Learning Decision Trees

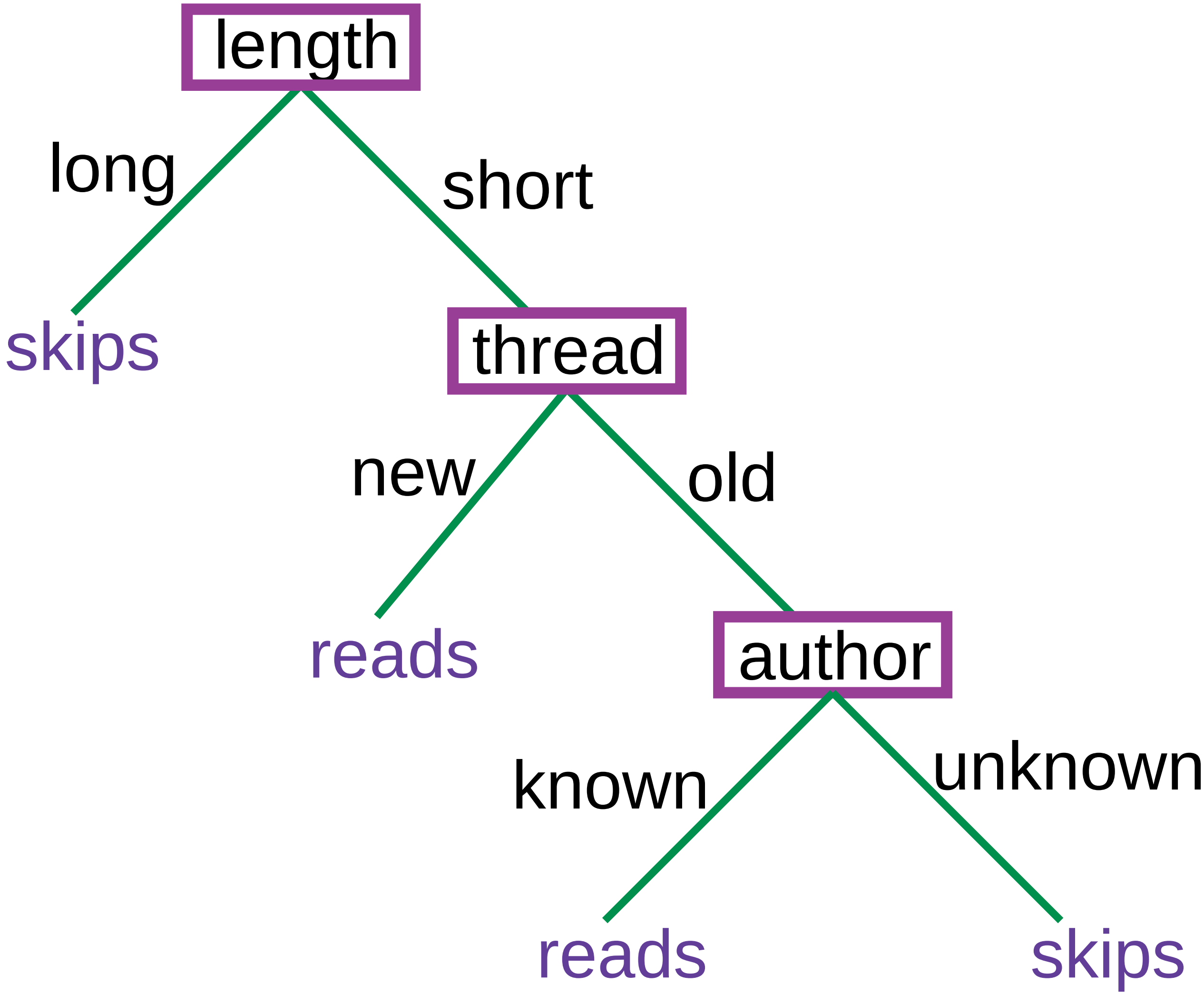
- Representation is a decision tree.
- Bias is towards simple decision trees.
- Search through the space of decision trees, from simple decision trees to more complex ones.

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

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

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.

