Decision Trees

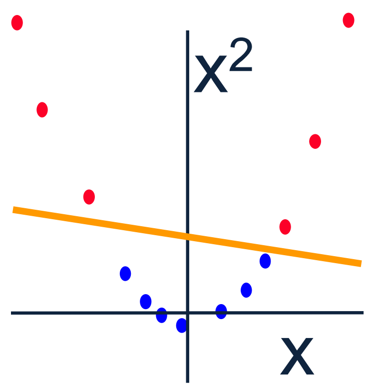
- Earlier, we decoupled the generation of the feature space from the learning.
- Argued that we can map the given examples into another space, in which the target functions are linearly separable.

Think about the Badges problem

- Do we always want to do it?
- How do we determine what are good mappings?

What's the best learning algorithm?

- The study of decision trees may shed some light on this.
- Learning is done directly from the given data representation.
- The algorithm ``transforms" the data itself.



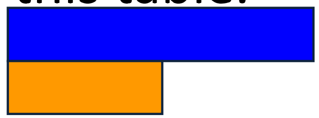
This Lecture

- Decision trees for (binary) classification
  - Non-linear classifiers
- Learning decision trees (ID3 algorithm)
  - Greedy heuristic (based on information gain)
    Originally developed for discrete features
  - Some extensions to the basic algorithm
- Overfitting
  - Some experimental issues

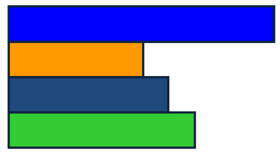
Representing Data

Representation

- Think about a large table, N attributes, and assume you want to know something about the people represented as entries in this table.
- E.g. own an expensive car or not;



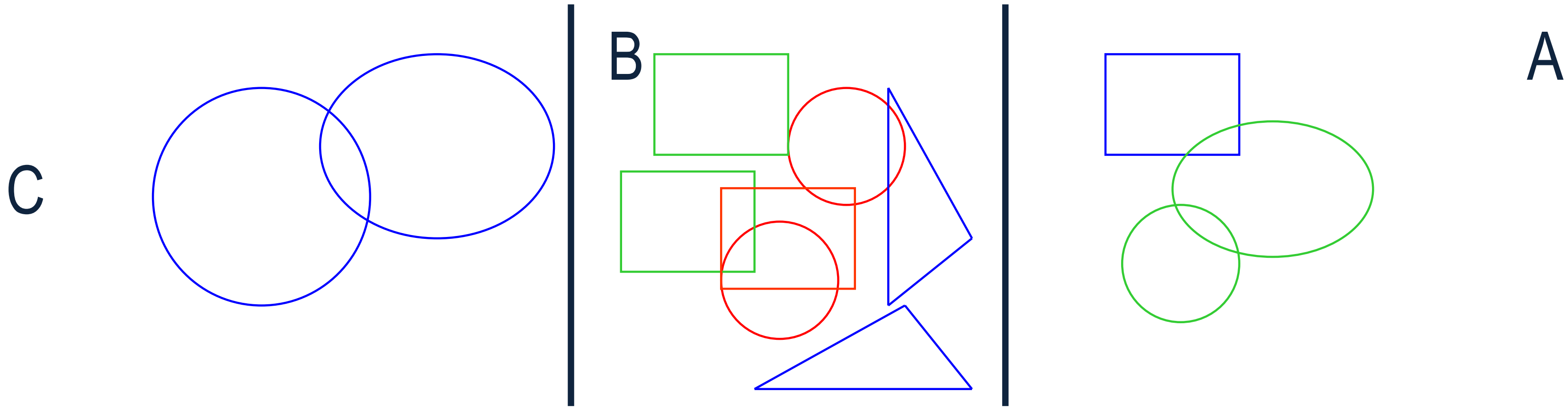
- Simplest way: Histogram on the first attribute – own
- Then, histogram on first and second (own & gender)



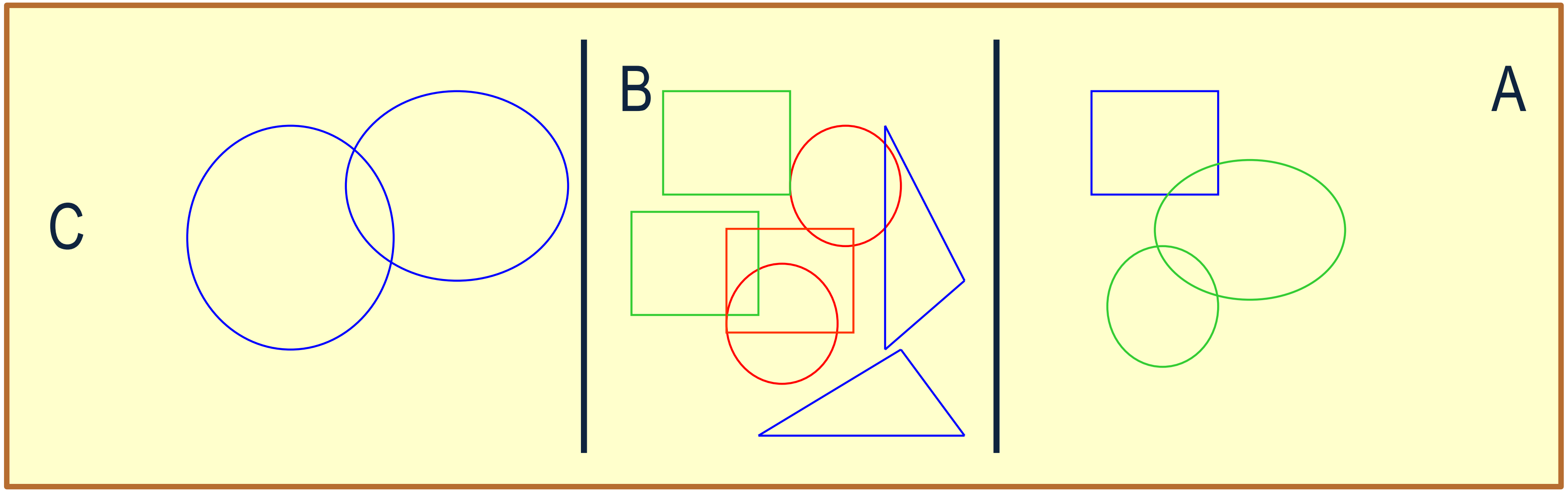
- But, what if the # of attributes is larger: N=16
- How large are the 1-d histograms (contingency tables) ? 16 numbers
- How large are the 2-d histograms? 16-choose-2 = 120 numbers
- How many 3-d tables? 560 numbers
- With 100 attributes, the 3-d tables need 161,700 numbers
  - **We need to figure out a way to represent data in a better way, and figure out what are the important attributes to look at first.**
  - **Information theory has something to say about it – we will use it to better represent the data.**

Decision Trees

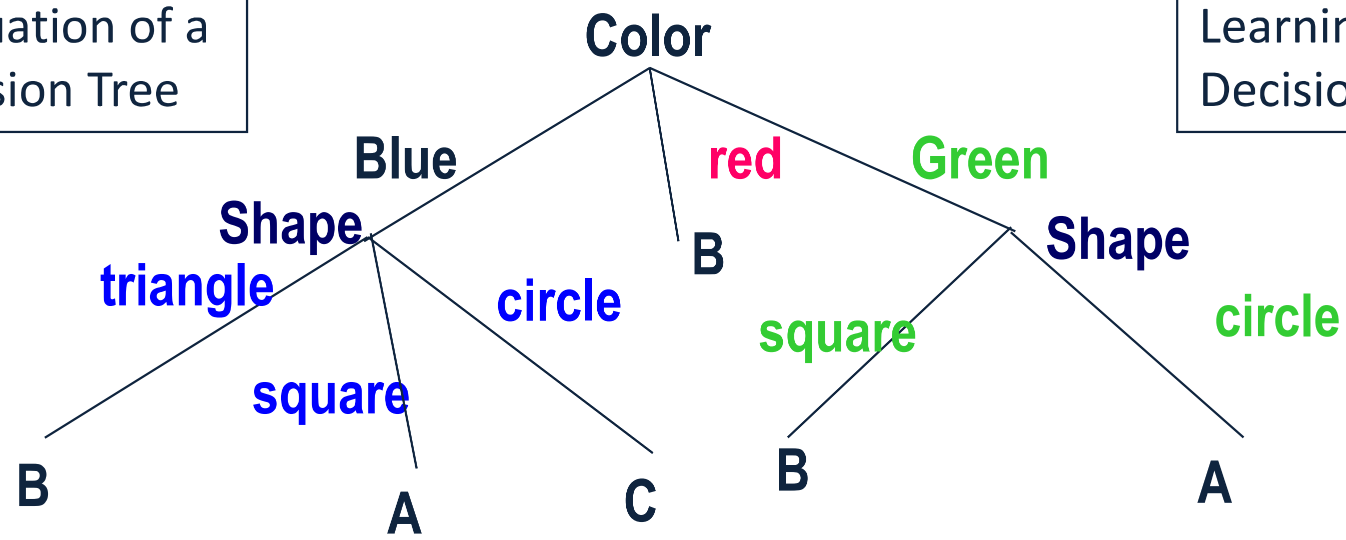
- A hierarchical data structure that represents data by implementing a divide and conquer strategy
- Can be used as a non-parametric classification and regression method
- Given a collection of examples, **learn a decision tree that represents it.**
- Use this representation to **classify new examples**



