ISE 599
Special Topics Applied Predictive Analytics

Mayank Kejriwal

Research Assistant Professor/Research Lead

Department of Industrial and Systems Engineering

Information Sciences Institute

USC Viterbi School of Engineering

kejriwal@isi.edu

Supervised machine learning

Basics

Given a set D of N items $\boldsymbol{x}_i$, each paired with an output value $y_i = f(\boldsymbol{x}_i)$, discover a function $h(\boldsymbol{x})$ which approximates $f(\boldsymbol{x})$

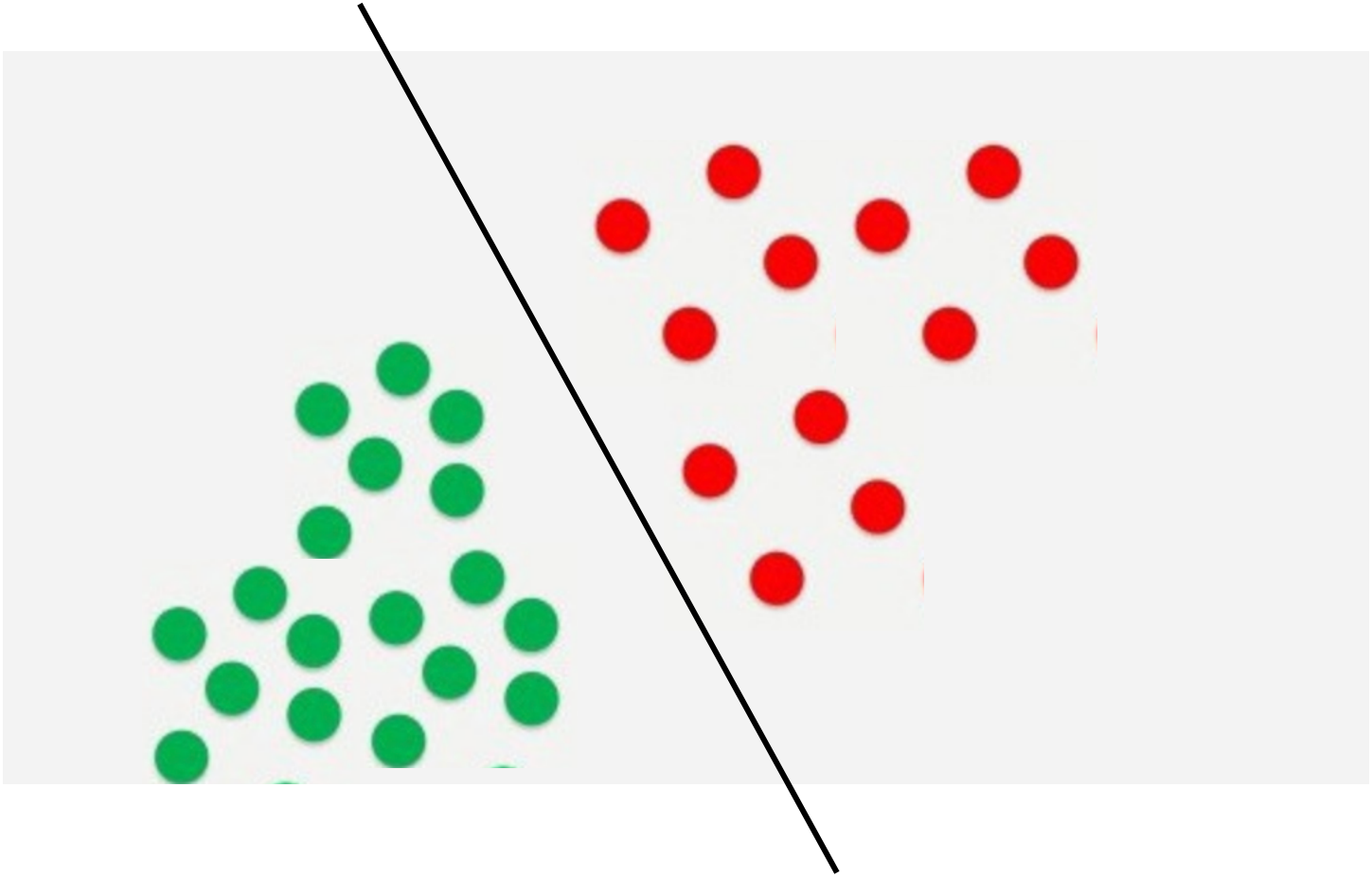
$$D = \{(\boldsymbol{x}_1, y_1), \ldots (\boldsymbol{x}_N, y_N)\}$$

