Summary of Ensemble Methods

- Bagging: a randomized algorithm based on bootstrapping
  - What is ***bootstrapping***
  - Concept of **Bias** vs **Variance**
  - Variance reduction
    - What learning algorithms will be good for bagging?
- Boosting:
  - Combine ***weak classifiers*** (i.e., slightly better than random)
  - Training using the same data set but different weights
    - How to update weights?
    - Are all classifiers equally weighted?
    - How to incorporate weights in learning (DT, KNN, Naïve Bayes)
  - One explanation for not overfitting: maximizing the margin
- Which is more sensitive to outliers: Boosting or bagging?

Feature Selection

Oct 29 2008

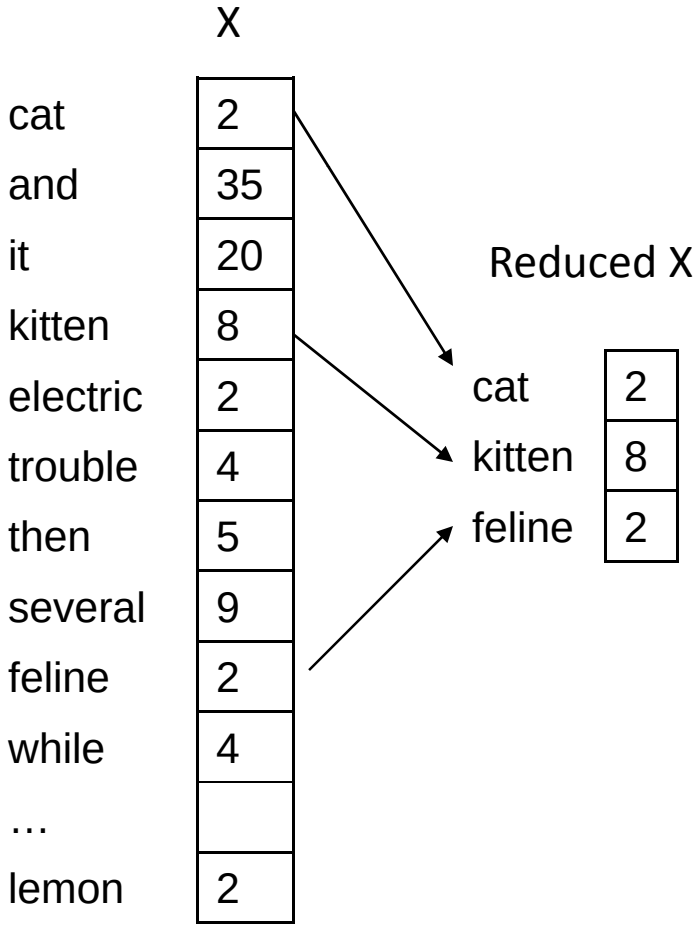
What is feature selection?

- Reducing the feature space by throwing out some of the features
  - Also called attribute selection

What is feature selection?

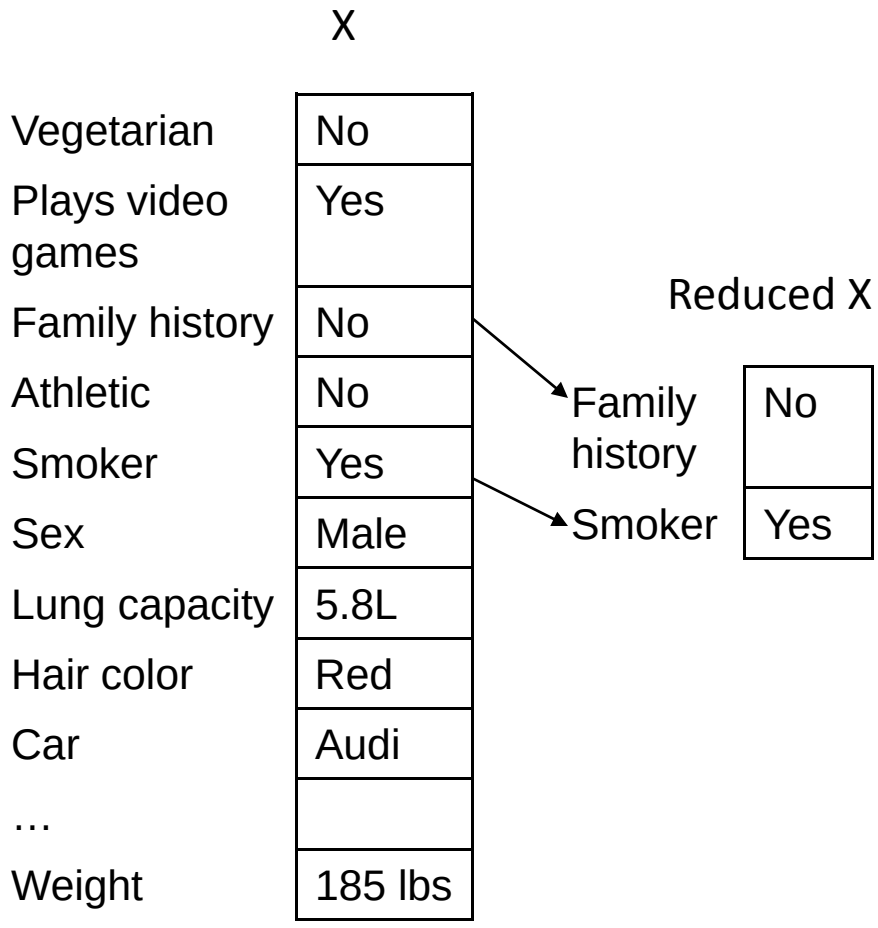
Task: classify whether a document is about cats