The Representation



Evaluation of a Decision Tree

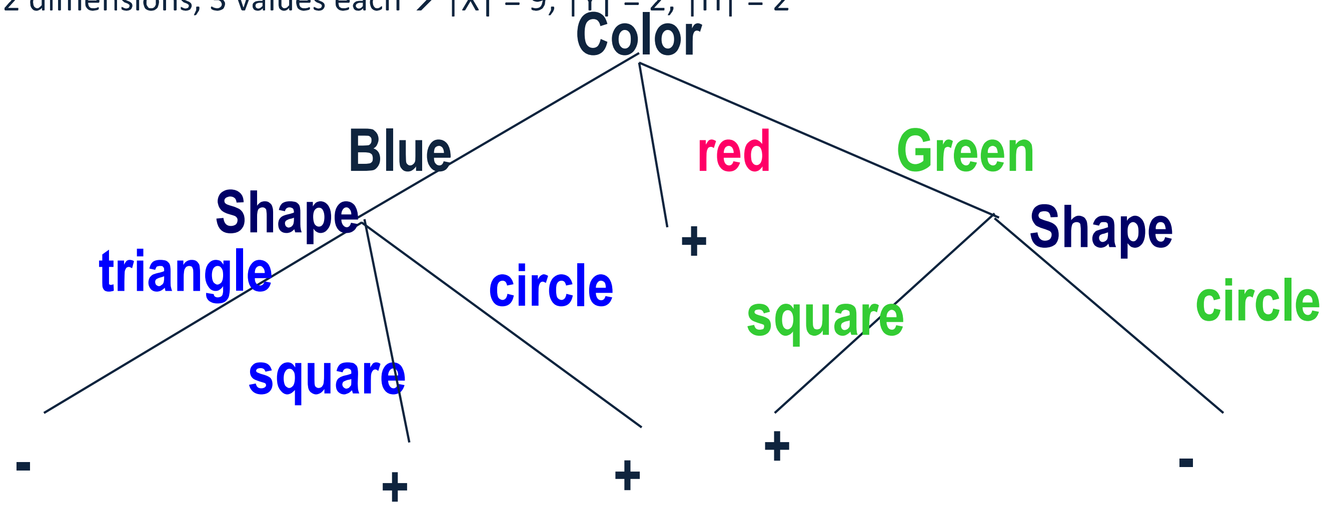


Learning a Decision Tree

Expressivity of Decision Trees

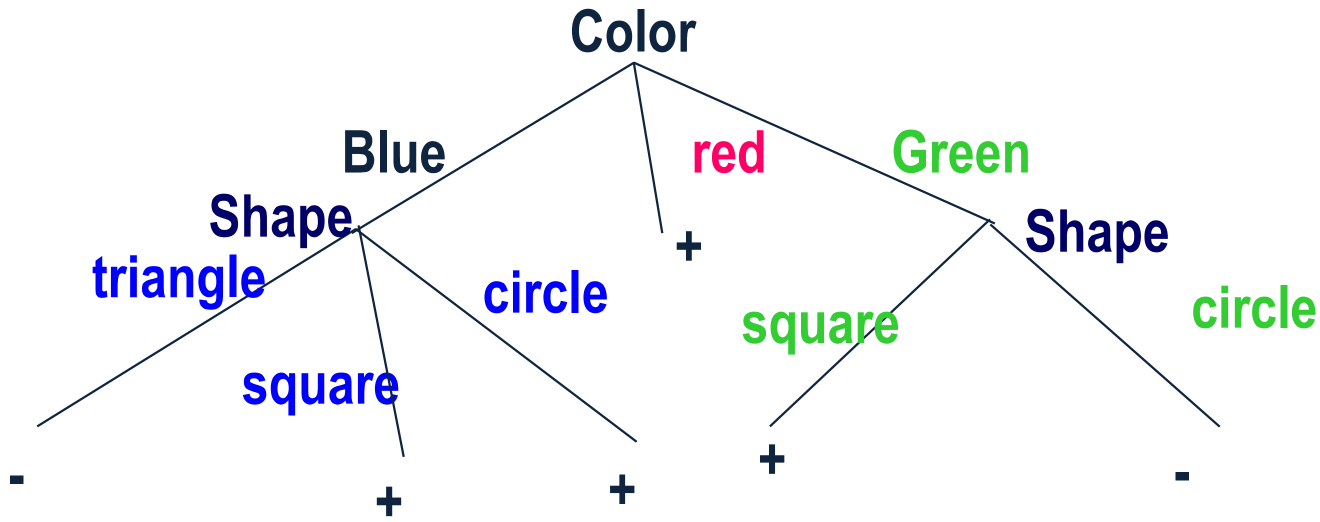
Decision Trees

- 2 dimensions, 3 values each→ |X| = 9; |Y| = 2; |H| = 2^9



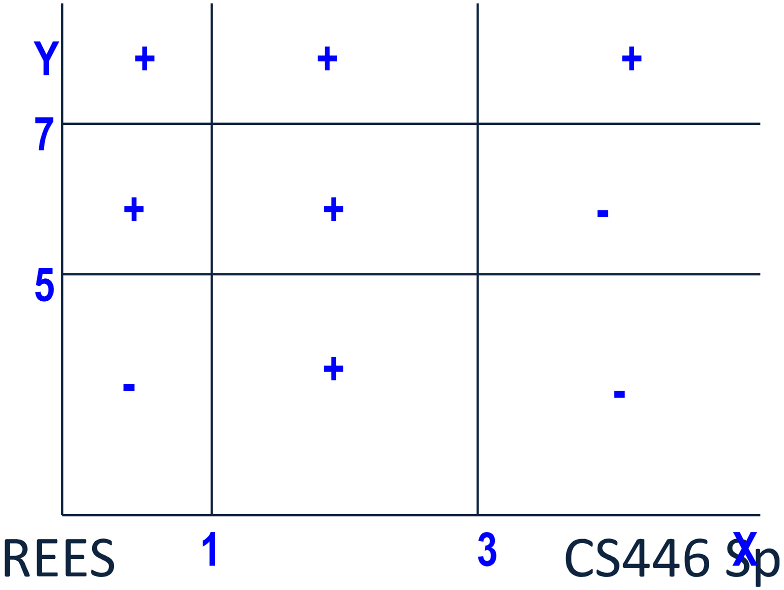
Decision Trees

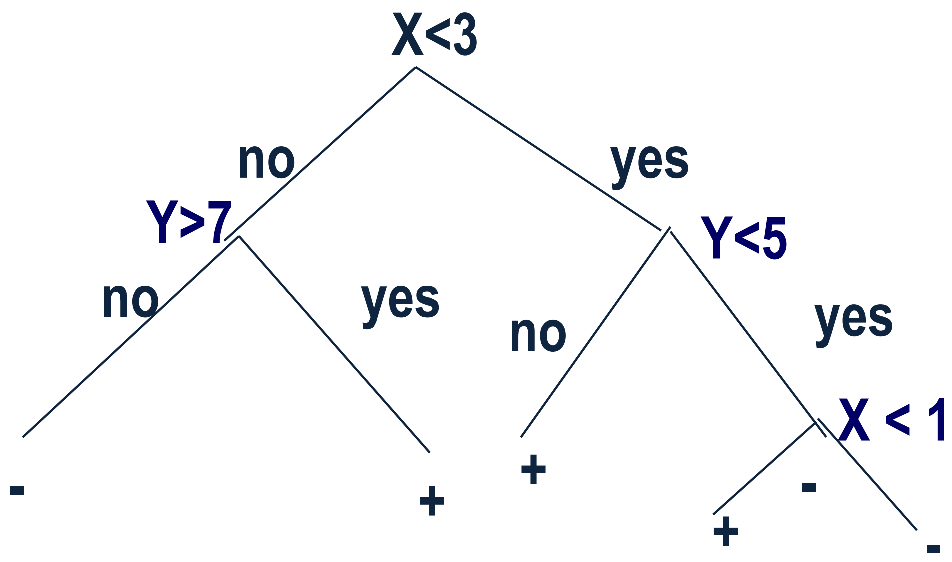
- Output is a discrete category. Real valued outputs are possible (regression trees)
- There are efficient algorithms for processing large amounts of data (but not too many features)
- There are methods for handling **noisy data** (classification noise and attribute noise) and for handling missing attribute values



Decision Boundaries

- Usually, instances are represented as attribute-value pairs (color=blue, shape = square, +)
- Numerical values can be used either by discretizing or by using thresholds for splitting nodes
- In this case, the tree divides the features space into axis-parallel rectangles, each labeled with one of the labels

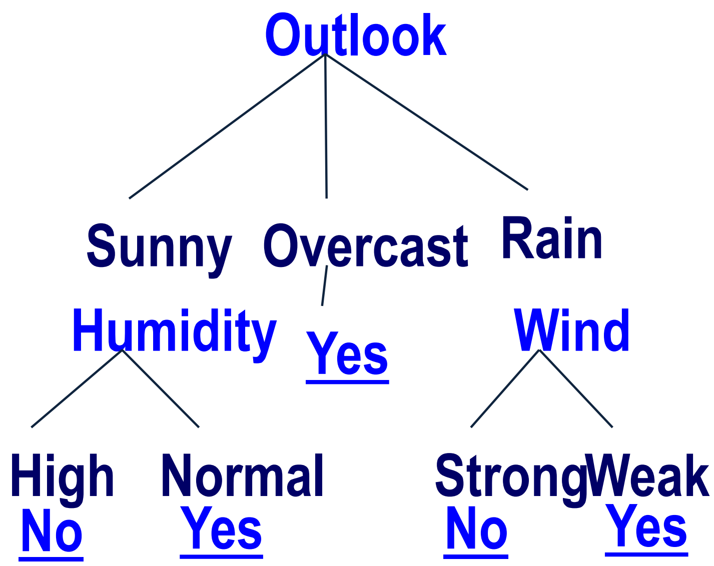




Decision Trees

- Can represent any Boolean Function
- Can be viewed as a way to compactly represent a lot of data.
- Natural representation: (20 questions)
- The evaluation of the Decision Tree Classifier is easy

- Clearly, given data, there are many ways to represent it as a decision tree.
- Learning a good representation from data is the challenge.



Will I play tennis today?

- **Features**
  - Outlook: {Sun, Overcast, Rain}
  - Temperature: {Hot, Mild, Cool}
  - Humidity: {High, Normal, Low}
  - Wind: {Strong, Weak}

- **Labels**
  - Binary classification task: Y = {+, -}

Will I play tennis today?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Outlook: S(unny), O(vercast), R(ainy)

Temperature: H(ot), M(edium), C(ool)

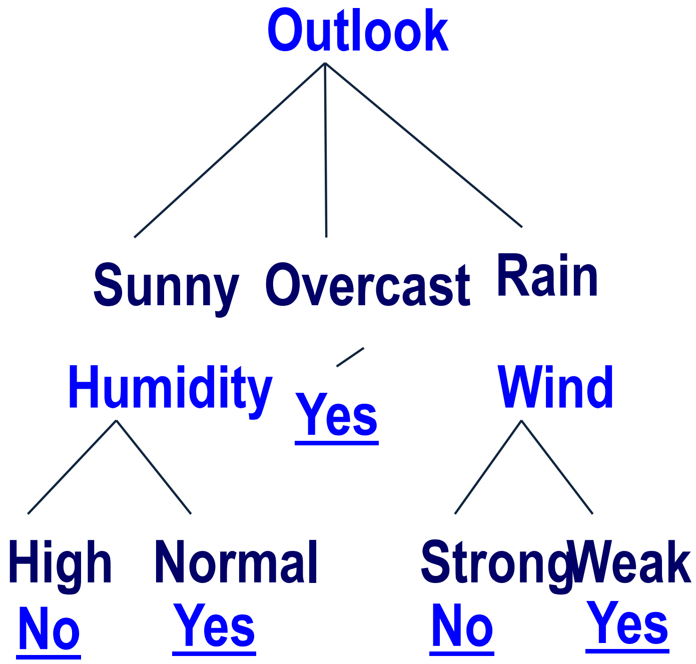
Humidity: H(igh), N(ormal), L(ow)

Wind: S(trong), W(eak)

Basic Decision Trees Learning Algorithm

- Data is processed in Batch (i.e. all the data available) Algorithm?
- Recursively build a decision tree top down.

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-



Basic Decision Tree Algorithm

ID3

- Let S be the set of Examples
  - Label is the target attribute (the prediction)
  - Attributes is the set of measured attributes
- ID3(*S*, Attributes, Label)

If all examples are labeled the same return a single node tree with Label

Otherwise Begin

A = attribute in Attributes that *best* classifies S (Create a Root node for tree)

for each possible value v of A

Add a new tree branch corresponding to A=v

Let *Sv* be the subset of examples in S with A=v

if *Sv* is empty: add leaf node with the common value of Label in S

why?

Else: below this branch add the subtree

For evaluation time

ID3(*Sv*, Attributes - {a}, Label)

End

Return Root

Picking the Root Attribute

- The goal is to have the resulting decision tree as small as possible (Occam's Razor)
  - But, finding the minimal decision tree consistent with the data is NP-hard
- The recursive algorithm is a greedy heuristic search for a simple tree, but cannot guarantee optimality.
- The main decision in the algorithm is the selection of the next attribute to condition on.

Picking the Root Attribute

Consider data with two Boolean attributes (A,B).

< (A=0,B=0), - >: 50 examples

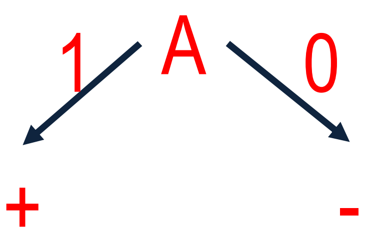
< (A=0,B=1), - >: 50 examples

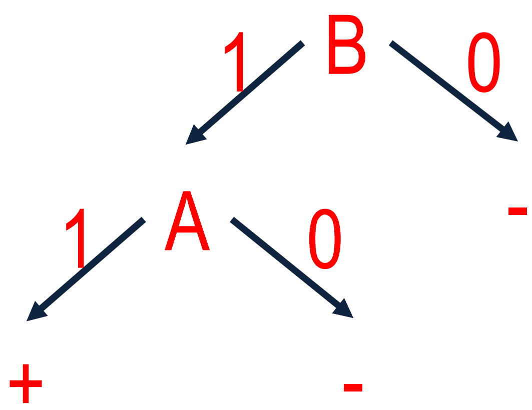
< (A=1,B=0), - >: 0 examples

< (A=1,B=1), + >: 100 examples

What should be the first attribute we select?

Splitting on A: we get purely labeled nodes.





Splitting on B: we don't get purely labeled nodes.

What if we have: <(A=1,B=0), - >: 3 examples

Picking the Root Attribute

Consider data with two Boolean attributes (A,B).

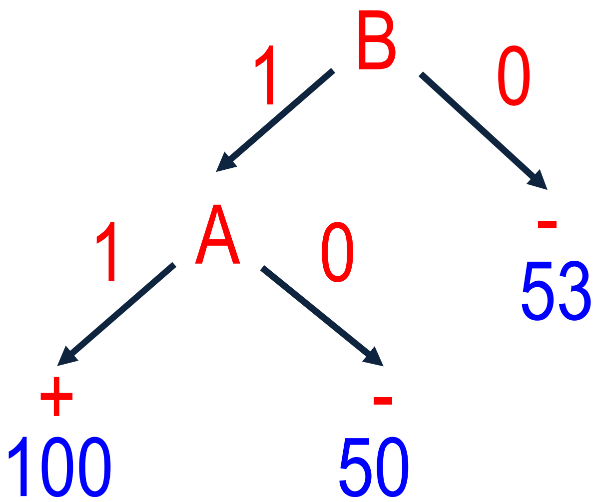
< (A=0,B=0), - >: 50 examples

< (A=0,B=1), - >: 50 examples

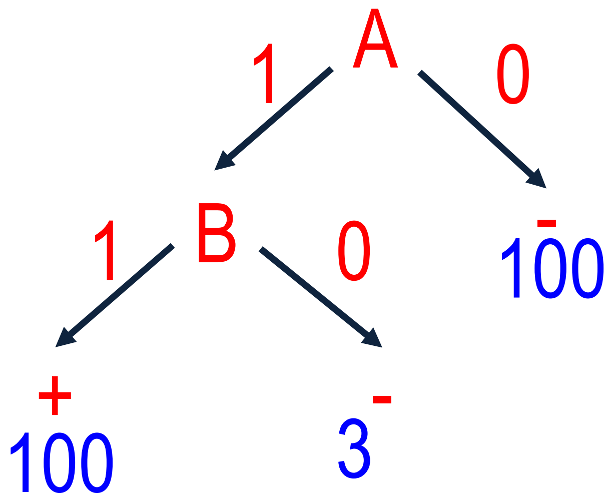
< (A=1,B=0), - >: ~~0 examples~~ 3 examples

< (A=1,B=1), + >: 100 examples

Trees looks structurally similar; which attribute should we choose?



Advantage A. But…
Need a way to quantify things



Picking the Root Attribute

- The goal is to have the resulting decision tree as small as possible (Occam's Razor)
- The main decision in the algorithm is the selection of the next attribute to condition on.
- We want attributes that split the examples to sets that are relatively pure in one label; this way we are closer to a leaf node.
- The most popular heuristics is based on information gain, originated with the ID3 system of Quinlan.

Entropy

- Entropy (impurity, disorder) of a set of examples, S, relative to a binary classification is:

$$\text{Entropy(S)} = -p_{+}\log(p_{+}) - p_{-}\log(p_{-})$$

- where P_+ is the proportion of positive examples in S and P_- is the proportion of negatives.
  - If all the examples belong to the same category: Entropy = 0
  - If all the examples are equally mixed (0.5, 0.5): Entropy = 1
  - Entropy = Level of uncertainty.

