Artificial Intelligence Foundation – JC3001

Lecture 37: Introduction to Machine Learning

Prof. Aladdin Ayesh (aladdin.ayesh@abdn.ac.uk)

Dr. Binod Bhattarai (binod.bhattarai@abdn.ac.uk)

Dr. Gideon Ogunniye, (g.ogunniye@abdn.ac.uk)

October 2025

Material adapted from:
Russell and Norvig (AIMA Book): Chapter 19 (19.1–19.3)
Sebastian Thrun (Stanford University / Udacity)
Andrew Ng (Stanford University / Coursera)

UNIVERSITY OF ABERDEEN

Course Progression

- Part 1: Introduction
  1. Introduction to AI ✓
  2. Agents ✓
- Part 2: Problem-solving
  1. Search 1: Uninformed Search ✓
  2. Search 2: Heuristic Search ✓
  3. Search 3: Local Search ✓
  4. Search 4: Adversarial Search ✓
- Part 3: Reasoning and Uncertainty
  1. Reasoning 1: Constraint Satisfaction ✓
  2. Reasoning 2: Logic and Inference ✓
  3. Probabilistic Reasoning 1: BNs ✓
  4. Probabilistic Reasoning 2: HMMs ✓
- Part 4: Planning
  1. Planning 1: Intro and Formalism ✓
  2. Planning 2: Algorithms & Heuristics ✓
  3. Planning 3: Hierarchical Planning ✓
  4. Planning 4: Stochastic Planning ✓
- Part 5: Learning
  1. **Learning 1: Intro to ML**
  2. Learning 2: Regression
  3. Learning 3: Neural Networks
  4. Learning 4: Reinforcement Learning
- Part 6: Conclusion
  1. Ethical Issues in AI
  2. Conclusions and Discussion

Objectives

- Overview of Machine Learning
- Types of Learning Algorithms
- Decision Trees

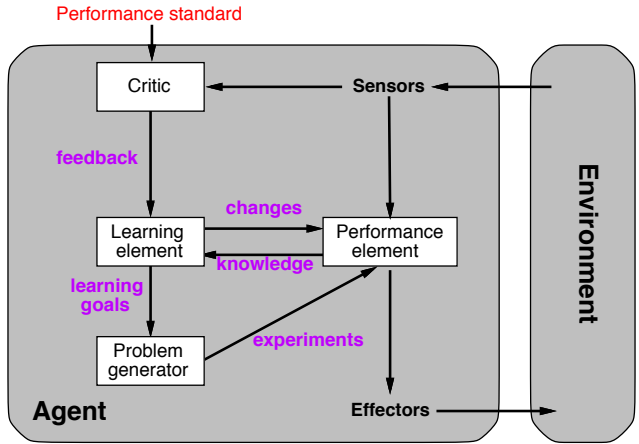
Outline

- We have explored how to make decisions over time in a number of settings
  - Via domain specific search
  - Via domain independent planning
  - Via stochastic planning
- However, all of these approaches assume a known static model of the environment.
- How can a machine use data to learn new models of the world?

Outline

- Learning is essential for unknown environments,
  i.e., when designer lacks omniscience
- Learning is useful as a system construction method,
  i.e., expose the agent to reality rather than trying to write it down
- Learning modifies the agent's decision mechanisms to improve performance

Learning Agents



- Arthur Samuel(1959). Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed.

- Arthur Samuel(1959). Machine Learning: Field of study that gives computers the ability to learn without being explicitly programmed.
- Tom Mitchell(1998). Well-posed Learning Problem: A computer program is said to learn from experience **E** with respect to some task **T** and some performance measure **P**, if its performance on **T**, as measured by **P**, improves with experience **E**.

Machine Learning Definition

> *A computer program is said to learn from experience **E** with respect to some task **T** and some performance measure **P**, if its performance on **T**, as measured by **P**, improves with experience **E**.*

Suppose your email program watches which emails you do or do not mark as spam, and based on that learns how to better filter spam.
What are each of the components of the learning problem?

- Classifying emails as spam or not spam
- Watching you label emails as spam or not spam
- The number (or fraction) of emails correctly classified as spam/not spam

Machine Learning Definition

> *A computer program is said to learn from experience **E** with respect to some task **T** and some performance measure **P**, if its performance on **T**, as measured by **P**, improves with experience **E**.*

Suppose your email program watches which emails you do or do not mark as spam, and based on that learns how to better filter spam.
What are each of the components of the learning problem?

- Classifying emails as spam or not spam $\Rightarrow$**T**
- Watching you label emails as spam or not spam
- The number (or fraction) of emails correctly classified as spam/not spam

> *A computer program is said to learn from experience **E** with respect to some task **T** and some performance measure **P**, if its performance on **T**, as measured by **P**, improves with experience **E**.*

Suppose your email program watches which emails you do or do not mark as spam, and based on that learns how to better filter spam.
What are each of the components of the learning problem?

- Classifying emails as spam or not spam $\Rightarrow$**T**
- Watching you label emails as spam or not spam $\Rightarrow$**E**
- The number (or fraction) of emails correctly classified as spam/not spam

> *A computer program is said to learn from experience **E** with respect to some task **T** and some performance measure **P**, if its performance on **T**, as measured by **P**, improves with experience **E**.*

Suppose your email program watches which emails you do or do not mark as spam, and based on that learns how to better filter spam.
What are each of the components of the learning problem?