Typically, the **input** values $\boldsymbol{x}$ are (real-valued or boolean) vectors: $\boldsymbol{x}_i \in R^n$ or $\boldsymbol{x}_i \in \{0,1\}^n$

The **output** values y are either boolean *(binary classification)*, elements of a finite set *(multiclass classification)*, or real *(regression)*

We already covered regression last time!

In its simplest form, think about a line separating positive from negative samples



Training and testing

Training: find $h(\boldsymbol{x})$

Given a training set D_{train} of items $(\boldsymbol{x}_i, y_i = f(\boldsymbol{x}_i))$, return a function $h(\boldsymbol{x})$ which approximates $f(\boldsymbol{x})$

Testing: how well does $h(\boldsymbol{x})$ generalize?

Given a test set D_{test} of items $\boldsymbol{x}_i$ that is disjoint from D_{train}, evaluate how close $h(\boldsymbol{x})$ is to $f(\boldsymbol{x})$.

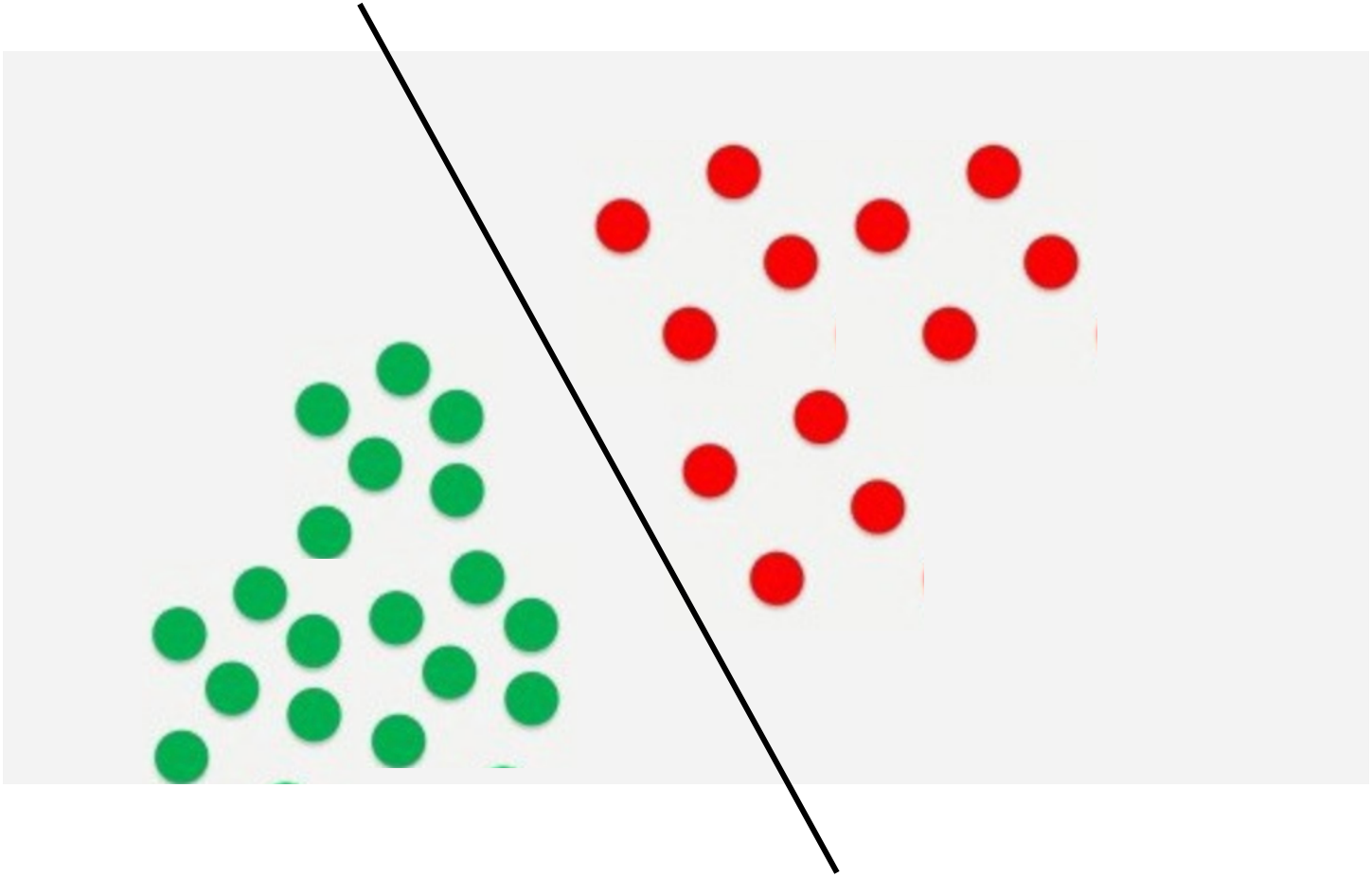
- (classification) accuracy: pctg. of $\boldsymbol{x}_i \in D_{test}$: $h(\boldsymbol{x}_i) = f(\boldsymbol{x}_i)$