In general, when p_i is the fraction of examples labeled i:

$$\text{Entropy}(\{p_1, p_2, \ldots p_k\}) = -\sum_{i=1}^{k} p_i \log(p_i)$$

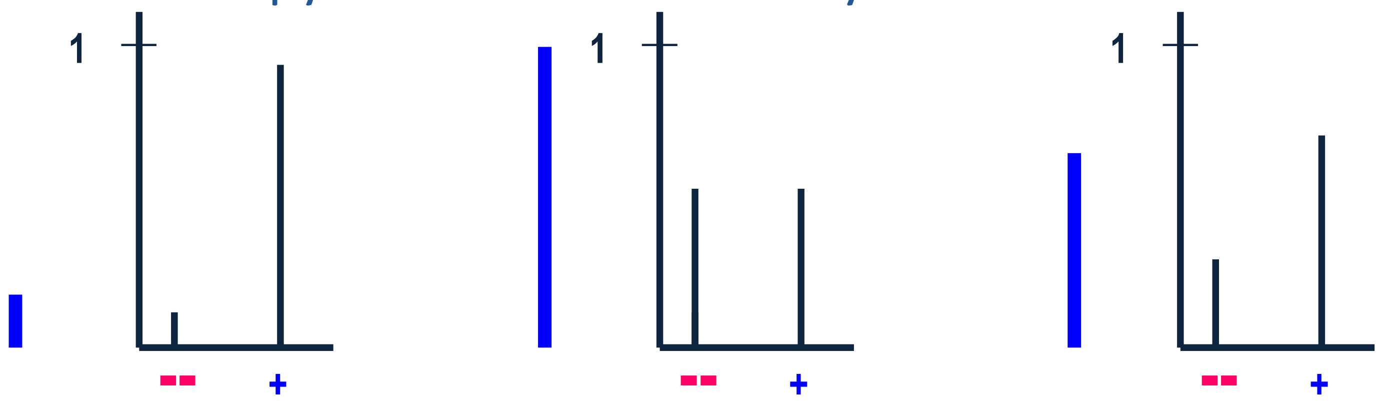
Entropy can be viewed as the number of bits required, on average, to encode the class of labels. If the probability for + is 0.5, a single bit is required for each example; if it is 0.8 -- can use less then 1 bit.

Entropy

- Entropy (impurity, disorder) of a set of examples, S, relative to a binary classification is:

$$\text{Entropy}(S) = -p_{+}\log(p_{+}) - p_{-}\log(p_{-})$$

- where P_+ is the proportion of positive examples in S and P_- is the proportion of negatives.
  - If all the examples belong to the same category: Entropy = 0
  - If all the examples are equally mixed (0.5, 0.5): Entropy = 1
  - Entropy = Level of uncertainty.



High Entropy – High level of Uncertainty

Low Entropy – No Uncertainty.

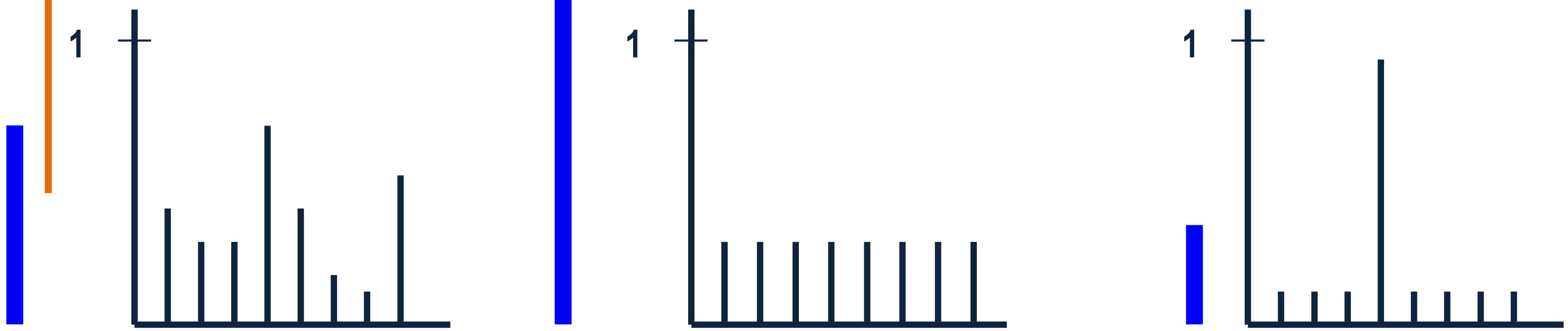
- Entropy (impurity, disorder) of a set of examples, S, relative to a binary classification is:

$$\text{Entropy}(S) = -p_{+}\log(p_{+}) - p_{-}\log(p_{-})$$

where p_{+} is the proportion of *positive* examples in S and p_{-} is the proportion of *negatives.*

If all the examples belong to the same category: Entropy = 0

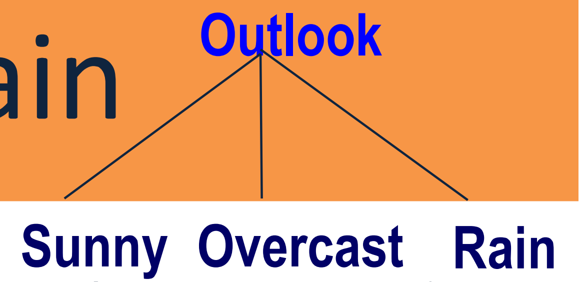
If all the examples are equally mixed (0.5, 0.5): Entropy = 1



High Entropy – High level of Uncertainty

Low Entropy – No Uncertainty.

Information Gain



- The information gain of an attribute a is the expected reduction in entropy caused by partitioning on this attribute

$$\text{Gain}(S,a) = \text{Entropy}(S) - \sum_{v \in \text{values}(a)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

- where S_v is the subset of S for which attribute a has value v, and the entropy of partitioning the data is calculated by weighing the entropy of each partition by its size relative to the original set
  - Partitions of low entropy (imbalanced splits) lead to high gain
- Go back to check which of the A, B splits is better

Will I play tennis today?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Outlook: S(unny), O(vercast), R(ainy)

Temperature: H(ot), M(edium), C(ool)

Humidity: H(igh), N(ormal), L(ow)

Wind: S(trong), W(eak)

Will I play tennis today?

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Current entropy:

$p = 9/14$

$n = 5/14$

$H(\mathrm{Y}) =$

$-(9/14)\log_2(9/14)$

$-(5/14)\log_2(5/14)$

≈ 0.94

Information Gain: Outlook

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Outlook = sunny:

$p = 2/5 \quad n = 3/5 \qquad \mathbf{H_S = 0.971}$

Outlook = overcast:

$p = 4/4 \quad n = 0 \qquad \mathbf{H_o = 0}$

Outlook = rainy:

$p = 3/5 \quad n = 2/5 \qquad \mathbf{H_R = 0.971}$

Expected entropy:

$(5/14)\times 0.971 + (4/14)\times 0$
$+ (5/14)\times 0.971 = \mathbf{0.694}$

Information gain:

$0.940 - 0.694 = \mathbf{0.246}$

Information Gain: Humidity

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Humidity = high:

$p = 3/7$ $n = 4/7$ $\mathbf{H_h = 0.985}$

Humidity = Normal:

$p = 6/7$ $n = 1/7$ $\mathbf{H_o = 0.592}$

Expected entropy:

(7/14)×0.985 + (7/14)×0.592= **0.7785**

Information gain:

0.940 – 0.151 **= 0.1515**

Which feature to split on?

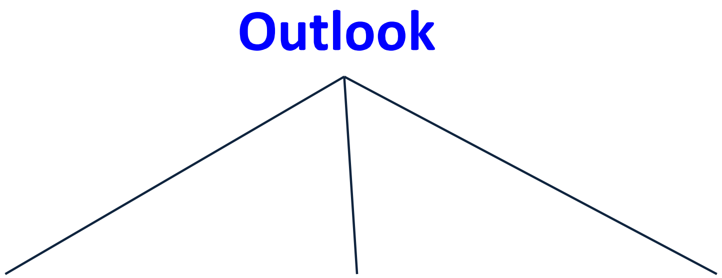
	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

Information gain:

Outlook: 0.246
Humidity: 0.151
Wind: 0.048
Temperature: 0.029

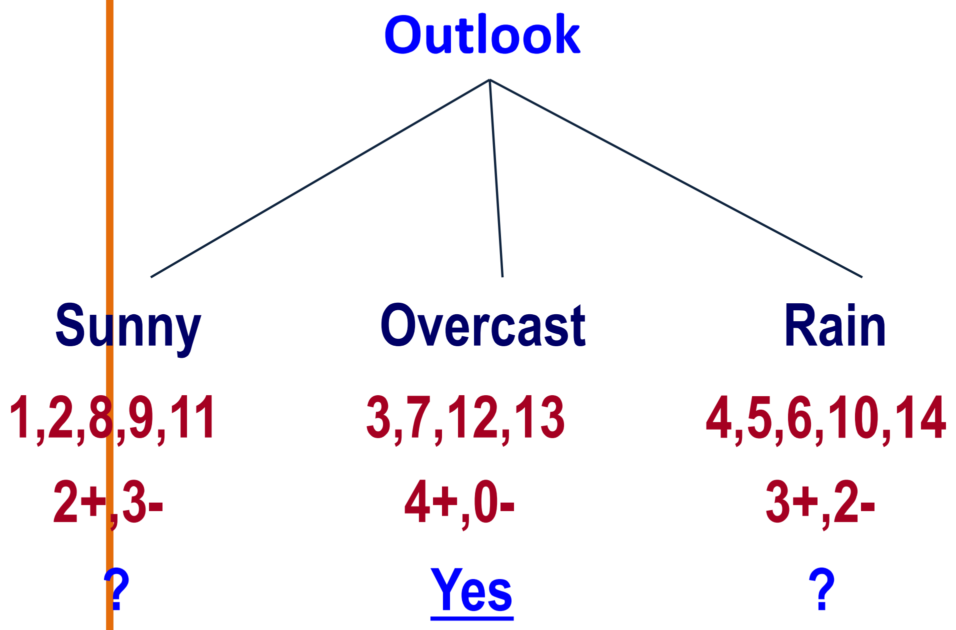
→ Split on Outlook

An Illustrative Example (III)



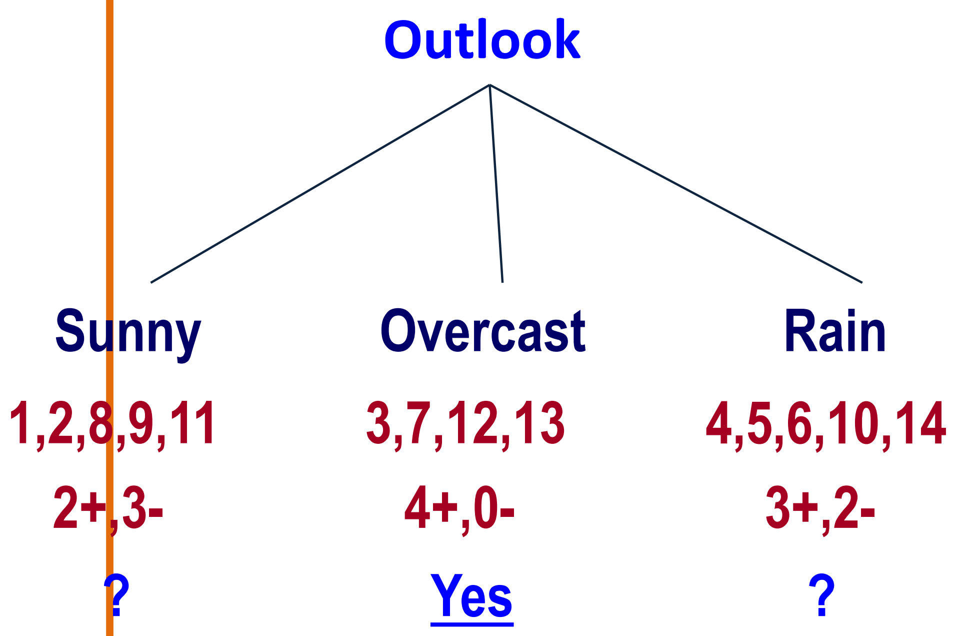
Gain(S,Humidity)=0.151
Gain(S,Wind) = 0.048
Gain(S,Temperature) = 0.029
Gain(S,Outlook) = 0.246

An Illustrative Example (III)



	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

An Illustrative Example (III)



Continue until:

- Every attribute is included in path, or,
- All examples in the leaf have same label

	O	T	H	W	Play?
1	S	H	H	W	-
2	S	H	H	S	-
3	O	H	H	W	+
4	R	M	H	W	+
5	R	C	N	W	+
6	R	C	N	S	-
7	O	C	N	S	+
8	S	M	H	W	-
9	S	C	N	W	+
10	R	M	N	W	+
11	S	M	N	S	+
12	O	M	H	S	+
13	O	H	N	W	+
14	R	M	H	S	-

An Illustrative Example (IV)

$Gain(S_{sunny}, Humidity) = .97-(3/5)\,0-(2/5)\,0 = .97$

$Gain(S_{sunny}, Temp) = .97-0-(2/5)\,1 = .57$

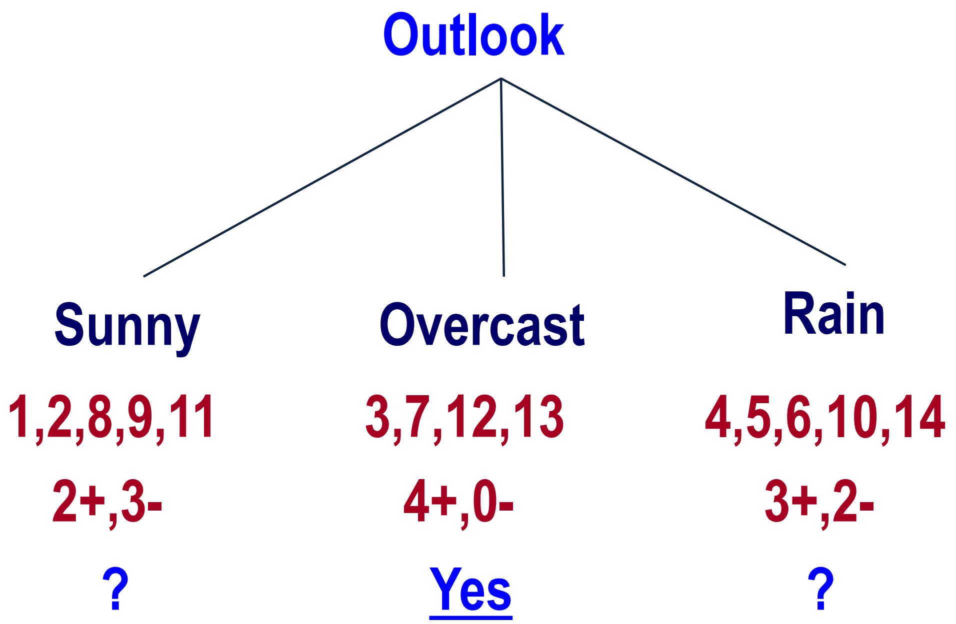
$Gain(S_{sunny}, Wind) = .97-(2/5)\,1-(3/5)\,.92 = .02$

