Theory of Machine learning

Peter Flach Book pp.124-126, Tom Mitchell, Machine Learning Chapter 7

We seek theory to relate:

- Probability of successful learning
- Number of training examples
- Complexity of hypothesis space
- Accuracy to which target concept is approximated
- Manner in which training examples presented

Two roles for Bayesian methods

Tom Mitchell, Machine Learning Chapter 6

- Provides practical learning algorithm
- Provides conceptual frameworks

  gold standard for evaluationg other learning algorithms

  insight to Occam;s razor

Brute Force MAP Hypothesis Learner

1. For each hypothesis h in H, calculate the posterior probability

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

2. Output the hypothesis h_{MAP} with the highest posterior probability

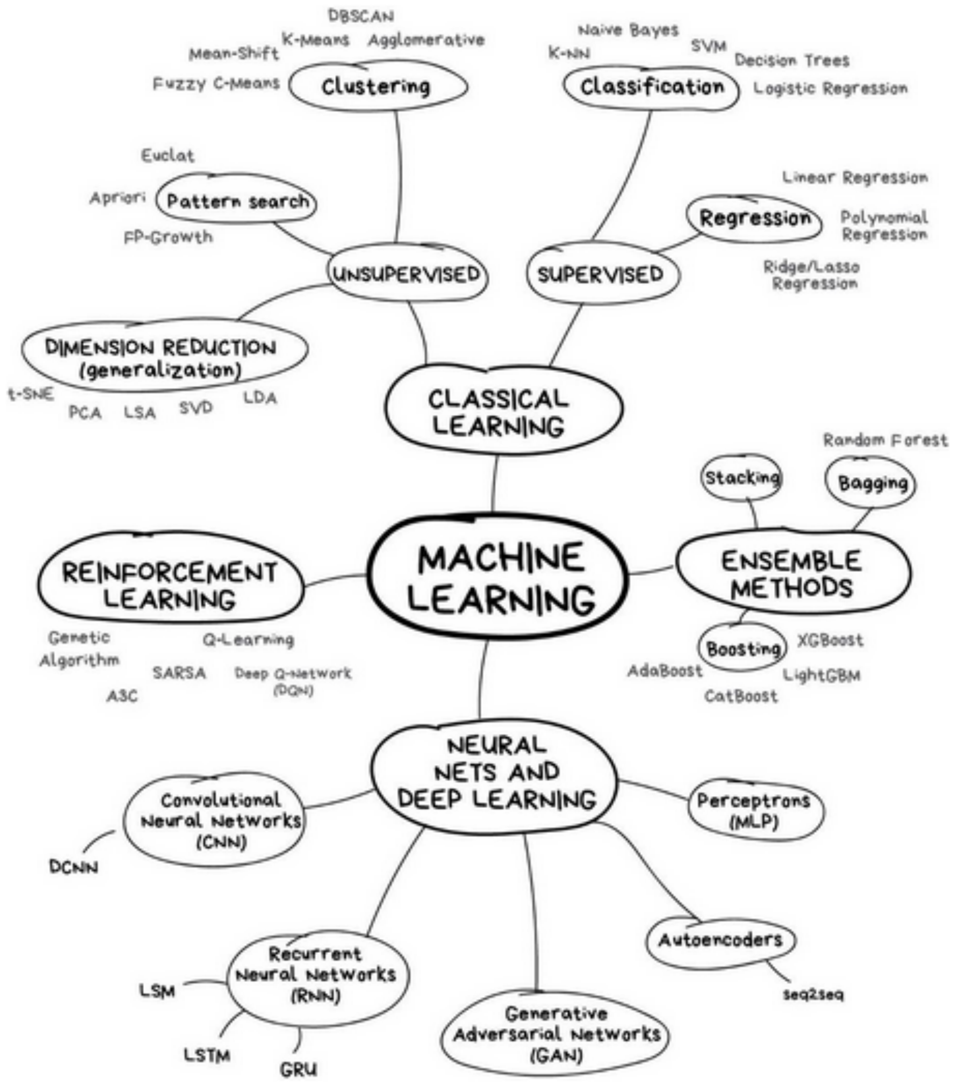
$$h_{MAP} = \underset{h \in H}{\operatorname{argmax}} P(h|D)$$

H … hypotheses
D … learning data
h_{MAP} … maximum a posteriori hypothesis

Bias-variance dilemma

- bias–variance dilemma: a low-complexity model suffers less from variability due to random variations in the training data, but
- may introduce a systematic bias that even large amounts of training data can't resolve;
- Example(s):
- on the other hand,
- a high-complexity model eliminates such bias but can suffer non-systematic errors due to variance.
- Example(s):

