Department of COMPUTER SCIENCE

WARWICK
THE UNIVERSITY OF WARWICK

Machine Learning
CS342

Lecture 5: Decision Tree Learning

Dr. Theo Damoulas
T.Damoulas@warwick.ac.uk

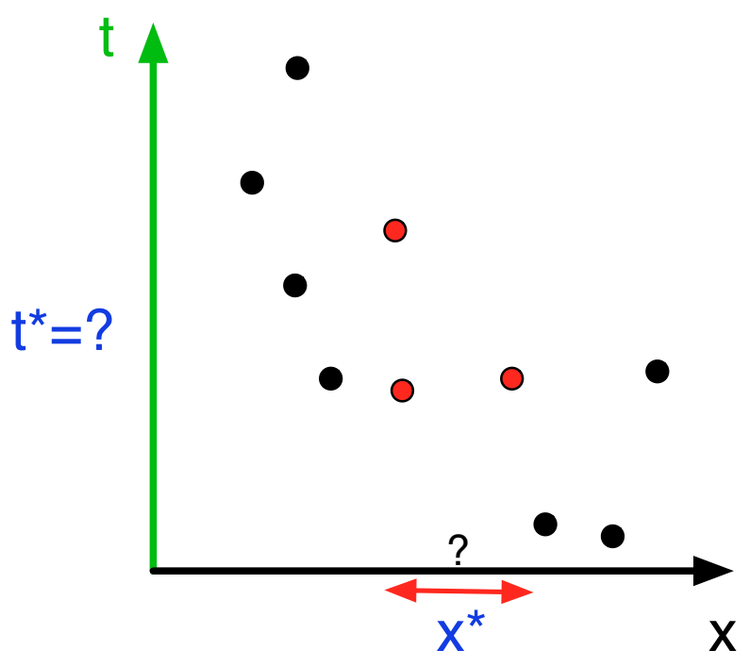
Office hours (CS 307)
Mon 16:00-17:00 Fri 16:00-17:00

Recap: Instance based learning

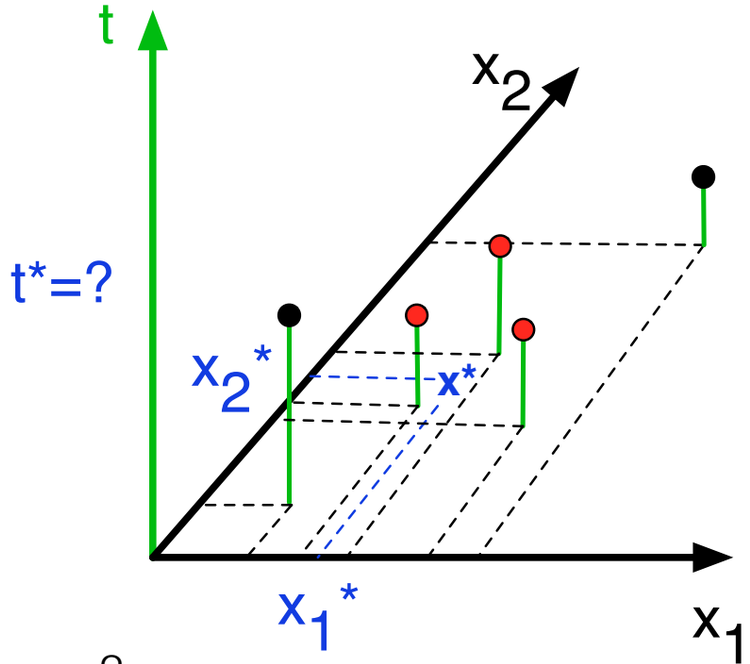
Instance-based learners are all the machine learning algorithms that ***do not*** construct an explicit description of the target function but store the training examples (instances) and use them, ***and a notion of distance*** from them, to generalise to unseen data.

Regression example

Univariate (x is 1D)



Multivariate (x is 2D)



Recap: Distances

Hamming distance

Categorical variables

$\mathbf{x}_i$	1	0	1	1
$\mathbf{x}_j$	0	1	1	1

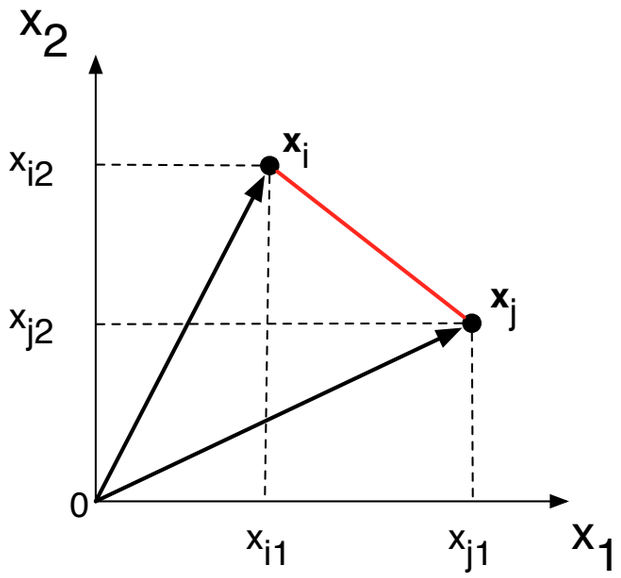
$$d(\mathbf{x}_i, \mathbf{x}_j) = \sum_{d=1}^{D} \begin{cases} 1 & \text{if } x_{id} \neq x_{jd} \\ 0 & \text{if } x_{id} = x_{jd} \end{cases}$$

Euclidean (L2) Distance between x_i and x_j?

$$\text{Euclidean } d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{d=1}^{D} (x_{id} - x_{jd})^2}$$

Input space
Univariate (x is 1-D scalar)