What is cross-validation?

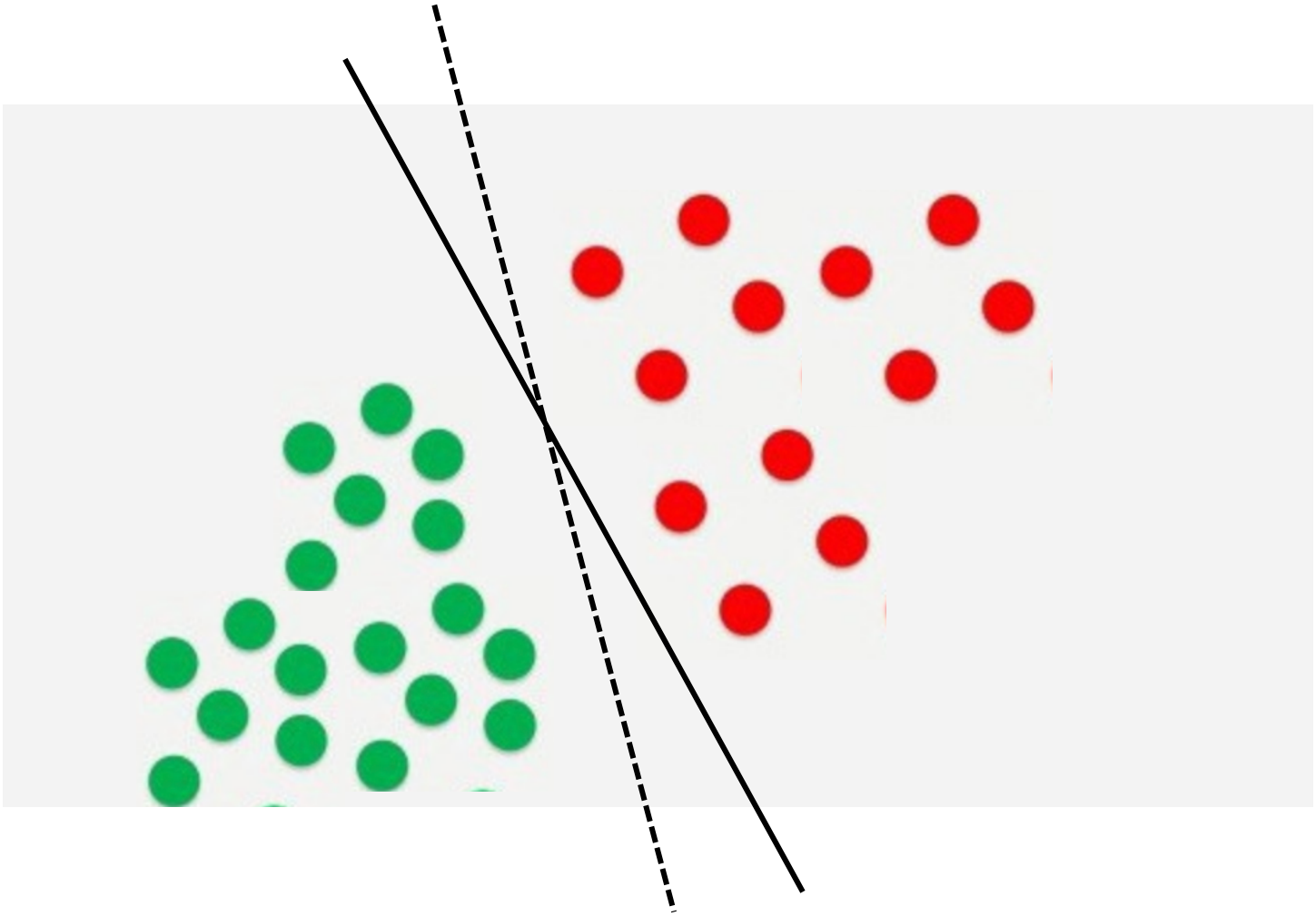
- N-fold (sometimes known as k-fold) cross-validation is a technique that allows the model to generalize better by avoiding parameter-specific overfitting

