super simple learning: 1R

1R is silly but surprisingly good for actual data sets!

```
for each attribute a
  for each value v of a, make rule:
   find most frequent class;
   rule assigns that class to v;
  calculate error rates of the rules;
choose rule set with min error rate
```

attribute	**rules**	**errors**	**total errors**
outlook	sunny --> no	2/5	4/14
	overcast --> yes	0/4	
	rainy --> yes	2/5	
temperature	hot --> no*	2/4	5/14
	mild --> yes	2/6	
	cool --> yes	1/4	
windy	false --> yes	2/8	5/14
	true --> no*	3/6	
humidity	hi --> no	3/7	4/14
	normal --> yes	1/7	

pick either rule set for *outlooook* or for *humidity* (if 2 classes are equally frequent (*), break ties arbitrarily)

another super simple learner: 1NN

given: .6 .2 .3 --> .7; .1 .1 .0 --> .5; .8 .7 .0 --> .1,
what's the output for .0 .1 .1 --> ?

- nearest neighbor is surprisingly good
  - basic idea:
    1. store all examples
    2. for any test case, predict that it has same output as the nearest / most similar example
- nearness is determined by Euclidean or Manhattan or Hamming distance

$$\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \qquad \sum_{i=1}^{n}\left|x_i - y_i\right|$$

k-nearest neighbors: average the output from the k - often 2 to 6 - nearest neighbors

naïve Bayes

- (naïvely) assumes all attributes are equally important and independent
  - count how often each attribute-value pair occurs with each value (*yes, no*) of *play*

outlook			**temp**			**humidity**			**windy**			**play**	
	yes	no		yes	no		yes	no		yes	no	yes	no
sunny	2	3	hot	2	2	hi	3	4	false	6	2	9	5
overcast	4	0	mild	4	2	normal	6	1	true	3	3		
rainy	3	2	cool	3	1								

---

sunny	2/9	3/5	hot	2/9	2/5	hi	3/9	4/5	false	6/9	2/5	9/14	5/14
overcast	4/9	0/5	mild	4/9	2/5	normal	6/9	1/5	true	3/9	3/5		
rainy	3/9	2/5	cool	3/9	1/5								

---

new day:

sunny cool hi true. What about *play*?

naïve Bayes, cont.

treating *sunny cool hi true play?* as equally important and **independent**

likelihood (*yes*) = 2/9 * 3/9 * 3/9 * 3/9 * 9/14 = .0053
likelihood (*no*) = 3/5 * 1/5 * 4/5 * 3/5 * 5/14 = .0206,
i.e. for new day, *no* is 4 times more likely than *yes*.

after normalizing,
p(*yes*|*e*) = .0053/(.0053+.0206) = 20.5%
p(*no*|*e*) = .0206/(.0053+.0206) = 79.5%

based on Bayes' Theorem:
p(*yes*|*e*)= (p(*yes*)*p(sunny|*yes*)*p(cool|*yes*)*p(hi|*yes*)*p(true|*yes*))/ p(e) = 20.5
(denominator p(e) disappears after normalizing…)

- missing values are unproblematic
- again: dependence among attributes (e.g. redundancy) yields poor results

the weather data, numeric

attributes

outlook {sunny, overcast, rainy}
temperature **real**
humidity **real**
windy {TRUE, FALSE}
play {yes, no}

data

sunny,85,85,FALSE,no
sunny,80,90,TRUE,no
overcast,83,86,FALSE,yes
rainy,70,96,FALSE,yes
rainy,68,80,FALSE,yes
rainy,65,70,TRUE,no
overcast,64,65,TRUE,yes
sunny,72,95,FALSE,no
sunny,69,70,FALSE,yes
rainy,75,80,FALSE,yes
sunny,75,70,TRUE,yes
overcast,72,90,TRUE,yes
overcast,81,75,FALSE,yes
rainy,71,91,TRUE,no

naïve Bayes on numeric data

- as before, counts for nominal attributes are normalized into probabilities but
- for each class and each numeric attribute, calculate mean and standard deviation
  - e.g. mean of *temperature* for class *yes* = 73 and its std dev = 6.2
  - assume data to be normally distributed
- prob density function for a normal distribution (mean μ, std dev σ) is

