In propositional logic, we have proved the following classes PAC learnable:

- Conjunctions and disjunctions on $n$ propositional symbols
- $k$-conjunctions and $k$-disjunctions
  - contain at most $k$ literals
  - $|F^k_{\text{conj}}| = |F^k_{\text{disj}}| = \mathcal{O}(n^k)$
- $k$-CNF and $k$-DNF
  - Conjunctions of clauses ($k$-disjunctions), resp. disjunctions of terms ($k$-conjunctions)
  - Poly number of $\mathcal{O}(n^k)$ different possible clauses (terms), so $k$-CNF and $k$-DNF as easy as conjunctions (disjunctions)
- $k$-term DNF and $k$-clause CNF
  - by $k$-CNF, resp. $k$-DNF, not by themselves!
- $k$-Decision lists

What about learnability in predicate logic?
Review of First-Order Predicate Logic Syntax

\[ \forall x \text{ human}(\text{father}(x)) \rightarrow \neg \text{species}(x, \text{dinosaur}) \]

Functions can be nested, e.g. \( \text{father}(\text{father}(\ldots \text{father}(x) \ldots)) \).
Operator precedence

To avoid too many parentheses, we set the following operator precedence:

1. negation $\neg$
2. conjunction $\land$
3. disjunction $\lor$
4. implication $\rightarrow$
5. quantifiers $\exists$, $\forall$
CNF Representation

Any first-order formula can be converted into a CNF, which is a conjunction of universally quantified clauses = clausal theory

Example:

\[ \forall x \ person(x) \rightarrow \exists y \ parent(y, x) \land \neg (\mother(y, x) \land \father(y, x)) \]

rewrite (note precedence of \( \land \) over \( \lor \)):

\[ \forall x \ \neg \person(x) \lor \exists y \ parent(y, x) \land \neg (\mother(y, x) \land \father(y, x)) \]

move negation straight to atoms:

\[ \forall x \ \neg \person(x) \lor \exists y \ parent(y, x) \land (\neg \mother(y, x) \lor \neg \father(y, x)) \]
skolemize:

$$\forall x \neg \text{person}(x) \lor \text{parent}(\text{par}(x), x) \land$$

$$\left( \neg \text{mother}(\text{par}(x), x) \lor \neg \text{father}(\text{par}(x), x) \right)$$

use the distribution principle $a \lor (b \land c) \equiv (a \lor b) \land (a \lor c)$

$$\forall x \left( \neg \text{person}(x) \lor \text{parent}(\text{par}(x), x) \right) \land$$

$$\left( \neg \text{person}(x) \lor \neg \text{mother}(\text{par}(x), x) \lor \neg \text{father}(\text{par}(x), x) \right)$$

Drop universal quantifiers since all variables are now quantified universally.

$$\left( \neg \text{person}(x) \lor \text{parent}(\text{par}(x), x) \right) \land$$

$$\left( \neg \text{person}(x) \lor \neg \text{mother}(\text{par}(x), x) \lor \neg \text{father}(\text{par}(x), x) \right)$$
CNF Representation (cont’d)

Rewrite as a set of clauses (implicitly connected by $\land$).

For readability, use the principle $\neg a \lor b \equiv a \rightarrow b$

$$
\begin{align*}
\text{person}(x) & \rightarrow \text{parent}(\text{par}(x), x) \\
\text{person}(x) \land \text{mother}(\text{par}(x), x) \land \text{father}(\text{par}(x), x) & \rightarrow \bot
\end{align*}
$$

$\bot$ is logical contradiction (‘false’).
Positive and Negative Literals

For any clause $C$ with literals $l_i \ (1 \leq i \leq j+k)$

$$C = \neg l_1 \lor \neg l_2 \lor \ldots \lor \neg l_j \lor l_{j+1} \lor l_{j+2} \lor \ldots \lor l_{j+k}$$

$$= l_1 \land l_2 \land \ldots \land l_j \rightarrow l_{j+1} \lor l_{j+2} \lor \ldots \lor l_{j+k}$$

denote

- $C^- = \{l_1, \ldots l_j\}$ (the set of atoms appearing as negative literals in $C$)
- $C^+ = \{l_{j+1}, \ldots l_{j+k}\}$ (those appearing as positive literals in $C$)
Substitution

A substitution is a mapping from a set of variables to terms.