Data: word counts in the document



Task: predict chances of lung disease

Data: medical history survey



Why feature selection?

- Motivation: try to find a simple model
  - Occam's razor: the simplest explanation that accounts for the data is the best
- Why not just use classifiers (learning algorithms) that are not (or less) sensitive to irrelevant features?
- Even such methods have trouble when faced with large number of irrelevant features
  - They are more prone to overfitting
  - This is because with large number of irrelevant features, it is more likely to have some chance structure in the data that the learning algorithm try to learn, thus overfit

Why do it?

- Case 1: We're interested in *features*—we want to know which are relevant
- Example: What causes a program to crash? [Alice Zheng '03, '04, '05]
  - Features are aspects of a single program execution
    - Which branches were taken?
    - What values did functions return?
  - Binary response variable: did the program crash?
  - Features that predict crashes well are probably bugs or bug-related
- Case 2: We're interested in *prediction;* features are not interesting in themselves, we just want a good predictor
- Example: Text classification
  - Features for all 10^5 English words, and maybe all word pairs
  - Common practice: throw in every feature you can think of, let feature selection get rid of useless ones
  - Training too expensive with all features
  - The presence of irrelevant features may cause overfitting

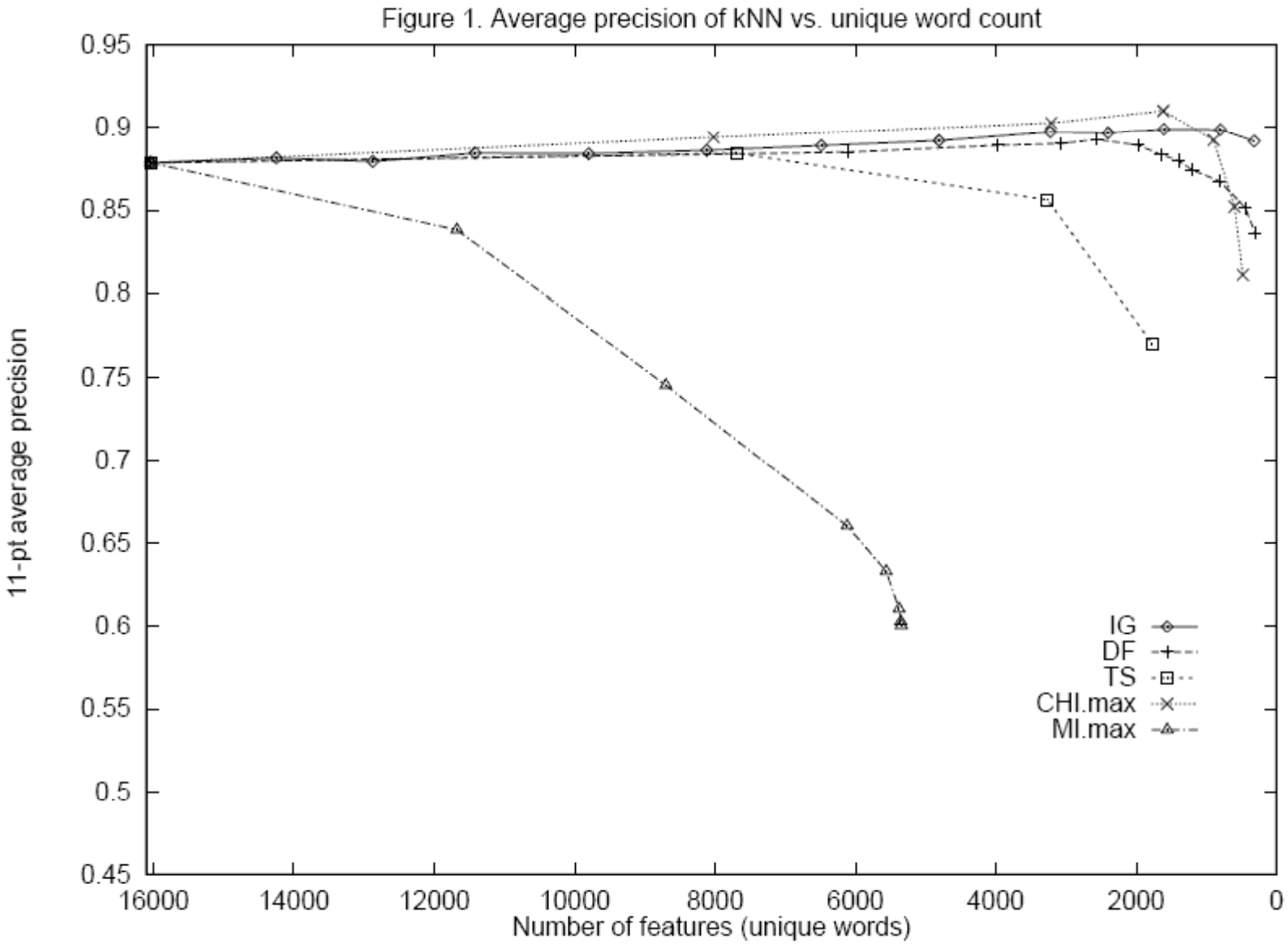
Filtering

Simple techniques for weeding out irrelevant features without considering the classifier that we are using

Filtering

- Basic idea: assign score to each feature f indicating how "related" x_f and y are.
  - Intuition: if $x^i_f = y^i$ for all i, then f is good no matter what our classifier is
  - Many popular scores including one we already know of:
    - Information gain
- Then somehow pick a fix number of highest scoring features to keep

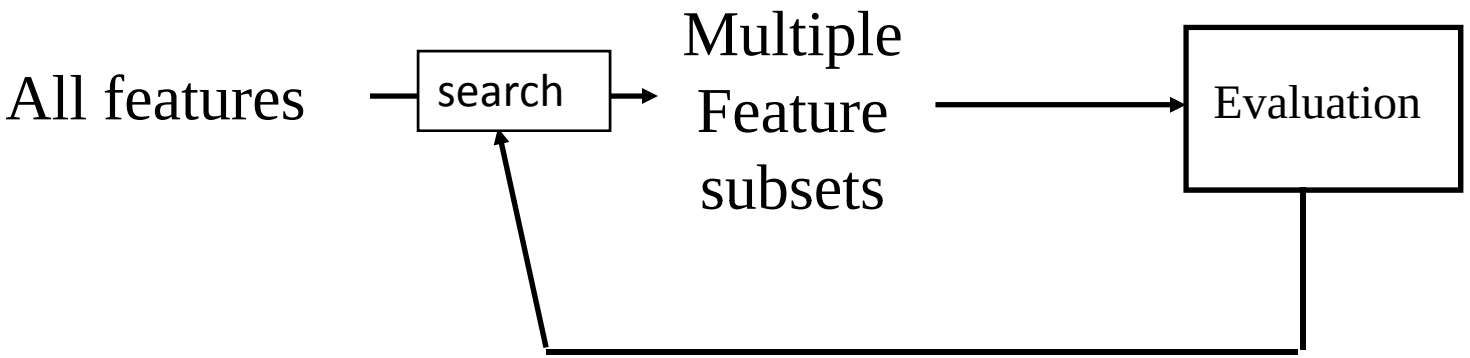
Comparison of filtering methods for text categorization [Yang and Pederson '97]



