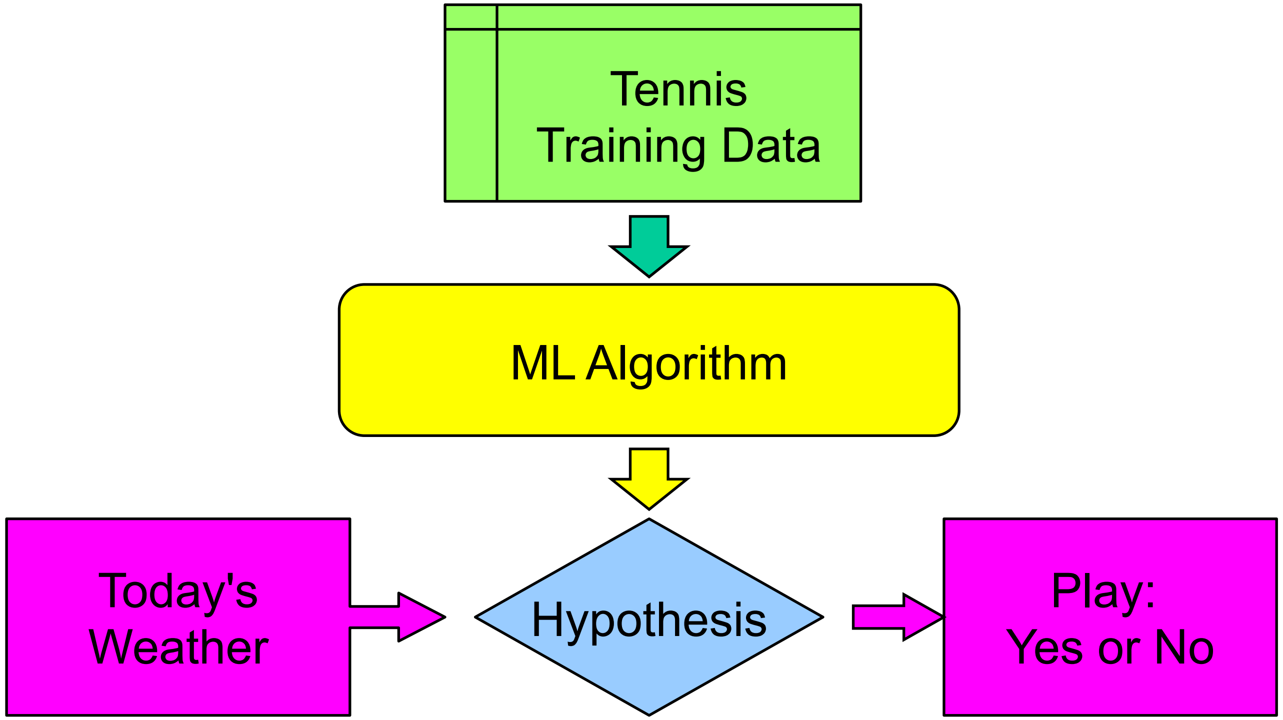
Topic 2: Information-based learning

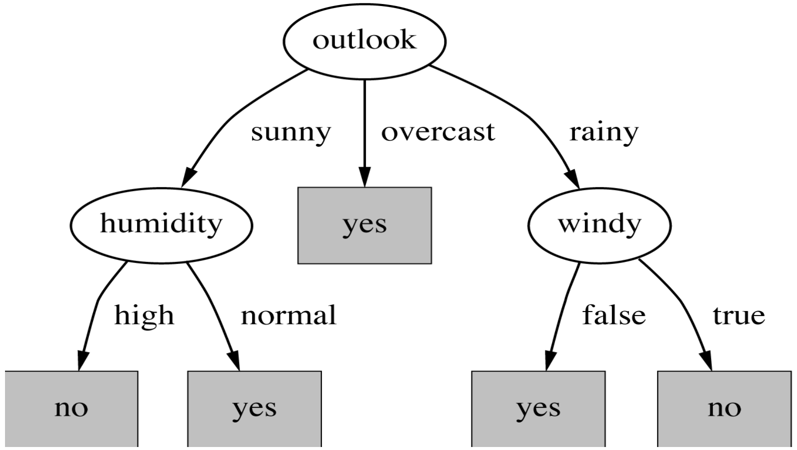
Part 6: The ID3 algorithm

Review: the supervised learning process



Anyone for Tennis?

