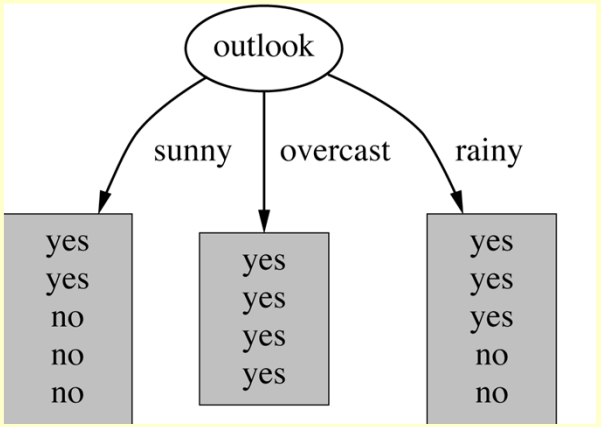
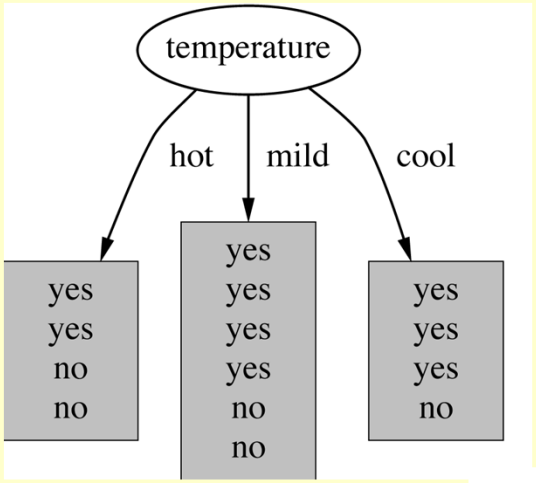
1R and tree stumps

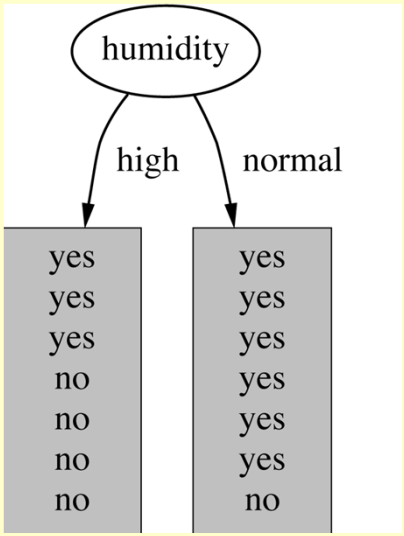
- Extremely simple learning methods that
  - Are always worth trying
  - Tell simple from real problems
  - Are used as components in some more complex learning algorithms
- 1R (one-attribute rule)

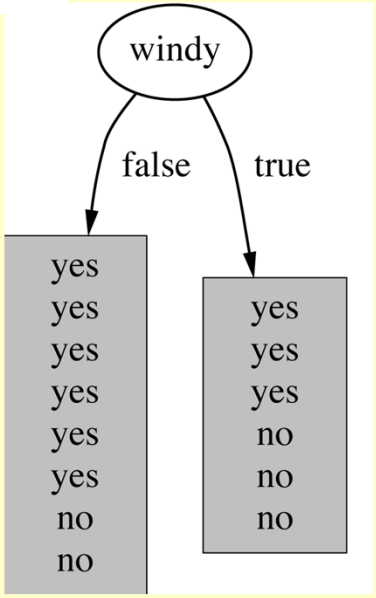
```
For each attribute,
  For each value of that attribute, make a rule as follows:
    count how often each class appears
    find the most frequent class
    make the rule assign that class to this attribute-value.
  Calculate the error rate of the rules.
Choose the rules with the smallest error rate.
```

Tree stumps: can work as 1R









Feature Selection [sec. 7.1 in Witten, Frank]

- Attribute-vector representation: coordinates of the vector are referred to as attributes *or features*
- *'curse* of dimensionality' – learning is search, and the search space increases drastically with the number of attributes
- Theoretical justification: We know from PAC theorems that this increase is exponential – [discuss]
- Practical justification: with divide-and-conquer algorithms the partition sizes decrease and at some point irrelevant attributes may be selected
- The task: find a subset of the original attribute set such that the classifier will perform *at least as well* on this subset as on the original set of attributes

three main approaches

- Manually: often unfeasible
- Filters: use the data alone, *independent of the classifier that will be used on this data* (aka scheme-independent selection)
- Wrappers: the FS process is wrapped in the classifier that will be used on the data

Filters - discussion

- Find the *smallest* attribute set in which all the instances are distinct. Problem: cost if exhaustive search used
- But learning and FS are related: in a way, the classifier already includes the ‘good’ (separating) attributes. Hence the idea:
- Use one classifier for FS, then another on the results. E.g. use a DT, and pass the data on to NB. Or use 1R for DT.