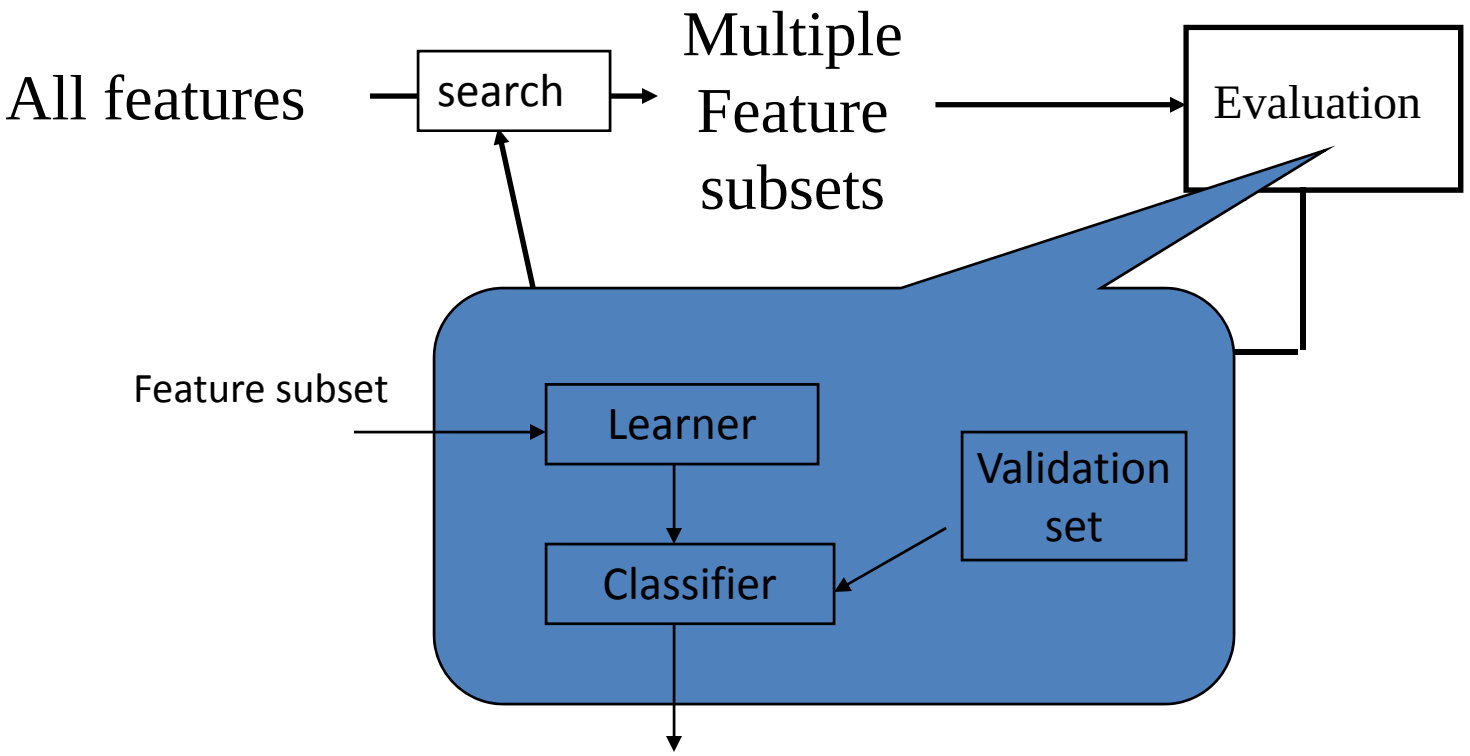
Filtering

- Advantages:
  - Very fast
  - Simple to apply
- Disadvantages:
  - Doesn't take into account which learning algorithm will be used.
  - Doesn't take into account interactions between features
    - Two features may each look bad, but jointly predict class well
- Suggestion: use filtering for initial screening