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

so,

$$f(temp = 66 \mid yes) = \frac{1}{\sqrt{2\pi}6.2} e^{-\frac{(66-73)^2}{2*6.2^2}} = 0.034$$

naïve Bayes on numeric data, cont.

given a new day:

sunny, 66, 90, true, **play = ?**

likelihood(*yes*) = 2/9 * .0340 * .0221 * 3/9 * 9/14
= .000036

likelihood(*no*) = 3/5 * .0291 * .0380 * 3/5 * 5/14
= .000136

p(*yes*) = .000036 / (.000036 + .000136) = 20.9%

p(*no*) = .000136 / (.000036 + .000136) = 79.1%

missing values are unproblematic
again: dependence among attributes (e.g. redundancy)
yields poor results

linear models for numeric data

- $x = w_0 + w_1 a_1 + w_2 a_2 + \ldots + w_k a_k$
  - where x is the class, a_i are attribute values, and w_i are weights
  - the weights are to be computed from data

- the predicted value for class x^i of instance i:
  $w_0 a_0^i + w_1 a_1^i + \ldots + w_k a_k^i = \sum_j^k w_j a_j^i$

linear regression chooses w_j to minimize errors over all training examples (x^i is ith instance's actual class)

minimize

$$\sum_{i=1}^{n} \left(x^i - \sum_{j=0}^{k} w_j a_j^i \right)^2$$

do a separate regression for each class, with ouput = 1 for instances in class, else 0. For new instance, pick class for which linear expr has largest value...

inductive learning

- task of induction:
  given a set of examples {x, *f*(x)} of *f*, return a function *h* (hypothesis) that approximates *f*.
- bias
  - a preference for one hypothesis over another, beyond mere consistency with the examples
  - often used bias: (syntactic) simplicity
- performance assessment (supervised)
  1) divide large set of examples into **test** and **training** sets; use training set to generate a hypothesis H; measure % of examples in test set that are correctly classified by H
  2) repeat 1) for different randomly selected training sets of different sizes, to obtain the algorithm's **learning curve** for a domain.

inductive learning, cont.

- in principle, there are infinitely many ways to fit a hypothesis to the given finitely many data.
  - Bias such as simplicity helps choose a hypothesis
  - see Goodman's 'perverse predicates':
    grue $=_{df}$ examined before time t and green or not so examined and blue
  - bleen $=_{df}$ examined before time t and blue or not so examined and green
    green $=_{df}$ examined before t and grue or not so examined and bleen ….
- how can we possibly know that our learning algorithm has produced a theory that will correctly predict future?
- **PAC** learning:

  a **simple** hyp consistent with sufficiently large set of examples is unlikely to be seriously wrong, i.e. is prob approx correct.

- if alg's **learning curve** is a **happy** graph with increasing training set size, the alg has picked up a pattern in the data: prediction quality goes up with size of training set

Occam's razor again

- **prefer simple hyp** because if a simple hyp fits many data, it's unlikely that this is a coincidence ….
  - this is a **bad** argument because
  - prefer hyp with exactly n nodes (or attrib arranged alphabetically etc) because such weird trees are also rare, so if it fits, unlikely to be coincidental ….
- simplicity depends on **internal representation**
  - what's simple for one learner may be complex for another - Razor suggests contradictory choices
- so?
  - organisms evolve
  - evolution can change representations easier than a learning algorithm
  - internal representations will evolve s.t. Occam's Razor is successful (**if** bias of learning algorithm is Razor!!)
  - organisms have evolved (roughly comparable!) **innate quality spaces**, and for these representations, the Razor's suggestions work to reduce mental clutter
  - Goodman's '**entrenchment theory**': projectible predicates are coextensive with predicates involved in past successful projections ….

Duhem's argument

- Theory T is too strong: $T \Vdash$ some false fact
- Theory T is too weak: $T \neg \Vdash$ some true fact

  $\Vdash$ undecidable in general, must be resource-bounded by some total recursive function

- if $T \Vdash$ some false fact, then try to localize <u>the</u> responsible hypothesis…
- Duhem's argument

  scientific theories are irrefutable in isolation.