Input space
Multivariate (**x** is 2-D vector)

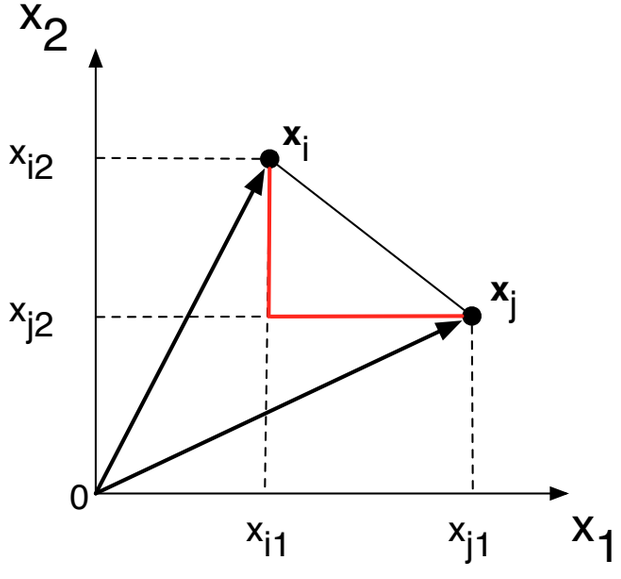


Manhattan (L1) Distance between x_i and x_j?

$$\text{Manhattan } d(\mathbf{x}_i, \mathbf{x}_j) = \sum_{d=1}^{D} |x_{id} - x_{jd}|$$

Input space
Univariate (x is 1-D scalar)

Input space
Multivariate (**x** is 2-D vector)



Recap: k-NN for Regression and Classification

Training:

- For each training example (input-output pair $\mathbf{x}_n, t_n$), add the example to the list *training_examples*

Prediction:

- Choose k, the number of neighbours we want
- Choose the distance function (e.g. Euclidean distance)
- Given a query instance **x*** to predict its output t*
  - Find $\mathbf{x}_1 \ldots \mathbf{x}_k$ the k instances that are **nearest** to **x*** using the selected distance
  - Return prediction:

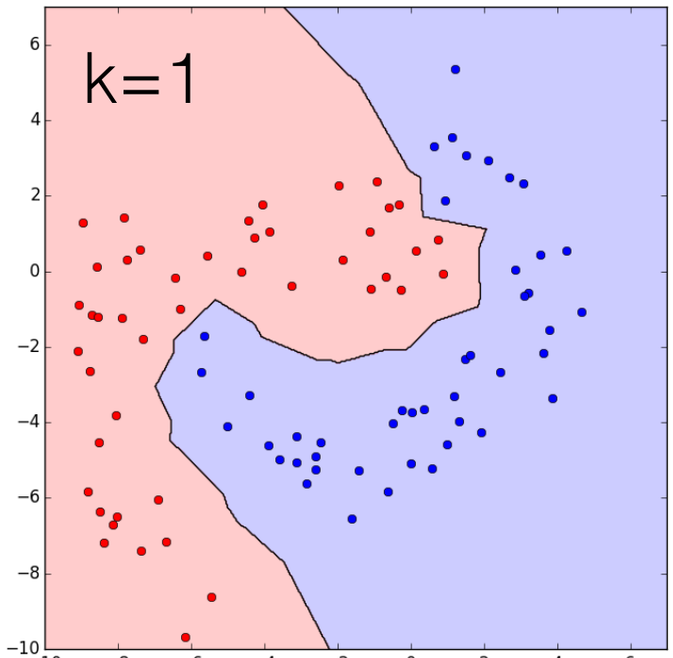
Regression $t^* \leftarrow \frac{\sum_{i=1}^{k} t_i}{k}$

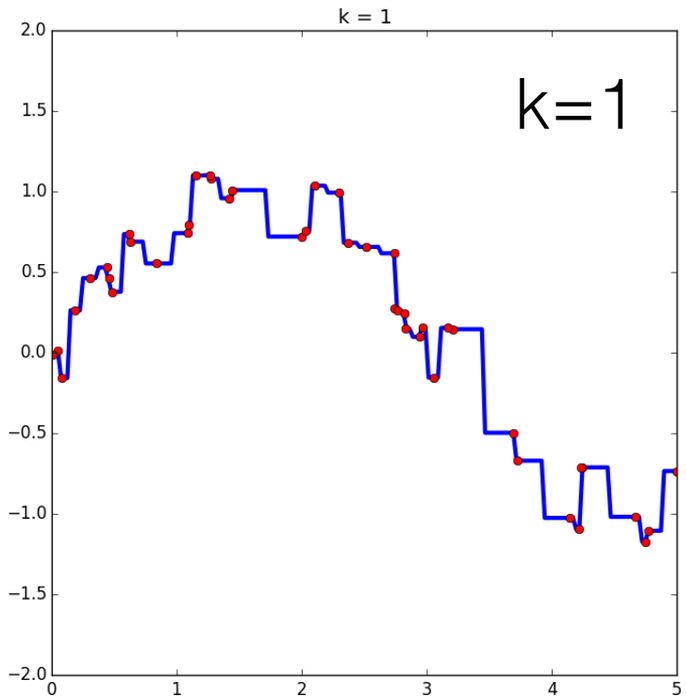
(Binary) Classification $t_n^* \leftarrow \underset{u \in \{-1,1\}}{\operatorname{argmax}} \sum_{i=1}^{k} \delta(u, t_i)$ where $\delta(a,b) = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{otherwise} \end{cases}$

Notes:

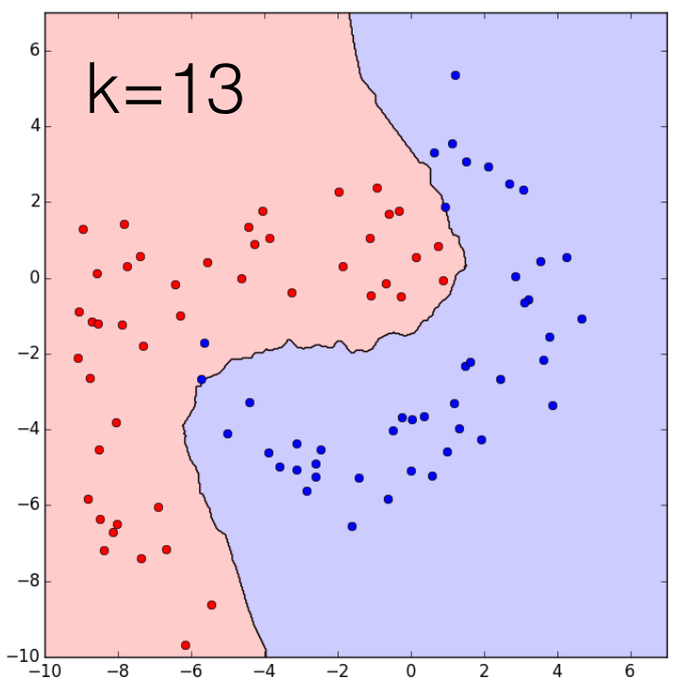
- i iterates over k neighbours
- t* is for single test point
- t^*_n is the *n*-th test point

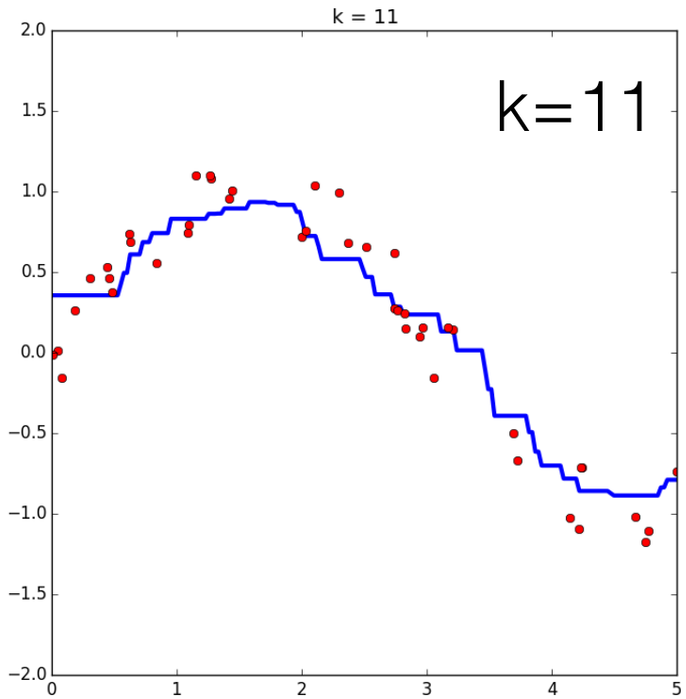
Recap: k-NN for Regression and Classification





Piece-wise linear





Use cross-validation to choose k

Recap: Distance-Weighted k-NN

- Weigh the vote of each neighbour by its distance to the observation
- Can help break ties when k is even
- Can help deal with noisy data and outliers

Regression:

$$t^* \leftarrow \frac{\sum_{k=1}^{K} w_i t_k}{\sum_{i=1}^{k} w_i}$$

$$w_i = \frac{1}{d(\mathbf{x}_i, \mathbf{x}^*)^2}$$

Classification:

$$t_n^* \leftarrow \underset{u \in \{-1,1\}}{\operatorname{argmax}} \sum_{i=1}^{k} w_i \delta(u, t_i)$$

$$\text{where } \delta(a,b) = \begin{cases} 1 & \text{if } a = b \\ 0 & \text{otherwise} \end{cases}$$

Recap: k-NN

- Vanilla k-NN will not perform well in high-D as distances “break” in high-D
- Assign weights for every dimension/attribute and learn via e.g. CV
- Tree data structures (e.g. k-d tree) to improve search time for neighbours
- High computational cost to store all the training data in big data settings

	Naive	k-d tree
NN Search:	$O(ND)$	$O(\log N)$

Decision Tree Learning

T. Mitchell Ch. 3. See module website

Task: *Classify Saturday mornings as suitable or not for playing tennis*

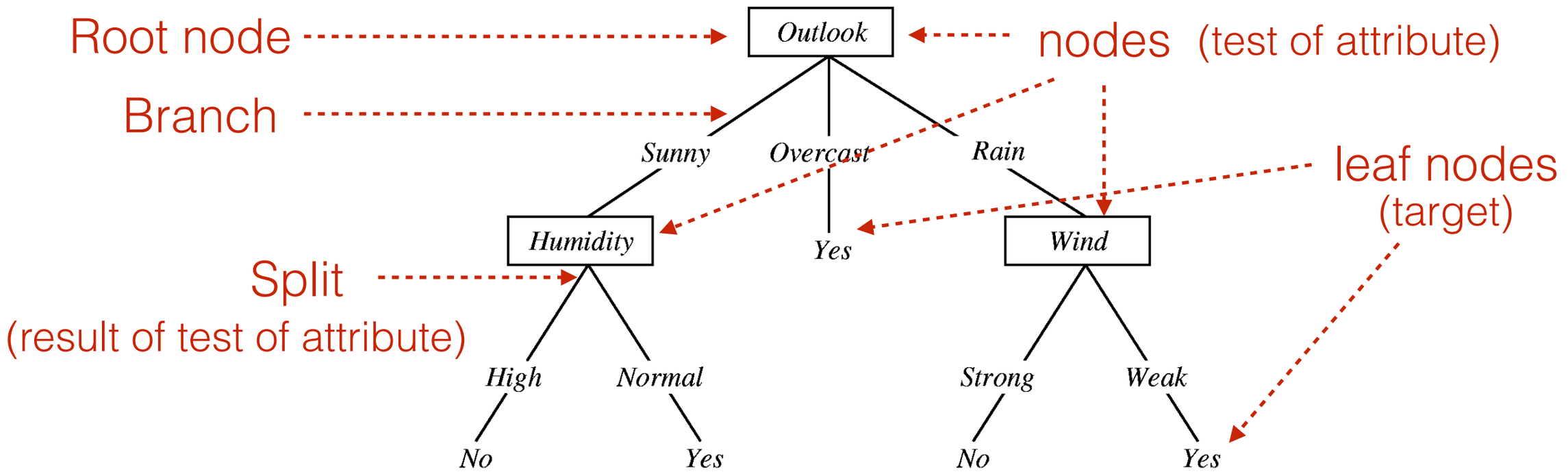
Nominal data:
Discrete-valued attributes