  – Split data into N equal-sized parts,
  – Run and evaluate N experiments
  – Report average accuracy, variance

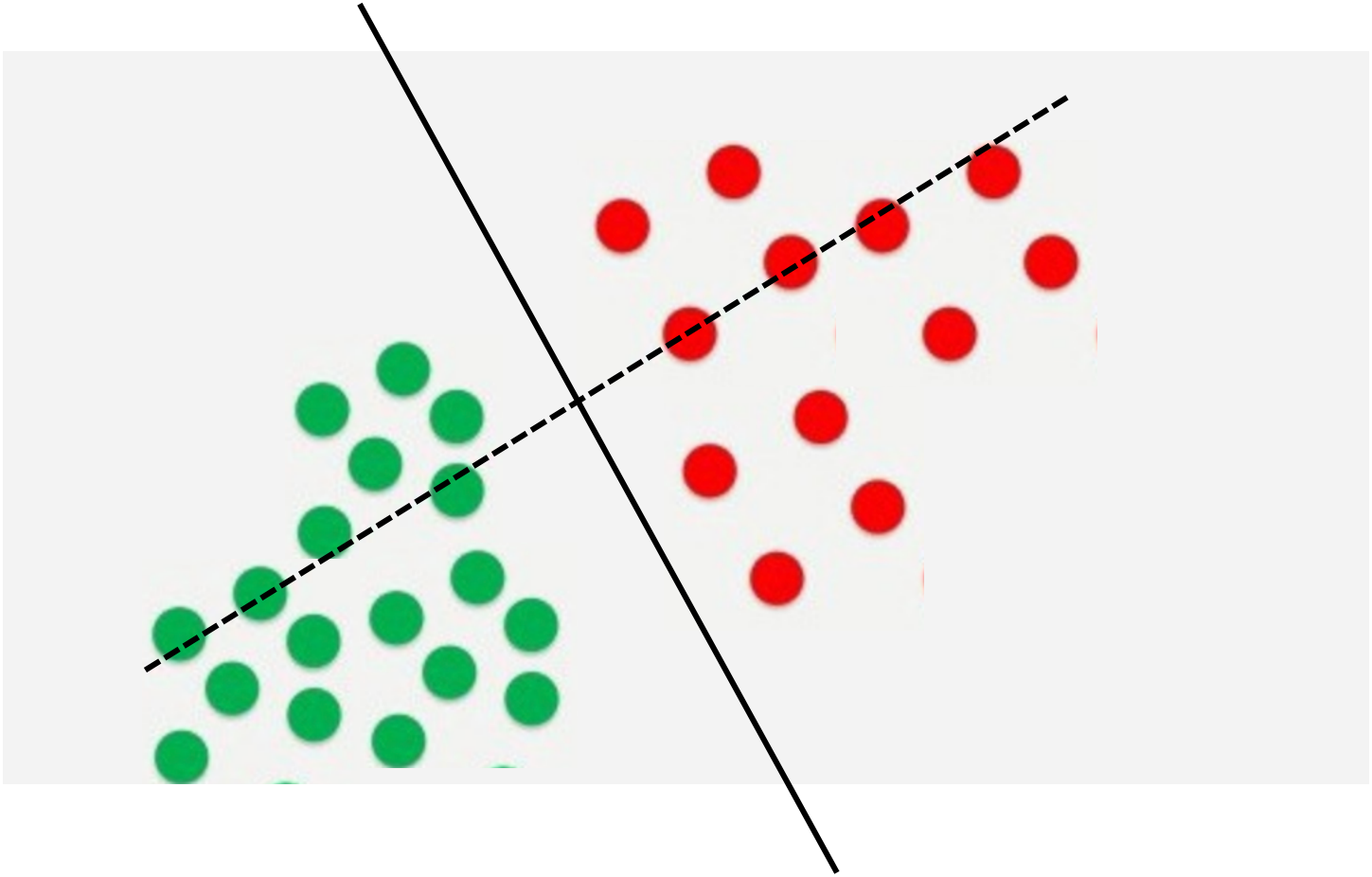
We had used this example before



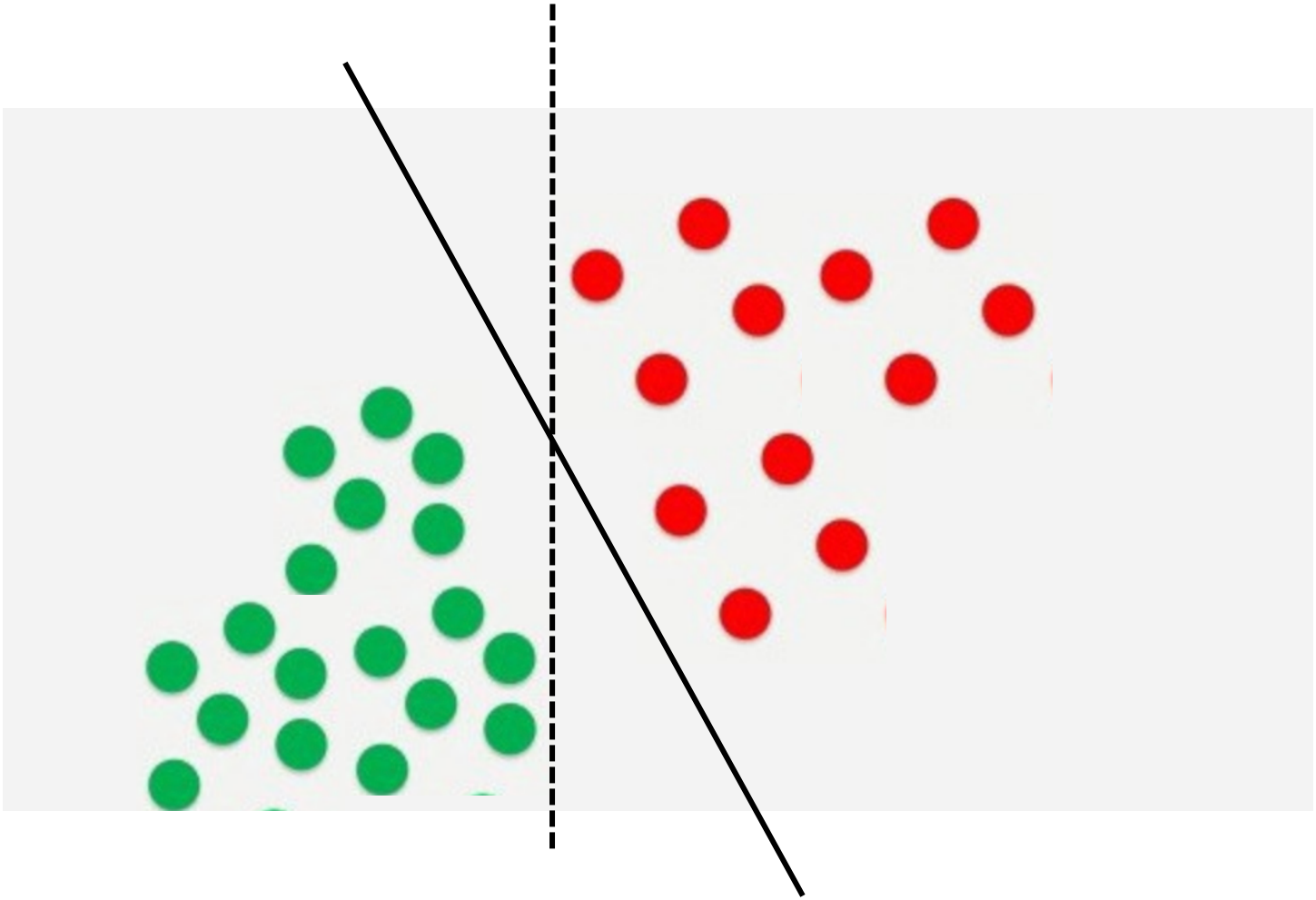
What about this one?