Filters cont'd: RELIEF [Kira, Rendell]

1. Initialize weight of all attrs to 0
2. Sample instances and check the similar ones.
3. Determine pairs which are in the same class (near hits) and in different classes (near misses).
4. For each hit, identify attributes with different values. Decrease their weight
5. For each miss, attributes with different values have their weight increased.
6. Repeat the sample selection and weighing (2-5) many times
7. Keep only the attrs with positive weight

Discussion: high variance unless the # of samples very high

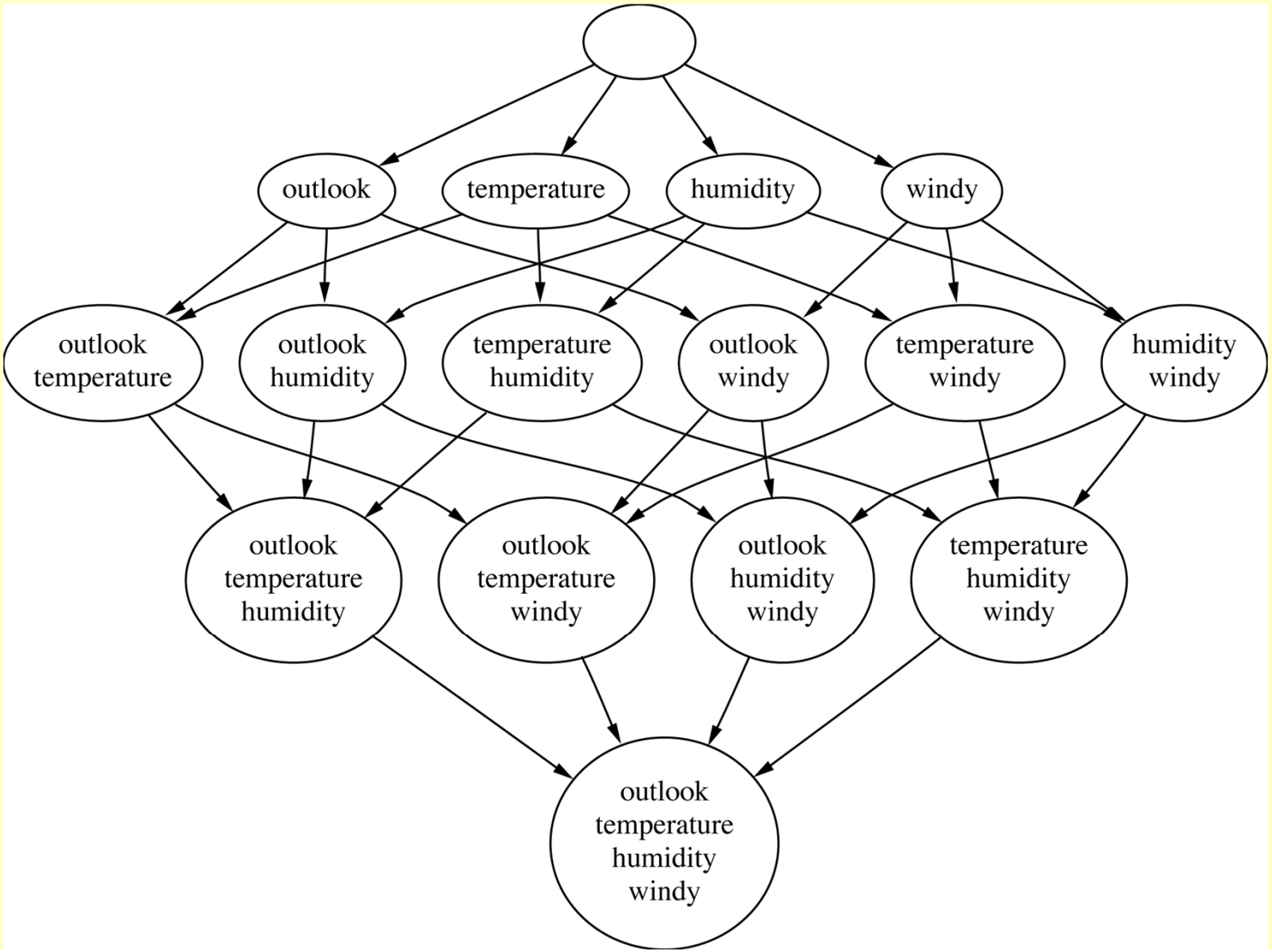
Deterministic RELIEF: use all instances and all hits and misses