Example

\[ \vartheta = \{ x \mapsto x, y \mapsto x, z \mapsto f(x, a) \} \]

Example of applying a substitution:

\[ C = p(w, x, y) \rightarrow q(z, a) \]

\[ C\vartheta = p(w, x, x) \rightarrow q(f(x, a), a) \]

\[ C^{-\vartheta} = \{ p(w, x, x) \} \]

A ground formula (or a set of atoms) is a formula (set of atoms) containing no variables.

A grounding substitution for a formula (or a set of atoms) \( \phi \) is a substitution \( \vartheta \) such that \( \phi\vartheta \) is ground.
Herbrand Interpretation and Model

Generalization of *truth-value assignment* in propositional logic.

Let $P$ be a finite set of predicates and $F$ be finite set of functions (including constants).

*Herbrand base* $\mathcal{H}$: set of all atoms that can be constructed with $P$ and $F$ (may be infinite)

*Herbrand interpretation* $I$: a subset of $\mathcal{H}$. (We will omit ‘Herbrand’.)

A clause $C$ is *true* in $I$ if for any grounding substitution $\vartheta$ for $C$

$$C^+ \vartheta \cap I \neq \{\} \text{ whenever } C^- \vartheta \subseteq I$$

An interpretation $I$ is *model* of a clausal theory $T$ if all clauses of $T$ are true in $I$; we write

$$I \models T$$
Interpretations and Models: Examples

Consider the clausal theory:

\[
\begin{align*}
\text{edge}(x, y) & \rightarrow \text{path}(x, y) \\
\text{edge}(x, z) \land \text{path}(z, y) & \rightarrow \text{path}(x, y)
\end{align*}
\]

The following interpretations are two of its models:

Herbrand:
\[
\{\text{edge}(a, b), \text{edge}(b, c), \text{path}(a, b), \text{path}(b, c), \text{path}(a, c)\}
\]

Non-Herbrand (informal):
\[
\begin{align*}
\{&\text{edge}(a, b), \text{edge}(b, c), \text{edge}(c, a) \\
&\text{path}(a, a), \text{path}(a, b), \text{path}(a, c), \\
&\text{path}(b, a), \text{path}(b, b), \text{path}(b, c) \\
&\text{path}(c, a), \text{path}(c, b), \text{path}(c, c)\}
\end{align*}
\]
Range-Restricted $jk$-theories

Clause $C$ is

- *range-restricted* if all variables in $C^+$ appear also in $C^-$.
- a *$jk$-clause* if it contains at most $k$ literals and each of them contains at most $j$ occurrences of predicate, variable and function symbols.

Example:

\[
\text{son}(x, y) \land \text{brother}(y, z) \rightarrow \text{uncle}(z, x)
\]

is a range-restricted $3,3$-clause, but not e.g. a range-restricted $3,2$-clause.

A range-restricted $jk$-theory is a set of range-restricted $jk$-clauses.
Learning Range-Restricted $jk$-Theories

Given finite sets $P, F$ of predicate/function symbols, observation space $X$ contains finite interpretations $x$ on the Herbrand base constructed with $P$ and $F$. The size of examples is given by the triple

$$|P|, |F|, n = \max\{|x| \mid x \in X\}$$

Hypothesis space $\mathcal{F}^{jk} =$

$$\{f_T \mid T \text{ is a range restricted } jk\text{-theory using only symbols from } P \text{ and } F\}$$

$$f_T(x) = 1 \text{ iff } x \models T$$

To see if $\mathcal{F}^{jk}$ is efficiently PAC-learnable, we will determine if

- $\ln |\mathcal{F}^{jk}|$ is polynomial.
- A consistent $f_T$ can be produced for any sample $S = \{x_1, \ldots, x_m\}$ in polynomial time.

Polynomial: in $1/\delta$, $1/\epsilon$, $|P|$, $|F|$, $n$
Cardinality of $\mathcal{F}^{jk}$

With $l$ different literals

$$c = \sum_{i=1}^{k} \binom{l}{i} = O(l^k)$$

different clauses containing at most $k$ literals can be constructed.

Atoms can be constructed with $|P|$ different predicate symbols.

With maximum atom size $j$, an atom has at most $j - 1$ places for function or variable symbols (exactly one place occupied by the predicate symbol).

There are $|F|$ different function symbols. Since any $jk$-theory contains at most $jk$ variables, there are $jk$ different variable symbols.

So $|P|(|F| + jk)^{j-1}$ different atoms, i.e. $l = 2|P|(|F| + jk)^{j-1}$ different literals can be used to form clauses.

Therefore $c$ is polynomial in $|P|$ and $|F|$ and so is $\ln |\mathcal{F}^{jk}| = \ln |2^n|$. 

