CMT311 Principles of Machine Learning

Computational Complexity of Learning

Angelika Kimmig
KimmigA@cardiff.ac.uk

01.11.2019

So far…

- Boolean concept learning
- More-general-than & version spaces
- ERM learning
- Sample complexity
- PAC-learnability & agnostic PAC-learnability
- Fundamental theorem of statistical learning

ERM Learning

with randomly labeled examples

the set of objects we want to label

domain set X

{0,1}

label set Y

the possible hypotheses

hypothesis space H

data-generation model

a probability distribution D over $X \times Y$

the learner does not know D

training data S

a finite sequence of objects from X labeled with elements of Y

ERM learner

hypothesis $h_S \in \arg\min_{h \in H} L_S(h)$

i.i.d. assumption, $S \sim D^m$: S contains m examples that are independently and identically distributed according to D

ERM Learning

- Fundamental theorem: if a class is learnable, any ERM learner will do the job (if given **sufficient data**)

sample complexity

- How to build ERM learners?

- Can we build **efficient** ERM learners?

computational complexity

Background: Computational Complexity of Algorithmic Tasks

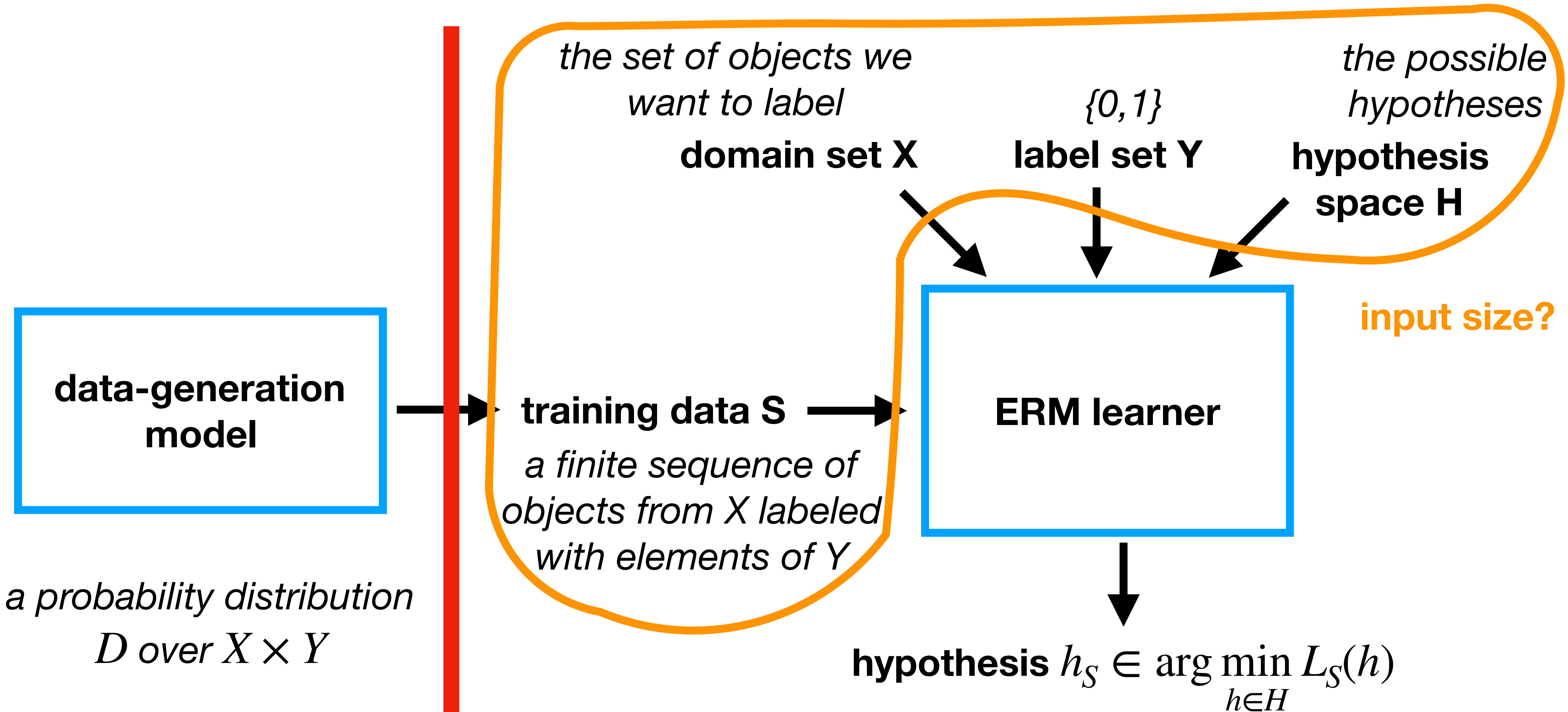
- Computational complexity studies how the **time** an algorithm takes to run depends on the **size of the input**
- Actual time in seconds depends on specific machine
- Asymptotic analysis ignores constant factors: for a given algorithm A, find a function f(n) such that there are constants n_0 and c such that for all inputs of size $n>n_0$, the actual runtime is at most $c*f(n)$
- "A is O(f(n))"
- Algorithms where f(n) is polynomial are generally considered efficient