ID	Outlook	Temp	Humidity	Windy	Play?
A	sunny	hot	high	false	no
B	sunny	hot	high	true	no
C	overcast	hot	high	false	yes
D	rainy	mild	high	false	yes
E	rainy	cool	normal	false	yes
F	rainy	cool	normal	true	no
G	overcast	cool	normal	true	yes
H	sunny	mild	high	false	no
I	sunny	cool	normal	false	yes
J	rainy	mild	normal	false	yes
K	sunny	mild	normal	true	yes
L	overcast	mild	high	true	yes
M	overcast	hot	normal	false	yes
N	rainy	mild	high	true	no



Review: inductive learning of a decision tree

Step 1
- For all attributes that have not yet been used in the tree, calculate their **entropy** and **information gain** values for the training samples

Step 2
- Select the attribute that has the highest information gain

Step 3
- Make a tree node containing that attribute

Repeat
- This node **partitions** the data:
  apply the algorithm **recursively** to each partition

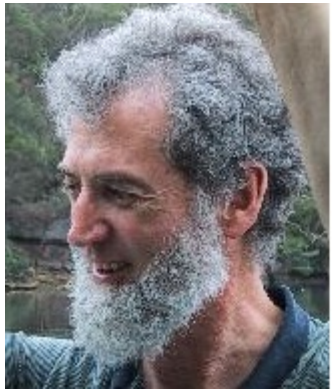
The ID3 algorithm

```
ID3(Examples, Attributes, Target):
Input: Examples:   set of classified examples
       Attributes: set of attributes in the examples
       Target:     classification to be predicted
if Examples is empty then return a Default class
else if all Examples have same class then return this class
else if all Attributes are tested then return majority class
else:
    let Best = attribute that best separates Examples relative to Target
    let Tree = new decision tree with Best as root node
    foreach value v_i of Best:
        let Examples_i = subset of Examples that have Best=v_i
        let Subtree = ID3(Examples_i, Attributes-Best, Target)
        add branch from Tree to Subtree with label v_i
    return Tree
```

BASE CASES

RECURSIVE CALL

Ross Quinlan, 1986



Based on Algorithm **Decision-Tree-Learning** in Russell & Norvig textbook

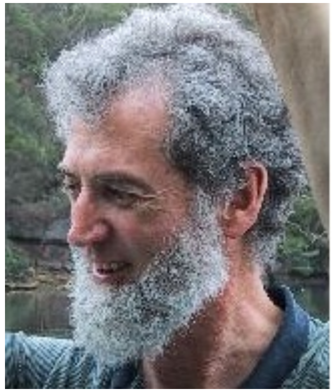
The ID3 algorithm

```
ID3(Examples, Attributes, Target):
Input: Examples:   set of classified examples
       Attributes: set of attributes in the examples
       Target:     classification to be predicted
if Examples is empty then return a Default class
else if all Examples have same class then return this class
else if all Attributes are tested then return majority class
else:
    let Best = attribute that best separates Examples relative to Target
    let Tree = new decision tree with Best as root node
    foreach value v_i of Best:
        let Examples_i = subset of Examples that have Best=v_i
        let Subtree = ID3(Examples_i, Attributes-Best, Target)
        add branch from Tree to Subtree with label v_i
    return Tree
```

BASE CASES

RECURSIVE CALL

Ross Quinlan, 1986



Based on Algorithm **Decision-Tree-Learning** in Russell & Norvig textbook

Topic 2: Information-based learning

Part 7: Issues in decision tree learning

Decision tree characteristics

- Popular because:
  - Relatively **easy** algorithm
  - **Fast**: greedy search without backtracking
  - **Comprehensible** output:
    important in decision-making (medical, financial, …)
  - **Practical**: discrete/numeric, irrelevant attributes, noisy data, …
- Expressiveness: what functions can a DT represent?
  - Technically, any Boolean function (propositional logic)
  - Some functions, however, require exponentially large tree
    (e.g. parity function)
  - Cannot consider relationships between two attributes