What Machine learning is

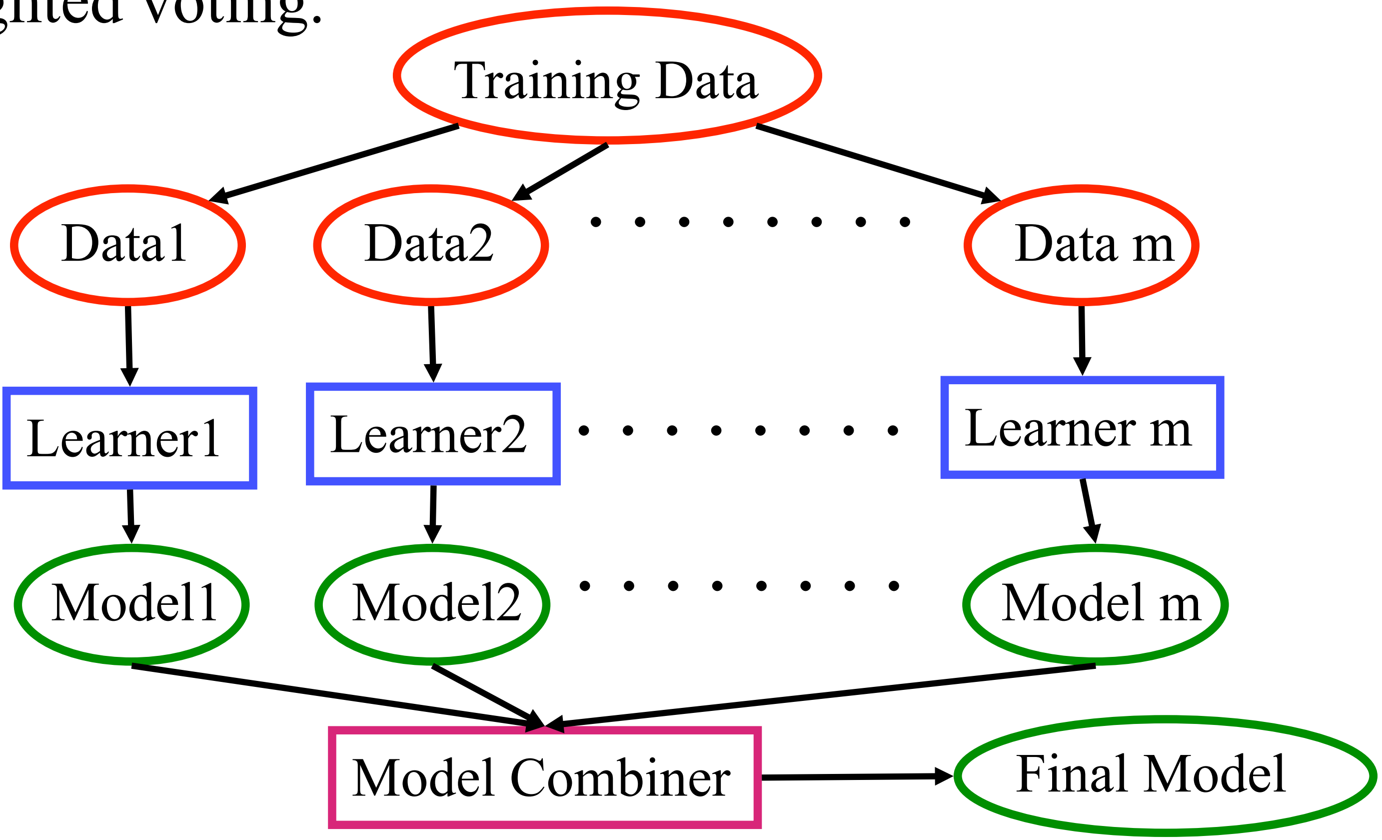


Ensembles

Based on Ray Mooney CS 391L
University of Texas at Austin

Learning Ensembles

- Learn multiple alternative definitions of a concept using different training data or different learning algorithms.
- Combine decisions of multiple definitions, e.g. using weighted voting.



Value of Ensembles

- When combing multiple ***independent*** and ***diverse*** decisions each of which is at least more accurate than random guessing, random errors cancel each other out, correct decisions are reinforced.
- Human ensembles are demonstrably better
  - How many jelly beans in the jar?: Individual estimates vs. group average.
  - Who Wants to be a Millionaire: Expert friend vs. audience vote.

