model of learning
- Very influential but different from mainstream machine learning
- Still useful as it often leads to simple proofs and analysis
- Often forms the basis for APS arguments

The Gold Paradigm

- In Gold's (1967) Identification in the Limit (IIL) paradigm a language consists of a set of strings
- There is a target language which is unknown to the learner T.
- The learner is presented with an infinite sequence of strings, possibly labeled as to whether they are grammatical or not
- At each step, the learner must produce a hypothesis $H_1, H_2, \ldots$
- As the learner gets more information, the hypotheses H_i should converge to the target T.

Alternative IIL Models

Two types of “presentation” of a language:

Positive data only

The input is a sequence of unlabelled examples from the languages: $s_1, s_2, \ldots$

- It never sees any sentences that are not in L
- Every sentence in L must appear at least once in the sequence.
- No other constraints at all on repetition, order etc.

Alternative IIL Models

Two types of “presentation” of a language:

Positive data only

The input is a sequence of unlabelled examples from the languages: $s_1, s_2, \ldots$

- It never sees any sentences that are not in L
- Every sentence in L must appear at least once in the sequence.
- No other constraints at all on repetition, order etc.

Positive and negative data

the examples are labeled as to whether they are in L or not in L
every possible string occurs in the sequence
no other constraints
Nothing like language acquisition

Convergence in the Gold Paradigm

- For a language L and a presentation of L the learner identifies in the limit the language L, if there is some N such that for all $n > N$, $G_n = G_N$, and G_N is a correct representation of L.
- IIL requires that a learner converge on the correct representation G_L of a language L in a finite but unbounded period of time.
- Alternatively, the learner only changes his hypothesis finitely many times, and ends on a correct hypothesis
- It only makes a finite number of mistakes.

IIL

IIL of a language

We say an algorithm *A* identifies in the limit a language *L*, if for every presentation of *L*, *A* converges in this sense.

IIL of a class of language

We say an algorithm *A* identifies in the limit a class of languages $\mathcal{L}$, if for every *L* in $\mathcal{L}$, *A* identifies in the limit *L*.

General properties

- Learnability is a property of classes of languages not languages

General properties

- Learnability is a property of classes of languages not languages
- If a class C of languages is not learnable, then any class that contains C is also not learnable.
- If a class C of languages is learnable then any smaller class is also learnable

Positive Evidence Only: The Class of Finite Languages

- The class of finite languages includes all and only languages with a finite number of strings.
- This class is itself infinite, as there are an infinite number of finite languages.

- **Gold Result 1:**
  The class of finite languages is identifiable in the limit on the basis of positive evidence.

The Rote learner

Prior knowledge/hypothesis class

Representation as a finite list or set
$G = \{w_1, w_2, \ldots, w_k\}$

Algorithm

Simply memorise the examples seen so far.

Positive Evidence Only: Finite Classes of Languages

- A finite class of languages $\mathcal{L}$ contains only a finite number of languages in the class.
- $\mathcal{L}$ may contain infinite languages, which are languages with an infinite number of strings.

- **Gold Result 2:**
  Any finite class of languages is identifiable in the limit on the basis of positive evidence.

Positive Evidence Only: Super Finite Languages

Negative result

- A super-finite class of languages is any class that contains all finite languages and at least one infinite language.
- The most influential Gold result proves the non-learnability in the limit of any such class of languages in the positive evidence only IIL model.
- **Gold Result 3:**
  Any class that is super-finite is not identifiable in the limit on the basis of positive evidence.

Corollary: the classes of regular/context-free/context-sensitive languages are not IIL from positive data.

Analysis of the Gold model

Disadvantages:

- The learner has to succeed under every presentation: even when it is adversarial.
- It doesn't incorporate complexity at all.
- It doesn't characterise learnability very well.

Advantages:

- It is easy to use: mathematically convenient.
- It fits well with discrete spaces.
- We can fix some of the limitations later on.

Acquisition

[Chomsky, 1973]

The fundamental empirical problem of linguistics is to explain how a person can acquire knowledge of language

Acquisition

[Chomsky, 1973]

The fundamental empirical problem of linguistics is to explain how a person can acquire knowledge of language

- The fundamental theoretical problem of linguistics is to have a theory of grammatical inference.

Bibliography I

Chomsky, N. (1973).
Conditions on transformations.
In Anderson, S. and Kiparsky, P., editors, *A festschrift for Morris Halle*, pages 232–286. Holt, Rinehart and Winston.

Clark, A. and Eyraud, R. (2007).
Polynomial identification in the limit of substitutable context-free languages.
Journal of Machine Learning Research, 8:1725–1745.

Harris, Z. (1964).
Distributional structure.
In Fodor, J. A. and Katz, J. J., editors, *The structure of language: Readings in the philosophy of language*, pages 33–49. Prentice-Hall.