- Classifying emails as spam or not spam $\Rightarrow$**T**
- Watching you label emails as spam or not spam $\Rightarrow$**E**
- The number (or fraction) of emails correctly classified as spam/not spam $\Rightarrow$**P**

- What?

- What?
  parameters,

- What?
  parameters, structure,

- What?
  parameters, structure, hidden concepts
- What from?

- What?
  parameters, structure, hidden concepts
- What from?

- What?
  parameters, structure, hidden concepts
- What from?
  supervised,

- What?
  parameters, structure, hidden concepts
- What from?
  supervised, unsupervised,

- What?
  parameters, structure, hidden concepts
- What from?
  supervised, unsupervised, reinforcement
- What for?

- What?
  parameters, structure, hidden concepts
- What from?
  supervised, unsupervised, reinforcement
- What for?

- What?
  parameters, structure, hidden concepts
- What from?
  supervised, unsupervised, reinforcement
- What for?
  prediction,

- What?
  parameters, structure, hidden concepts
- What from?
  supervised, unsupervised, reinforcement
- What for?
  prediction, diagnostics,

- What?
  parameters, structure, hidden concepts
- What from?
  supervised, unsupervised, reinforcement
- What for?
  prediction, diagnostics, summarisation
- How?

- What?
  parameters, structure, hidden concepts
- What from?
  supervised, unsupervised, reinforcement
- What for?
  prediction, diagnostics, summarisation
- How?

- What?
  parameters, structure, hidden concepts
- What from?
  supervised, unsupervised, reinforcement
- What for?
  prediction, diagnostics, summarisation
- How?
  passive,

- What?
  parameters, structure, hidden concepts
- What from?
  supervised, unsupervised, reinforcement
- What for?
  prediction, diagnostics, summarisation
- How?
  passive, active,

- What?
  parameters, structure, hidden concepts
- What from?
  supervised, unsupervised, reinforcement
- What for?
  prediction, diagnostics, summarisation
- How?
  passive, active, online,

- What?
  parameters, structure, hidden concepts
- What from?
  supervised, unsupervised, reinforcement
- What for?
  prediction, diagnostics, summarisation
- How?
  passive, active, online, offline,
- Outputs?

- What?
  parameters, structure, hidden concepts
- What from?
  supervised, unsupervised, reinforcement
- What for?
  prediction, diagnostics, summarisation
- How?
  passive, active, online, offline,
- Outputs?

- What?
  parameters, structure, hidden concepts
- What from?
  supervised, unsupervised, reinforcement
- What for?
  prediction, diagnostics, summarisation
- How?
  passive, active, online, offline,
- Outputs?
  classification,

- What?
  parameters, structure, hidden concepts
- What from?
  supervised, unsupervised, reinforcement
- What for?
  prediction, diagnostics, summarisation
- How?
  passive, active, online, offline,
- Outputs?
  classification, regression

Design of learning element is dictated by

- what type of performance element is used
- which functional component is to be learned

Example scenarios:
Supervised learning: correct answers for each instance
Unsupervised learning: instances we aim to group by similarity
Reinforcement learning: occasional rewards

Design of learning element is dictated by

- what type of performance element is used
- which functional component is to be learned
- how that functional component is represented

Example scenarios:
Supervised learning: correct answers for each instance
Unsupervised learning: instances we aim to group by similarity
Reinforcement learning: occasional rewards

Design of learning element is dictated by

- what type of performance element is used
- which functional component is to be learned
- how that functional component is represented
- what kind of feedback is available

Example scenarios:
Supervised learning: correct answers for each instance
Unsupervised learning: instances we aim to group by similarity
Reinforcement learning: occasional rewards

- Learning over a number of training examples of the form:

$$x_1\ x_2\ x_3\ \ldots\ x_n\ \rightarrow\ y$$

  - a feature vector where each x_i is the value of a specific attribute
  - x_i then refers to the same attribute in all training examples
  - y is a **target label**

- Learning over a number of training examples of the form:

$$x_1\ x_2\ x_3\ \ldots\ x_n\ \rightarrow\ y$$

  - a feature vector where each x_i is the value of a specific attribute
  - x_i then refers to the same attribute in all training examples
  - y is a **target label**
- Example, factors that affect how much a driver spends in car repairs:

$$Driver\ Age, Gender, Time\ licensed, \#\ of\ violations \rightarrow Cost$$

- Learning over a number of training examples of the form:

$$x_1\ x_2\ x_3\ \ldots\ x_n\ \rightarrow\ y$$

  - a feature vector where each x_i is the value of a specific attribute
  - x_i then refers to the same attribute in all training examples
  - y is a **target label**

- Example, factors that affect how much a driver spends in car repairs:

$$Driver\ Age, Gender, Time\ licensed, \#\ of\ violations \rightarrow Cost$$

- Learning occurs over a number of examples

$$\left|\begin{array}{ccccccc} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & \ldots & x_n^{(1)} & \rightarrow & y^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & \ldots & x_n^{(2)} & \rightarrow & y^{(2)} \\ & & & \vdots & & & \\ x_1^{(m)} & x_2^{(m)} & x_3^{(m)} & \ldots & x_n^{(m)} & \rightarrow & y^{(m)} \end{array}\right| data$$