Dealing with noisy or missing data

- What about inconsistent ("noisy") data?
  - Use majority class (line 7 of ID3 alg.)

```
7. else if all Attributes are tested then return majority class
```

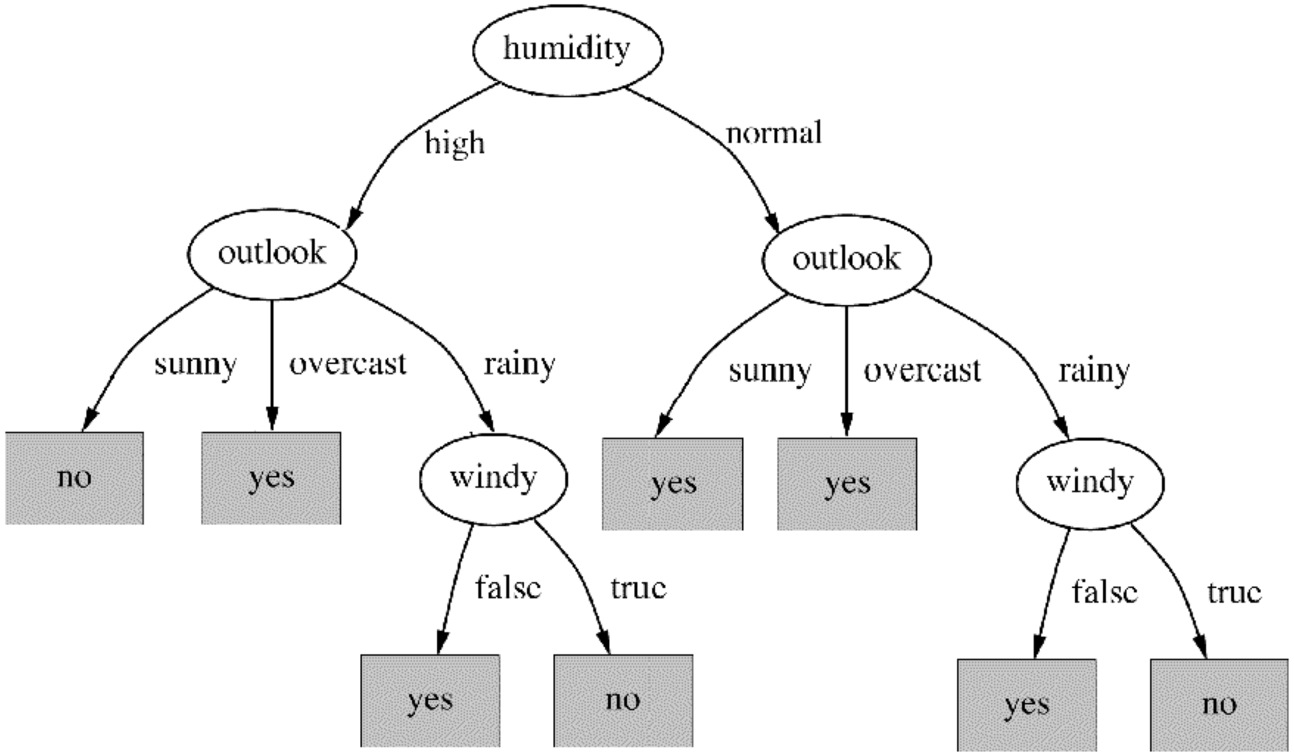
  - or interpret as probabilities
  - or return “average” target feature value

- What about missing data?
  - Given a complete decision tree, how should one classify an example that is missing one of the test attributes?
  - How should one modify the information gain formula when some training examples have unknown values for an attribute?
  - Could assign the most common value among the training examples that reach that node
  - Or could assume the attribute has all possible values, weighting each value according to its frequency among the training examples that reach that node (considered/used in C4.5 algorithm)

Instability of decision trees

- Hypothesis found is sensitive to training set used
  - consequence of greedy search
- Replace one example:
  - new one consistent with original tree
- Some algorithmic modifications to reduce the instability of decision tree learning were proposed by Li and Belford in their 2002 paper “Instability of decision tree classification algorithms”.
- Li and Belford’s main idea is to alter the attribute selection procedure, so that the tree learning algorithm is less sensitive to some % of the training dataset being replaced.

