Learning

Learning is the ability to improve one's behavior based on experience.

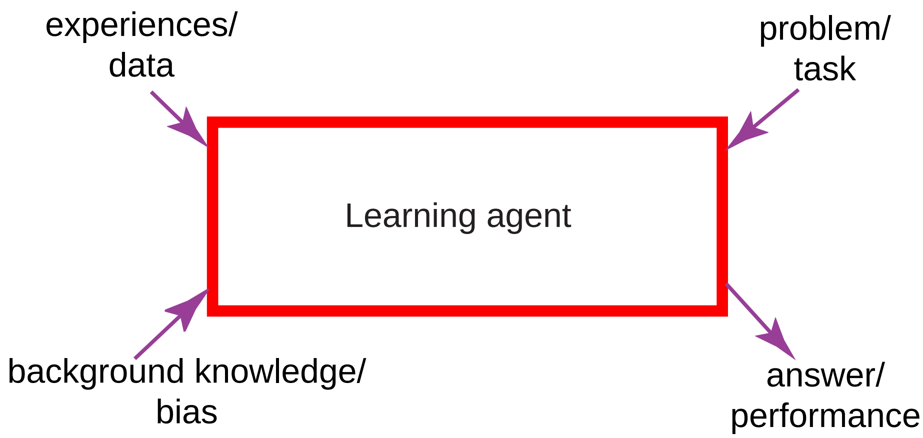
- The range of behaviors is expanded: the agent can do more.
- The accuracy on tasks is improved: the agent can do things better.
- The speed is improved: the agent can do things faster.

Components of a learning problem

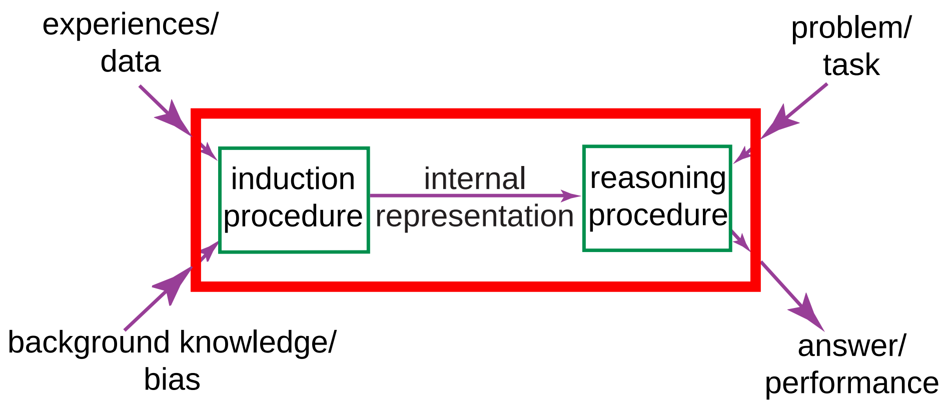
The following components are part of any learning problem:

- $\boxed{\text{task}}$ The behavior or task that's being improved.
  For example: classification, acting in an environment
- $\boxed{\text{data}}$ The experiences that are being used to improve performance in the task.
- $\boxed{\text{measure of improvement}}$ How can the improvement be measured?
  For example: increasing accuracy in prediction, new skills that were not present initially, improved speed.

Learning task



Learning architecture



Choosing a representation

- The richer the representation, the more useful it is for subsequent problem solving.
- The richer the representation, the more difficult it is to learn.

Common Learning Tasks

- Supervised classification Given a set of pre-classified training examples, classify a new instance.
- Unsupervised learning Find natural classes for examples.
- Reinforcement learning Determine what to do based on rewards and punishments.
- Analytic learning Reason faster using experience.
- Inductive logic programming Build richer models in terms of logic programs.

Example Classification Data

	Action	Author	Thread	Length	Where
e1	skips	known	new	long	home
e2	reads	unknown	new	short	work
e3	skips	unknown	old	long	work
e4	skips	known	old	long	home
e5	reads	known	new	short	home
e6	skips	known	old	long	work

We want to classify new examples on property *Action* based on the examples' *Author*, *Thread*, *Length*, and *Where*.