What about this one?



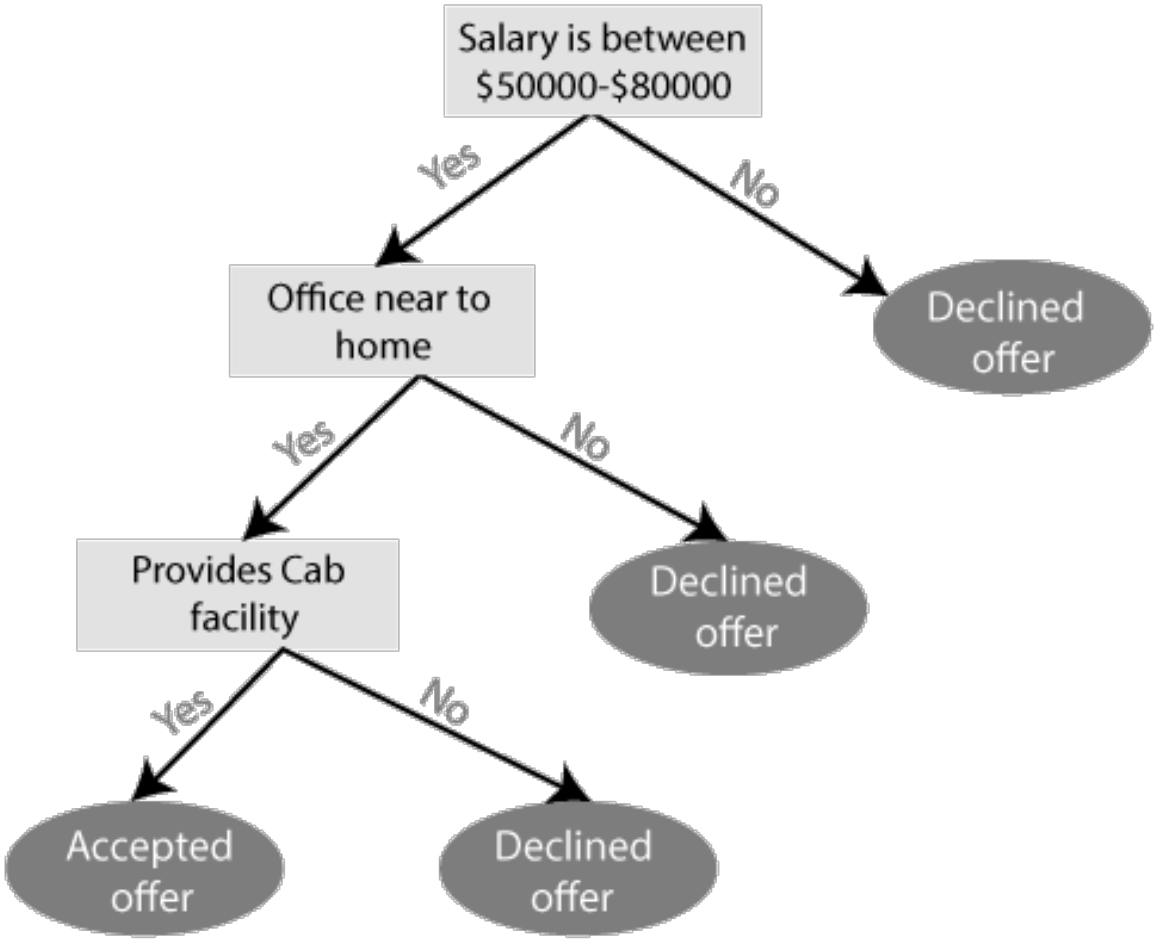
And this?