ID	Outlook	Temp	Humidity	Windy	Play?
~~C~~	~~overcast~~	~~hot~~	~~high~~	~~false~~	~~yes~~
O	sunny	hot	normal	true	yes



Pruning

- Overfitting occurs in a predictive model when the hypothesis learned makes predictions which are based on spurious patterns in the training data set. Consequence: poor generalisation to new examples.
- Overfitting may happen for a number of reasons, including sampling variance or noise present in the dataset
- **Tree pruning** may be used to combat overfitting in decision trees
- Tree pruning can lead to induced trees which are inconsistent with the training set
- Generally, there are two different approaches to pruning:
  - Pre-pruning (e.g. <= target # of examples in a partition, limiting tree depth, creating a new node only when information gain is above a threshold, statistical tests such as Chisquare(χ^2)
  - Post-pruning (e.g. target # of examples, compare error rate for model on a validation dataset with and without a given subtree; only keep a subtree if it improves the error rate, statistical tests such as χ^2, reduced error pruning (Quinlan, 1987))

Topic 2: Information-based learning

Part 8: ID3 extensions and related algorithms

Continuous-valued attributes

- What about continuous-valued attributes?
  - Pick threshold value *T* for attribute A, and test whether *A >T*
  - Information Gain can be used to decide which *T* is best
  - Could select *T* at a midpoint where classification changes

54 **85**

Temp	40	48	60	72	80	90
Play?	No	No	Yes	Yes	Yes	No

Selecting the best attribute: alternative metrics (1)

- Earlier, we introduced the concept of information gain, which we can use as a metric for the discriminatory power of an attribute
- Information gain does have some drawbacks; it tends to favour attributes that can take on a large number of different values
- One alterative is to use the **information gain ratio**
  - dividing the information gain for an attribute by the amount of information used to determine the value of the attribute.

$$\text{GainRatio}(S, A) = \frac{\text{Gain}(S, A)}{\sum_{v \in \text{Values}(A)} -p_v \log_2 p_v}$$

- The divisor of this fraction measures the amount of information used to compute the gain value, and is the entropy of S with respect to A

Selecting the best attribute: alternative metrics (2)

- Another alternative is to use the **Gini index** instead of entropy as a measure of the impurity of a set
  - Italian statistician and sociologist Corrado Gini and published in 1912 (social inequality, e.g. income or wealth inequality
  - 0 is the least inequality, and 1 is the highest inequality.

$$\mathrm{Gini}(S) = 1 - \sum_{i=1}^{n} {p_i}^2$$

- Then the gain for a feature may be calculated based on the reduction in the Gini index (rather than a reduction in entropy):

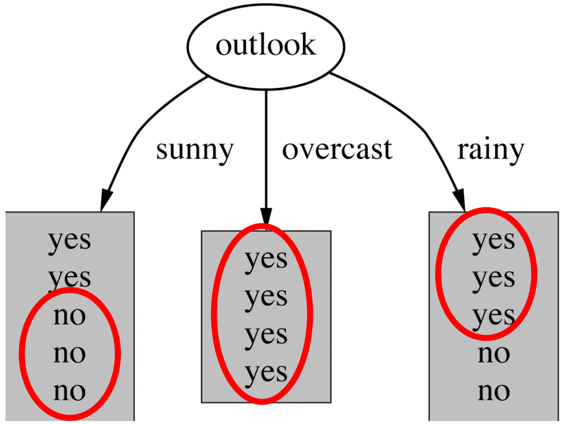
$$\mathrm{GiniGain}(S, A) = \mathrm{Gini}(S) - \sum_{v \in \mathrm{Values}(A)} \frac{|S_v|}{|S|} \mathrm{Gini}(S_v)$$

- Information gain, information gain ratio, or GiniGain?

