

## Growing a Decision Tree

Given:

- set of (binary) features,  $\mathcal{F}$
- set of target classes
- training set,  $\mathcal{T}$ , with class assignments

Return  $\mathbf{Train}(\mathcal{F}, \mathcal{T})$

Function  $\mathbf{Train}(S, T)$ : returns tree

1. Calculate probability  $p$  over  $T$ , entropy,  $H(p)$
2. For all features  $F_i$  in  $S$ :
  - (a) partition  $T$  into  $T_{Y,i}$  and  $T_{N,i}$ ,
  - (b) calculate probabilities,  $p_{Y,i}$  and  $p_{N,i}$  and their entropies
  - (c) calculate *information gain*,  $G_i$ :

$$G_i = H(p) - \left( \frac{|T_{Y,i}|}{|T|} H(p_{Y,i}) + \frac{|T_{N,i}|}{|T|} H(p_{N,i}) \right)$$

3. Choose feature  $F$  that maximizes  $G$
4. Return tree that splits on  $F$ , with subtrees,  $\mathbf{Train}(S \setminus F, T_Y)$  and  $\mathbf{Train}(S \setminus F, T_N)$ .

## Growing a Decision Tree

### Advantages:

- works better than cosine method,
- easy to comprehend resulting classifier (but be careful how you interpret them!).

### Disadvantages:

- training phase,
- greedy algorithm,
- overkill for linearly separable problems.

### C4.5:

- can handle some non-binary features,
- prunes (with threshold),
- supports cross-validation.