Consistent $f_T$

A $f_T$ consistent with a sample $S$ is produced by a generalization algorithm.

$$
\phi = \land_{i=1}^{c'} C_i \quad \{\text{conjunction of all range-restricted } jk\text{-clauses formed using } P, F \text{ and } jk \text{ different variables}\}
$$

for each example $(x, 1) \in S$ do

for $i = 1 \ldots c'$ do

if $x \not\models C_i$ then

delete $C_i$ from $\phi$

return $\phi$

Number $c$ from the previous slide considered all $jk$-clauses, not only those range-restricted, so $c' \leq c$.

Similar to the generalization algorithm for learning propositional conjunctions. Also here only positive examples (interpretations) are used.
Efficiency of the Generalization Algorithm

The algorithm makes $mc'$ steps ($m = |S|$) where $c'$ is polynomial, i.e. polynomial number of steps.

It needs polynomial time if each step takes polynomial time, i.e. if checking $x \models C$ for interpretation $x$ and a range-restricted $jk$-clause $C$ takes polynomial time.

Checking $x \models C_i$ requires

1. finding all substitutions $\vartheta$ so that $C^-\vartheta \subseteq x$
2. checking if $C^+\vartheta \cap x \neq \{\}$ for each such $\vartheta$
Finding \( \vartheta \) so that \( C^- \vartheta \subseteq x \)

A tree-search

Example: \( C^- = \{p(x,y), q(y,z)\} \), \( x = \{p(a,b), p(b,c), q(b,c), q(b,d)\} \)

\[
\begin{align*}
\vartheta_1 &= \{x \mapsto a, y \mapsto b\} \\
q(y,z)\vartheta_1 &= q(b,z) \\
\vartheta_{11} &= \{z \mapsto c\} \\
q(b,z)\vartheta_{11} &= q(b,c) \\
\vartheta_2 &= \{x \mapsto b, y \mapsto c\} \\
q(y,z)\vartheta_2 &= q(c,z) \\
\vartheta_{12} &= \{z \mapsto d\} \\
q(b,z)\vartheta_{12} &= q(b,d)
\end{align*}
\]

Solutions:

\[
\begin{align*}
\vartheta &= \vartheta_1 \cup \vartheta_{11} = \{x \mapsto a, y \mapsto b, z \mapsto c\} \\
\vartheta &= \vartheta_1 \cup \vartheta_{11} = \{x \mapsto a, y \mapsto b, z \mapsto d\}
\end{align*}
\]
PAC-learnability of $\mathcal{F}^{jk}$

The tree has depth at most $k$ and branching factor at most $n$ (maximum size of interpretation $x$).

Thus it has at most $n^k$ vertices.

The atom in each vertex has at most $j$ arguments so the tree can be searched in $O((jn)^k)$ units of time.

For each resulting $\vartheta$, $C^+ \vartheta \cap x \neq \emptyset$ can be checked in $O(jn)$ units of time since $C$ is range-restricted and thus $C^+ \vartheta$ is ground.

Therefore, checking $x \models C_i$ takes polynomial time.

In summary, the generalization algorithm runs in time polynomial in $|P|, |F|, n$.

Since also $\ln |\mathcal{F}^{jk}|$ is polynomial in $|P|$ and $|F|$, $\mathcal{F}^{jk}$ is efficiently PAC-learnable.
Expressivness of $\mathcal{F}^{jk}$

$\mathcal{F}^{1k} = \mathcal{F}^{k}\text{-CNF}$ (propositional $k$-CNF), but for $j > 1$, $\mathcal{F}^{jk}$ is much more expressive. For example, the concept of a directed path in a graph can be (recursively) expressed in $\mathcal{F}^{3,3}$:

$$
\begin{align*}
\text{edge}(x,y) & \rightarrow \text{path}(x,y) \\
\text{edge}(x,z) \land \text{path}(z,y) & \rightarrow \text{path}(x,y)
\end{align*}
$$

the concept be efficiently PAC-learned from example interpretations such as

$$
\{\text{edge}(a,b), \text{edge}(b,c), \text{path}(a,b), \text{path}(b,c), \text{path}(a,c)\}
$$

and

$$
\{\text{edge}(a,b), \text{edge}(b,c), \text{edge}(c,a), \text{path}(a,a), \text{path}(a,b), \text{path}(a,c), \text{path}(b,a), \text{path}(b,b), \text{path}(b,c), \text{path}(c,a), \text{path}(c,b), \text{path}(c,c)\}
$$
Inductive Logic Programming

The concept can be written as a Prolog program:

\[
\begin{align*}
\text{path}(X,Y) & : \text{edge}(X,Y).
\text{path}(X,Y) & : \text{edge}(X,Z), \text{path}(Z,Y).
\end{align*}
\]

Thus algorithms for learning in predicate logic allow to *induce* (= learn) some Prolog *programs* from examples.