Examples

- Checking whether a given element appears in a given list by going over the elements one by one
  - input size n = length of the list
  - O(n)
- Sorting a list of length n using bubblesort is $O(n^2)$

ERM Learning

with randomly labeled examples



the learner does not know D

i.i.d. assumption, $S \sim D^m$: S contains m examples that are independently and identically distributed according to D

Input size for ERM

- number of training examples?
  - no: can ignore any examples beyond sample complexity
- key parameters:
  - target accuracy ϵ
  - desired confidence δ of reaching that accuracy
  - dimensionality of domain set X (size of an example)
  - measure of complexity of hypothesis space H
- Only allow classes H where making a prediction for a new example using some $h \in H$ does not take longer than learning

finite hypothesis classes: ERM by exhaustive search

- hyp = none; loss = 1;
- for h in H
  - compute $L_S(h) = \dfrac{\left| \{i \in \{1,\ldots,m\} \mid h(x_i) \neq y_i\} \right|}{m}$
  - if $L_S(h)$<loss then loss = $L_S(h)$; hyp = h; endif
- return hyp

Finite Classes

- Assume we have the minimum number of examples needed to PAC-learn the class, i.e., the smallest m with $m \geq \frac{\log(|H|/\delta)}{\epsilon}$
- ERM by exhaustive search is $O(|H|\frac{\log(|H|/\delta)}{\epsilon})$
- Can get expensive: e.g., if the size of H is exponential in the length of an example

A basic learner: FIND-S

- set *h* to the most specific hypothesis in *H*
- for each positive *x* in *D*
  - for each constraint *a* in *h*
    - if *x* does not satisfy *a* then replace *a* in *h* by the next more general constraint *a'* that is satisfied by *x*
- return *h*

Rectangles

- Consider the class of axis aligned rectangles in $\mathbb{R}^n$, where each hypothesis is described by parameters $(a_1, \ldots, a_n, b_1, \ldots, b_n)$ and a point $(x_1, \ldots, x_n)$ is labeled 1 if for all i, $a_i \leq x_i \leq b_i$, and 0 otherwise

- n measures the complexity of the class

- FIND-S PAC-learns this class in time O(mn), where m is the sample size

- Cannot be efficiently learned in the agnostic setting (unless P=NP)

Boolean conjunctions

- For domain set $X = \{0,1\}^n$, a Boolean conjunction is a formula of the form $x_{i_1} \wedge \cdots \wedge x_{i_k} \wedge \neg x_{j_1} \wedge \cdots \wedge \neg x_{j_r}$ for some indices $i_1, \ldots, i_k, j_1, \ldots j_r$ between 1 and n

- Such a conjunction labels an example 1 if $x_{i_1} = \ldots = x_{i_k} = 1$ and $x_{j_1} = \ldots = x_{j_r} = 0$, and 0 otherwise

- Size of class is at most $1 + 3^n$, sample complexity is linear in n

- FIND-S PAC-learns this class in time O(mn), where m is the sample size

- Cannot be efficiently learned in the agnostic setting (unless P=NP)

3-Term DNF

- A 3-term DNF is a formula of the form $h(x) = A_1(x) \vee A_2(x) \vee A_3(x)$ where each $A_i(x)$ is a Boolean conjunction over X

- h(x)=1 if at least one of the three conjunctions is 1, and 0 if all three are 0

- Size of class is at most 3^{3n}, sample complexity is linear in n

- Learning such a 3-term DNF is hard even in the realisable case

- But: the class of functions represented by 3-term DNF can be learned efficiently if we use a different representation