- Learning over a number of training examples of the form:

$$x_1\ x_2\ x_3\ \ldots\ x_n\ \rightarrow\ y$$

  - a feature vector where each x_i is the value of a specific attribute
  - x_i then refers to the same attribute in all training examples
  - y is a **target label**
- Example, factors that affect how much a driver spends in car repairs:

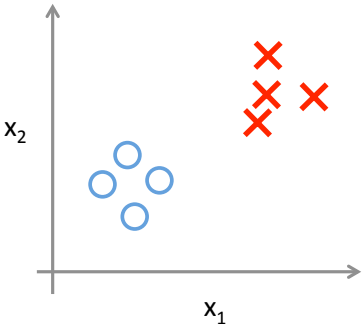
$$Driver\ Age, Gender, Time\ licensed, \#\ of\ violations \rightarrow Cost$$

- Learning occurs over a number of examples

$$\left|\begin{array}{ccccccc} x_1^{(1)} & x_2^{(1)} & x_3^{(1)} & \ldots & x_n^{(1)} & \rightarrow & y^{(1)} \\ x_1^{(2)} & x_2^{(2)} & x_3^{(2)} & \ldots & x_n^{(2)} & \rightarrow & y^{(2)} \\ & & & \vdots & & & \\ x_1^{(m)} & x_2^{(m)} & x_3^{(m)} & \ldots & x_n^{(m)} & \rightarrow & y^{(m)} \end{array}\right| data$$

Derive a function $f(X^{(m)}) = Y^{(m)}$ to predict, for any x, $f(x) = y$

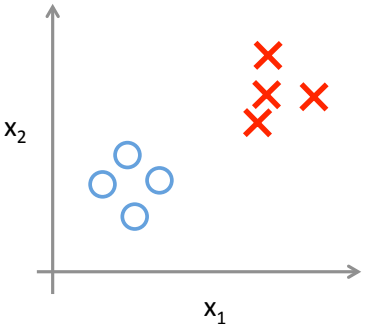
Supervised Learning



- Labelled examples
- From $(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \ldots (\vec{x}^{(m)}, y^{(m)})$, derive a function $f(X^{(m)}) = Y^{(m)}$

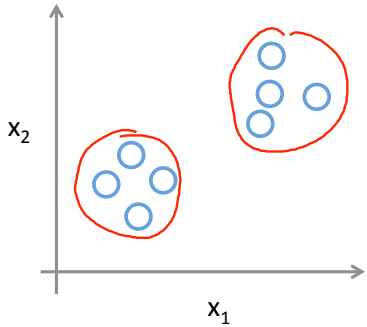
Supervised vs Unsupervised Learning

Supervised Learning



- Labelled examples
- From $(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \ldots (\vec{x}^{(m)}, y^{(m)})$, derive a function $f(X^{(m)}) = Y^{(m)}$

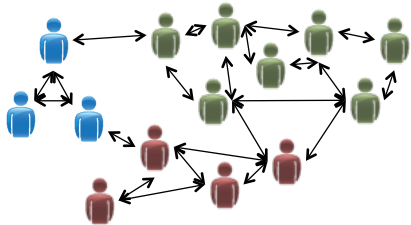
Unsupervised Learning



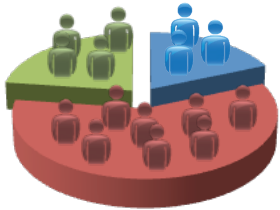
- Unlabeled examples
- From $\vec{x}^{(1)}, \vec{x}^{(2)}, \vec{x}^{(m)}, \ldots, \vec{x}^{(m)}$, identify classes x_c and derive a function to calculate $P(X = x_c)$

Applications of Unsupervised Learning

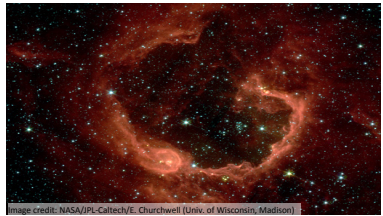
Organize computing clusters



Social network analysis



Market segmentation



Astronomical data analysis

Credit to Andrew Ng (Coursera) and NASA

Of the following examples, which would you address using a supervised (S) learning algorithm and which would you address using an unsupervised (U) learning algorithm?

( ) Given email labeled as spam/not spam, learn a spam filter.

( ) Given a set of news articles found on the web, group them into set of articles about the same story.

( ) Given a database of customer data, automatically discover market segments and group customers into different market segments.

( ) Given a dataset of patients diagnosed as either having diabetes or not, learn to classify new patients as having diabetes or not.

Of the following examples, which would you address using a supervised (S) learning algorithm and which would you address using an unsupervised (U) learning algorithm?

(S) Given email labeled as spam/not spam, learn a spam filter.

( ) Given a set of news articles found on the web, group them into set of articles about the same story.

( ) Given a database of customer data, automatically discover market segments and group customers into different market segments.