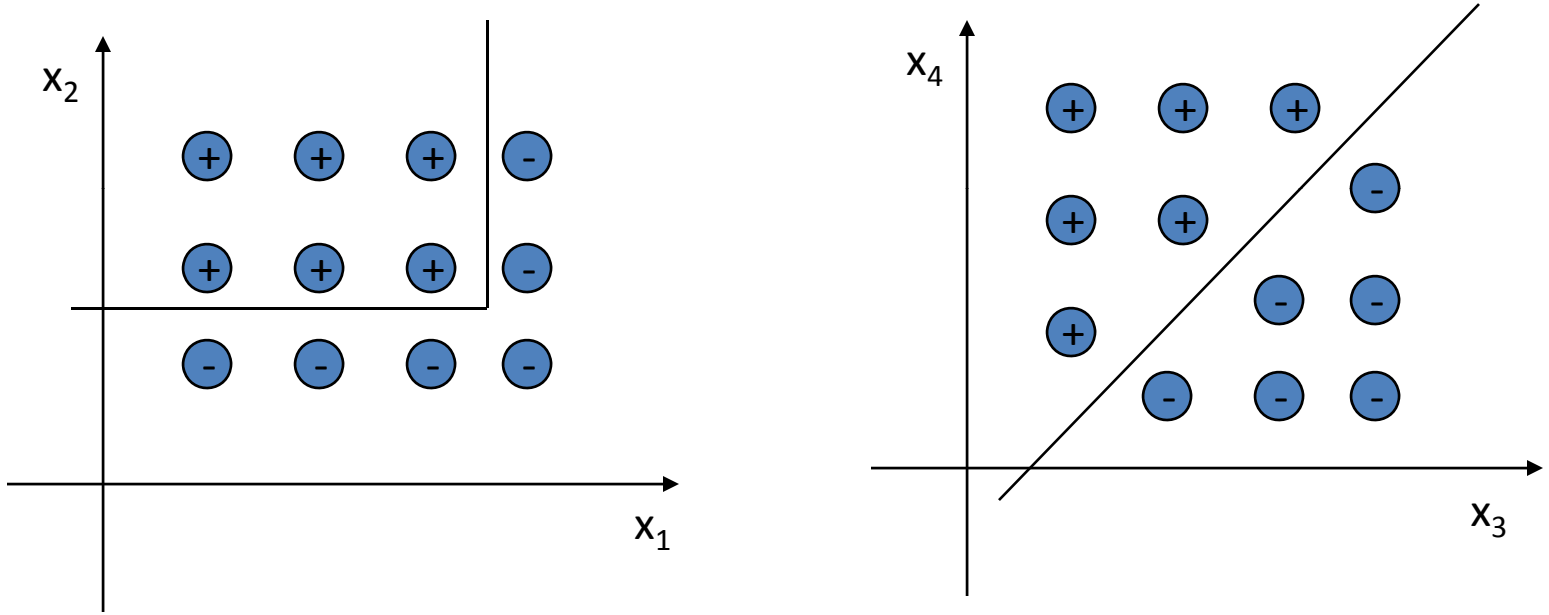
Wrapper Approach



Wrapper Approach

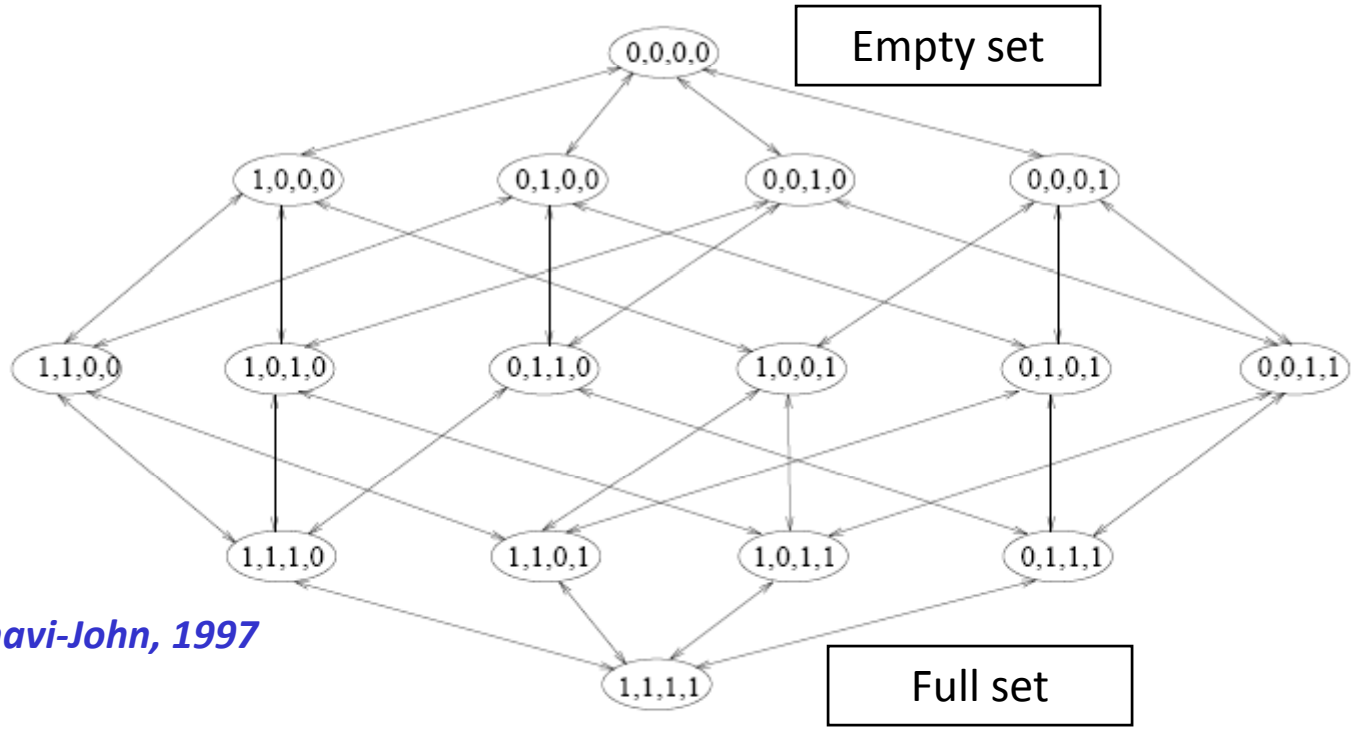


Why include the learning algorithm in the loop?