Related algorithms [1]

- 1R
  - Decision tree with just one rule
  - Introduced in a paper by Robert C. Holte (1993).

    "Very Simple Classification Rules Perform Well on Most Commonly Used Datasets", Computer Science Department, University of Ottawa



- Decision Stump
  - 1 rule with 1 test

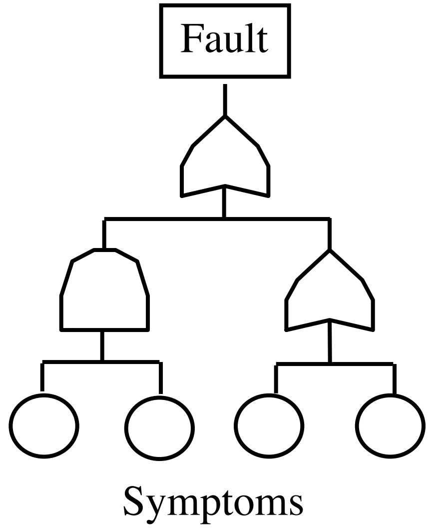
    Outlook = Overcast => YES

    Outlook != Overcast => YES

These are deliberately simple variants that are used within other algorithms (meta-learning; ensembles). Often referred to as "weak learners".

Related algorithms [2]

- Decision Lists
  - A set of rules (predicate logic), describing the hypothesis, that are followed in the given order
- C4.5 Rules
  - Alternative representation of C4.5 decision trees
- PART: Rules constructed with *partial* DTs
  - Can be more readable than standard DTs
- IFT: Induction of Fault Trees



Decision tree software

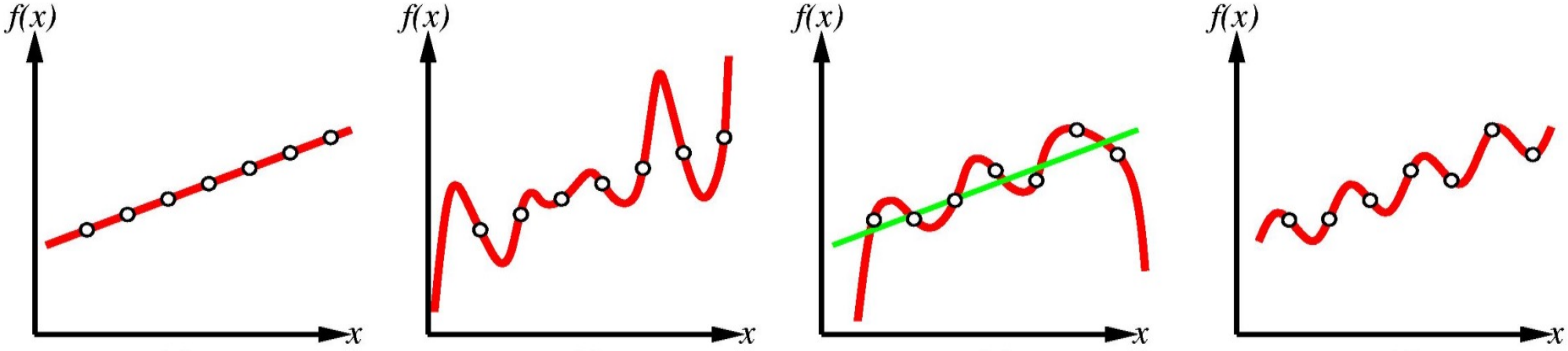
- C4.5
  - Original implementation (C language, command line), available on Ross Quinlan's website (https://www.rulequest.com/Personal/)
  - Deals with missing values in the data, high-branching attributes (e.g. ID in the weather data), pruning to avoid overfitting, converting a decision tree to a list of rules
- C5.0
  - Commercial version from RuleQuest Research, with improvements over C4.5
- WEKA software
  - Accompanies book by Witten & Frank, Data Mining: Practical Machine Learning Tools and Techniques
  - Java implementations of many ML algorithms, including C4.5 (mysteriously called J48)
  - Easy-to-use front end and utilities
- Many other implementations in Python and R…

Topic 2: Information-based learning

Part 9: Supervised learning considerations

Supervised Learning Considerations [1]

- Various hypotheses can be consistent with observations but inconsistent with each other:
  Which one should we choose?