( ) Given a dataset of patients diagnosed as either having diabetes or not, learn to classify new patients as having diabetes or not.

Of the following examples, which would you address using a supervised (S) learning algorithm and which would you address using an unsupervised (U) learning algorithm?

(S) Given email labeled as spam/not spam, learn a spam filter.

(U) Given a set of news articles found on the web, group them into set of articles about the same story.

( ) Given a database of customer data, automatically discover market segments and group customers into different market segments.

( ) Given a dataset of patients diagnosed as either having diabetes or not, learn to classify new patients as having diabetes or not.

Of the following examples, which would you address using a supervised (S) learning algorithm and which would you address using an unsupervised (U) learning algorithm?

(S) Given email labeled as spam/not spam, learn a spam filter.

(U) Given a set of news articles found on the web, group them into set of articles about the same story.

(U) Given a database of customer data, automatically discover market segments and group customers into different market segments.

( ) Given a dataset of patients diagnosed as either having diabetes or not, learn to classify new patients as having diabetes or not.

Of the following examples, which would you address using a supervised (S) learning algorithm and which would you address using an unsupervised (U) learning algorithm?

(S) Given email labeled as spam/not spam, learn a spam filter.

(U) Given a set of news articles found on the web, group them into set of articles about the same story.

(U) Given a database of customer data, automatically discover market segments and group customers into different market segments.

(S) Given a dataset of patients diagnosed as either having diabetes or not, learn to classify new patients as having diabetes or not.

Outline

4 Overfitting

- Consists of trying to find the hypothesis h that is most probable

$$h^* = \arg\max_{h \in \mathcal{H}} P(h \mid data)$$

- Consists of trying to find the hypothesis h that is most probable

$$h^* = \arg\max_{h \in \mathcal{H}} P(h \mid data)$$

- Consists of trying to find the hypothesis h that is most probable

$$h^* = \arg\max_{h \in \mathcal{H}} P(h \mid data)$$

by Bayes Rule, this is equivalent to

$$h^* = \arg\max_{h \in \mathcal{H}} P(data \mid h) P(h)$$

- Trade-off between :
  - expressiveness of a hypothesis space
  - complexity of finding a good hypothesis within that space

- Consists of trying to find the hypothesis h that is most probable

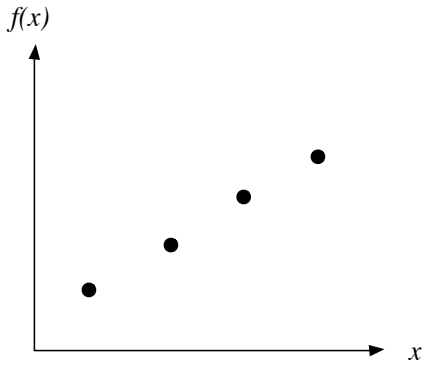
$$h^* = \arg\max_{h \in \mathcal{H}} P(h \mid data)$$

by Bayes Rule, this is equivalent to

$$h^* = \arg\max_{h \in \mathcal{H}} P(data \mid h) P(h)$$

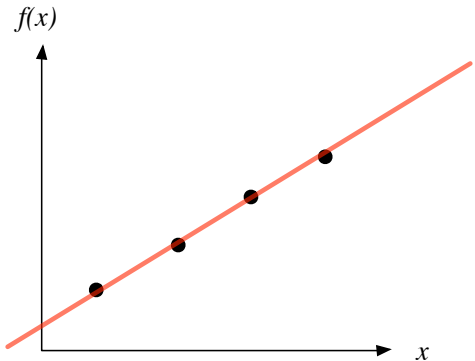
- Trade-off between :
  - expressiveness of a hypothesis space
  - complexity of finding a good hypothesis within that space
- Examples:
  - Linear function versus high-degree polynomial
  - Boolean function in propositional logic versus FOL
  - Java Programs/Turing Machines (Generally Undecidable)

Curve Fitting Example



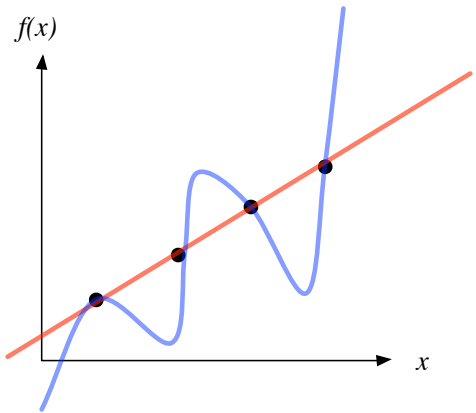
- Given the following set of points.

Curve Fitting Example



- Given the following set of points.
- And two curves learned from the example points.

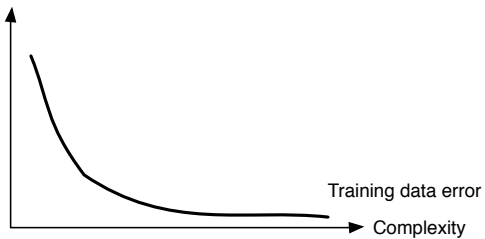
Curve Fitting Example



- Given the following set of points.
- And two curves learned from the example points.
- Which curve represents a better learned function for the example points?