Different learning algorithms may work well with different feature subsets

Exhaustive search is expensive

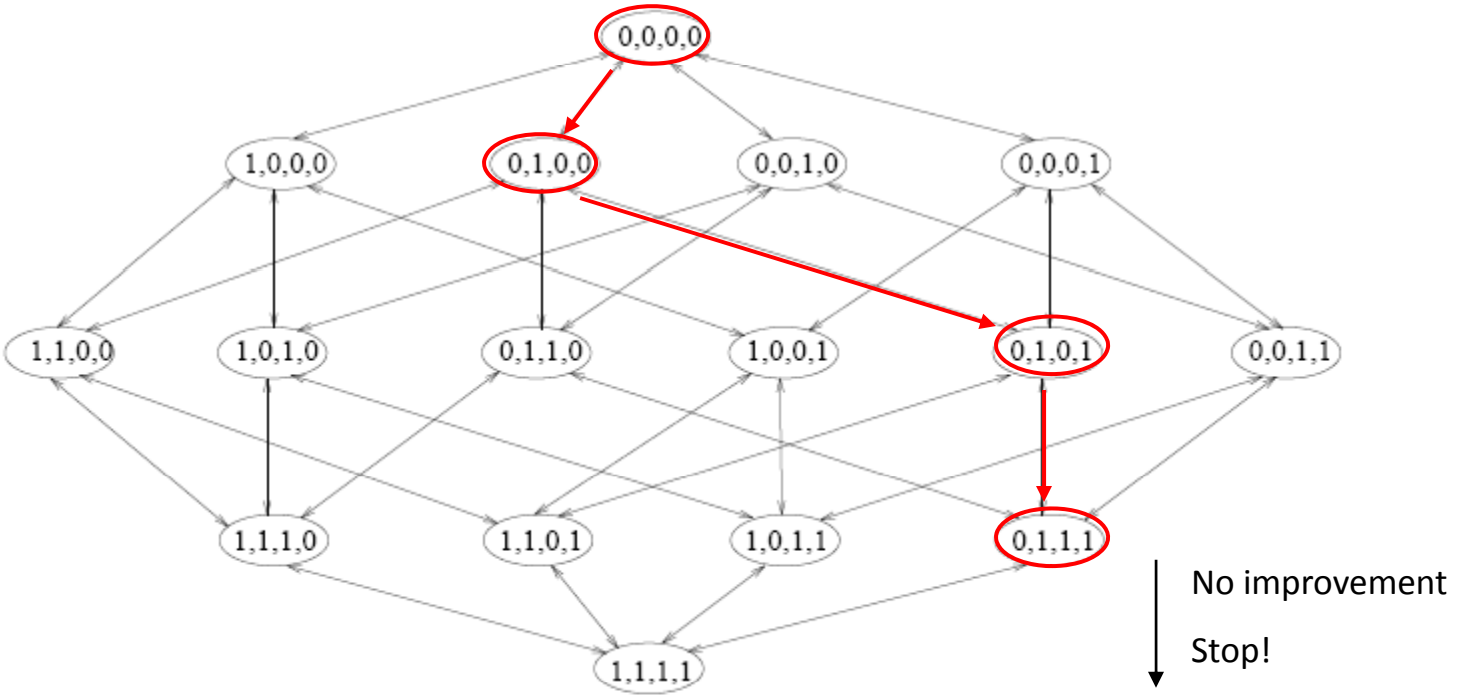


Kohavi-John, 1997

N features, 2^N possible feature subsets!