Outlook

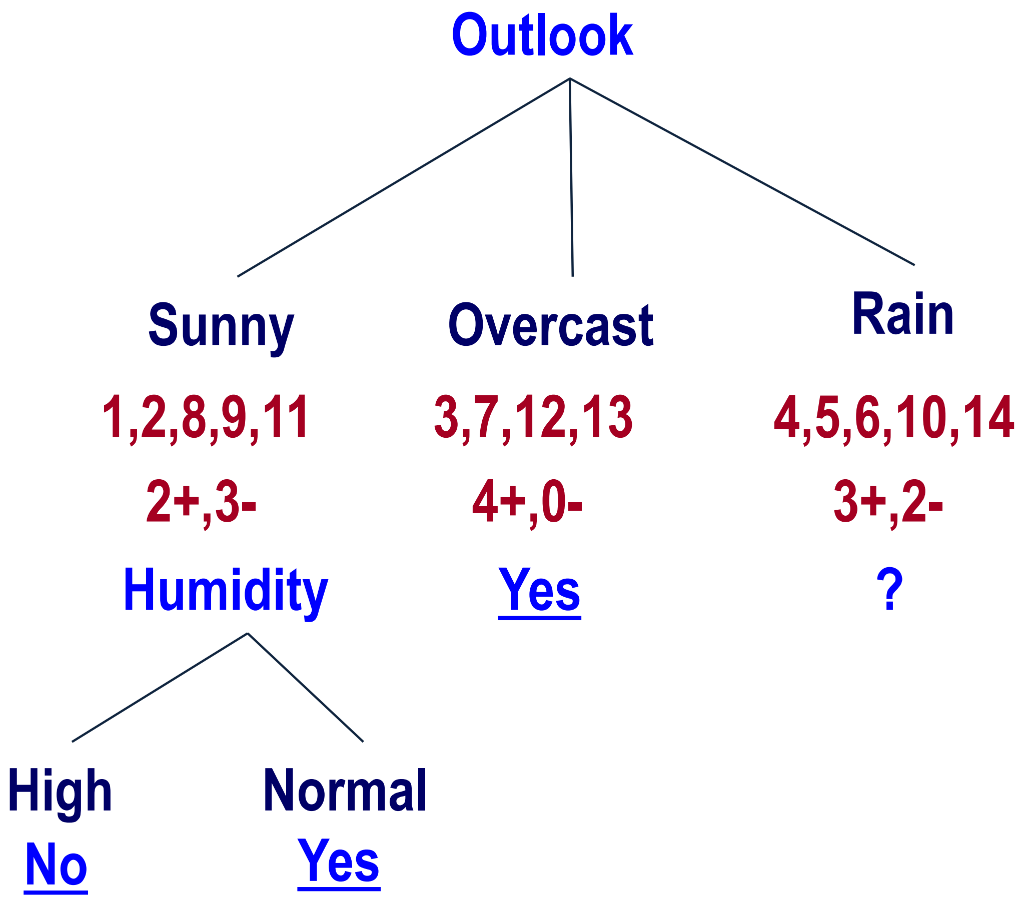
Sunny	Overcast	Rain
1,2,8,9,11	3,7,12,13	4,5,6,10,14
2+,3-	4+,0-	3+,2-
?	<u>Yes</u>	?

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	High	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

An Illustrative Example (V)



An Illustrative Example (V)



induceDecisionTree(S)

- 1. Does S uniquely define a class?

  if all $s \in S$ have the same label y: **return** S;

- 2. Find the feature with the most information gain:

  $i = \text{argmax}_{\,i}\ \mathit{Gain}(S, X_i)$

- 3. Add children to S:

  for *k* **in** $\text{Values}(X_i)$:

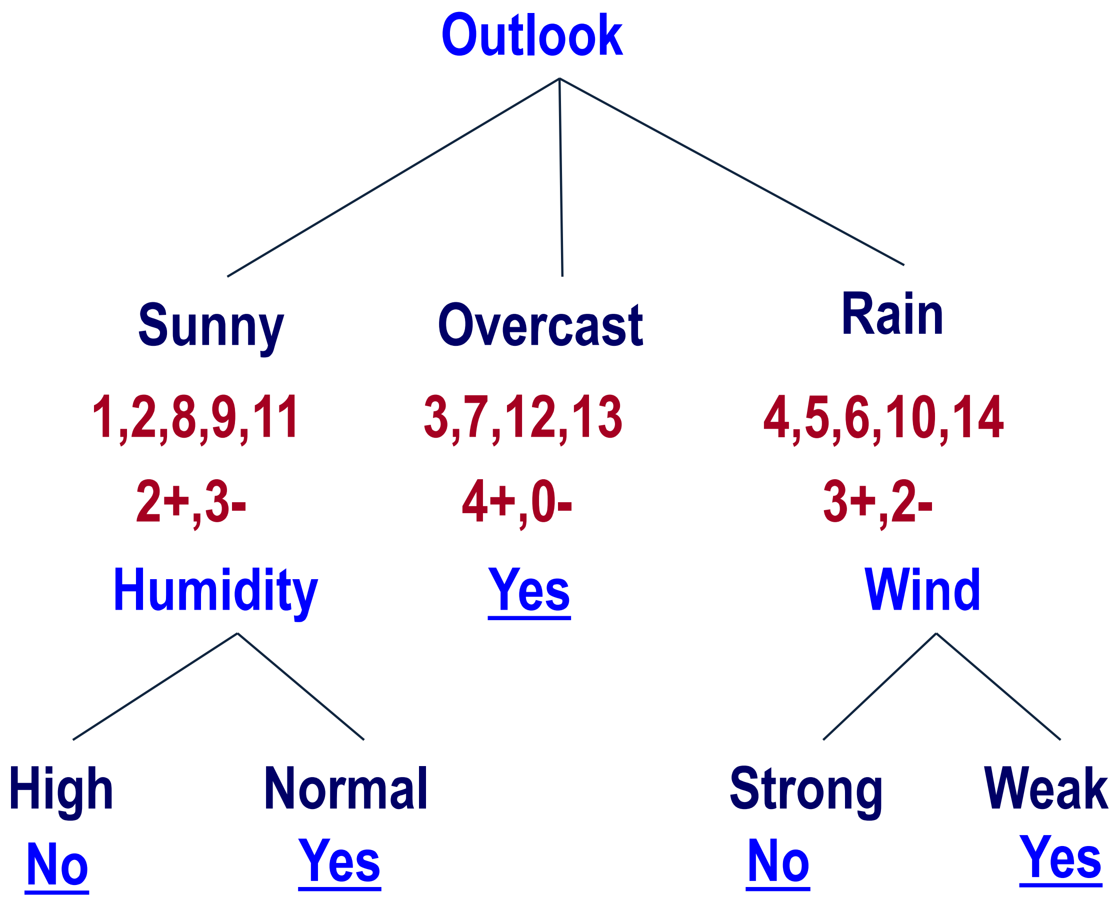
  $\quad S_k = \{s \in S \mid x_i = k\}$

  $\quad \text{addChild}(S, S_k)$

  $\quad \text{induceDecisionTree}(S_k)$

  return S;

An Illustrative Example (VI)



Hypothesis Space in Decision Tree Induction

- Conduct a search of the space of decision trees which can represent all possible discrete functions. (pros and cons)
- Goal: to find the best decision tree
  - Best could be “smallest depth”
  - Best could be “minimizing the expected number of tests”
- Finding a minimal decision tree consistent with a set of data is NP-hard.
- Performs a greedy heuristic search: hill climbing without backtracking
- Makes statistically based decisions using all data

Today's key concepts

- Learning decision trees (ID3 algorithm)
  - Greedy heuristic (based on information gain)
    Originally developed for discrete features
- Overfitting
  - What is it? How do we deal with it?

How can this be avoided with linear classifiers?

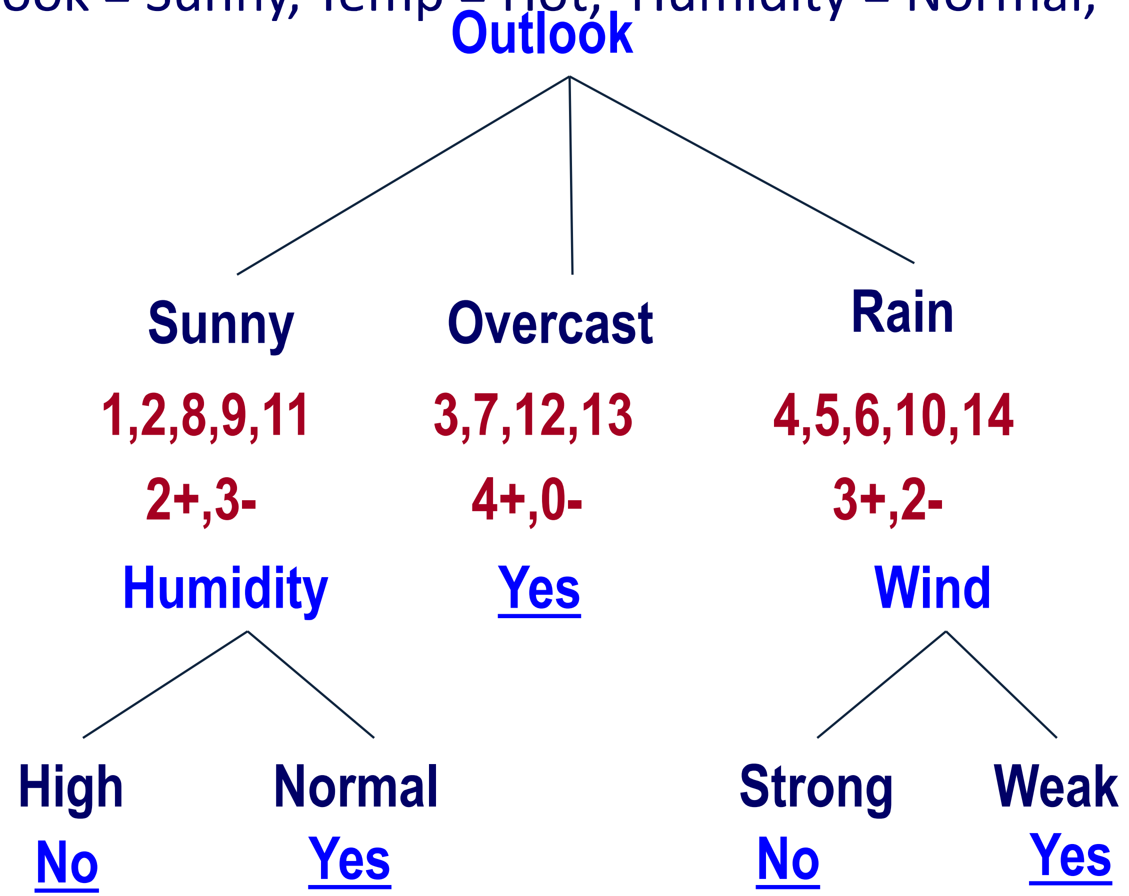
- Some extensions of DTs
- Principles of Experimental ML

History of Decision Tree Research

- Hunt and colleagues in Psychology used full search decision tree methods to model human concept learning in the 60s
- Quinlan developed ID3, with the information gain heuristics in the late 70s to learn expert systems from examples
- Breiman, Freidman and colleagues in statistics developed CART (classification and regression trees simultaneously)
- A variety of improvements in the 80s: coping with noise, continuous attributes, missing data, non-axis parallel etc.
- Quinlan's updated algorithm, C4.5 (1993) is commonly used (New: C5)
- Boosting (or Bagging) over DTs is a very good general purpose algorithm