A different approach

- View attribute selection as a search in the space of all attributes
- Search needs to be driven by some heuristic (*evaluation criterion*)
- This could be some measure of the discrimination ability of the result of search, or
- Cross-validation, on the part of the training set put aside for that purpose. This means that the classifier is wrapped in the FS process, hence the name *wrapper* (*scheme-specific selection*)

Greedy search example

- A single attribute is added (forward) or deleted (backward)
- Could also be done as best-first search or beam search, or some randomized (e.g. genetic) search



Wrappers

- Computationally expensive (k-fold xval at each search step)
- backward selection often yields better accuracy
  - x-val is just an optimistic estimation that may stop the search prematurely –
  - in backward mode attr sets will be larger than optimal
  - Forward mode may result in better comprehensibility
- Experimentally FS does particularly well with NB on data on which NB does not do well
  - NB is sensitive to redundant and dependent (!) attributes
  - Forward selection with *training set* performance does well [Langley and Sage 94]

Discretization

- Getting away from numerical attrs
- We know it from DTs, where numerical attributes were sorted and splitting points between each two values were considered
- Global (independent of the classifier; eg. 1R in ch. 4 of Witten) and local (different results in ea tree node) schemes exist
- What is the result of discretization: a value of an nominal attribute
- Even if the learner cannot use the order info, this information could still be conveyed: a discretized attribute with *k* values is converted into *k-1* binary attributes – for the i-th value of the discretized attr. encountered in the data, the first *i-1* attributes := *true*, remaining := *false*
- *Supervised* and *unsupervised discretization*