Play Tennis?	Weather Outlook	Humidity	Wind	Temperature
Yes	Sunny	Normal	Weak	Medium
Yes	Overcast	High	Strong	Medium
No	Sunny	High	Weak	Medium
Yes	Overcast	Normal	Weak	Medium
No	Rain	High	Strong	Medium
No	Rain	Normal	Strong	Medium
Yes	Rain	High	Weak	Medium
No	Sunny	High	Weak	Medium
…	…	…	…	…

What should we predict if:
<Outlook = Sunny, Temperature = Hot, Humidity = High, Wind = Strong>

Decision Tree Learning

Lets say ***we have learned*** the following rules (somehow) and formed a tree



What about now? easier?

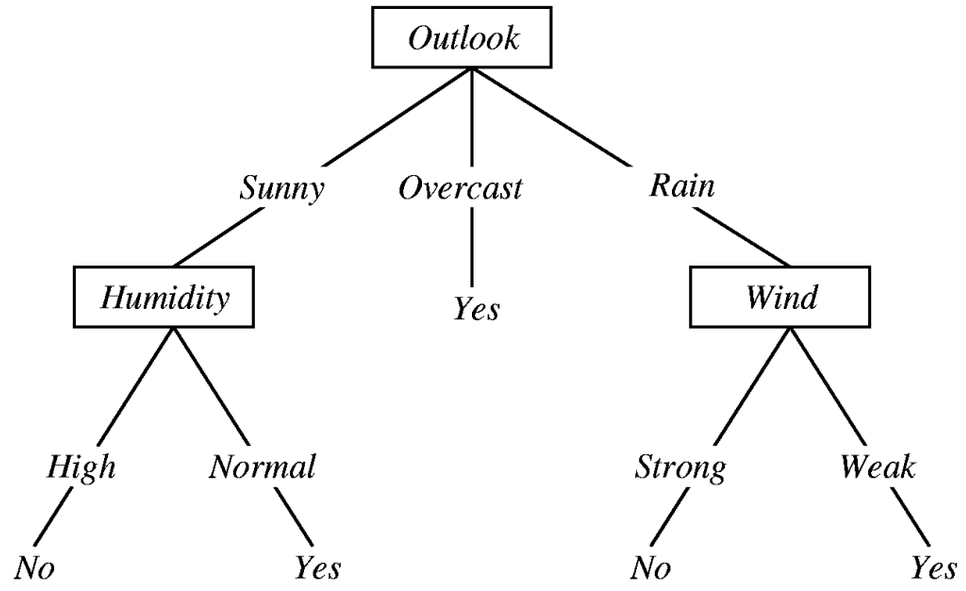
<Outlook = Sunny, Temperature = Hot, Humidity = High, Wind = Strong>

This is a DT. **It represents a disjunction of conjunctions**

But how do we learn such a tree from data? what do we need to learn?

Decision Tree Learning

Imagine splitting our training data through this tree



We need to learn:

- Which attribute goes to which node? “best” attributes go higher up
- How many attributes to use (the depth of the tree)
- What are the splitting criteria - e.g. continuous attributes?
- How many splits at each node?
- When should we place a leaf node? Can a leaf node be “impure”?

Further issues:

- If a leaf node is “impure” (non-zero training error) how is the target decided?
- If the tree is too large how do we “prune” it back?
- How can we handle missing data

Decision Tree Learning: ID3 (Iterative Dichotomiser 3)

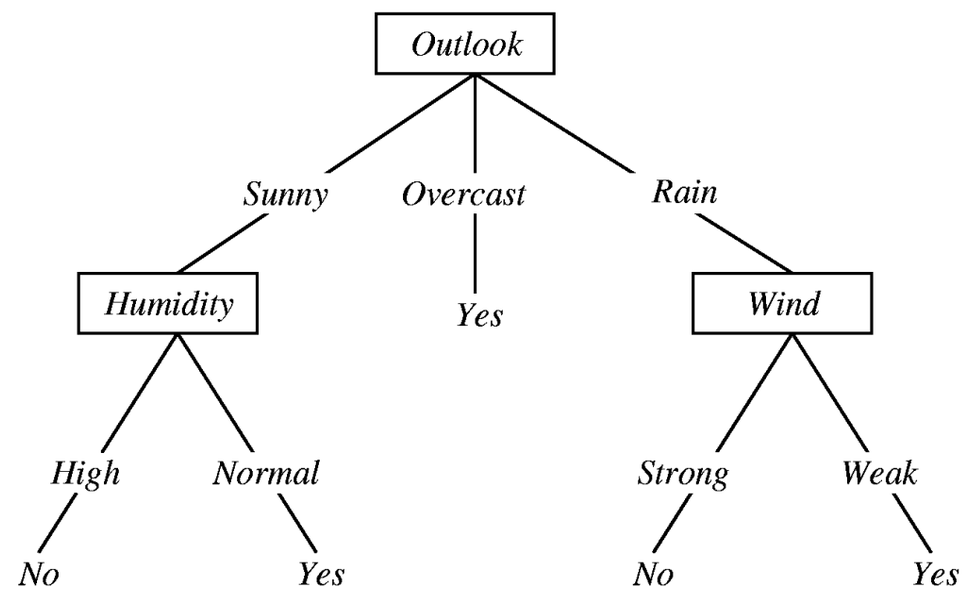
One of the first DT algorithms focused on ***nominal attributes*** and classification