  If P_1 & P_2 & … $P_n \Vdash O$, and O is false, logic only tells us that **some** P_i must go, but there is no way to find out which one with logic alone. **Any** theory can be saved by making adjustments elsewhere, and there are always several theories involved in deriving an observation statement O! (epistemological wholism) i.e. **there are no crucial experiments to pinpoint hyp responsible for clash**

- for general, incremental learning, some form of anti-Duhem is necessary!

inductive learning, cont.

- paradigm for inductive inference
  - given
    - observational statements, facts F
    - tentative (maybe empty) inductive hypothesis
    - background knowledge BK (assumptions, constraints), incl. preference criterion

    find
    - an inductive assertion H, s.t. H & BK ==> F, i.e. H ➠➠ F, F ➠➠ H, and H is intelligible
- preference criterion
  - pref: <(c1, t1), (c2, t2)...>

    evaluate all H with criterion c1 and retain those that score within threshold t1 from best, **then** apply c2 to remainder ...
  - criteria: # of operators and predicates (syntactic simplicity), cost of evaluating H, cost of measuring predicate values....

inductive generalization

- generalization

  E ::> K {event E falls under concept K} ➠ D ::> K if E ⇒ D

  p & q ⇒ p becomes p & q ::> K ➠ p ::> K (brown and round objs are balls ➠ round objs are balls)

- types of generalizations
  - **dropping conditions**

    ctx & S ::> K ➠ ctx ::> K
  - **adding alternatives**

    ctx1 ::> K ➠ ctx1 **v** ctx2 ::> K

    eg. color = red ➠ color = red **v** blue..
  - **climbing generalization tree**

    shape(P) = triangle ::> K, shape(P) =rectangle ::>K ➠ shape(P) = polygon ::>K
  - **turning constants into variables**

    F(a), F(b),... ➠ ∀v F(v) (for descriptive gen.)

    F(a) & F(b) ::> K ➠ ∃ v F(v) ::> K (acquisition)
  - **turning conjunctions into disjunctions**

    p & q ::> K ➠ p **v** q ::> K

inductive generalization, cont.

- more types of generalizations
  - **extending quantification domain**

    $\forall v\ F(v) ::> K \Rrightarrow \exists\ v\ F(v) ::> K$
  - **inductive resolution**

    $p\ \&\ F1 ::> K, \neg\ p\ \&\ F2 ::> K \Rrightarrow F1 \vee F2 ::> K$

    (if company and good food, eat a lot; if no company and hungry, eat a lot; thus, good food or hungry, eat a lot)
  - extension against; for discriminant descriptions

    $ctx1\ \&\ L = R1 ::> K, ctx2\ \&\ L = R2 ::> \neg\ K$

    $\Rrightarrow\ L \neq R2 ::> K$
  - **constructive induction**: use of descriptors not present in original observations (BK or learned)
    - $ctx\ \&\ F1 ::> K, F1 \Rightarrow F2 \Rrightarrow ctx\ \&\ F2 ::> K$
    - count quantified variables; new predicates: #v in a description satisfying COND
    - count arguments of a predicate; new predicates: #arg in relation satisfying COND
    - ……

inductive learning from examples

- learning from examples with ➠➠ and ➠➠

  G(e | E, m): set of all maximally general expressions that cover event e and do not cover negative examples in E, limited to the m best descriptions according to preference criterion.

1. randomly select event e from **pos.**
2. generate G(e|**neg**,m), using generalization rules, heuristics, previously learned concepts, etc..
3. in G(e|**neg**,m), find most preferred description D.
4. if D covers **pos** completely, go to 6.
5. reduce **pos** to contain only events not covered by D;
   go to 1.
6. optionally apply contraction rules; disjunction of all descriptions D is a complete and consistent concept description.

star-learning ...

1. put individual selectors from **e** onto list PS (partial star, may cover some elements in neg). Single selectors from **e** are generalized via dropping condition rule, etc. Order elements via pref1 = <(-negcover, t1), (poscover, t2)> (where negcover = # of neg examples covered by expression in PS. pref1 minimizes negcover, max poscover!)