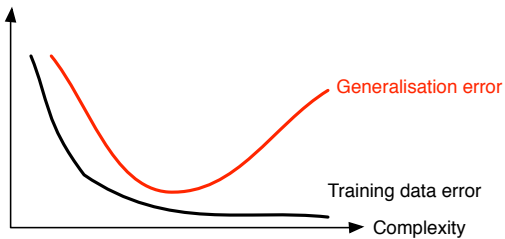
Everything else being equal, choose the less complex hypothesis.

$$Fit \longleftrightarrow Complexity$$



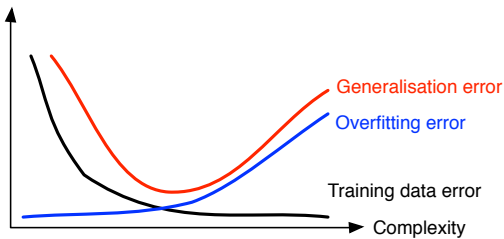
Everything else being equal, choose the less complex hypothesis.

$$Fit \longleftrightarrow Complexity$$



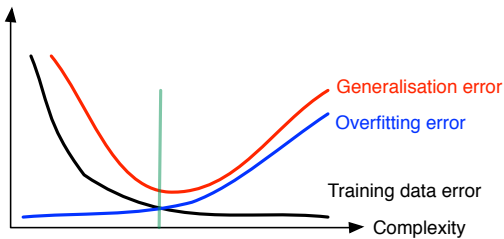
Everything else being equal, choose the less complex hypothesis.

$$Fit \longleftrightarrow Complexity$$

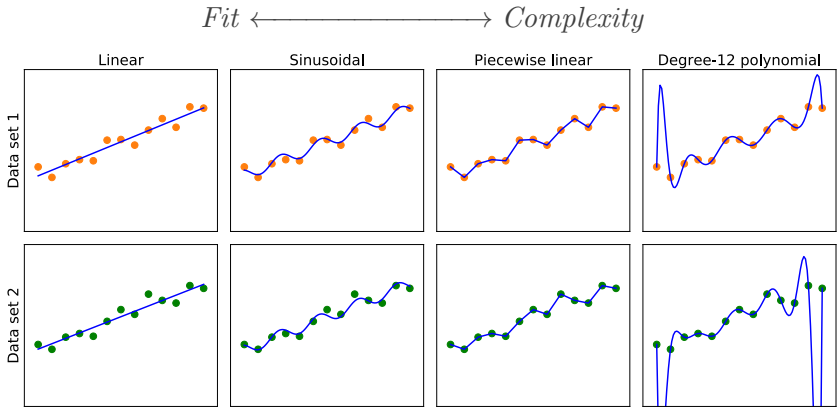


Everything else being equal, choose the less complex hypothesis.

$$Fit \longleftrightarrow Complexity$$

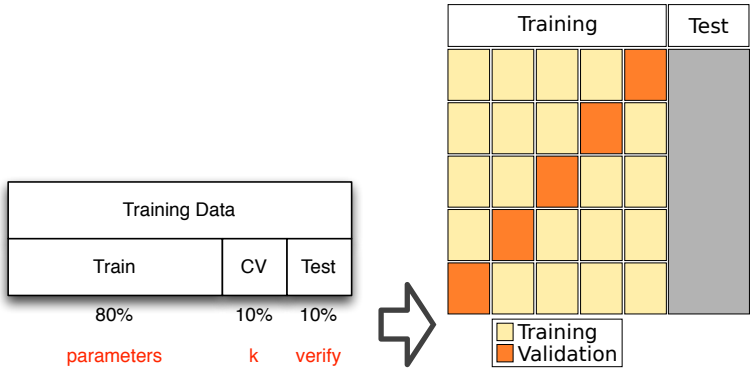


Everything else being equal, choose the less complex hypothesis.



Overfitting Prevention in General

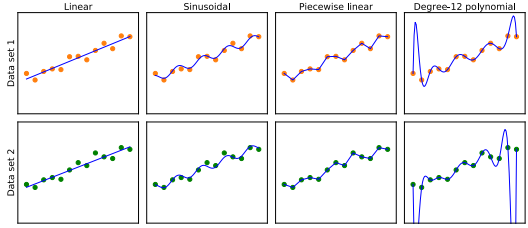
Cross validation



Assumption (of most algorithms):
Data is Independent and Identically Distributed (IID)

Issues in Supervised Learning

4 Overfitting



- Bias: tendency of a hypothesis to deviate from expectation in different training sets
- Variance: amount of change in a hypothesis due to changes in data
  - Bias-Variance tradeoff: choosing between good fit in training vs. test
- Underfitting: a hypothesis that has a high error in training data
- Overfitting: a hypothesis that has a high error in test data