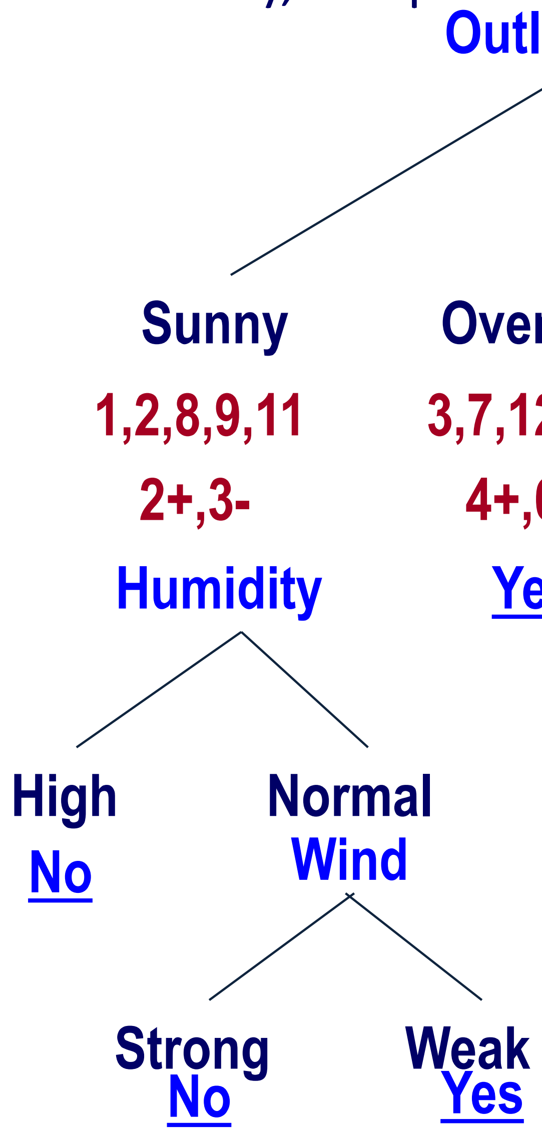
Example

Outlook = Sunny, Temp = Hot, Humidity = Normal, Wind = Strong, NO



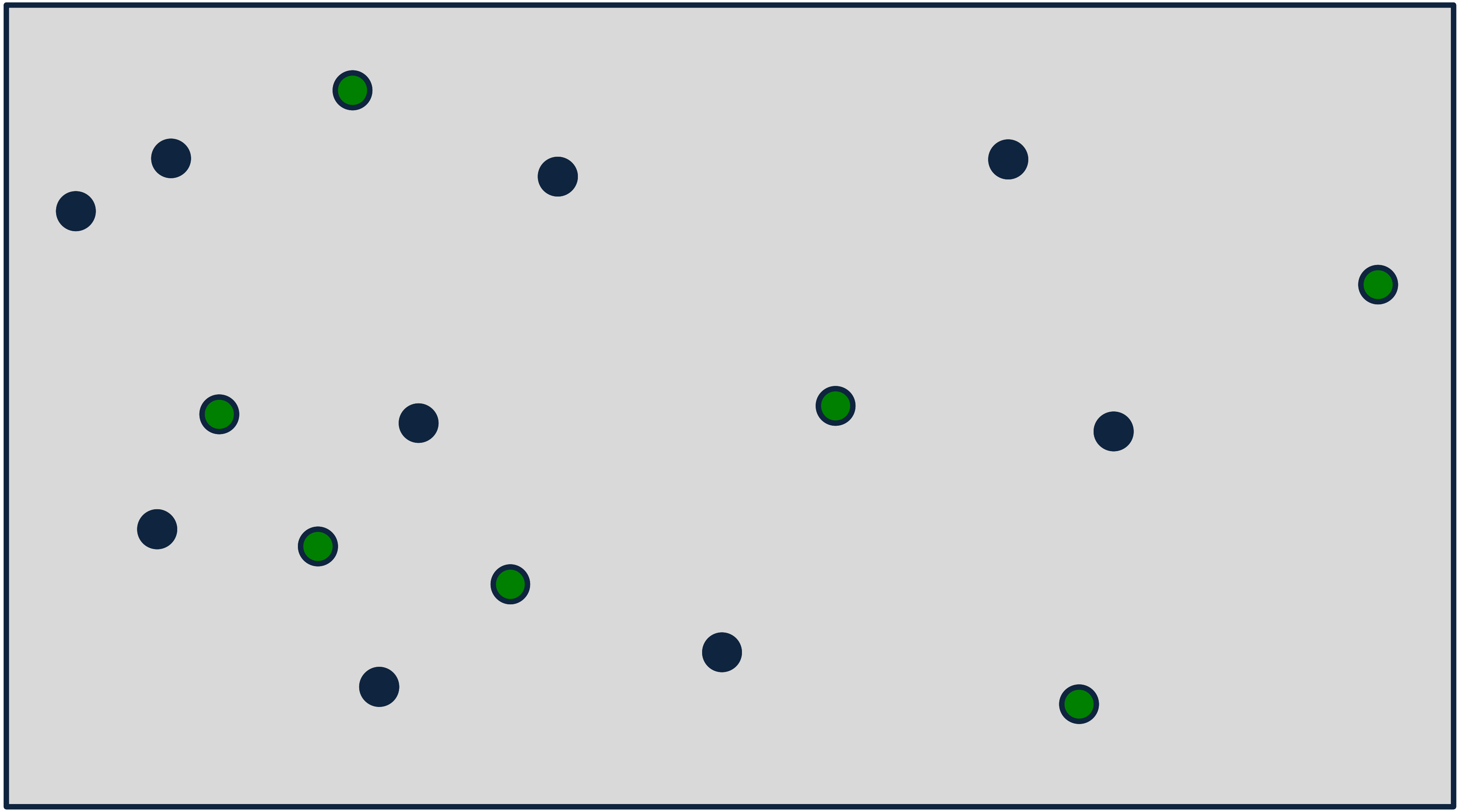
Overfitting - Example

Outlook = Sunny, Temp = Hot, Humidity = Normal, Wind = Strong, NO

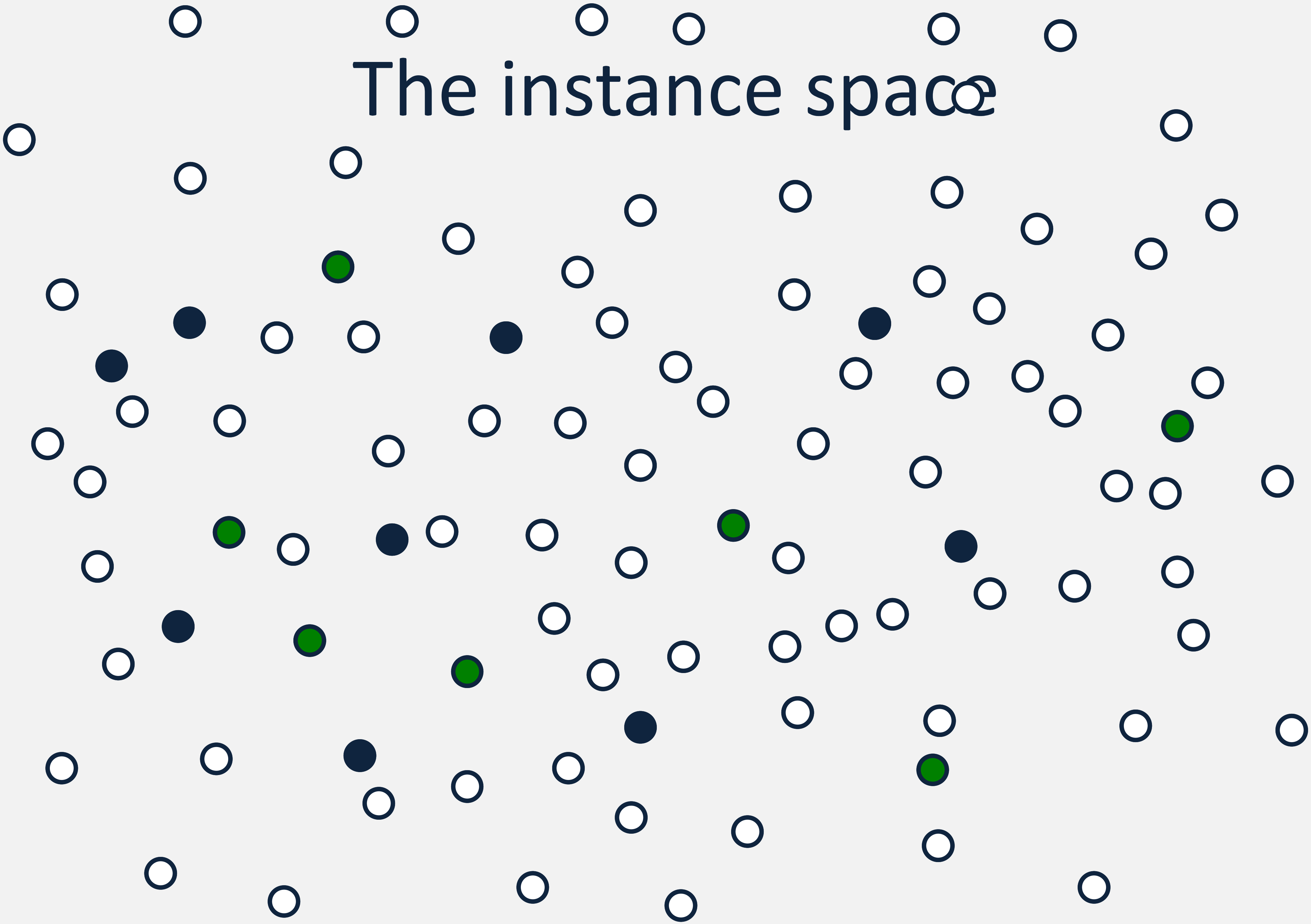


This can always be done – may fit noise or other coincidental regularities

Our training data

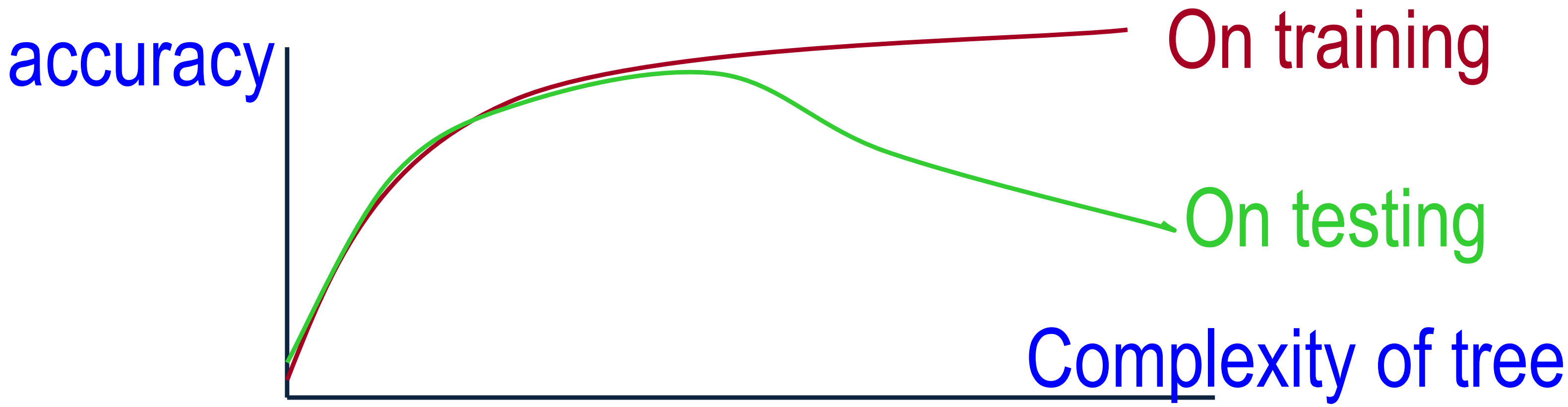


The instance space



Overfitting the Data

- Learning a tree that classifies the training data perfectly may not lead to the tree with the best generalization performance.
  - There may be noise in the training data the tree is fitting
  - The algorithm might be making decisions based on very little data
- A hypothesis h is said to overfit the training data if there is another hypothesis h', such that h has a smaller error than h' on the training data but h has larger error on the test data than h'.



Reasons for overfitting

- Too much variance in the training data
  - Training data is not a representative sample of the instance space
  - We split on features that are actually irrelevant
- Too much noise in the training data
  - Noise = some feature values or class labels are incorrect
  - We learn to predict the noise
- In both cases, it is a result of our will to minimize the empirical error when we learn, and the ability to do it (with DTs)

Pruning a decision tree

- Prune = remove leaves and assign majority label of the parent to all items
- Prune the children of S if:
  - all children are leaves, and
  - the accuracy on the validation set does not decrease if we assign the most frequent class label to all items at S.

Avoiding Overfitting

- Two basic approaches

How can this be avoided with linear classifiers?

  - Pre-pruning: Stop growing the tree at some point during construction when it is determined that there is not enough data to make reliable choices.
  - Post-pruning: Grow the full tree and then remove nodes that seem not to have sufficient evidence.
- Methods for evaluating subtrees to prune
  - Cross-validation: Reserve hold-out set to evaluate utility
  - Statistical testing: Test if the observed regularity can be dismissed as likely to occur by chance
  - Minimum Description Length: Is the additional complexity of the hypothesis smaller than remembering the exceptions?
- This is related to the notion of regularization that we will see in other contexts – keep the hypothesis simple.

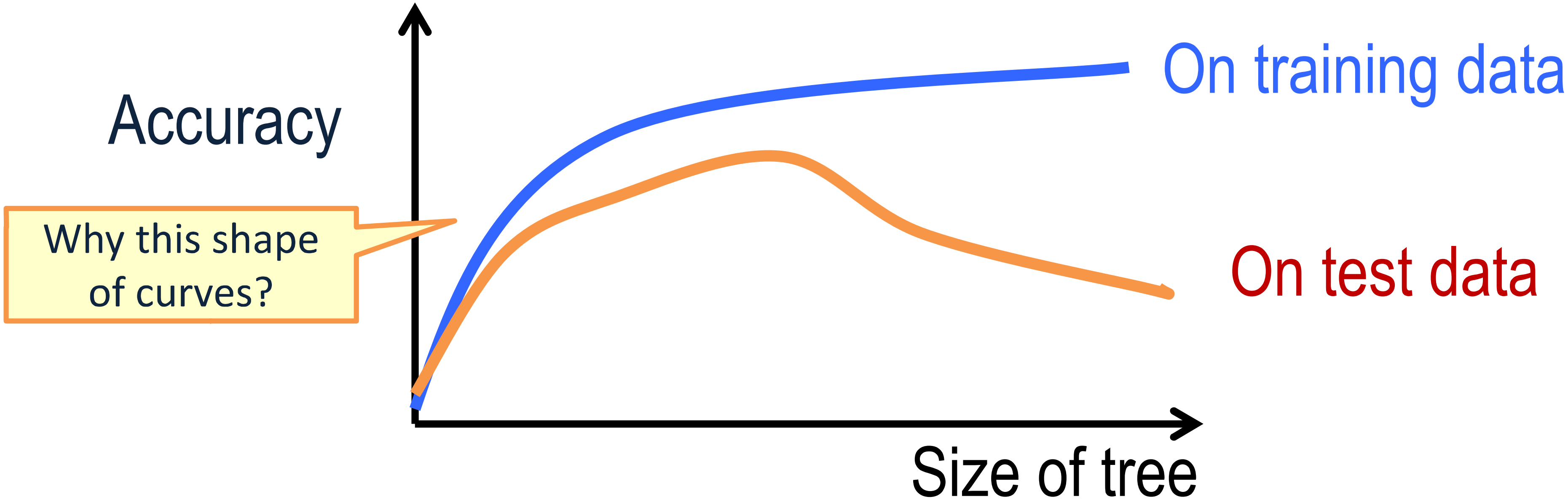
Hand waving, for now.