3-Term DNF

- Each 3-term DNF over variables $x_1, \ldots, x_n$ can be written as a Boolean conjunction over variables $z_1, \ldots, z_{(2n)^3}$
  - $$A_1 \vee A_2 \vee A_3 = \bigwedge_{u \in A_1, v \in A_2, w \in A_3} (u \vee v \vee w)$$
  - There are $(2n)^3$ triplets of literals over variables $x_1, \ldots, x_n$: introduce a variable z_i for each
- Sample complexity of learning conjunctions over the z_i is cubic in n
- Not every such conjunction corresponds to a 3-term DNF, but in many cases, performance is more important than form

Summary

- The statistical learning framework provides a formal view on machine learning:
  - from a **statistical** perspective, any PAC-learnable class can be learned using ERM in both the realisable and agnostic case
  - from a **computational** perspective, the difference between the realisable and agnostic can be huge
- In practice, we often need the agnostic case, and thus have to resort to learning algorithms different from ERM

CMT311 Topics

- Learning Theory
- Logic & Learning
- Probabilistic Graphical Models
- Statistical Relational Learning

Logic & Learning

- Concept Learning
- Rule Learning
- Knowledge in Learning

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
human	warm	yes	no	no	mammal
python	cold	no	no	no	reptile
salmon	cold	no	no	yes	fish
whale	warm	yes	no	yes	mammal
frog	cold	no	no	sometimes	amphibiar
komodo	cold	no	no	no	reptile
bat	warm	yes	yes	no	mammal
pigeon	warm	no	yes	no	bird
cat	warm	yes	no	no	mammal
leopard shark	cold	yes	no	yes	fish
turtle	cold	no	no	sometimes	reptile
penguin	warm	no	no	sometimes	bird
porcupine	warm	yes	no	no	mammal
eel	cold	no	no	yes	fish
salamander	cold	no	no	sometimes	amphibiar
gila monster	cold	no	no	no	reptile
platypus	warm	no	no	no	mammal
owl	warm	no	yes	no	bird
dolphin	warm	yes	no	yes	mammal
eagle	warm	no	yes	no	bird

Name	Blood Type	Give Birth	Can Fly	Live in Water	Class
lemur	warm	yes	no	no	?
turtle	cold	no	no	sometimes	?
dogfish shark	cold	yes	no	yes	?

R1: if (Give Birth = no) & (Can Fly = yes), then bird
R2: if (Give Birth = no) & (Live in Water = yes), then fish
R3: if (Give Birth = yes) & (Blood Type = warm), then mammal
R4: if (Give Birth = no) & (Can Fly = no), then reptile
R5: if (Live in Water = sometimes), then amphibian

Example

days described by
< Sky, AirTemp, Humidity, Wind, Water, Forecast >

Sunny / Cloudy / Rainy

Warm / Cold

Normal / High

Strong / Weak

Warm / Cool

Same / Change

labels 1 / 0 indicating whether someone enjoys their favourite sport on that day

domain set X

label set Y

training data D

<Sunny, Warm, Normal, Strong, Warm, Same> 1
<Sunny, Warm, High, Strong, Warm, Same> 1
<Rainy, Cold, High, Strong, Warm, Change> 0
<Sunny, Warm, High, Strong, Cool, Change> 1

learner

hypothesis h

conjunction of attribute constraints using ?, -, specific values

Rule Learning

- Each hypothesis in the sports example can be seen as a conjunctive rule describing exactly the positive class:

<?,?,Normal,?,Warm,Same>

if (Humidity = Normal) & (Water=Warm) & (Forecast = Same) **then** 1

<?,?,?,?,?,?>

if TRUE **then** 1

<-,-,-,-,-,->

if FALSE **then** 1

We use TRUE and FALSE as the empty and impossible conjunction, respectively

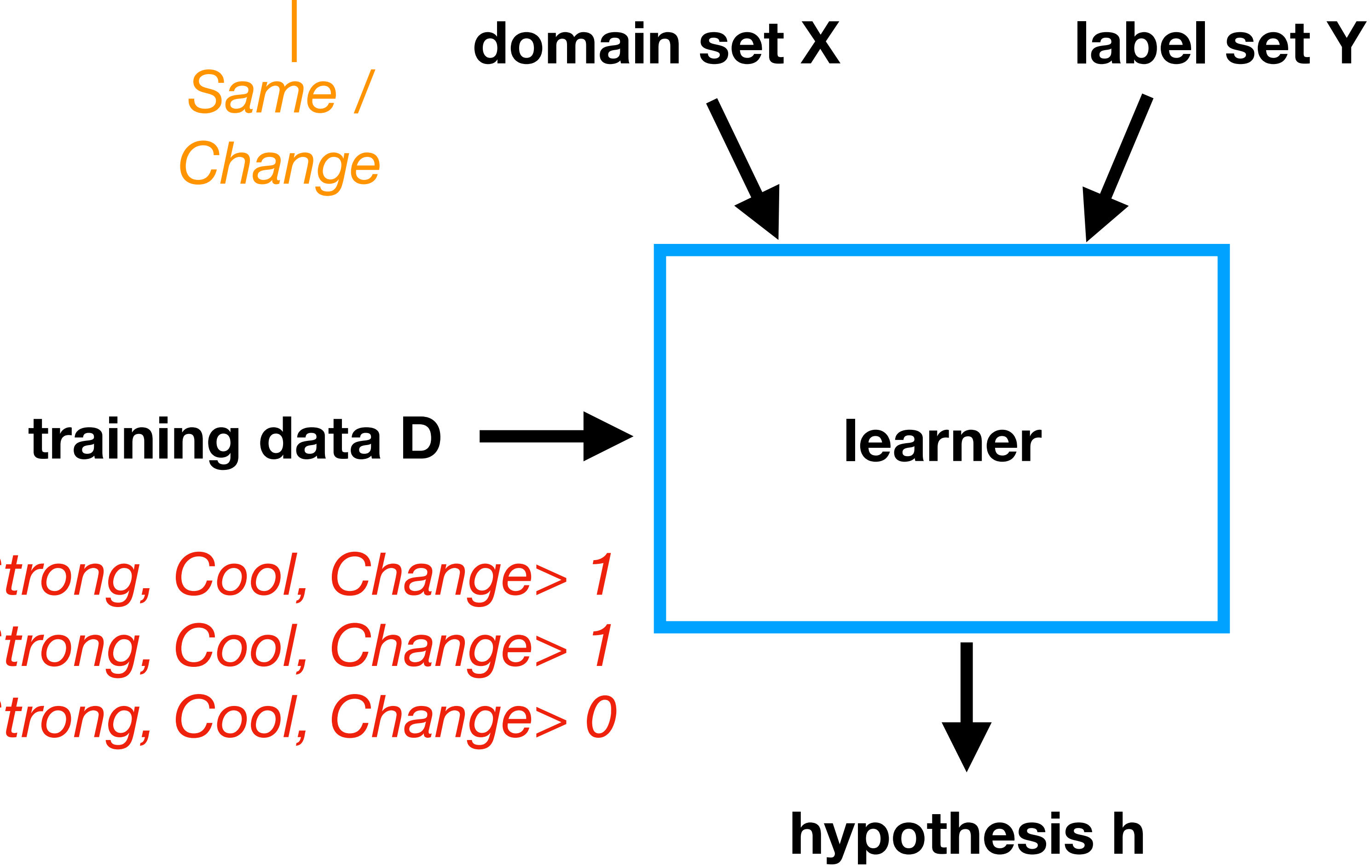
- In realistic settings, one such rule will rarely be enough, and we'd like to learn a set of rules instead

No correct h in H

days described by
< Sky, AirTemp, Humidity, Wind, Water, Forecast >

Sunny / Cloudy / Rainy
Warm / Cold
Normal / High
Strong / Weak
Warm / Cool
Same / Change

labels 1 / 0 indicating whether someone enjoys their favourite sport on that day



<Sunny, Warm, Normal, Strong, Cool, Change> 1
<Cloudy, Warm, Normal, Strong, Cool, Change> 1
<Rainy, Warm, Normal, Strong, Cool, Change> 0

conjunction of attribute constraints

days described by
< Sky, AirTemp, Humidity, Wind, Water, Forecast >

<Sunny, Warm, Normal, Strong, Cool, Change> 1
<Cloudy, Warm, Normal, Strong, Cool, Change> 1
<Rainy, Warm, Normal, Strong, Cool, Change> 0

if (Sky = Sunny) & (AirTemp=Warm) & (Humidity=Normal) & (Wind = Strong) & (Water=Cool) & (Forecast = Change) **then** 1

if (Sky = Cloudy) & (AirTemp=Warm) & (Humidity=Normal) & (Wind = Strong) & (Water=Cool) & (Forecast = Change) **then** 1

More-general-than

- Let h_j and h_k be two Boolean-valued functions defined over X.
- Then h_j is **more general than or equal to** h_k, $h_j \geq_g h_k$, if and only if $\forall x \in X : h_k(x) = 1 \rightarrow h_j(x) = 1$
- h_j is **strictly more general than** h_k, $h_j >_g h_k$, if and only if $h_j \geq_g h_k$ and $h_k \ngeq_g h_j$
- h_j is **more specific than** h_k if and only if h_k is more general than h_j
- note: these notions are **independent** of the target concept