Outline

5 Decision Trees

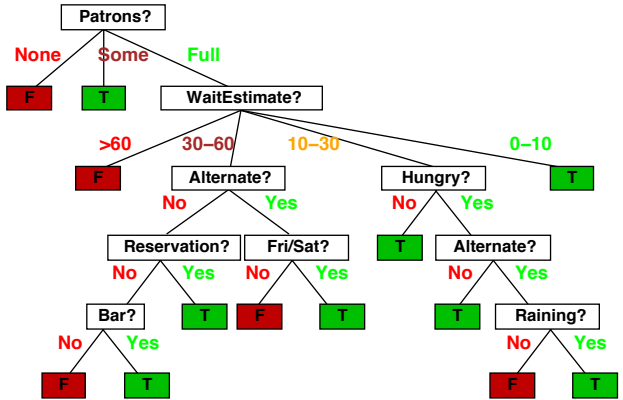
Attribute-based representation

- Examples described by attribute values (Boolean, discrete, continuous, etc.)
  E.g., situations where I will/won't wait for a table:

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*T*	*F*	*F*	*T*	*Some*	*$$$*	*F*	*T*	*French*	*0–10*	*T*
X_2	*T*	*F*	*F*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*30–60*	*F*
X_3	*F*	*T*	*F*	*F*	*Some*	*$*	*F*	*F*	*Burger*	*0–10*	*T*
X_4	*T*	*F*	*T*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*10–30*	*T*
X_5	*T*	*F*	*T*	*F*	*Full*	*$$$*	*F*	*T*	*French*	*>60*	*F*
X_6	*F*	*T*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Italian*	*0–10*	*T*
X_7	*F*	*T*	*F*	*F*	*None*	*$*	*T*	*F*	*Burger*	*0–10*	*F*
X_8	*F*	*F*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Thai*	*0–10*	*T*
X_9	*F*	*T*	*T*	*F*	*Full*	*$*	*T*	*F*	*Burger*	*>60*	*F*
X_{10}	*T*	*T*	*T*	*T*	*Full*	*$$$*	*F*	*T*	*Italian*	*10–30*	*F*
X_{11}	*F*	*F*	*F*	*F*	*None*	*$*	*F*	*F*	*Thai*	*0–10*	*F*
X_{12}	*T*	*T*	*T*	*T*	*Full*	*$*	*F*	*F*	*Burger*	*30–60*	*T*

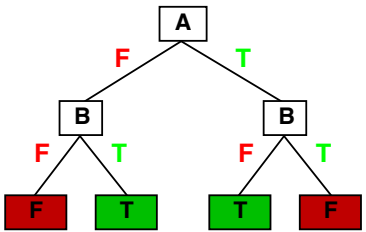
- **Classification** of examples is **positive** (T) or **negative** (F)

One possible representation for hypotheses
E.g., here is the “true” tree for deciding whether to wait:



- Decision trees can express any function of the input attributes.
  E.g., for Boolean functions, truth table row $\rightarrow$ path to leaf:

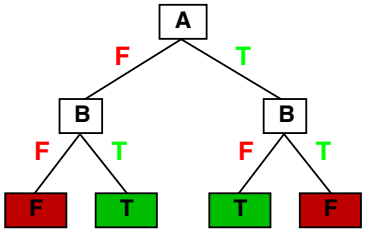
A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



- Trivially, there is a consistent decision tree for any training set
  w/ one path to leaf for each example (unless f nondeterministic in x)
  but it probably won't generalize to new examples

- Decision trees can express any function of the input attributes.
  E.g., for Boolean functions, truth table row $\rightarrow$ path to leaf:

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



- Trivially, there is a consistent decision tree for any training set
  w/ one path to leaf for each example (unless f nondeterministic in x)
  but it probably won't generalize to new examples
- Prefer to find more compact decision trees

- *How many distinct decision trees with n Boolean attributes*

- *How many distinct decision trees with n Boolean attributes*
  - = number of Boolean functions

- *How many distinct decision trees with n Boolean attributes*
  - = number of Boolean functions
  - = number of distinct truth tables with 2^n rows = 2^{2^n}

- *How many distinct decision trees with n Boolean attributes*
  - = number of Boolean functions
  - = number of distinct truth tables with 2^n rows = 2^{2^n}
- E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 functions (many more trees)

Hypothesis spaces

- *How many distinct decision trees with n Boolean attributes*
  - = number of Boolean functions
  - = number of distinct truth tables with 2^n rows = 2^{2^n}
- E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 functions (many more trees)
- *How many purely conjunctive hypotheses (e.g., $Hungry \wedge \neg Rain$)*

- *How many distinct decision trees with n Boolean attributes*
  - = number of Boolean functions
  - = number of distinct truth tables with 2^n rows = 2^{2^n}
- E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 functions (many more trees)
- *How many purely conjunctive hypotheses (e.g., $Hungry \wedge \neg Rain$)*
- Each attribute can be in (positive), in (negative), or out
  $\Longrightarrow$ 3^n distinct conjunctive hypotheses
- More expressive hypothesis space
  - increases chance that target function can be expressed (Good)
  - increases number of hypotheses consistent w/ training set
    $\Longrightarrow$ may get worse predictions (Bad)

- In general, decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances.
- Each path from the tree root to a leaf corresponds to a conjunction of attribute tests.
- If we are trying to learn an arbitrarily large boolean function, the decision tree may be exponential in the number of attributes.
- But decision trees are well suited to certain types of problems.