(1) Data with exact linear hypothesis

(2) Same data: Exact 7th-Order polynomial hyp.

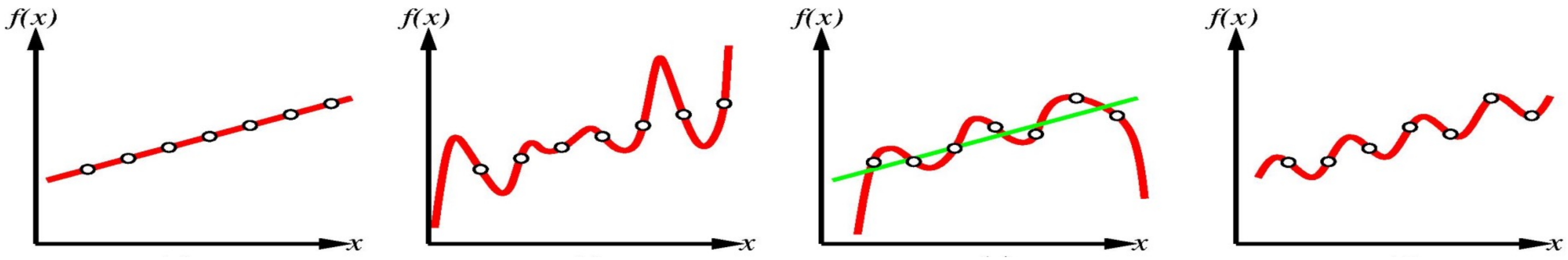
(3) Different data: Exact 5th-order poly and approx. linear hyp.

(4) Same data: Exact sinusoidal hypothesis

- General form of a polynomial: $f(x) = a + bx + cx^2 + dx^3 + \cdots$

Supervised Learning Considerations [2]

- Various hypotheses can be consistent with observations but inconsistent with each other: Which one should we choose?



(1) Data with exact linear hypothesis

(2) Same data: Exact 7th-Order polynomial hyp.

(3) Different data: Exact 5th-order poly and approx. linear hyp.

(4) Same data: Exact sinusoidal hypothesis

- One solution: Ockham's Razor principle (William Ockham – 14th Century):
  - Prefer *simplest* hypothesis consistent with data
    - Example: Elvis death hypothesis.
  - Definitions of simplicity (& consistency) may subject to debate
  - Depends strongly on how hypotheses are expressed

Supervised Learning Considerations [3]

- Hypothesis language is too limited?
  - Might be **unable** to find hypothesis that exactly matches 'true' function
  - If true function is more complex than what hypothesis can express, it will **underfit** the data
  - Saw this in previous slide, 3rd and 4th figures

- Hypothesis language cannot exactly match true function?
  - there will be a trade-off between **complexity** of hypothesis and how well it **fits the data**

Supervised Learning Considerations [4]

- Hypothesis language is very **expressive**?
  - Its search space is very large and the **computational complexity** of finding a good hypothesis will be high
  - Also need a large amount of data to avoid **overfitting**

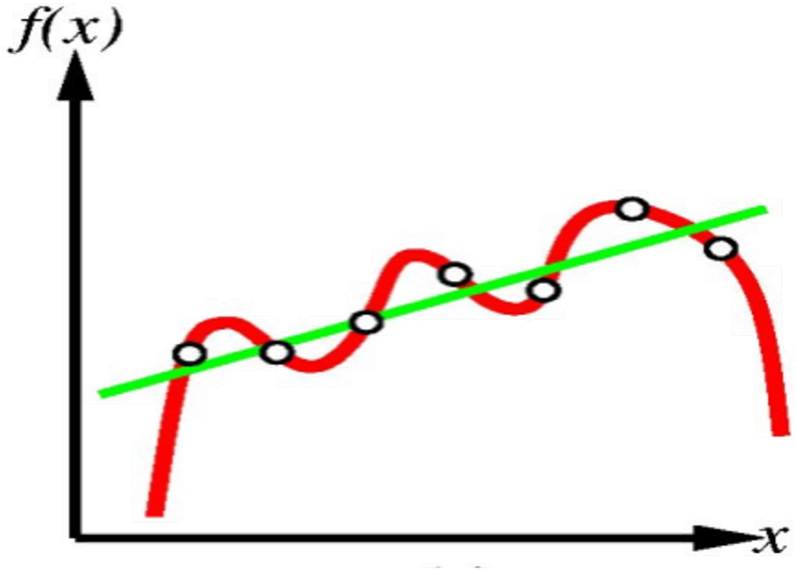
- What can decision trees express?
  - Will learn about other algorithms that express hypotheses differently
  - In general, would like to use an algorithm for a problem that can express the true underlying function