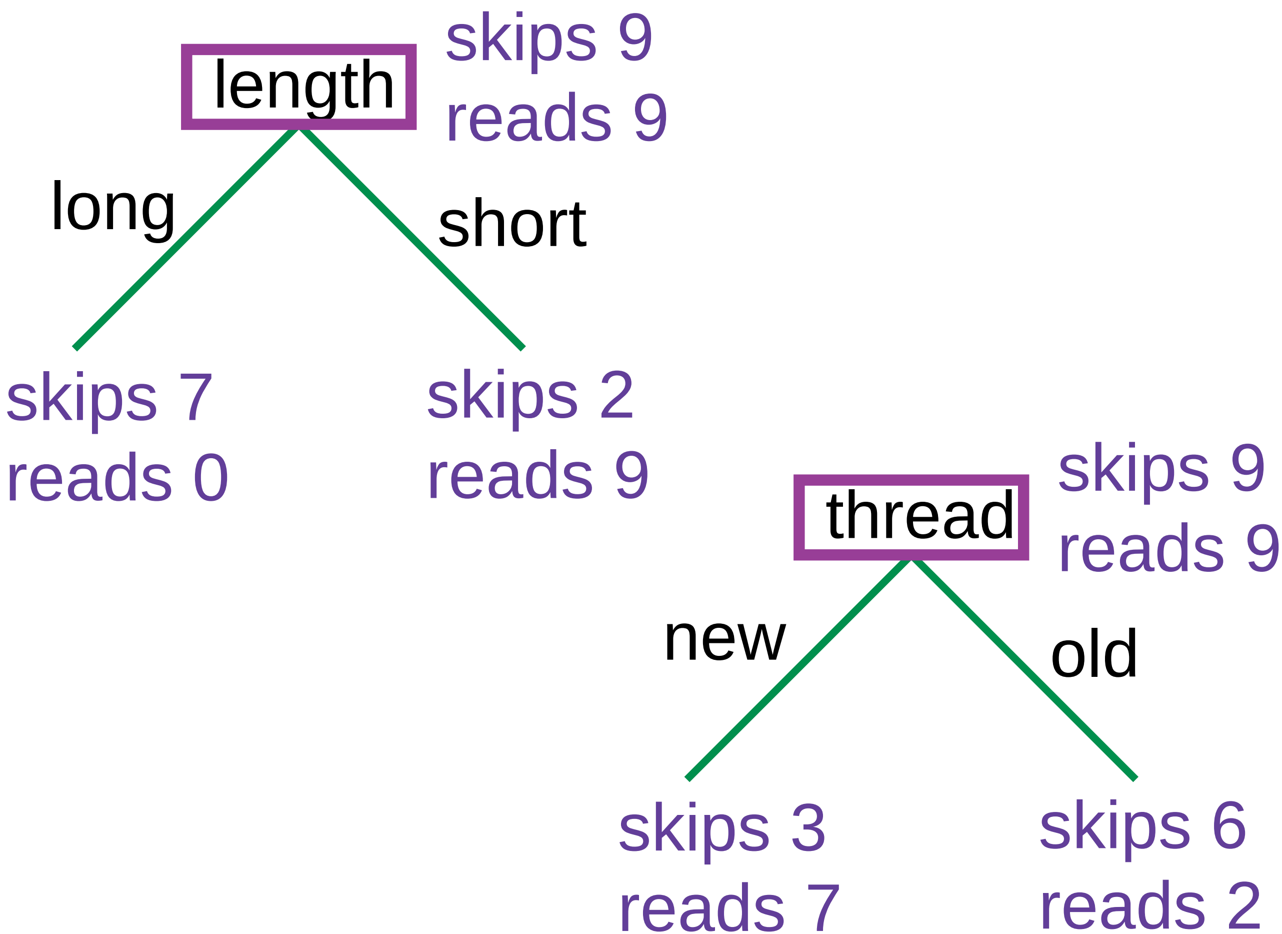
Decision tree learning: Boolean attributes

% $dtlearn(Goal, Examples, Attributes, DT)$ given $Examples$

% and $Attributes$ construct decision tree DT for $Goal$.

$$
\begin{array}{l}
dtlearn(Goal, Exs, Atts, Val) \leftarrow \\
\qquad all\_examples\_agree(Goal, Exs, Val). \\
dtlearn(Goal, Exs, Atts, if(Cond, YT, NT)) \leftarrow \\
\qquad examples\_disagree(Goal, Exs) \wedge \\
\qquad select\_split(Goal, Exs, Atts, Cond, Rem\_Atts) \wedge \\
\qquad split(Exs, Cond, Yes, No) \wedge \\
\qquad dtlearn(Goal, Yes, Rem\_Atts, YT) \wedge \\
\qquad dtlearn(Goal, No, Rem\_Atts, NT).
\end{array}
$$

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.