Comparing Conjunctive Rules

- h_j is **more general than or equal to** h_k, $h_j \geq_g h_k$, if and only if $\forall x \in X : h_k(x) = 1 \rightarrow h_j(x) = 1$

- Adding conditions to a conjunctive rule makes it more specific, dropping conditions makes it more general.

if TRUE **then** 1

$\geq_g$

if (Humidity=Normal) & (Wind = Strong) & (Forecast = Change) **then** 1

$\geq_g$

if (Sky = Cloudy) & (AirTemp=Warm) & (Humidity=Normal) & (Wind = Strong) & (Water=Cool) & (Forecast = Change) **then** 1

$\geq_g$

if FALSE **then** 1

Comparing Sets of Conjunctive Rules

- To make a set of conjunctive rules **more general**, we can
  - make one or more **rules** in the set **more general**, or
  - **add** one or more rules to the set
- To make a set of conjunctive rules more **specific**, we can
  - make one or more **rules** in the set **more specific**, or
  - **remove** one or more rules from the set

if TRUE **then** 1
if TRUE **then** 1

= **if** TRUE **then** 1

$\geq_g$

if (Sky = Sunny) **then** 1
if (Sky = Cloudy) **then** 1

$\geq_g$

if (Sky = Sunny) **then** 1
if (Sky = Cloudy) & (AirTemp=Warm) & (Humidity=Normal) & (Wind = Strong) **then** 1

$\geq_g$

if (Sky = Sunny) & (AirTemp=Warm) & (Humidity=Normal) **then** 1
if (Sky = Cloudy) & (AirTemp=Warm) & (Humidity=Normal) & (Wind = Strong) & (Water=Cool) & (Forecast = Change) **then** 1

$\geq_g$

if (Sky = Sunny) & (AirTemp=Warm) & (Humidity=Normal) **then** 1

$\geq_g$

if FALSE **then** 1

CANDIDATE-ELIMINATION

- G = set of maximally general hypotheses in H
- S = set of maximally specific hypotheses in H
- for each training example d
    - if d is positive
        - remove from G any h inconsistent with d
        - for each s in S that is not consistent with d
            - remove s from S
            - add to S all minimal generalisations h of s such that h is consistent with d and some member of G is more general than h
            - remove from S any h that is more general than some h’ in S
    - if d is negative
        - remove from S any h inconsistent with d
        - for each g in G that is not consistent with d
            - remove g from G
            - add to G all minimal specialisations h of g such that h is consistent with d and some member of S is more specific than h
            - remove from G any h that is less general than some h’ in G

Learning Rule Sets

- Given this generality order, we could e.g. use CANDIDATE-ELIMINATION to implement ERM for rule sets
- But remember overfitting…
- A popular solution is to use **sequential covering**: build the set one rule at a time, build each rule one condition at a time
  - Prefers shorter hypotheses

Learning Sets of Rules: Sequential Covering

- Start with the empty rule set
- While there are positive examples in the training data S
  - R = *if TRUE then 1*
  - while R covers some negative example in S
    - add a condition to R such that the extended rule covers some positive example(s) in S

assumes realisability

  - remove all positives covered by R from S
  - add R to the rule set

Choosing literals

- Typically greedy search, using some scoring function measuring how well the extended rule separates positive from negative examples

- e.g., difference between number of positives and negatives covered

< Sky, AirTemp, Humidity, Wind, Water, Forecast >

<Sunny, Warm, Normal, Strong, Warm, Same> 1
<Sunny, Warm, High, Strong, Warm, Same> 1
<Rainy, Cold, High, Strong, Cool, Change> 0
<Sunny, Warm, High, Strong, Cool, Change> 1
<Rainy, Warm, High, Weak, Warm, Same> 1
<Rainy, Cold, High, Strong, Warm, Change> 1
<Sunny, Cold, Normal, Strong, Cool, Change> 0
<Rainy, Warm, Normal, Weak, Warm, Change> 1

Rule Sets

- Rule sets explicitly encode **knowledge** about the domain
- So far, we considered the most basic case:
  - attribute-value pairs in the conjunctive body
  - a single target class to predict
- What about more complex domains?
  - Objects, attributes, relations
  - Auxiliaries

Using Datalog Notation

predicates constants

- Represent information about the domain using **facts:**

  father(bob,ann). father(bob,mary). male(bob). female(ann).
  mother(jane,mary). mother(alice,bob).