Supervised Learning Considerations [5]

- But don't forget:
  we never know the true underlying function
- E.g. To avoid problem with poorly fitting data from a previous slide
  - Could change algorithm so that, as well as searching for coefficients of polynomials, it tries combinations of trig. functions (sin, cos, tan)
  - Learning problem will become enormously more complex, but will it solve our problems?
  - Probably not: you could easily think up some different kind of mathematical function, to generate a new dataset that the algorithm still cannot represent perfectly.
- For this reason, often use relatively simple hypothesis languages, in the absence of special knowledge about domain
  - more complex languages don't come with any real guarantees
  - more simple languages correspond to easier searching.

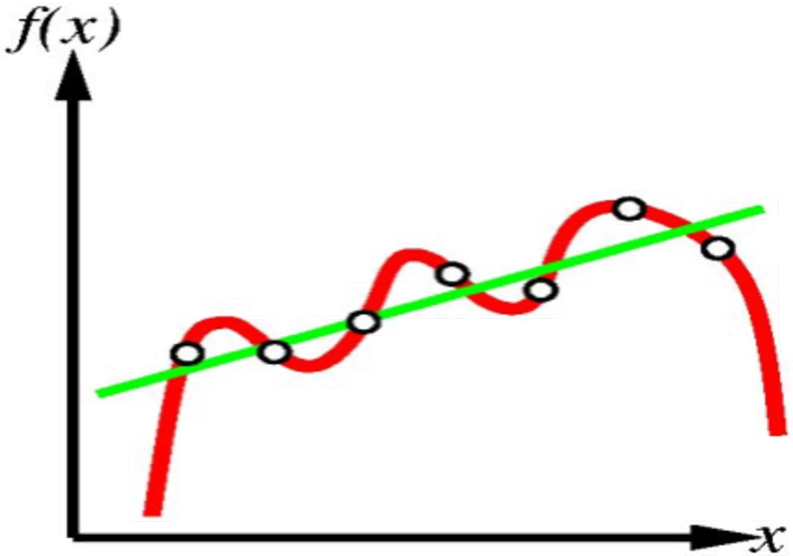
Noise, Overfitting and Underfitting [1]

- NOISE:
  imprecise or incorrect attribute values or labels
  - Can't always quantify it, but should know from situation if it is present
  - E.g. labels may require subjective judgement or values may come from imprecise measurements