Feedback

Learning tasks can be characterized by the feedback given to the learner.

- Supervised learning What has to be learned is specified for each example.
- Unsupervised learning No classifications are given; the learner has to discover categories and regularities in the data.
- Reinforcement learning Feedback occurs after a sequence of actions.

Measuring Success

- The measure of success is not how well the agent performs on the training examples, but how well the agent performs for new examples.
- Consider two agents:
  - $\boxed{P}$ claims the negative examples seen are the only negative examples. Every other instance is positive.
  - $\boxed{N}$ claims the positive examples seen are the only positive examples. Every other instance is negative.
- Both agents correctly classify every training example, but disagree on every other example.

Bias

- The tendency to prefer one hypothesis over another is called a bias.
- Saying a hypothesis is better than N's or P's hypothesis isn't something that's obtained from the data.
- To have any inductive process make predictions on unseen data, you need a bias.
- What constitutes a good bias is an empirical question about which biases work best in practice.

Learning as search

- Given a representation and a bias, the problem of learning can be reduced to one of search.
- Learning is search through the space of possible representations looking for the representation or representations that best fits the data, given the bias.
- These search spaces are typically prohibitively large for systematic search. Use hill climbing.
- A learning algorithm is made of a search space, an evaluation function, and a search method.

Noise

- Data isn't perfect:
  - some of the attributes are assigned the wrong value
  - the attributes given are inadequate to predict the classification
  - there are examples with missing attributes
- overfitting occurs when a distinction appears in the data, but doesn't appear in the unseen examples. This occurs because of random correlations in the training set.

Characterizations of Learning

- Find the best representation given the data.
- Delineate the class of consistent representations given the data.
- Find a probability distribution of the representations given the data.

Learning Decision Trees

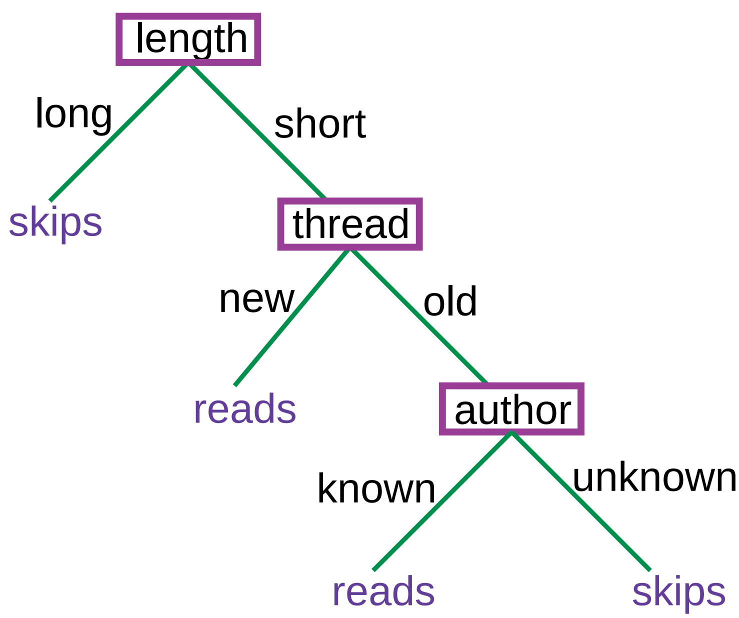
- Representation is a decision tree.
- Bias is towards simple decision trees.
- Search through the space of decision trees, from simple decision trees to more complex ones.

Decision trees

A decision tree is a tree where:

- The nonleaf nodes are labeled with attributes.
- The arcs out of a node labeled with attribute A are labeled with each of the possible values of the attribute A.
- The leaves of the tree are labeled with classifications.