Next: a brief detour into explaining generalization and overfitting

The i.i.d. assumption

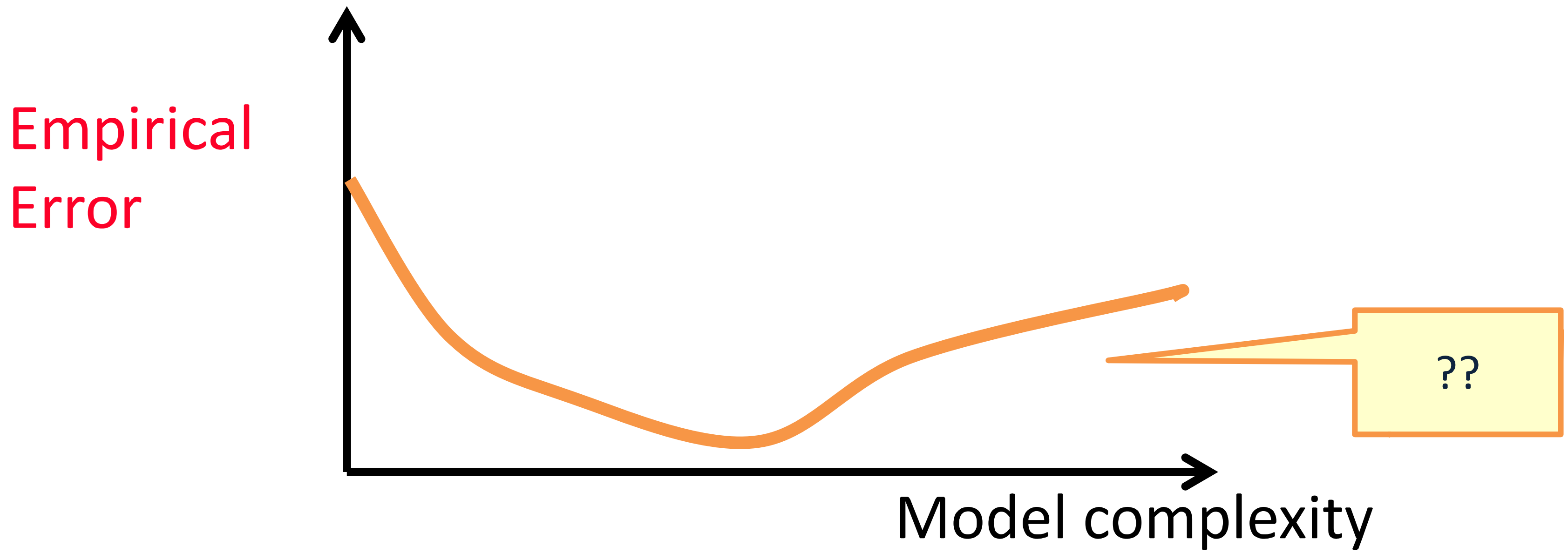
- Training and test items are independently and identically distributed (i.i.d.):
  - There is a distribution $P(\mathbf{X}, Y)$ from which the data $\mathcal{D} = \{(\mathbf{x}, y)\}$ is generated.
    - Sometimes it's useful to rewrite $P(\mathbf{X}, Y)$ as $P(\mathbf{X})P(Y|\mathbf{X})$
      Usually $P(\mathbf{X}, Y)$ is unknown to us (we just know it exists)
  - Training and test data are samples drawn from the *same* $P(\mathbf{X}, Y)$: they are identically distributed
  - Each $(\mathbf{x}, y)$ is drawn independently from $P(\mathbf{X}, Y)$

Overfitting



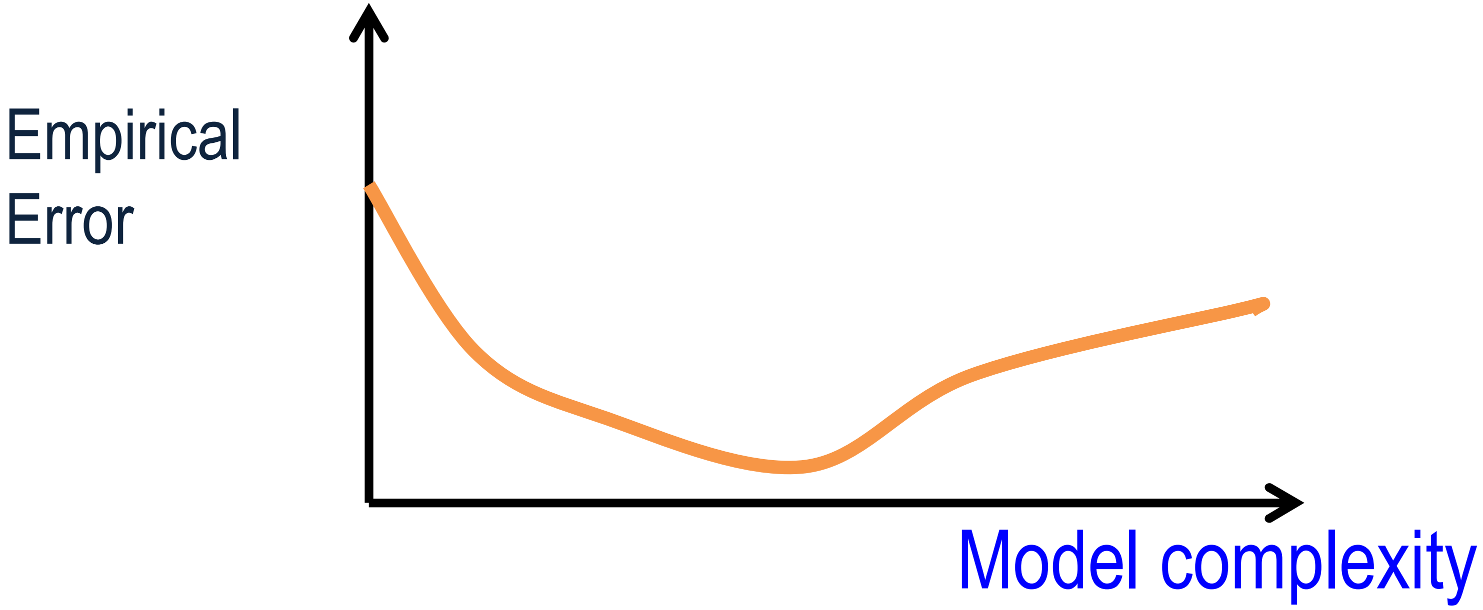
- A decision tree overfits the training data when its accuracy on the training data goes up but its accuracy on unseen data goes down

Overfitting



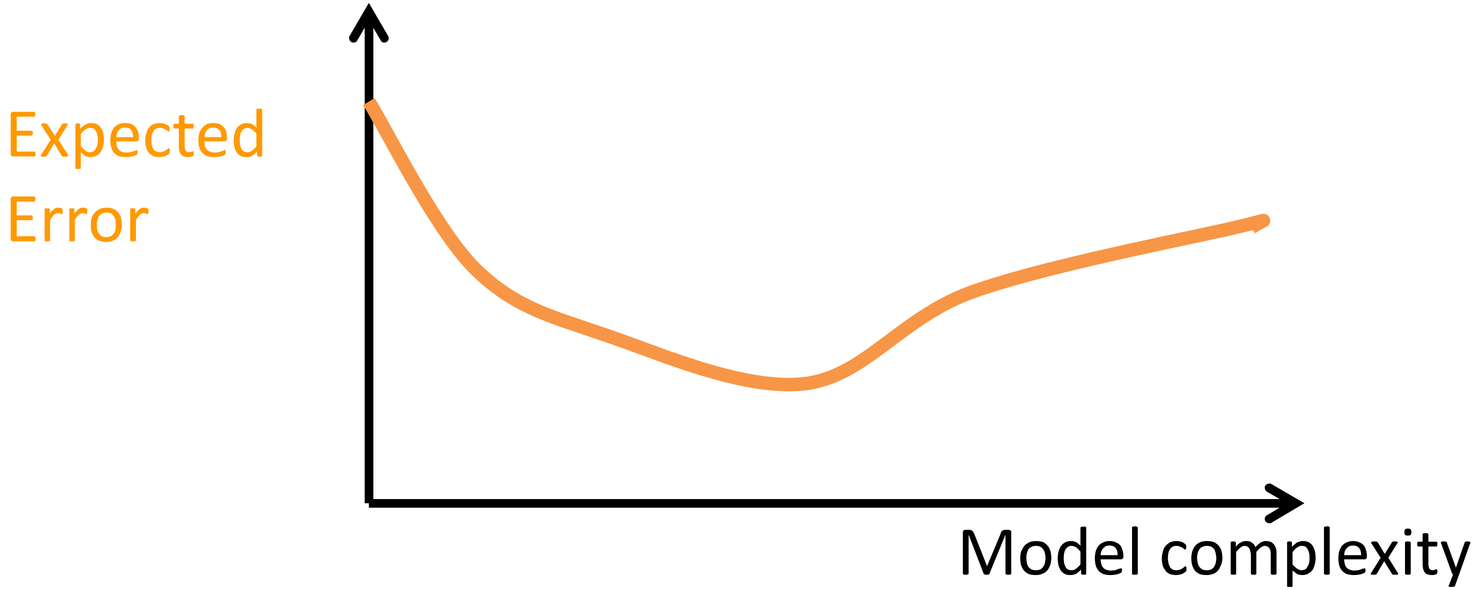
- Empirical error (= on a given data set):
  The percentage of items in this data set are misclassified by the classifier *f*.

Overfitting



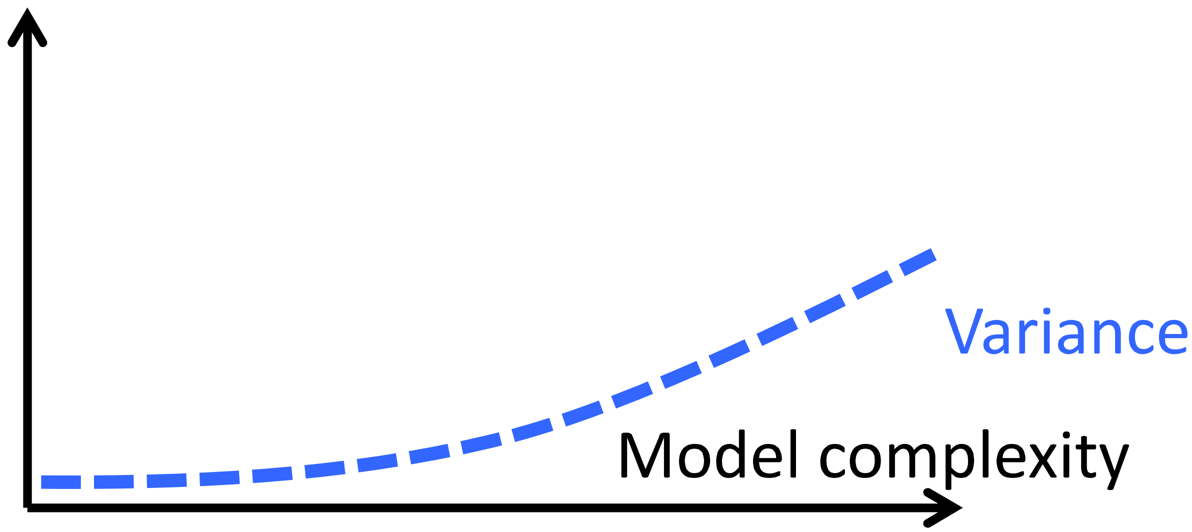
- Model complexity (informally):
  How many parameters do we have to learn?
  - Decision trees: complexity = #nodes

Overfitting



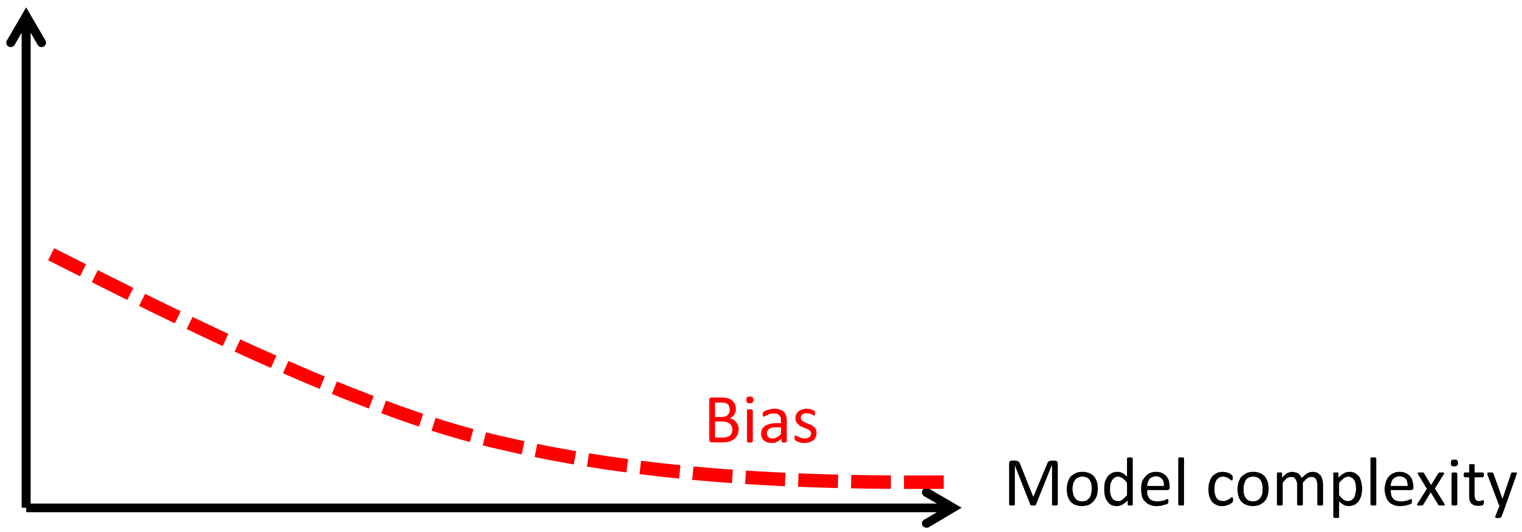
- Expected error:
  What percentage of items drawn from $P(\mathbf{x},y)$ do we expect to be misclassified by f?
- (That's what we really care about – generalization)