When to use Decision Trees

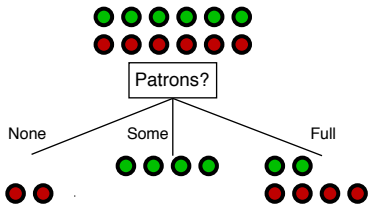
- Decision trees work well when
  - Instances are represented by attribute-value pairs with a fixed set of attributes. Works best when each attribute takes on a small number of disjoint possible values. Extensions required to handle numerical values.
  - The target function has discrete output values (e.g. yes/no, true/false, red/green/blue). Extensions allow real-valued outputs, but this usage is not typical.
  - Disjunctive descriptions may be required (e.g. I will do this if x or if y).
  - Training data may contain errors.
  - Training data may contain missing attribute values.
- It is easy to use an existing decision tree, but how can a decision tree be automatically created?

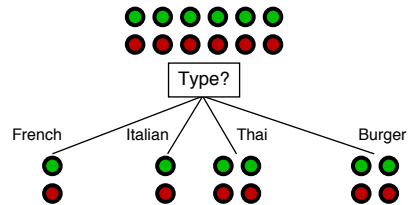
- Aim: find a small tree consistent with the training examples
- Idea: (recursively) choose "most significant" attribute as root of (sub)tree

function Decision-Tree-Learning($examples$, $attributes$, $parent\_examples$) **returns** a tree
  if $examples$ is empty **then return** Plurality-Value($parent\_examples$)
  else if all $examples$ have the same classification **then return** the classification
  else if $attributes$ is empty **then return** Plurality-Value($examples$)
  else
    $A \leftarrow \arg\max_{a \in attributes}$ Importance(a, $attributes$)
    $tree \leftarrow$ a new decision tree with root test A
    for each value v_k of A **do**
      $exs \leftarrow \{e : e \in examples$ **and** $e.A = v_k\}$
      $subtree \leftarrow$ Decision-Tree-Learning(exs, $attributes - A$, $examples$)
      add a branch to $tree$ with label $(A = v_k)$ and subtree $subtree$

Choosing an attribute

- Idea: a good attribute splits the examples into subsets that are (ideally) "all positive" or "all negative"





- $Patrons?$ is a better choice—gives information about the classification

- We need to obtain a measure of how good a split an attribute achieves.
- It should be maximal when the attribute is perfect, and minimal when it is useless.
- The expected amount of information makes a suitable measure.
  - Consider a coin toss. Finding out that the coin landed on heads provides you with some information.
  - But if the coin was loaded, this might be expected, so less information is provided.
  - Information theory measures information in bits; one bit is enough to answer a yes/no question about which one has no idea.
  - If $v_1 \dots v_n$ are the possible answers to a question, then the total information content can be measured as follows (**entropy**):

$$H(V) = \sum_{k=1}^{n} -P(v_k) \log_2 P(v_k)$$

- For decision tree learning, we are trying to answer the question of how to classify a single example.
- An estimate of the initial probabilities of possible answers before testing attributes is given by the ratio of positive to negative examples in the training set.
- Then the amount of information contained in a correct answer, for a binary variable with probability q is

$$B(q) = \left(\frac{p}{p+n}\right)$$

Information Gain

- Now an attribute A with d possible values:
  - divides the training set E into d subsets $E_1, \ldots E_d$; and
  - each subset E_k has p_k positive and n_k negative values.
- Amount of information needed to categorise the remaining examples is $B(p_k/(p_k + n_k))$
- A randomly chosen example from the training set has the kth value for the attribute with probability $(p_k + n_k)/(p + n)$
- So on average, after testing attribute A, we need the following amount of information to classify the example:

$$Remainder(A) = \sum_{k=1}^{d} \frac{p_k + n_k}{p + n} B\left(\frac{p_k}{p_k + n_k}\right)$$

- The information gain from the attribute is the difference between the original information requirement and the new requirement.

$$Gain(A) = B\left(\frac{p}{p + n}\right) - Remainder(A)$$

Information Gain (cont)

- We select the attribute that has the largest information gain.
- Suppose we have p positive and n negative examples at the root
  $\Longrightarrow B(q) = \left(\frac{p}{p+n}\right)$ bits needed to classify a new example
  E.g., for 12 restaurant examples, $p = n = 6$ so we need 1 bit
- An attribute splits the examples E into subsets E_k, each of which (we hope) needs less information to complete the classification
- Let E_k have p_k positive and n_k negative examples
  $\Longrightarrow (p_k + n_k)/(p + n)$ bits needed to classify a new example
  $\Longrightarrow$ expected number of bits per example over all branches is