We need something faster!

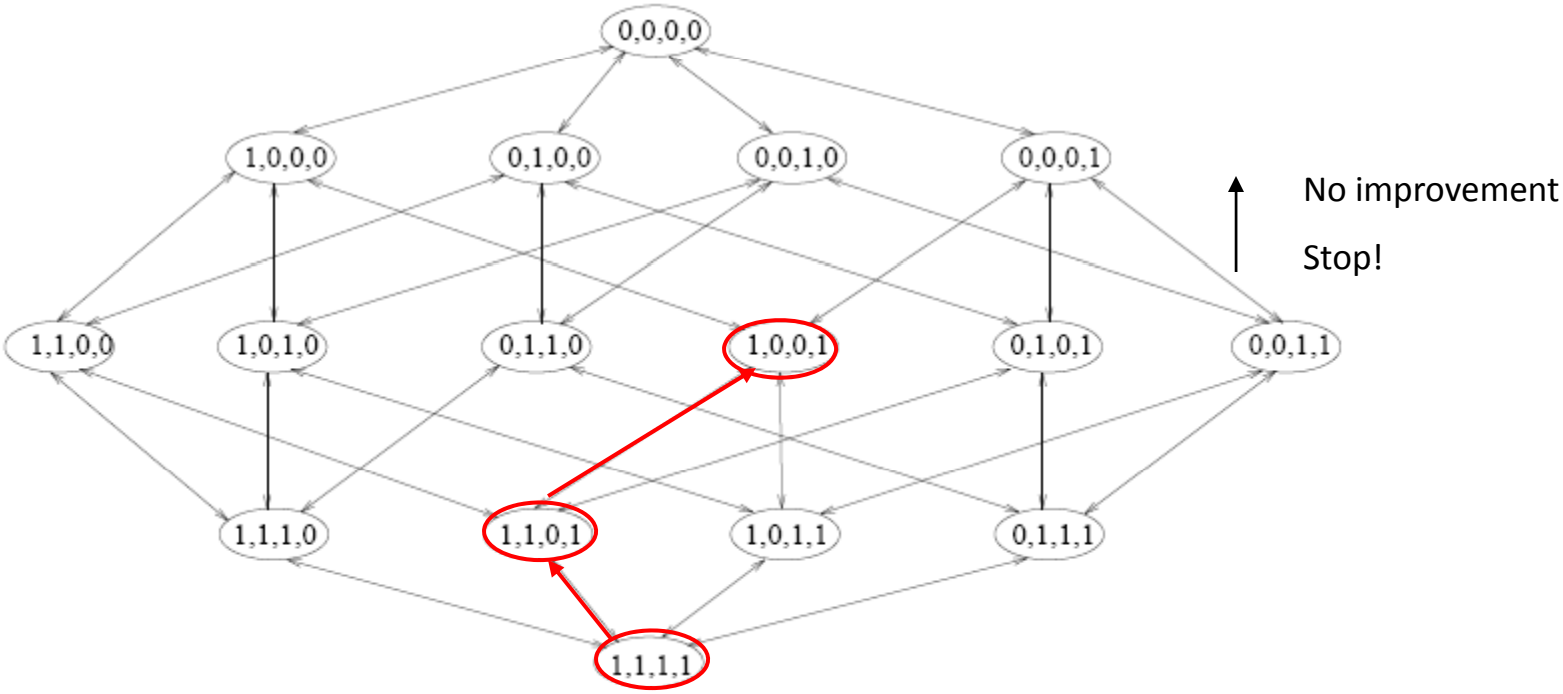
Greedy search: forward selection



Forward selection

```
Initialize s={}
Do:
   Add feature to s
   which improves Score(s) most
While score(s) can be improved
```