Homogenous Ensembles

- Use a single, arbitrary learning algorithm but manipulate training data to make it learn multiple models.
  - Data1 ≠ Data2 ≠ … ≠ Data m
  - Learner1 = Learner2 = … = Learner m
- Different methods for changing training data:
  - Bagging: Resample training data
  - Boosting: Reweight training data

Bagging

- Create ensembles by repeatedly randomly resampling the training data (Brieman, 1996).
- Given a training set of size *n*, create *m* samples of size *n* by drawing *n* examples from the original data, ***with replacement***.
  - Each ***bootstrap sample*** will on average contain 63.2% of the unique training examples, the rest are replicates.
- Combine the *m* resulting models using simple majority vote.
- Decreases error by decreasing the variance in the results due to ***unstable learners***, algorithms (like decision trees) whose output can change dramatically when the training data is slightly changed.

Bagging : Algorithms

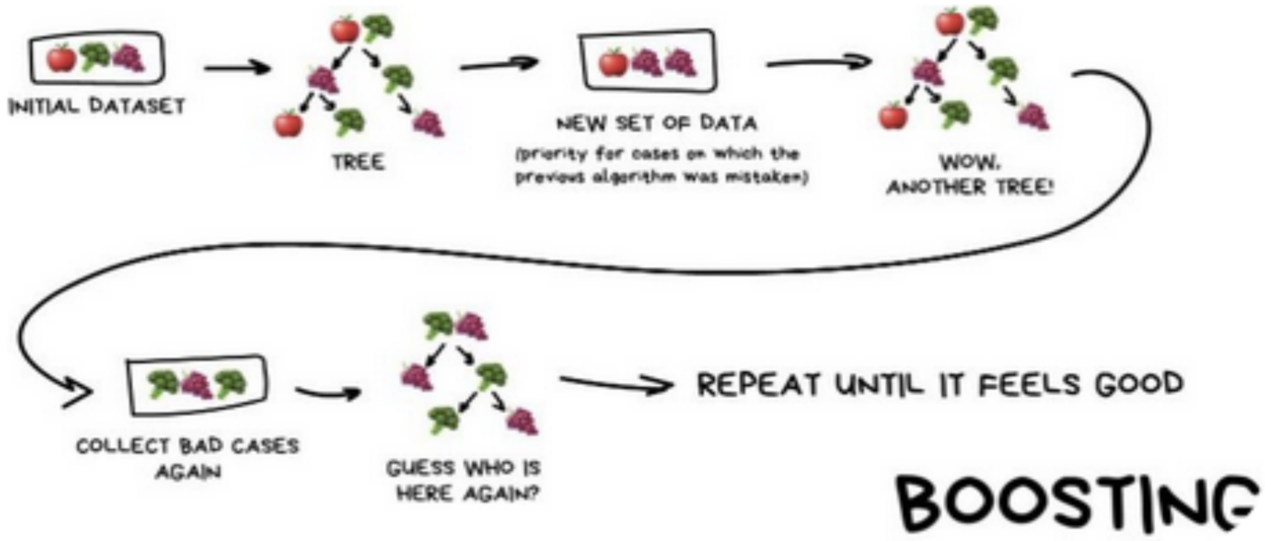
Algorithm Bagging($D, T, \mathscr{A}$) – train an ensemble of models from bootstrap samples.

Input : data set D; ensemble size T; learning algorithm $\mathscr{A}$.

Output : ensemble of models whose predictions are to be combined by voting or averaging.

for $t = 1$ to T **do**
    build a bootstrap sample D_t from D by sampling $|D|$ data points with replacement;
    run $\mathscr{A}$ on D_t to produce a model M_t;
end
return $\{M_t | 1 \leq t \leq T\}$

Boosting



- Originally developed by computational learning theorists to guarantee performance improvements on fitting training data for a ***weak learner*** that only needs to generate a hypothesis with a training accuracy greater than 0.5 (Schapire, 1990; Goedel Prize)

Boosting

- Revised to be a practical algorithm, AdaBoost, for building ensembles that empirically improves generalization performance (Freund & Shapire, 1996).

- Examples are given weights. At each iteration, a new hypothesis is learned and the examples are reweighted to focus the system on examples that the most recently learned classifier got wrong.