ID3(Observations, Targets, Attributes)

- Create a Root node for the tree
- If all observations are class +1, return single-node tree Root with label +1
- If all observations are class -1, return single-node tree Root with label -1
- If Attributes is empty, return the single-node tree Root with label the most common value in Targets
- Else begin:
    - A ⟵ *best* attribute from Attributes (***highest Information Gain***)
    - The decision attribute for Root ⟵ A
    - For each possible vale u_i of A:
        - Add a new tree branch below Root for A = u_i
        - S_u_i ⟵ Subset of Observations with A = u_i
        - If S_u_i is empty:
            - Add leaf node with label the most common value in Targets
        - Else add below branch
          ID3(S_u_i, Targets, Attributes - {A}))

Decision Tree Learning: ID3 and Information Gain

Recursive greedy top-down algorithm so “best” attribute at top



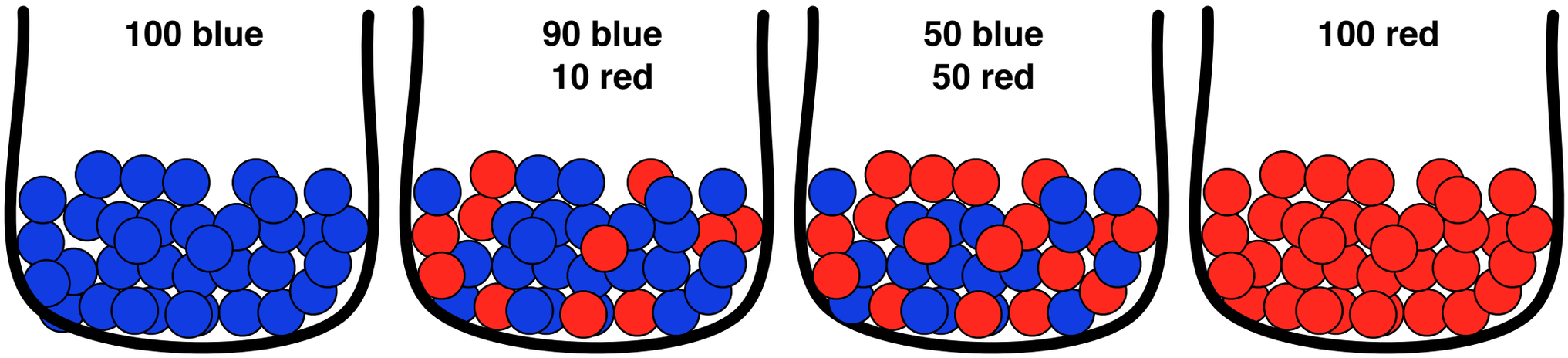
- Our training examples $\{\mathbf{x}_n, t_n\}_{n=1}^{N}$ are used to learn the tree
- Classification so in theory we want leaf nodes that are pure (e.g. only observations that belong to the same class, e.g. “Yes”)
- A good attribute is one that is **discriminative** for the classes
- Subset of observations in each branch are as “homogeneous” as possible in their targets

Decision Tree Learning: ID3 and Information Gain

Entropy as a measure of "homogeneity" of a set S of examples that can take one of C values

$$\text{Entropy}(S) = -\sum_{i=1}^{C} p_i \log_2 p_i$$

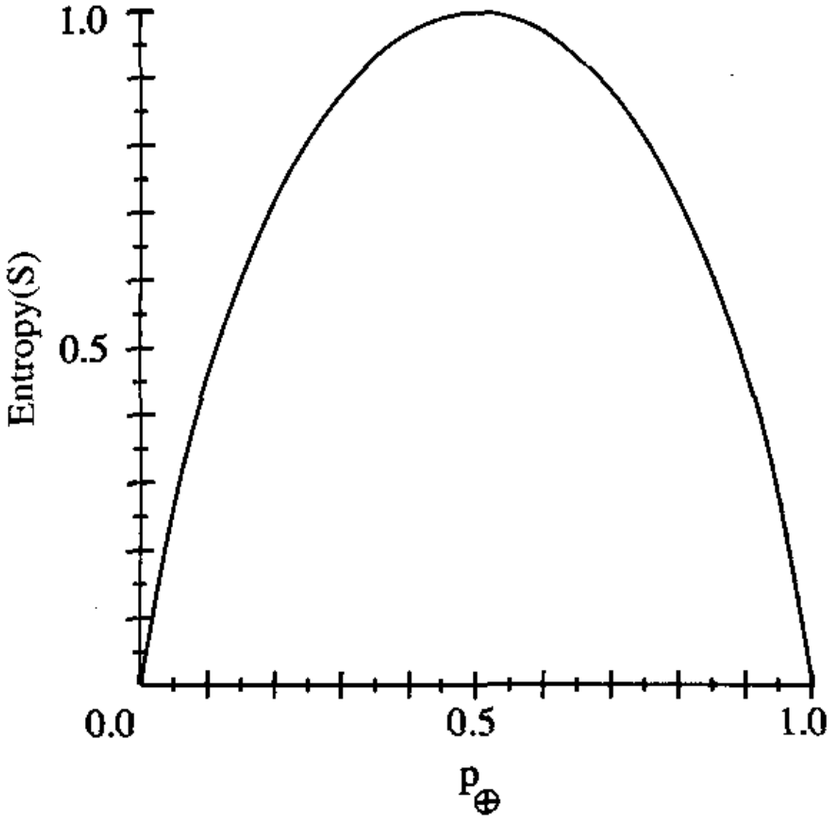
where p_i frequency of items with type i in set S