2. expand PS: constructive induction, heuristics, generaliz, etc

3. insert each new selector into PS in proper place according to pref1; remove from PS all but most preferred m selectors.

4. test descriptions in PS for consistency and completeness:

consistent: negcover = 0; complete: poscover = # pos

Remove cons and complete descriptions from PS and put on **solutions**; stop when size of **solutions** = parameter **sol**.

Remove cons and incomplete descriptions from PS and put on **consistent**. If size of **consistent** > parameter **cons** go to 6.

5. specialize each description in PS by appending a single selector from original PS. Appended selectors must have lower preference than last selector in conjunction. Then rank via pref1 and keep m.

repeat 4 and 5 while size of **consistent** $\leq$ parameter **cons.**

6. generalize each expr on **consistent**

7. rank generalizations with global **pref** defined in BK. To get discriminant description, max **poscover** and max **simplicity**.

generic rule induction

create initial rule set

while rules are not good enough do

generate new rules from old ones or modify/reorder existing ones

specific to general or vv or both? Operators? Heuristics? BGK?

evaluate rules on training data

preference criteria; go for pos coverage, min neg coverage; simplicity...

eliminate low-scoring rules

keep 1 or more rules in each cycle?

select new training instances…

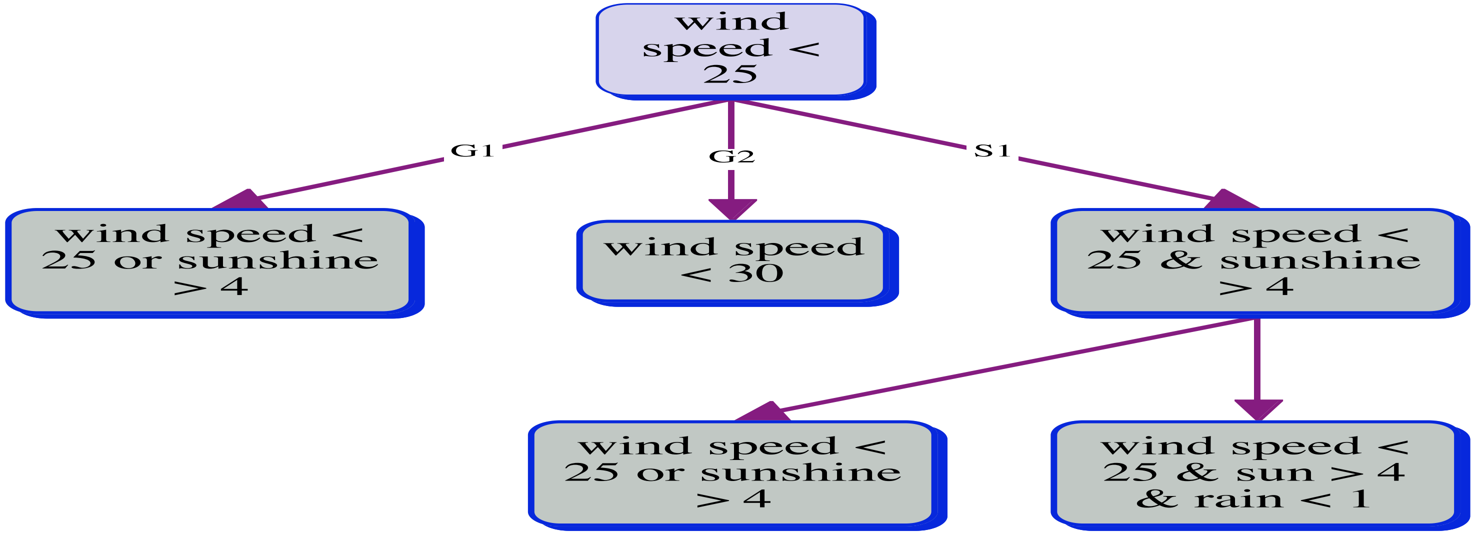
endwhile

typical rule induction ...