$$\sum_{k=1}^{d} \frac{p_k + n_k}{p + n} B\left(\frac{p_k}{p_k + n_k}\right)$$

Information gain in the Restaurant Example

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	*T*	*F*	*F*	*T*	*Some*	*$$$*	*F*	*T*	*French*	*0–10*	*T*
X_2	*T*	*F*	*F*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*30–60*	*F*
X_3	*F*	*T*	*F*	*F*	*Some*	*$*	*F*	*F*	*Burger*	*0–10*	*T*
X_4	*T*	*F*	*T*	*T*	*Full*	*$*	*F*	*F*	*Thai*	*10–30*	*T*
X_5	*T*	*F*	*T*	*F*	*Full*	*$$$*	*F*	*T*	*French*	*>60*	*F*
X_6	*F*	*T*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Italian*	*0–10*	*T*
X_7	*F*	*T*	*F*	*F*	*None*	*$*	*T*	*F*	*Burger*	*0–10*	*F*
X_8	*F*	*F*	*F*	*T*	*Some*	*$$*	*T*	*T*	*Thai*	*0–10*	*T*
X_9	*F*	*T*	*T*	*F*	*Full*	*$*	*T*	*F*	*Burger*	*>60*	*F*
X_{10}	*T*	*T*	*T*	*T*	*Full*	*$$$*	*F*	*T*	*Italian*	*10–30*	*F*
X_{11}	*F*	*F*	*F*	*F*	*None*	*$*	*F*	*F*	*Thai*	*0–10*	*F*
X_{12}	*T*	*T*	*T*	*T*	*Full*	*$*	*F*	*F*	*Burger*	*30–60*	*T*

$$Gain(A) = B\left(\frac{p}{p+n}\right) - Remainder(A)$$

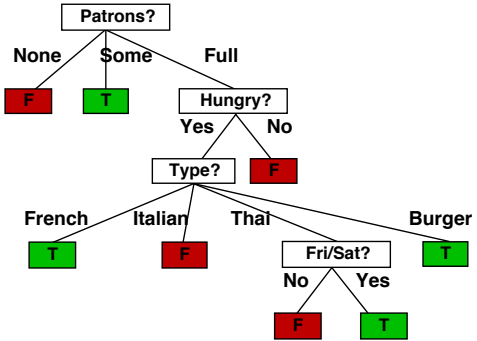
$$Gain(Patrons) = 1 - \left[\frac{2}{12}B\left(\frac{0}{2}\right) + \frac{4}{12}B\left(\frac{4}{4}\right) + \frac{6}{12}B\left(\frac{2}{6}\right)\right] \approx 0.541 \; bits$$

$$Gain(Type) = 1 - \left[\frac{2}{12}B\left(\frac{1}{2}\right) + \frac{2}{12}B\left(\frac{1}{2}\right) + \frac{4}{12}B\left(\frac{2}{4}\right) + \frac{4}{12}B\left(\frac{2}{4}\right)\right] = 0 \; bits$$

$\overset{38/49}{\Longrightarrow}$ choose the attribute that maximises information gain

Example (cont)

Decision tree learned from the 12 examples:



Substantially simpler than “true” tree—a more complex hypothesis is not justified by small amount of data

Problems with Decision Tree Learning

- Our basic algorithm grows each branch of the tree just deeply enough to perfectly classify the training examples.
- If there is noise in the data, or we have too few training examples to produce a representative sample, this is a problem.
- Our solution would then produce trees that over fit the training examples.

- Two main approaches to avoiding overfitting (in decision trees):
  - We can stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data.
  - We can allow the tree to over fit the data, and then post prune the tree (i.e. remove some branches).
- In practice the second approach seems to work better as its difficult to estimate when to stop growing the tree.

- We know we need to prune or stop the tree at a certain size, but how do we determine what the correct size is?
  - We could use a separate set of examples to evaluate the utility of post-pruning nodes from the tree.
  - Use statistical tests to estimate whether expanding/pruning a node will produce an improvement beyond the training set.
  - Use an explicit measure of complexity of the tree, halting tree growth when this is exceeded.
- The first of these is the most common, and is referred to as a training and validation set approach.
  - It separates data into two sets, a training set used to form the learned hypothesis, and a validation set used to evaluate the accuracy of the hypothesis and to evaluate the impact of pruning.
  - Idea: random errors in the training set are not likely to be exhibited in the validation set.

Reduced Error Pruning

- Each node in the tree is a candidate for pruning.
- Pruning removes the subtree rooted at the node, turning the node into a leaf node.
- The value of this leaf node is the most common classification of the training examples associated with the node.
- Nodes are removed only if the pruned tree performs no worse than the original over the validation set.
- Given a choice of multiple nodes to remove, the one which most increases accuracy is removed.
- This approach works very well if large amounts of data are available.

- Pruning does not only allow us to improve performance by overcoming over fitting
- We can now naturally handle noisy data
- There are still several issues we ideally need to handle
  - Continuous value in input attributes
  - Continuous value in output attributes (we can mix decision trees with regression to handle this).
  - Attributes with large domains (handled by changing our gain measure)
  - Attributes with differing costs (handled by adding a cost function to the gain measure)
  - Missing data

- Dealing with continuous value input is relatively simple.
- We discretise the attribute into the learned tree.
- E.g. split at value c, going left (or returning an answer) if the input is less than c.
- But how do we decide what the threshold should be?
- We pick it in such a way so as to maximise information gain.

- Simplest approach to missing data is to assign it the most common value among training examples at that node.
- A more complex approach could assign this value based on the probability of observed values in non-missing data.

Outline

Introduction to Machine Learning Summary

- Introduction to Machine Learning
- Supervised vs. Unsupervised Learning
- Decision Trees
  - Information Gain
  - Pruning

Any Questions.