Therefore they are called algorithms of *inductive logic programming (ILP)*. ILP has applications in problems where hypotheses are learned from structured data such as in biochemistry.
Learning from Interpretations

The learning principles explained so far are referred to as the ILP setting of learning from (finite) interpretations.

Some concepts have only infinite models. E.g.

\[
\begin{align*}
\text{even}(s(s(0))) \\
\text{even}(x) & \rightarrow \text{even}(s(s(x)))
\end{align*}
\]

(the unary function s produces the successor of its argument)

so they cannot be learned from finite interpretations.
Learning from Clauses

An alternative setting of ILP is known as learning from clauses or learning from entailment.

Here $x \in X$ are first-order predicate clauses and $\mathcal{F}$ contains hypotheses $f_T$ in the form of first-order clausal theories $T$ such that

$$f_T(x) = 1 \text{ iff } T \vdash x$$

where the $\vdash$ is the *logical entailment* relation. $T \vdash x$ holds iff all models of $T$ are also models of $x$. For example:

$$\text{human}(\text{Sokrates})$$
$$\text{human}(x) \rightarrow \text{mortal}(x) \vdash \text{mortal}(\text{Sokrates})$$

Samples $S$ from $X$ are assumed non-contradictory, i.e. no positive (negative) example entails a negative (positive) example.
Learning from Clauses: Example

Consider sample $S$ containing the following positive examples

\[
\begin{align*}
x_1 & = \text{even}(s(s(s(s(s(0)))))) \\
x_2 & = \text{even}(s(s(x))) \rightarrow \text{even}(s(s(s(s(x)))))
\end{align*}
\]

and the negative example

\[
x_3 = \text{even}(s(s(x)))
\]

Then the intended hypothesis $f_T$

\[
\begin{align*}
\text{even}(s(s(0))) \\
\text{even}(x) \rightarrow \text{even}(s(s(x)))
\end{align*}
\]

is consistent with $S$ ($T \vdash x_1$, $T \vdash x_2$, $T \nvdash x_3$), which can be shown e.g. by resolution. So can we learn from clauses what we cannot learn from interpretations?
Learning from Clauses: Problems

Main problems causing the inability to PAC-learn general clausal theories from clauses.

1. **Infinite VC-dimension.** General clausal theories have the expressive power of a Turing machine. Any (finite) sample $S$ can be shattered e.g. by the trivial theory

$$S = \bigwedge_i x_i$$

where $x_i$ are clauses taken from the positive examples of $S$. This problem can be avoided by making $\mathcal{F}$ finite, for example $\mathcal{F} = \mathcal{F}^{jk}$.

2. **Undecidability of entailment**

$$T \vdash x$$

is generally undecidable (even if $T$ is a single clause)

This makes it impossible to verify if a hypothesis is consistent with a sample, therefore finding a consistent hypothesis is also generally impossible.
$\theta$-Subsumption

If $T$ is a single clause, $T \vdash x$ can be approximated by the $\theta$-subsumption relation $\preceq_\theta$.

$T \preceq_\theta x$ holds iff there is a substitution $\theta$ such that all literals of $T\theta$ are also in $x$.

Checking $\theta$-subsumption decidable but $\text{NP-complete}$. Maximum size of the two clauses must be fixed for efficient checking.

Example:

\[ T = \text{path}(x, z) \land \text{edge}(z, y) \rightarrow \text{path}(x, y) \]
\[ x = \text{path}(x, a) \land \text{edge}(a, y) \land \text{edge}(b, y) \rightarrow \text{path}(x, y) \]

Here both $T \vdash x$ and $T \preceq_\theta x$ (for $\theta = \{z \mapsto a\}$)
$\vartheta$-Subsumption (cont’d)

However, in:

\[
\begin{align*}
T &= p(x, y, z) \rightarrow p(y, z, x) \\
x &= p(x, y, z) \rightarrow p(z, x, y)
\end{align*}
\]

$T \vdash x$ (can be checked by resolving $T$ with $T$) but $T \not\leq_{\vartheta} x$.

$\vartheta$-Subsumption is an approximation to entailment in that

- $T \leq_{\vartheta} x$ implies $T \vdash x$
- $T \vdash x$ implies $T \leq_{\vartheta} x$ only if $T$ and $x$ are not self-resolving