Boosting: Basic Algorithm

- General Loop:

  Set all examples to have equal uniform weights.
  For t from 1 to T do:
  Learn a hypothesis, h_t, from the weighted examples
  Decrease the weights of examples h_t classifies correctly
- Base (weak) learner must focus on correctly classifying the most highly weighted examples while strongly avoiding over-fitting.
- During testing, each of the T hypotheses get a weighted vote proportional to their accuracy on the training data.

Note on ensemble construction

- Ensemble construction can be defined as a learning problem
- given the predictions of some base classifiers as features, learn a meta-model that best combines their predictions.
- E.g. in Bagging, what classifiers to use and with what weights (weighted voting)
- In Boosting we could learn the weights rather than deriving them from each base model’s error rate.

Random Forests

- an ensemble of classification or regression random trees.

- each Random tree is constructed by a
  - different bootstrap sample from the original data
  - with a subset of features
- 1/3 of all samples are left out (a cause of bootstrap) – OOB (out of bag) data – for classification error estimation
- majority voting, = a variant of bagging

Ensembles and bias-variance dilemma

- Bagging decreases variance
  variance -> variance/num_of_ensembleMembers

- Boosting decreases bias
  (as hypothesis complexity is increasing)

Rule learning

Based partially on J. Fürnkranz ML course, U. Darmstadt

Example

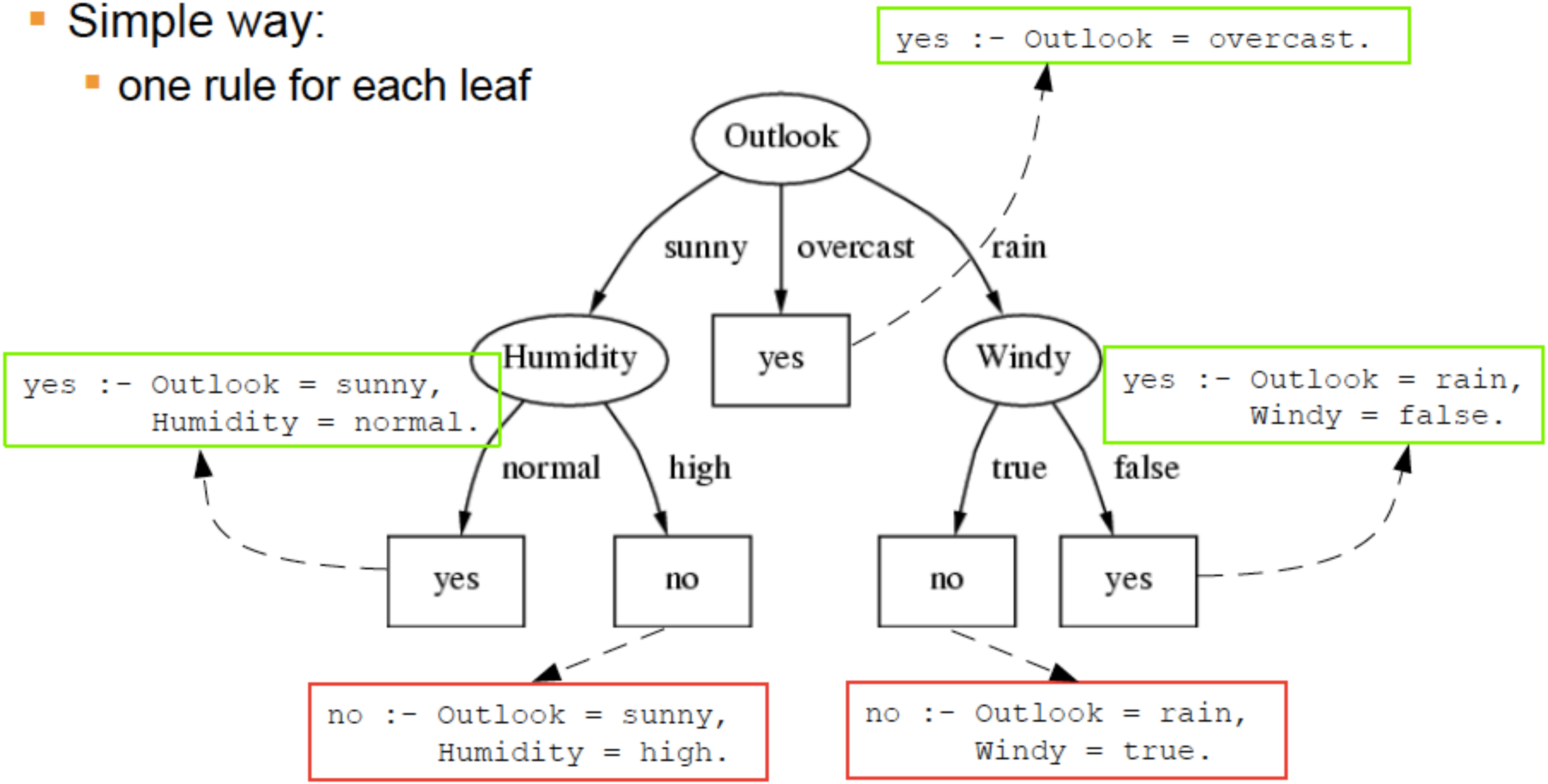
```
@relation weather.symbolic

@attribute outlook {sunny, overcast, rainy}
@attribute temperature {hot, mild, cool}
@attribute humidity {high, normal}
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}

@data
sunny,hot,high,FALSE,no
sunny,hot,high,TRUE,no
overcast,hot,high,FALSE,yes
rainy,mild,high,FALSE,yes
rainy,cool,normal,FALSE,yes
rainy,cool,normal,TRUE,no
overcast,cool,normal,TRUE,yes
…
```