- learning = searching thru space of all possible descriptions / hypotheses, for **useful** descriptions
- given DB of weather records and indication whether next day was fine:
  - day1: rainfall 0mm, sunshine 10.8 hr, windspeed 6km/h, …. nextday fine
  - day2: rainfall 08 mm, sunshine 4 hr, windspeed 10 km/h, …. nextday bad
    etc etc….
    output a rule to predict whether next day is fine…

- start with arbitrary initial description
- apply transformation operators to generate successors
- evaluate successors (% of correct 'predictions')
- best-first search: generate successors only from nodes/rules with high evaluation scores

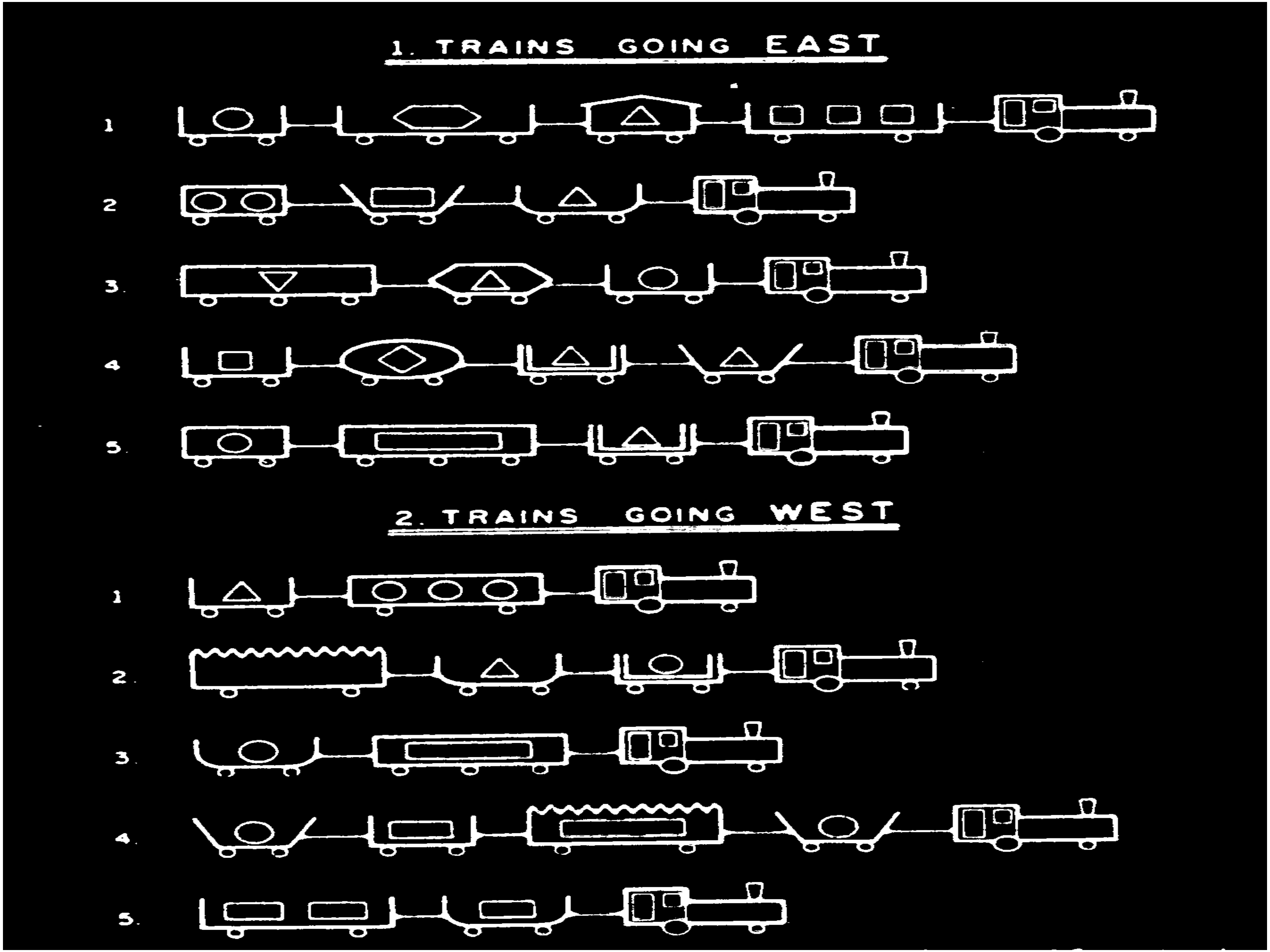
more weather ...



description-transforming operators:

G1: add or-condition
G2: increase <-constant
G3: drop and-condition
G4: change and to or
G5: decrease >-constant
…….? ………

S1: decrease <-constant
S2: add and-condition
S3: change or to and
S4: increase >-constant
S5: drop or condition
…….?……...