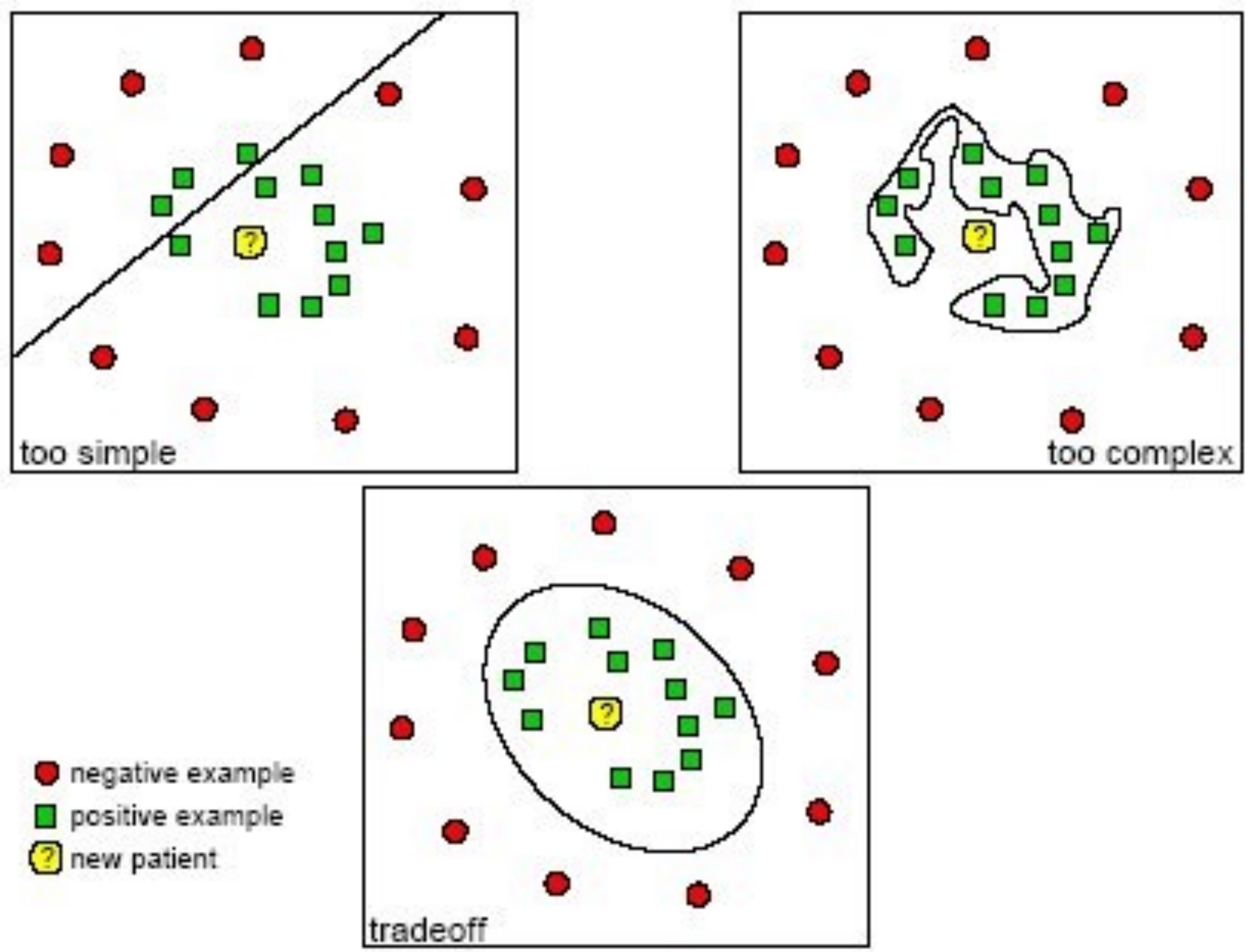
Noise, Overfitting and Underfitting [2]

- If the data might have noise, harder to decide which hypothesis is best:
  - Linear hypothesis could not fit to it, but polynomial could
  - But which would really be the better choice?



- Complex classification methods prone to **overfitting**; simple ones prone to **underfitting**
  - If you increase complexity of hypothesis, you increase ability to fit to the data, but might also increase risk of overfitting

Illustration of Underfitting & Overfitting



Detecting Underfitting & Overfitting

- Previous slides have illustrated concepts only
  - In general, cannot visualise very high dimensional data: -=> can’t directly observe overfitting/underfitting

- Main symptom of underfitting:
  - Poor performance even on the training data
- Main symptom of overfitting:
  - *Much* better performance on the training data than on independent test data
  - (Slightly better performance is to be expected)

Topic 2: Information-based learning

Part 10:
Review of topic

Learning Objectives Review

After completing this topic successfully, you will be able to …

1. Explain what supervised learning is
2. Distinguish it from unsupervised learning and reinforcement learning
3. Describe in detail an algorithm for decision tree induction
4. Demonstrate the application of decision tree induction to a data set
5. List related algorithms
6. Discuss high-level concepts such as choice of hypothesis language, overfitting, underfitting and noise