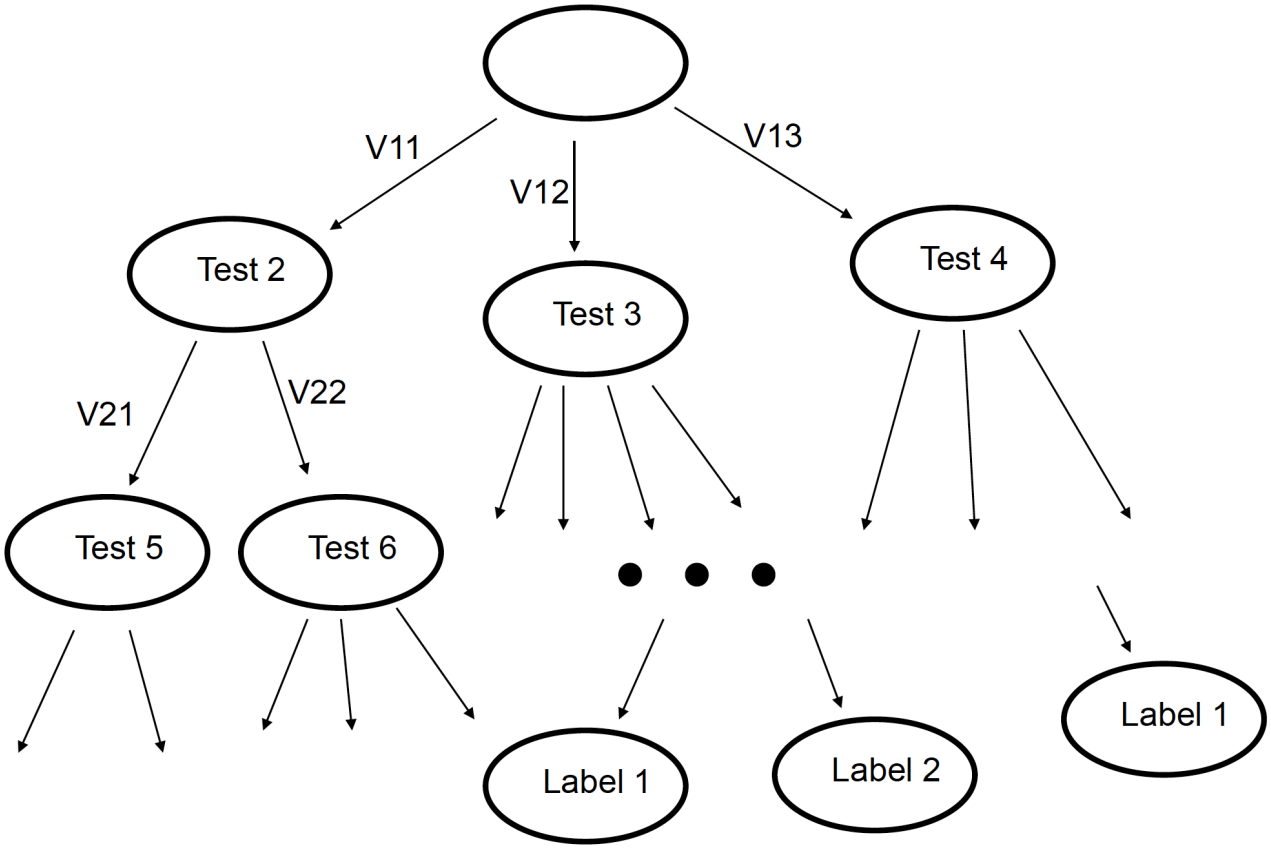
Decision Trees

Example

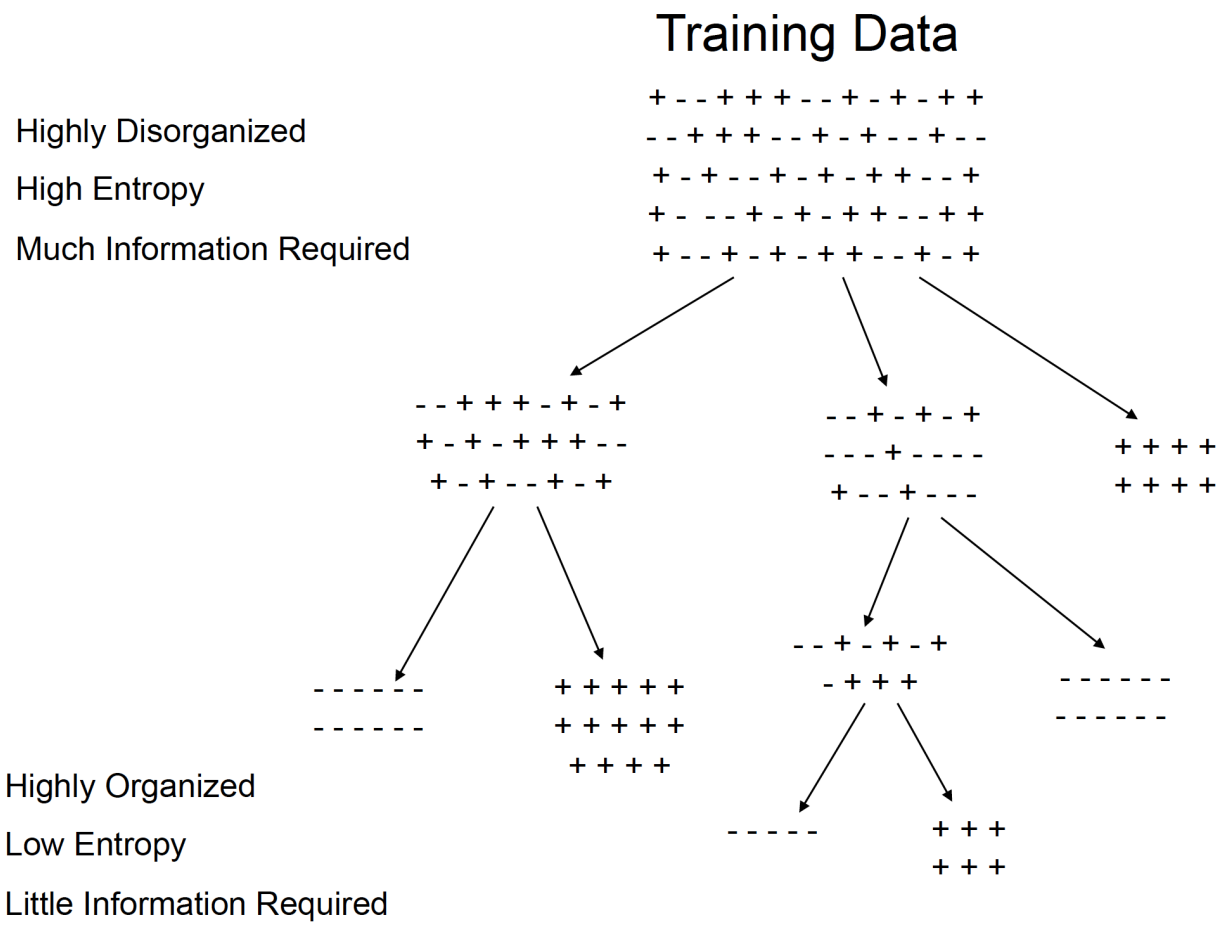
Will I accept this job offer?



Abstract view



Suppose we just assume binary-label problem (E.g., sentiment analysis)



How do we find ‘good’ decision trees?

- All trees cannot be enumerated (why?)
  - With n Boolean attributes (‘features’) how many possible examples can we have? Answer: 2^n
  - One decision tree assigns ‘true’ to one ‘subset’ of these 2^n examples, which means there are 2^(2^n) different subsets of examples!
  - This is the same number of possible decision trees…
  - Try to work this out with small values of n (=10,20…) to see how quickly it explodes
- Need to do greedy or local search
- Important thing to remember is that ‘splits’ must be informative
  - After split, need to be more certain about which label to assign to the data points that meet the test

Criterion for splitting decision tree nodes: information gain

Idea: subtract information required after split from the information required before the split.

Information required before the split: $H(S_b)$

Information required after the split:
$P(S_{a1}) \cdot H(S_{a1}) + P(S_{a2}) \cdot H(S_{a2}) + P(S_{a3}) \cdot H(S_{a3})$
$P(S_{a1})$: use sample counts

Information Gain = $\mathbf{H}(S_b) - \sum_i \mathbf{H}(S_{ai}) \frac{|S_{ai}|}{|S_b|}$