Example Decision Tree



Equivalent Logic Program

$prop(Obj, user\_action, skips) \leftarrow$
$\quad prop(Obj, length, long).$
$prop(Obj, user\_action, reads) \leftarrow$
$\quad prop(Obj, length, short) \wedge prop(Obj, thread, new).$
$prop(Obj, user\_action, reads) \leftarrow$
$\quad prop(Obj, length, short) \wedge prop(Obj, thread, old) \wedge$
$\quad prop(Obj, author, known).$
$prop(Obj, user\_action, skips) \leftarrow$
$\quad prop(Obj, length, short) \wedge prop(Obj, thread, old) \wedge$
$\quad prop(Obj, author, unknown).$

Issues in decision-tree learning

- Given some data, which decision tree should be generated? A decision tree can represent any discrete function of the inputs.
- You need a bias. Example, prefer the smallest tree. Least depth? Fewest nodes? Which trees are the best predictors of unseen data?
- How should you go about building a decision tree? The space of decision trees is too big for systematic search for the smallest decision tree.

Searching for a Good Decision Tree

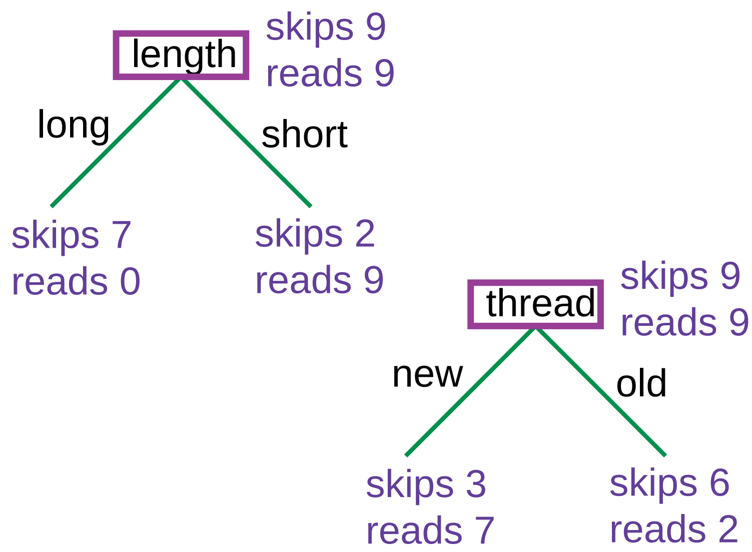
- The input is a target attribute (the *Goal*), a set of examples, and a set of attributes.
- Stop if all examples have the same classification.
- Otherwise, choose an attribute to split on,
  - for each value of this attribute, build a subtree for those examples with this attribute value.

Decision tree learning: Boolean attributes

% $dtlearn(Goal, Examples, Attributes, DT)$ given *Examples*
% and *Attributes* construct decision tree *DT* for *Goal*.

$dtlearn(Goal, Exs, Atts, Val) \leftarrow$
$\quad all\_examples\_agree(Goal, Exs, Val).$
$dtlearn(Goal, Exs, Atts, if(Cond, YT, NT)) \leftarrow$
$\quad examples\_disagree(Goal, Exs) \wedge$
$\quad select\_split(Goal, Exs, Atts, Cond, Rem\_Atts) \wedge$
$\quad split(Exs, Cond, Yes, No) \wedge$
$\quad dtlearn(Goal, Yes, Rem\_Atts, YT) \wedge$
$\quad dtlearn(Goal, No, Rem\_Atts, NT).$

Example: possible splits



Using this algorithm in practice

- Attributes can have more than two values. This complicates the trees.
- This assumes attributes are adequate to represent the concept. You can return probabilities at leaves.
- Which attribute to select to split on isn't defined. You want to choose the attribute that results in the smallest tree. Often we use information theory as an evaluation function in hill climbing.
- Overfitting is a problem.

Handling Overfitting

- This algorithm gets into trouble overfitting the data. This occurs with noise and correlations in the training set that are not reflected in the data as a whole.
- To handle overfitting:
  - You can restrict the splitting, so that you split only when the split is useful.
  - You can allow unrestricted splitting and prune the resulting tree where it makes unwarranted distinctions.