Variance of a learner (informally)



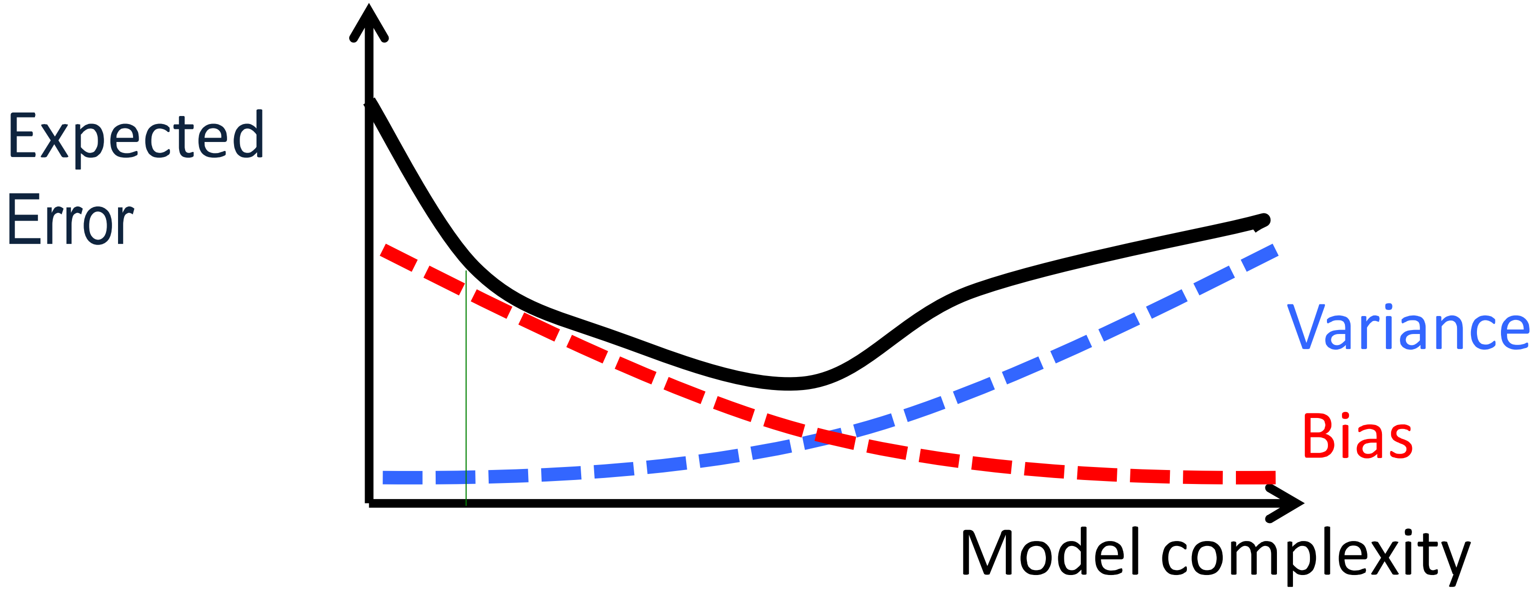
- How susceptible is the learner to minor changes in the training data?
  - (i.e. to different samples from P(X, Y))
- Variance increases with model complexity
  - Think about extreme cases: a hypothesis space with one function vs. all functions.
  - The larger the hypothesis space is, the more flexible the selection of the chosen hypothesis is as a function of the data.
  - More accurately: for each data set D, you will learn a different hypothesis h(D), that will have a different true error e(h); we are looking here at the variance of this random variable.

Bias of a learner (informally)



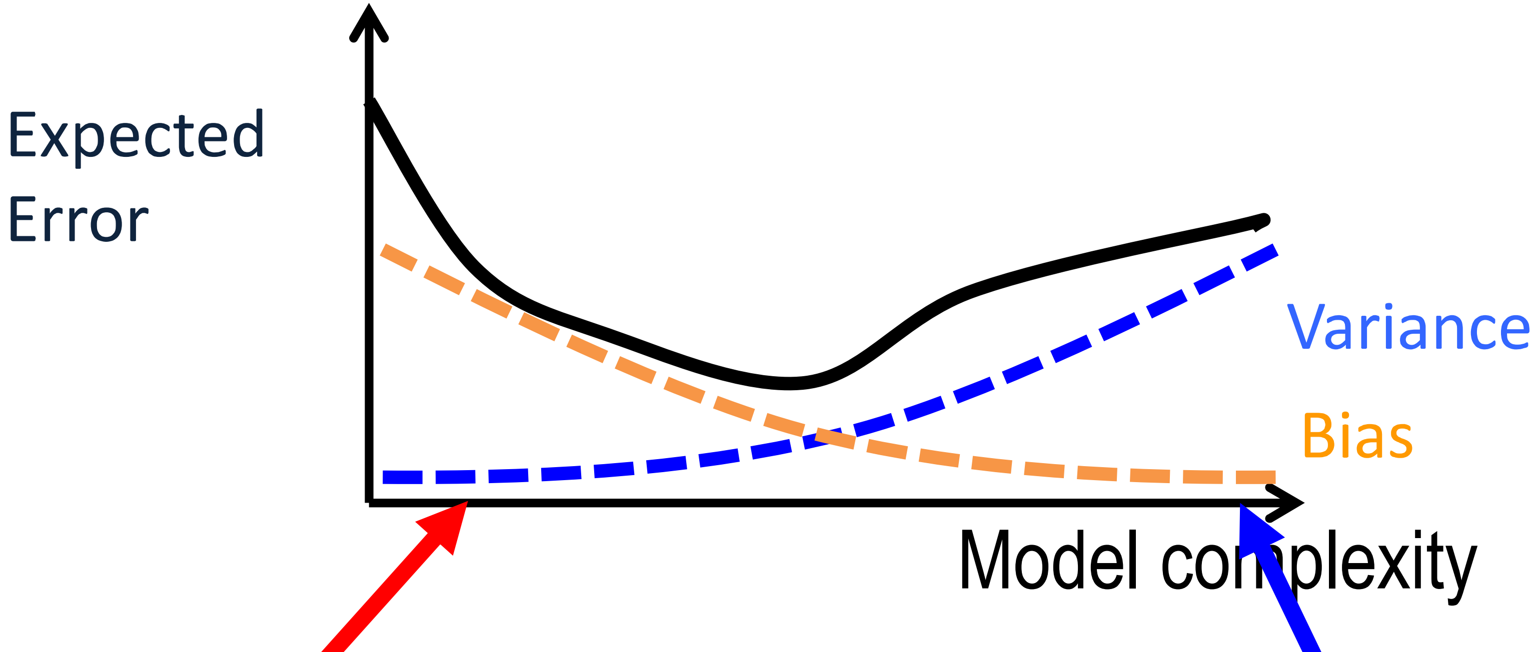
- How likely is the learner to identify the target hypothesis?
- Bias is low when the model is expressive (low empirical error)
- Bias is high when the model is (too) simple
  - The larger the hypothesis space is, the easiest it is to be close to the true hypothesis.
  - More accurately: for each data set D, you learn a different hypothesis h(D), that has a different true error e(h); we are looking here at the difference of the mean of this random variable from the true error.

Impact of bias and variance



- Expected error ≈ bias + variance

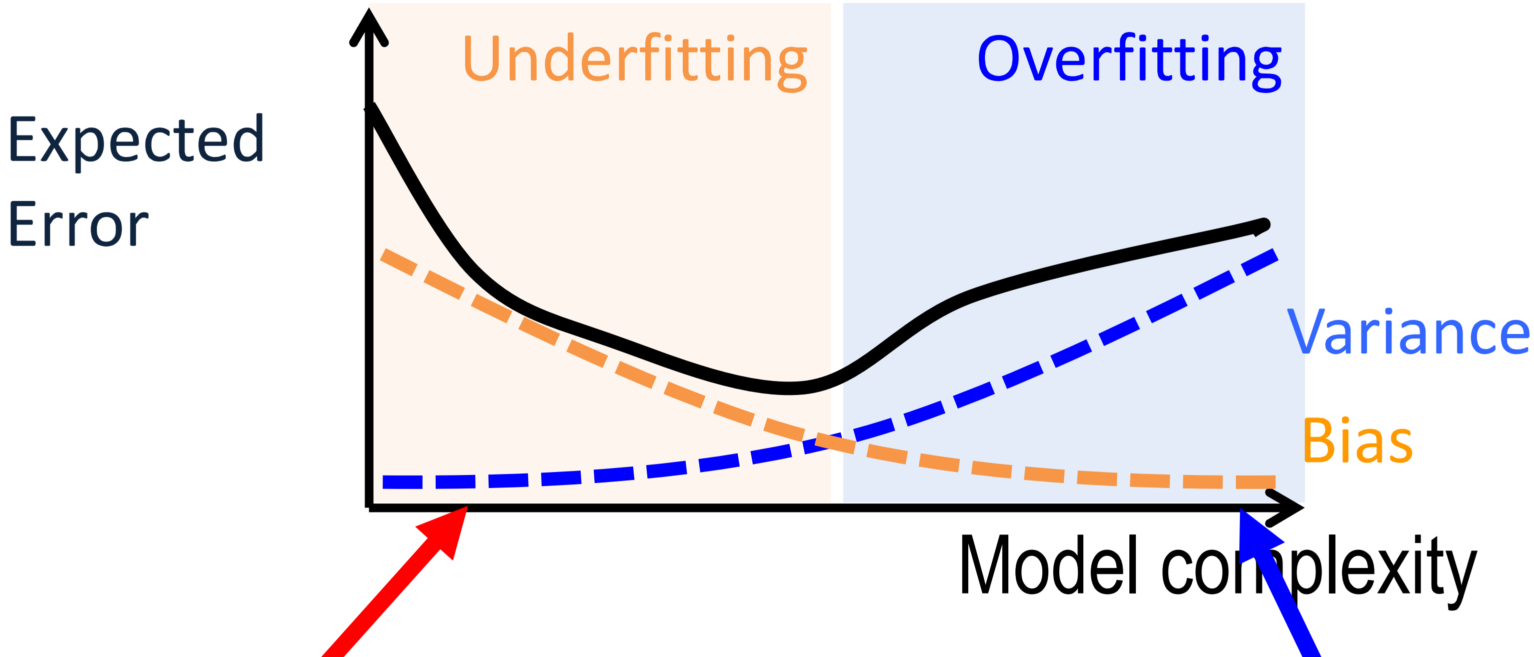
Model complexity



- **Simple models:**
  High bias and low variance

Complex models:
High variance and low bias

Underfitting and Overfitting



- **Simple models:**
  High bias and low variance
- **Complex models:**
  High variance and low bias
- This can be made more accurate for some loss functions.
- We will develop a more precise and general theory that trades expressivity of models with empirical error

Avoiding Overfitting

How can this be avoided with linear classifiers?

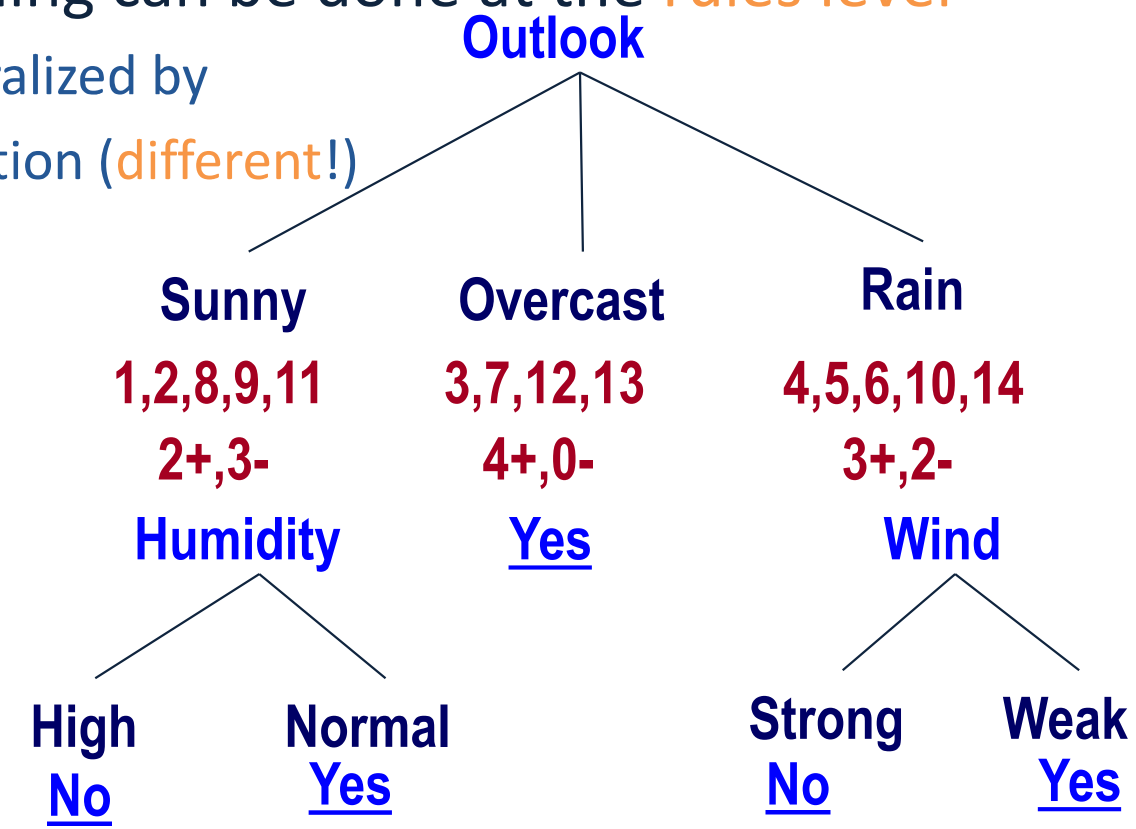
- Two basic approaches
  - Pre-pruning: Stop growing the tree at some point during construction when it is determined that there is not enough data to make reliable choices.
  - Post-pruning: Grow the full tree and then remove nodes that seem not to have sufficient evidence.
- Methods for evaluating subtrees to prune
  - Cross-validation: Reserve hold-out set to evaluate utility
  - Statistical testing: Test if the observed regularity can be dismissed as likely to occur by chance
  - Minimum Description Length: Is the additional complexity of the hypothesis smaller than remembering the exceptions?
- This is related to the notion of regularization that we will see in other contexts – keep the hypothesis simple.