From trees to rules

- Simple way:
  - one rule for each leaf



C4.5rules

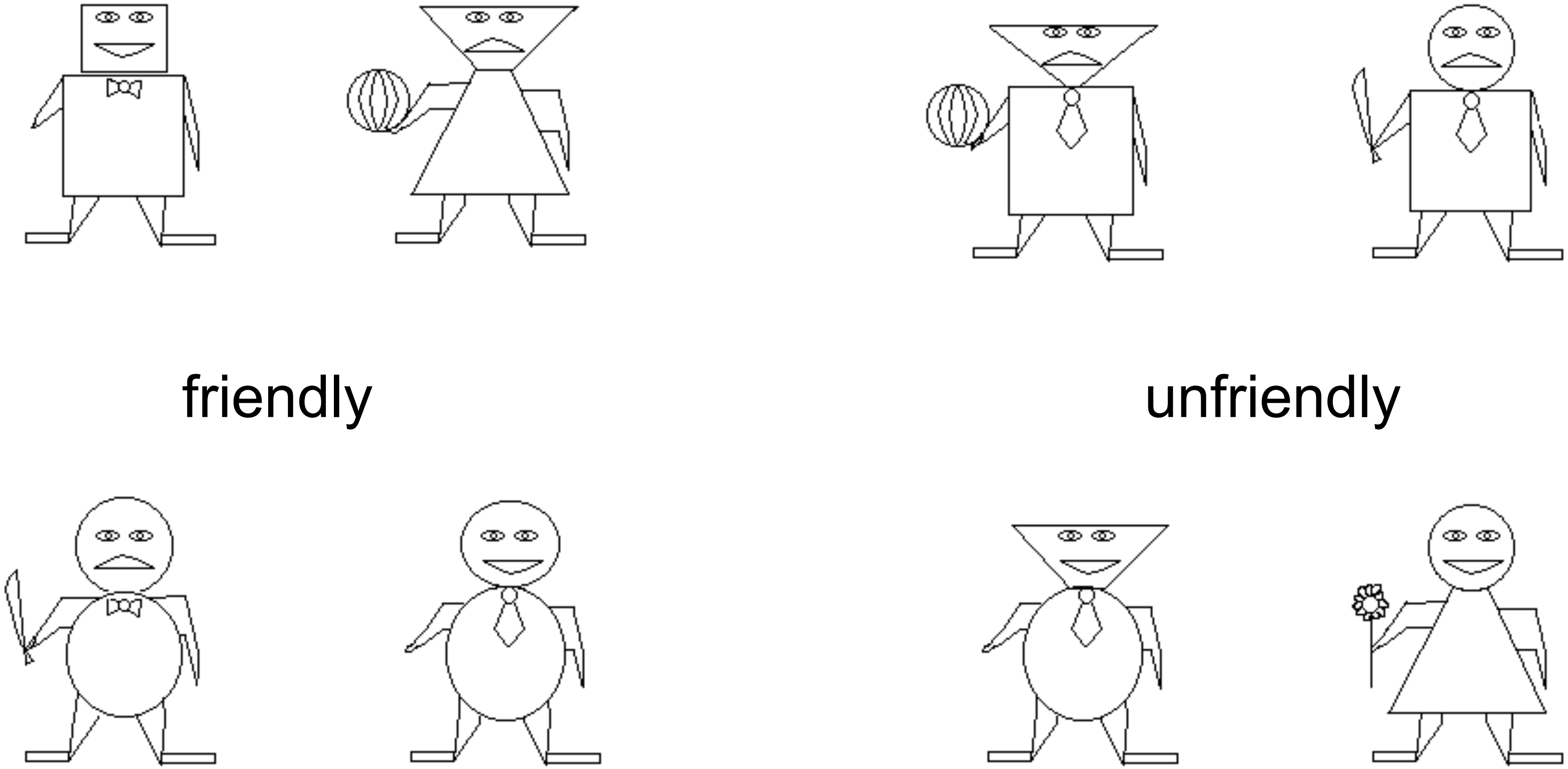
- C4.5rules:
  - greedily prune conditions from each rule if this reduces its estimated error
    - Can produce duplicate rules
    - Check for this at the end
  - Then look at each class in turn
    - consider the rules for that class
    - find a "good" subset (guided by MDL)
    - rank the subsets to avoid conflicts
  - Finally, remove rules (greedily) if this decreases error on the training data