- Represent general knowledge about the domain using **rules**:

  parent(X,Y) :- father(X,Y). variables
  parent(X,Y) :- mother(X,Y).
  grandmother(X,Y) :- mother(X,Z), parent(Z,Y).

if conjunction

Using Datalog Notation

- Each predicate corresponds to a Boolean concept over its arguments, e.g.,
  male maps one constant to {0,1}
  father maps pairs of constants to {0,1}

- Facts list cases known to map to 1.

- Rules allow us to derive additional cases that map to 1.

- Rules can be recursive:
  ancestor(X,Y) :- parent(X,Y).
  ancestor(X,Y) :- parent(X,Z), ancestor(Z,Y).

< Sky, AirTemp, Humidity, Wind, Water, Forecast >

<Sunny, Warm, Normal, Strong, Warm, Same> 1
<Sunny, Warm, High, Strong, Warm, Same> 1
<Rainy, Warm, Normal, Weak, Warm, Change> 0

```
sky(ex1,sunny). airTemp(ex1,warm). humidity(ex1,normal).
wind(ex1,strong). water(ex1, warm). forecast(ex1,same).
sky(ex2,sunny). airTemp(ex2,warm). humidity(ex2,high).
wind(ex2,strong). water(ex2, warm). forecast(ex2,same).
sky(ex3,sunny). airTemp(ex3,warm). humidity(ex3,normal).
wind(ex3,strong). water(ex3, warm). forecast(ex3,same).
```

```
enjoy(ex1).
enjoy(ex2).
```

if (Sky=Sunny) & (AirTemp=Warm) **then** 1
if (Sky=Rainy) & (Water=Warm) **then** 1

```
enjoy(X) :- sky(X,sunny), airTemp(X,warm).
enjoy(X) :- sky(X,rainy),water(X,warm).
```

Learning Datalog Rules

- The sequential coverage algorithm can be adapted to learn rules for a single predicate (-> the FOIL algorithm)
- To incrementally build knowledge bases:
  - The first knowledge base consists of just the known facts.
  - Then, iterate:
    - Learn rules for the next predicate based on the current knowledge base.
    - Add these rules to the current knowledge base.
- Learning such rules is also known as **inductive logic programming** (ILP)

FOIL

- Start with the empty rule set
- While there are positive examples target(c1,..,cn) in the training data S
    - R = *target(X1,…,Xn) :- true.*
    - while R covers some negative example in S
        - add a condition to R such that the extended rule covers some positive example(s) in S
    - remove all positives covered by R from S
    - add R to the rule set

FOIL

- Adding conditions to rules:
    - add a condition that reuses at least one variable already in the rule, or the negation of such a condition
    - add equality or inequality conditions between variables in the rule, or between a variable in the rule and a constant

Knowledge base:

father(bob,ann). father(bob,mary). male(bob). female(ann). mother(jane,mary).
mother(alice,bob). mother(alice,eve). female(eve). mother(eve,tom).
…

Training data:
(sibling(ann,mary),1), (sibling(eve,bob),1), (sibling(bob,ann),0), (sibling(eve,eve),0)

sibling(X,Y) :- true.

sibling(X,Y) :- mother(X,Y).

sibling(X,Y) :- female(X).

sibling(X,Y) :- mother(Z,X).

sibling(X,Y) :- mother(Z,X), Y≠Z.

sibling(X,Y) :- mother(Z,X), mother(Z,Y).

sibling(X,Y) :- mother(Z,X), mother(Z,Y), X≠Y.

sibling(X,Y) :- not father(X,Y).

sibling(X,Y) :- X=Y.

...

FOIL

- The knowledge base can contain rules, whose head predicates can be used in new rules to achieve more compact definitions. E.g., with
  parent(X,Y) :- father(X,Y).
  parent(X,Y) :- mother(X,Y).
  in the knowledge base, we could use *parent* to learn *sibling*.

- FOIL can use type information to restrict the search space.

- To deal with noise, FOIL does not grow individual rules beyond a certain limit.

CMT311 Topics

- Learning Theory
- Logic & Learning
- Probabilistic Graphical Models
- Statistical Relational Learning

Reading Material

- Computational complexity of learning:
  chapter 8 of Understanding Machine Learning

- Rule learning:
  (parts of) chapter 19 of Russell & Norvig