Hand waving, for now.

Next: a brief detour into explaining generalization and overfitting

Trees and Rules

- Decision Trees can be represented as Rules
  - (outlook = sunny) and (humidity = normal) then YES
  - (outlook = rain) and (wind = strong) then NO
- Sometimes Pruning can be done at the rules level
  - Rules are generalized by erasing a condition (different!)



Continuous Attributes

Additional Issues I

- Real-valued attributes can, in advance, be discretized into ranges, such as *big, medium, small*
- Alternatively, one can develop splitting nodes based on thresholds of the form A<c that partition the data into examples that satisfy A<c and A>=c. The information gain for these splits is calculated in the same way and compared to the information gain of discrete splits.
- How to find the split with the highest gain?
  - For each continuous feature A:
    - Sort examples according to the value of A
    - For each ordered pair (x,y) with different labels
      - Check the mid-point as a possible threshold, i.e.

$$S_{a \leq x}, S_{a \geq y}$$

Continuous Attributes

Additional Issues I

- Example:
  - Length (L): 10 15 21 28 32 40 50
  - Class: - + + - + + -
  - Check thresholds: L < 12.5; L < 24.5; L < 45
  - Subset of Examples= {…}, Split= k+,j-

- How to find the split with the highest gain ?
  - For each continuous feature A:
    - Sort examples according to the value of A
    - For each ordered pair (x,y) with different labels
      - Check the mid-point as a possible threshold. I.e,

        $S_{a \leq x}$, $S_{a \geq y}$

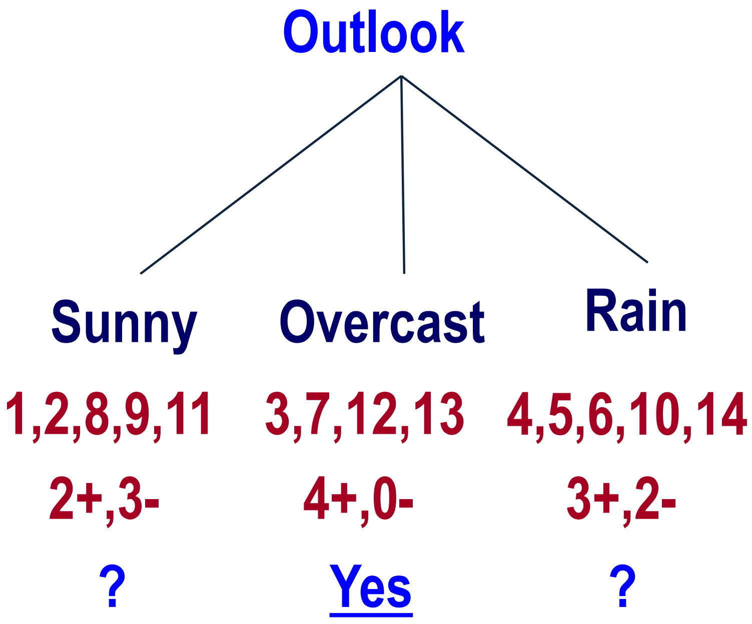
Missing Values

Additional Issues II

- Diagnosis = < fever, blood_pressure,…, blood_test=?,…>
- Many times values are not available for all attributes during training or testing (e.g., medical diagnosis)
- Training: evaluate Gain(S,a) where in some of the examples a value for a is not given

Missing Values

$$Gain(S,a) = Ent(S) - \sum \frac{|S_v|}{|S|} Ent(S_v)$$



$Gain(S_{sunny}, Temp) =$ .97- 0-(2/5) 1 = .57

$Gain(S_{sunny}, Humidity) =$

- Fill in: assign the most likely value of X_i to **s**: argmax $_k$ $P(X_i = k)$: Normal
  - 97-(3/5) Ent[+0,-3] -(2/5) Ent[+2,-0] = .97
- Assign fractional counts $P(X_i = k)$ for each value of X_i to **s**
  - .97-(2.5/5) Ent[+0,-2.5] - (2.5/5) Ent[+2,-.5] < .97

Other suggestions?

Day	Outlook	Temperature	Humidity	Wind	PlayTennis
1	Sunny	Hot	High	Weak	No
2	Sunny	Hot	High	Strong	No
8	Sunny	Mild	???	Weak	No
9	Sunny	Cool	Normal	Weak	Yes
11	Sunny	Mild	Normal	Strong	Yes

Missing Values

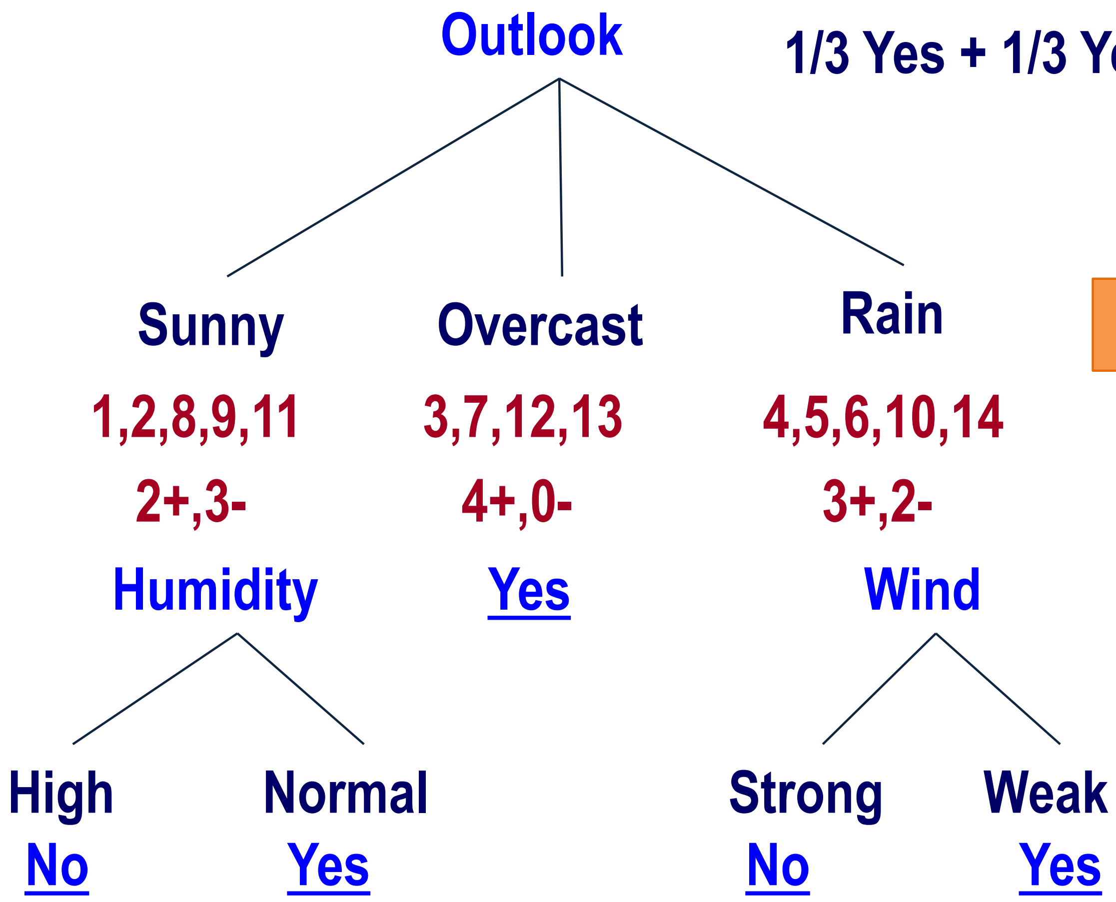
Additional Issues II

- Diagnosis = < fever, blood_pressure,…, blood_test=?,…>
- Many times values are not available for all attributes during training or testing (e.g., medical diagnosis)
- Training: evaluate Gain(S,a) where in some of the examples a value for a is not given
- Testing: classify an example without knowing the value of a

Missing Values

Outlook = Sunny, Temp = Hot, Humidity = ???, Wind = Strong, label = ?? Normal/High

Outlook = ???, Temp = Hot, Humidity = Normal, Wind = Strong, label = ??



1/3 Yes + 1/3 Yes +1/3 No = Yes

Other suggestions?

Other Issues

Other Issues

- Attributes with different costs
  - Change information gain so that low cost attribute are preferred
    - Dealing with features with different # of values
- Alternative measures for selecting attributes
  - When different attributes have different number of values information gain tends to prefer those with many values
- Oblique Decision Trees
  - Decisions are not axis-parallel
- Incremental Decision Trees induction
  - Update an existing decision tree to account for new examples incrementally (Maintain consistency?)

Decision Trees as Features

Features

- Rather than using decision trees to represent the target function it is becoming common to use small decision trees as features
- When learning over a large number of features, learning decision trees is difficult and the resulting tree may be very large

  → (over fitting)
- Instead, learn small decision trees, with limited depth.
- Treat them as “experts”; they are correct, but only on a small region in the domain. (what DTs to learn? same every time?)
- Then, learn another function, typically a linear function, over these as features.
- Boosting (but also other linear learners) are used on top of the small decision trees. (Either Boolean, or real valued features)

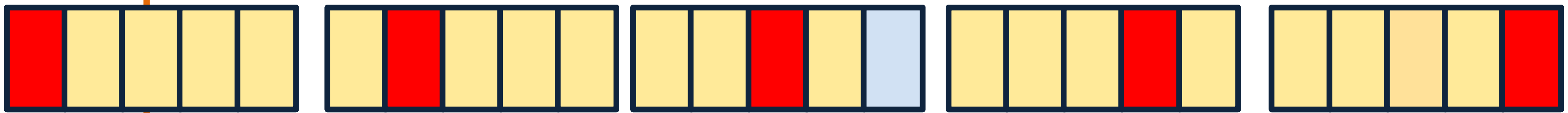
Experimental Machine Learning

- Machine Learning is an Experimental Field and we will spend some time (in Problem sets) learning how to run experiments and evaluate results
  - First hint: be organized; write scripts
- Basics:
  - Split your data into two (or three) sets:
    - Training data (often 70-90%)
    - Test data (often 10-20%)
    - Development data (10-20%)
- You need to report performance on test data, but you are not allowed to look at it.
  - You are allowed to look at the development data (and use it to tweak parameters)

N-fold cross validation

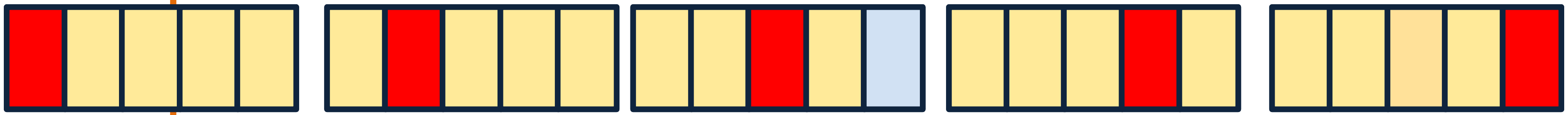
- Instead of a single test-training split:

- Split data into N equal-sized parts



- Train and test N different classifiers
- Report average accuracy and standard deviation of the accuracy

Evaluation: significance tests



- You have two different classifiers, A and B
- You train and test them on the same data set using N-fold cross-validation
- For the n-th fold:

  accuracy(A, n), accuracy(B, n)

  p_n = accuracy(A, n) - accuracy(B, n)
- Is the difference between A and B's accuracies significant?

Hypothesis testing

- You want to show that hypothesis H is true, based on your data
  - (e.g. H = "classifier A and B are different")
- Define a null hypothesis H_0
  - (H_0 is the contrary of what you want to show)
- H_0 defines a distribution $P(m \mid H_0)$ over some statistic
  - e.g. a distribution over the difference in accuracy between A and B
- Can you refute (reject) H_0?

Rejecting H_0

- H_0 defines a distribution $P(M\ /H_0)$ over some statistic M
  - (e.g. M= the difference in accuracy between A and B)
- Select a significance value S
  - (e.g. 0.05, 0.01, etc.)
  - You can only reject H0 if $P(m\ /H_0) \leq S$
- Compute the test statistic m from your data
  - e.g. the average difference in accuracy over your N folds
- Compute $P(m\ /H_0)$
- Refute H_0 with $p \leq S$ if $P(m\ /H_0) \leq S$

Paired t-test

- Null hypothesis (H_0; to be refuted):
  - There is no difference between A and B, i.e. the expected accuracies of A and B are the same
- That is, the expected difference (over all possible data sets) between their accuracies is 0:

  H_0: $E[p_D] = 0$
- We don't know the true $E[p_D]$
- N-fold cross-validation gives us N samples of p_D

Paired t-test

- Null hypothesis H_0: $E[diff_D] = \mu = 0$
- m: our estimate of μ based on N samples of $diff_D$

$$m = 1/N \sum_n \mathrm{diff}_n$$

- The estimated variance S^2:

$$S^2 = 1/(N-1) \sum_{1,N} (\mathrm{diff}_n - m)^2$$

- Accept Null hypothesis at significance level a if the following statistic lies in $(-t_{a/2,\, N-1}, +t_{a/2,\, N-1})$

$$\frac{\sqrt{N}m}{S} \sim t_{N-1}$$

Decision Trees - Summary

- Hypothesis Space:
  - Variable size (contains all functions)
  - Deterministic; Discrete and Continuous attributes
- Search Algorithm
  - ID3 - batch
  - Extensions: missing values
- Issues:
  - What is the goal?
  - When to stop? How to guarantee good generalization?
- Did not address:
  - How are we doing? (Correctness-wise, Complexity-wise)