Greedy strategy: backward elimination

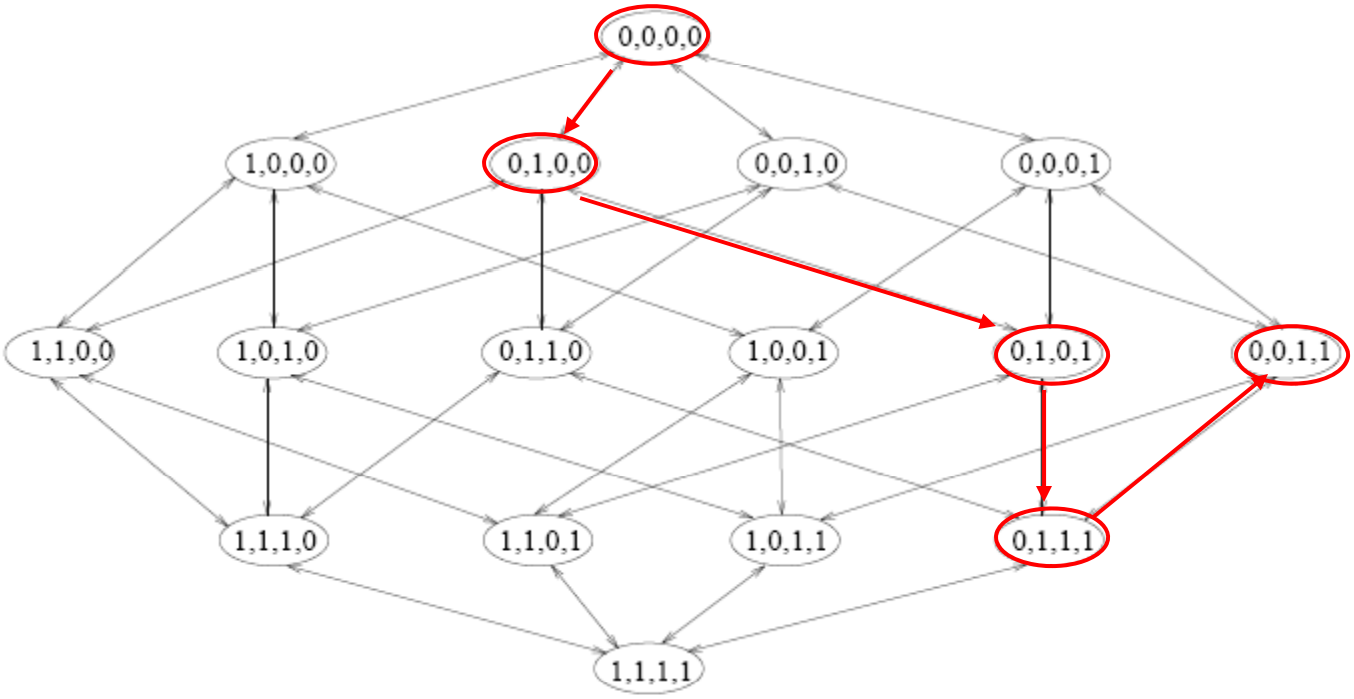


Backward elimination

```
Initialize s={all features}
Do:
   Delete feature from s
   which improves Score(s) most
While score(s) can be improved
```

Comparisons

- Which of these two methods do you expect to be faster:
  - Forward selection
- Which of them do you expect to perform better:
  - Backward elimination
  - Better at finding interacting features
  - But frequently too expensive to fit the large models at the beginning of search
- Both can be too greedy
  - Why?
  - How to improve?



Go forward, then backward

Back and forth until no improvement!

Feature selection summary

- Filter approaches – consider one feature at a time
- Wrapper approaches – search through feature subsets and include learning algorithm in selection process
- Wrapper is more powerful but much more expensive
- Filter can be good for initial screening

Road map to the rest of the term

- Wed Nov 5th – midterm (cover contents up to today)
- Unsupervised learning and pattern discovery – starting Friday 31th, 2-3 weeks
- Reinforcement learning – 2-3 weeks
- We will have a guest lecture on automatic speech recognition on 17th of Nov