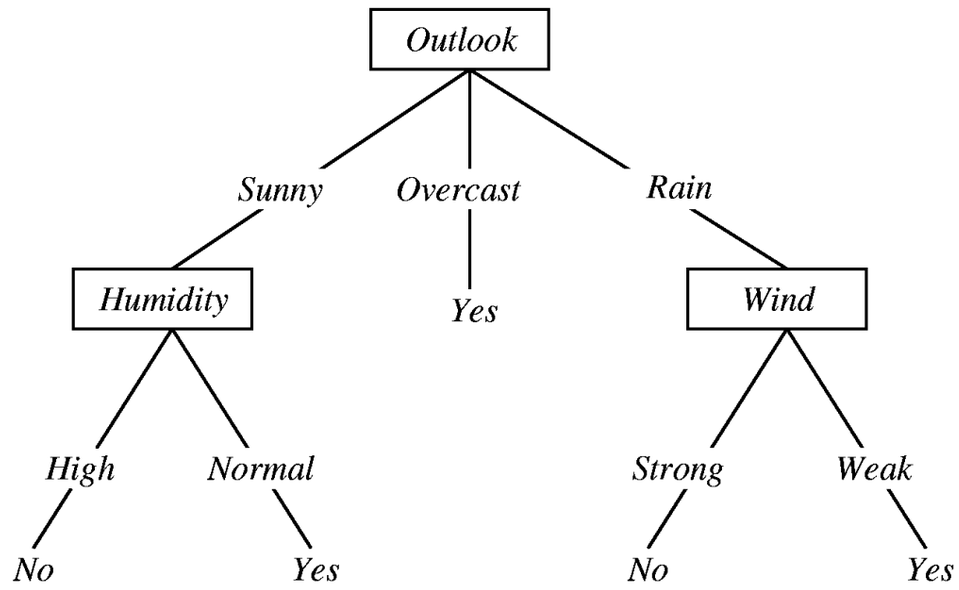


Decision Tree Learning: ID3 and Information Gain

We could use Entropy directly as a measure of how good an attribute is to create “homogeneity” but we can do something even better

We want to see the improvement in homogeneity at every node so we compute the **reduction in Entropy** between 2 consecutive nodes!





Decision Tree Learning: ID3 and Information Gain

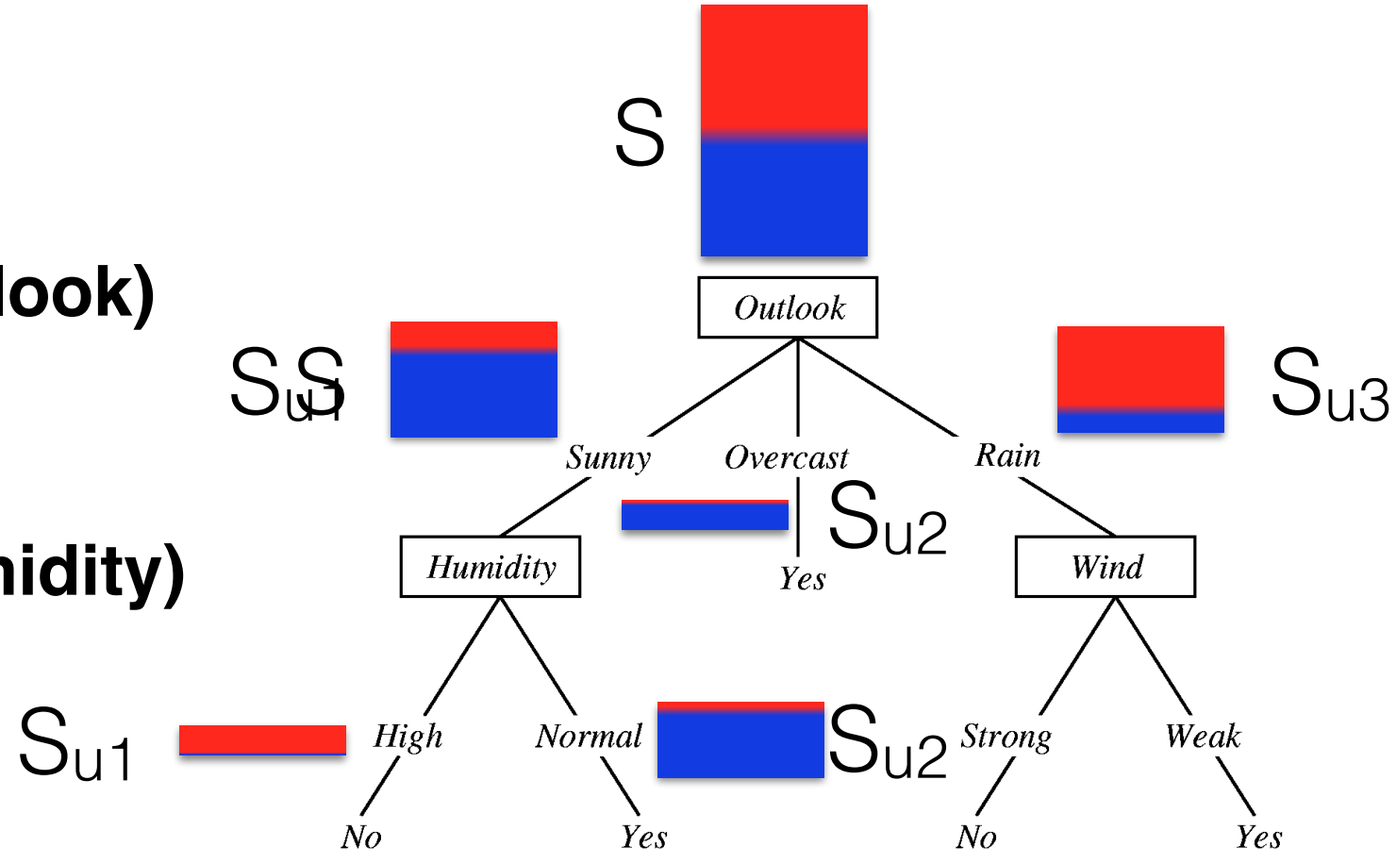
Information Gain = Relative Entropy = Kullback-Leibler (KL) Divergence