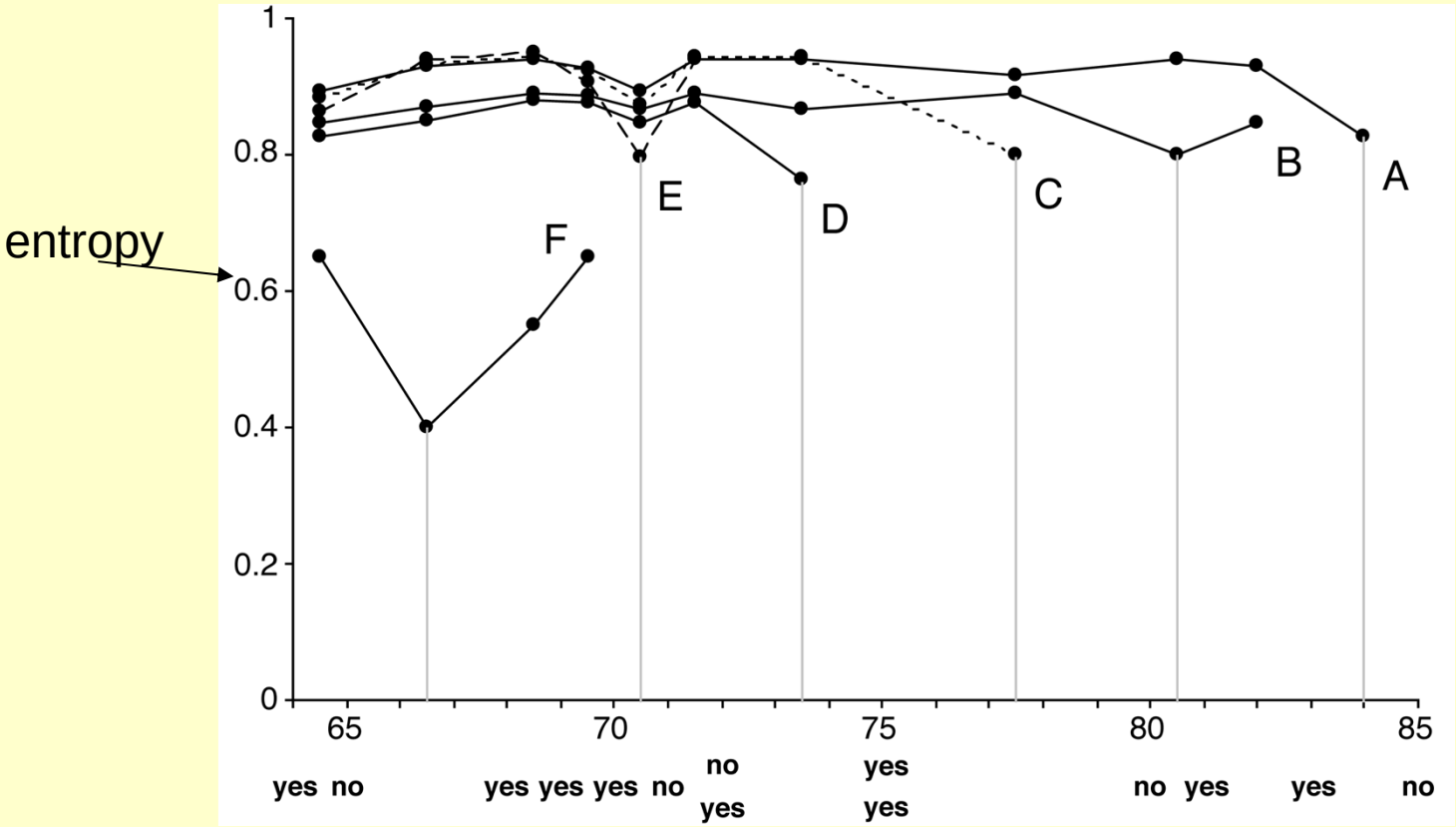
Unsupervised discretization

- Fixed –length intervals (equal interval binning): eg (*max-min*)/*k*
  - How do we know *k*?
  - May distribute instances unevenly
- Variable-length intervals, ea containing the same number of instances (equal frequency binning, or *histogram equalization*)

Supervised discretization

Example of Temperature attribute in the play/don't play data

- A recursive algorithm using information measure/ We go for the cut point with lowest information (cleanest subset)

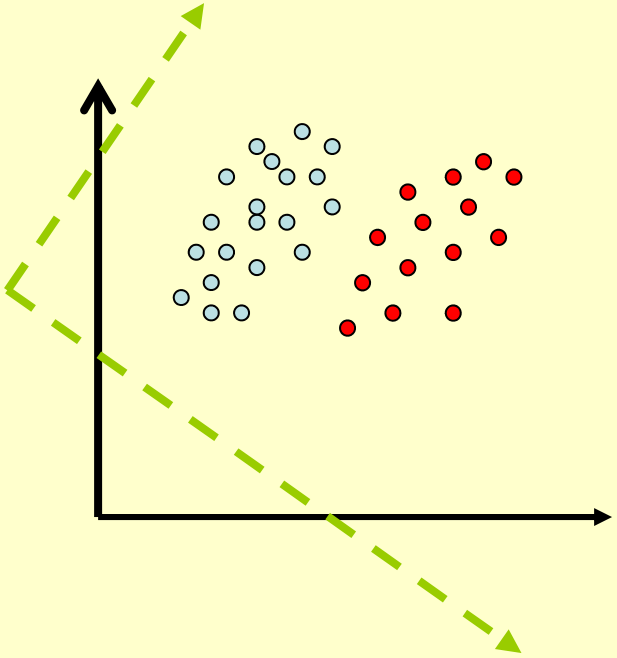


Supervised discretization cont'd

- What's the right stopping criterion?
- How about MDL? Compare the info to transmit the label of ea instance *before* the split, with the info to transmit the *split point* = $log_2(N-1)$ bits, + label info for points *below* and info for points *above; N* is the number of instances.
- Ea. instance costs 1 bit *before* the split (what is its class), and slightly > 0 bits *after* a good split
- Split is useful if the info. gain for the split exceeds a value *v(N,k,entropy of E, of E1 and E2, k1, k2)* (see p. 301 in Witten). This is the Irani, Fayyad 93 method

Principal component analysis (an attribute engineering technique)

- Sometimes, the existing attributes ("coordinates, axis") do not separate well into classes:



Basic idea: detect the direction of the largest variance in the data, build an axis in that.
Build the other axis as orthogonal to the first axis

PCA cont'd

- But how to generalize in multiple dimensions (more than 2 attributes?)
- $$\mathrm{cov}(X_i, X_j) = \frac{\sum_{k=1}^{n} (X_{ik} - \overline{X_i})(X_{jk} - \overline{X_j})}{(n-1)}$$
- Use covariance matrix of the original attributes. In fact, if we normalize the data by subtracting the mean, we have data with the mean average, then the covariance matrix $C_X = 1/n\ XX^T$

PCA cont’d

- Find the [unit] eigenvectors (vectors that are left unchanged when multiplied by the covariance matrix) and eigenvalues (multiplier factors)
- Eigenvectors with largest eigenvalues are the principal component

PCA cont'd

- Algebraically, why the eigenvalues of covariance:
- we transform *linearly* the original data into the data expressed in terms of new axes by matrix multiplication:
- $Y = PX$ where X is the original data, Y is the newly represented data
- such that $C_Y = 1/n YY^T$ is a diagonal matrix with eigenvalues as elements

PCA cont'd

- But $C_Y = 1/nYY^T = 1/n(PX)(PX)^T=P(1/nXX^T)P^T=PC^XP^T$
- So we need to find eigenvalues of the covariance matrix to identify principal components