Introduction to Inductive Logic Programming

In collaboration with Olga Štěpánková

Example:

Can we recognize robots after short experience?



Example:
Robots and an atribute-value description

head	smile	neck	body	In hand	friendly
Circle	ne	Tie	Rectangle	Sword	no
Rectangle	ano	Butterfly	Rectangle	Nothing	yes
Circle	ne	Butterfly	Circle	Sword	yes
Triangle	ne	Tie	Rectangle	Ball	no
Circle	ano	Nothing	Triangle	Flower	no
Triangle	ne	Nothing	Triangle	Ball	yes
Triangle	ano	Tie	Circle	Nothing	no
Circle	ano	Tie	Circle	Nothing	yes

Example: hypothesis and testing

In the form of a decision tree

```
if  neck = butterfly   then  yes
         =  nothing   then
                       if  head  = triangle then yes
                                              else no
         = tie         then
                       if  body = rectangle  then  no
                                               else
                                               if head = circle then  yes
                                                                 else  no
```

head	smile	neck	body	in hand	friendly
circle	no	tie	circle	sword	yes
triangle	yes	nothing	rectangle	nothing	yes

Example: hypothesis and testing (cont.)

Using a relation of equality

if neck = body then yes
else no

head	smile	neck	body	in hand	friendly
circle	no	tie	circle	sword	yes
triangle	yes	nothing	rectangle	nothing	no

Both trees classify the learning examples in the same way but they differ on testing set.

When an attribute-value representation is insufficcient?

- Examples do not have a uniform description (e.g. are of a different length)
- A structure of examples is important
- Domain knowledge is (multi-)relational

Inductive logic programming: Basic task

(Muggleton94)

A set of positive E+ and negative E- examples
Domain knowledge B (a logic program)

goal: to find a logic program P that together with B covers
(almost all) positive examples and
not cover (almost no) negative example

+: much more flexible
data of any structure can be processed

-: some effort needed
more time consuming(even though << NeuroN)

Example

Example: find a path in an oriented graph

path(X,Y) :- edge(X,Y).

path(X,Y) :- path(X,U),edge(U,Y).

edge(1,2). edge(1,3). edge(2,3). edge(2,4). …

= domain knowledge

Specialization and generalization

A formula G is a **specialization of** a formula G iff

F is a logical consequence of G

G |= F (any model of G is also a model of F).

Specialization operator (refinement operator)

assigns to a clause a set of all its specializations

Most of ILP systems use two basic operations of specialization

binding two variables

spec(path(X, Y)) = path(X, X)

adding a goal into a clause body

spec(path(X,Y)) = (path(X,Y):-edge(U,V))

and also

substitution a variable with a constant

spec(number(X)) = number(0)

substitution a variable with a most general term

spec(number(X) = number(s(Y)) .

Example: path(From,To) in a graph

Learning set

positive examples : path(1,2). path(1,3). path(1,4). path(2,3).

negative examples: path(2,1). path(2,5).

Domain knowledge

edge(1,2). edge(1,3). edge(2,3). edge(2,4).

Specialization (refinement) tree

path(X,Y).

path(X,X).

path(X,Y) :- edge(Z,U).

path(X,Y) :- edge(X,U).

path(X,Y) :- edge(X,Y).

path(X,Y):-path(Z,U).

path(X,Y):-path(X,U).

…

path(X,Y):-path(X,U),edge(V,W).

path(X,Y):-path(X,U),edge(X,W).

…

path(X,Y):-path(X,U),edge(U,W).

…

path(X,Y):-path(X,U),edge(U,Y).