$$\text{I.Gain}(S, A) = \text{Entropy}(S) - \sum_i \frac{|S_{u_i}|}{|S|} \text{Entropy}(S_{u_i})$$

where A is the attribute, u_i is the possible values of A, S_{u_i} is the subset of S where $A = u_i$

I.Gain(S,Outlook)

I.Gain(S,Humidity)



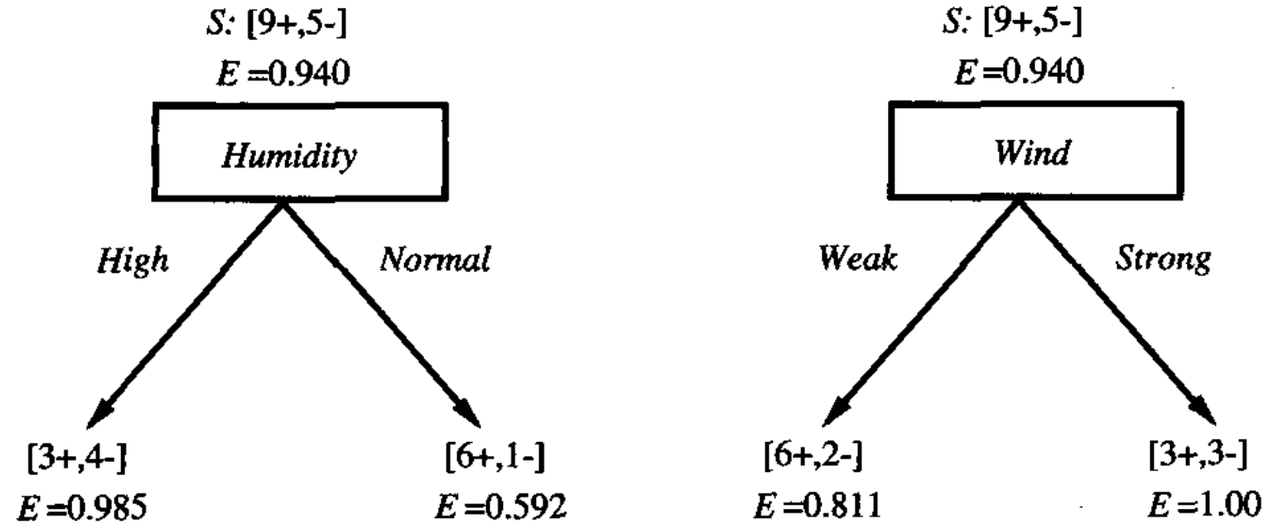
Department of COMPUTER SCIENCE

T. Mitchell Ch. 3. See module website

Decision Tree Learning: An example

$$\text{I.Gain}(S, A) = \text{Entropy}(S) - \sum_i \frac{|S_{u_i}|}{|S|} \text{Entropy}(S_{u_i})$$

Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No



We choose the attribute with the **highest** Information Gain
= biggest reduction in Entropy

T. Mitchell Ch. 3. See module website

Decision Tree Learning: Hypothesis space

What is the hypothesis space? Is ID3 optimal?

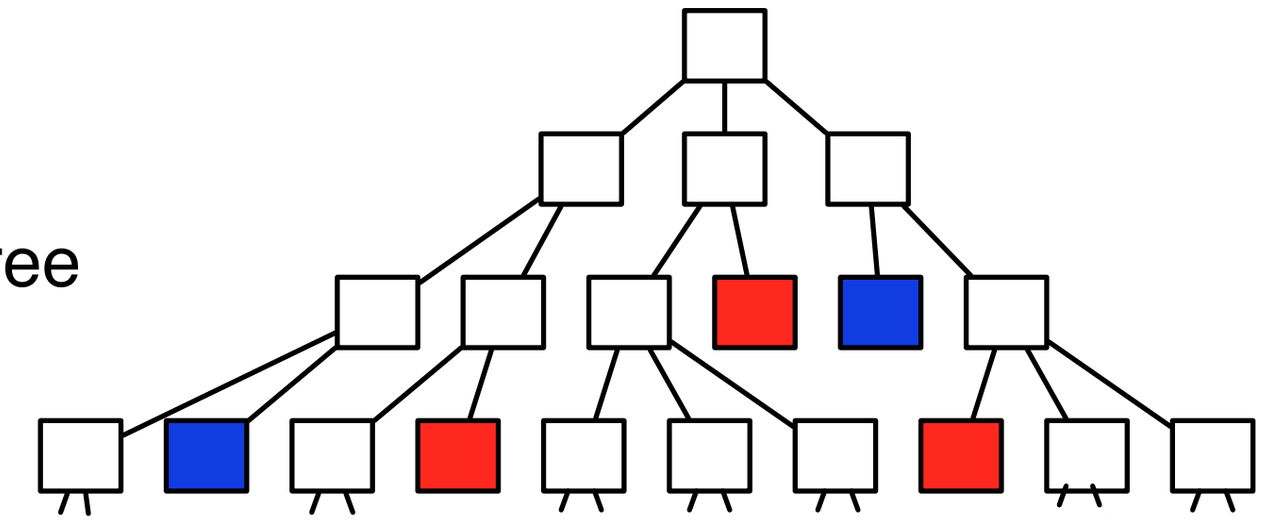
- Hypothesis space is a ***complete*** space (any finite discrete valued function) that includes target function
- ***Hill-climbing (greedy) search*** with IG as the evaluation function
- ID3 is greedy so the search is ***incomplete*** and only retains current best hypothesis - no ***back-tracking*** to avoid local solutions
- ***Inductive bias:*** preference of shorter trees over larger trees and high IG attributes closer to the root
- Shorter trees = simpler hypothesis. Prefer the simplest hypothesis that explains the data. ***Occam's razor principle***:

Among competing hypotheses, prefer the one with the fewest assumptions.

Decision Tree Learning: Complexity & Overfitting

So how do we vary the complexity of our DTs?

Model Complexity: Depth of the tree

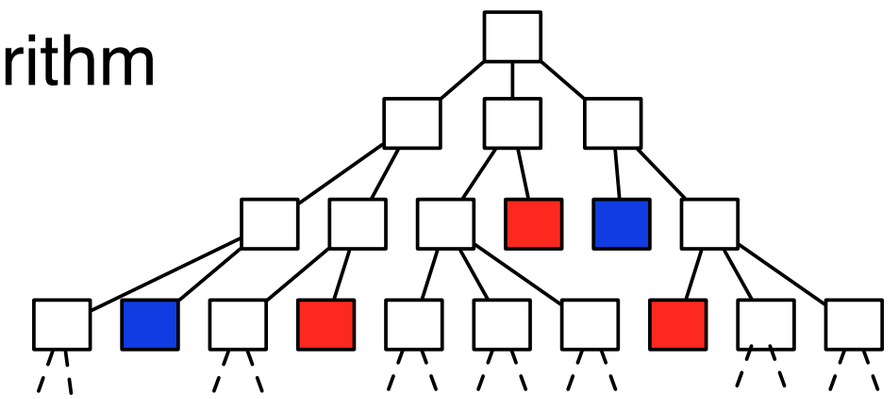


Two general approaches to avoid overfitting in trees:

1) ***Stop growing*** the tree earlier, before perfect classification on training data
2) Allow the tree to overfit the training data and then ***post-prune*** it

Decision Tree Learning: Complexity & Overfitting

Specifically for ID3 algorithm



Stopping Approach:

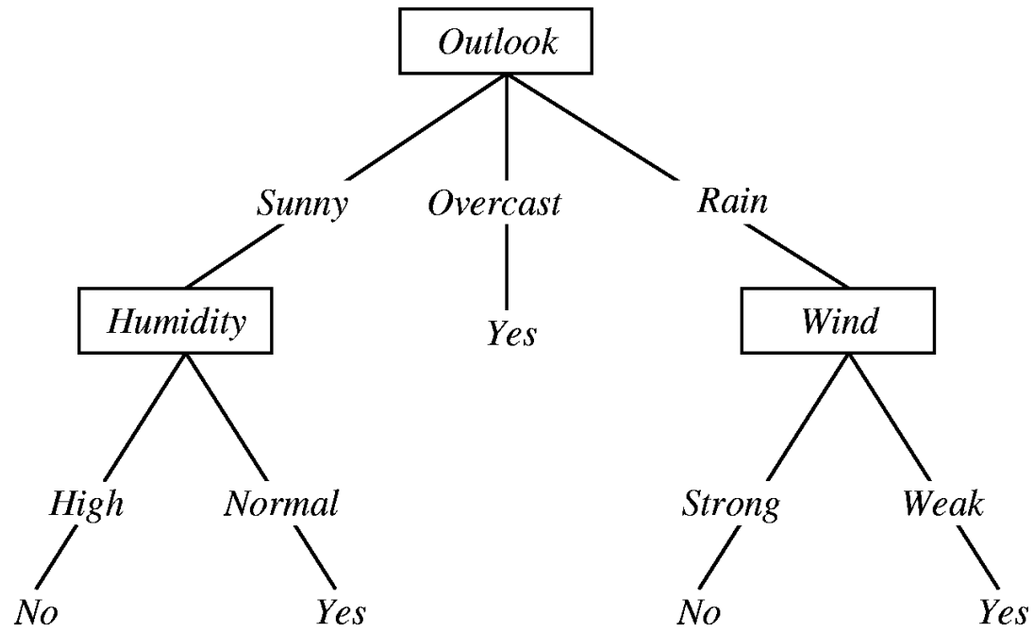
- Stop when no statistical significant change in Entropy (small IG)
- Can use a statistical test known as chi-square test (next lectures)

Pruning Approaches:

- ***"Reduced error pruning"***: Propose nodes for removal and use (cross-)validation set to examine effect. Node removals that are found to improve performance on validation set are removed iteratively.
- ***"Rule post-pruning"***: Convert tree into set of rules and prune each rule.

Decision Tree Learning: from Discrete to Continous

So far ID3 on discrete-valued attributes (e.g. Humidity: high or low)



Most real-world data has continuous attributes (e.g. Temperature)

What would you do?

Main idea: Threshold the continuous attribute to transform to discrete!
Next Lecture!