1. TRAINS GOING EAST
1
2
3.
4
5.
2. TRAINS GOING WEST
1
2.
3.
4.
5.

trains going east and west

- given predicates
  - *infront, length, carShape (rectangle, Ushaped, ellipse, jaggedTop …), containsLoad, loadShape (circle, triangle, etc.), numberOfPartsInLoad, numberOfWheels*
- create more predicates like
  - numberOfCarsInTrain, positionOfCar, etc.
- then generalize and eliminate inconsistencies, then order by **simplicity**
- **if a train contains a car which is short and has closed top, then east, else west**
- **if a train contains a car whose load is a triangle, and load of car behind is polygon, then east, else west**
- **if there are 2 cars, or if there is a jagged-top car, then west, else east**
- **if there are more than 2 kinds of freight then east, else west**

least (most cowardly) generalization

- look at each example and form the **least generalization** (LG) of the example and the partial concept description discovered so far.

  only examines all pos examples once, ignores neg examples; thus it produces **characteristic** rather than **discriminating** descriptions.

- **LG** is the **dual** of the **most general unifier** of 2 literals

  e.g. LG (p(a,f(c)), p(b,f(b))) = p (X, f(Y))

  if K is {words}, G_1, G_2 are generalizations of K, then G_2 is an LG of K iff

  1) for every word V in K, $G_2 \geq_g V$ ('$\geq_g$' means 'is more general than') and
  2) if for every V in K, $G_1 \geq_g V$, then $G_1 \geq_g G_2$.

$G_1 \geq_g G_2$ iff $G_1\sigma = G_2$ for some substitution instance σ.

e.g. $p(X,X,f(g(Y)) \geq_g p(a(b),a(b),f(g(X)))$: substitute a(b) for X and X for Y.

clause $C_1 \geq_g C_2$, i.e. C_1 subsumes C_2 iff $C_1\sigma \subseteq C_2$ for some substitution instance σ.

e.g. $p(X) \vee p(f()) \geq_g p(f())$: substitute f() for X.

algorithm for least generalization

init with 2 literals V_1, V_2, and empty σ; $V_i := W_i$;

• try to find terms t_1, t_2, $t_1 \neq t_2$, in same place in V_1, V_2, such that either they start with different function letters or at least one of them is a variable;

if there is no such t_i, stop; V_1 is a LG of $\{W_1, W_2\}$, $V_1 = V_2$, and $V_i \sigma_i = W_i$;

choose variable x distinct from any in V_i, and replace occurrences of t_i by x;

update σ_i with $\{t_i \mid x\}$;

repeat from •

$V_1 = p(f(a(),g(Y)),X,g(Y))$, $V_2 = p(h(a(),g(X)),X,g(X))$

$t_1 := Y$; $t_2 := X$; introduce new variable Z

$V_1 = p(f(a(),g(Z)),X,g(Z))$, $V_2 = p(h(a(),g(Z)),X,g(Z))$; $\sigma_1 = \{Y|Z\}$, $\sigma_2 = \{X|Z\}$; repeat;

$t_1 := f(a(),g(Z))$; $t_2 := h(a(),g(Z))$; introduce new variable T

$V_1 = p(T,X,g(Z)) = V_2$; $\sigma_1 = \{f(a(),g(Z))|T\}\{Y|Z\} = \{f(a(),g(Y))|T, Y|Z\}$,

$\sigma_2 = \{h(a(),g(Z))|T\}\{X|Z\} = \{h(a(),g(Y))|T, X|Z\}$; repeat;

stop: $p(T,X,g(Z))$ is LG