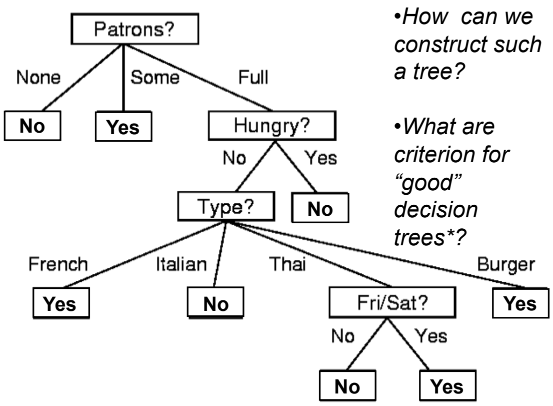
Lecture 23: Learning 2

Victor R. Lesser
CMPSCI 683
Fall 2010

Today's Lecture

- Continuation of Decision Tree Algorithms for Classification
  - How do we construct them
- Neural Networks



•How can we construct such a tree?

•What are criterion for "good" decision trees?*

Constructing the Decision Tree

Construct a root node that *includes all the examples*, then for each node:

1. if there are both positive and negative examples, choose the best attribute to split them.
2. if all the examples are pos (neg) answer yes (no).
3. if there are no examples for a case (no observed examples) then choose a default based on the majority classification at the parent. – suppose you split on attribute raining
   - *No case of raining -yes* when hungry- yes, alternate - yes
   - No (x2,x10) Yes (x1,x4,x12) when hungry- yes, alternate - yes
4. if there are no attributes left but we have both pos and neg examples, this means that the selected features are not sufficient for classification or that there is error in the examples. (can use majority vote.)

Decision Tree Algorithm*

- Basic idea is to build the tree *greedily*.
  - Decisions once made are not revised
  - No search -- WHY?
- Choose "most significant attribute" to be the root. Then split the dataset based on values of attribute, and repeat process on new terminal leaves.
  - One step look ahead
- Define "significance" using information theory (based on information gain or "entropy").

Finding the smallest decision tree is an intractable problem

Choosing the Best Attribute Based on Information Theory*

- Expected amount of information provided by an attribute
  - *Similar to the concept of value of perfect information?*
- Amount of information content in a set of examples
  - v_i possible answers, $P(v_i)$ probability of occurring

$$I(P(v_1),\ldots,P(v_n)) = \sum_{i=1}^{n} -P(v_i)\log_2 P(v_i)$$

  - *Example 12 cases, 6 pos, 6 neg; information 1 bit*

$$I\left(\frac{p}{p+n},\frac{n}{p+n}\right) = -\frac{p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

  - *Info content after splitting, v values of Attribute A*

$$remainder(A) = \sum_{i=1}^{v}\frac{p_i+n_i}{p+n} I\left(\frac{p_i}{p_i+n_i},\frac{n_i}{p_i+n_i}\right)$$

Choosing the Best Attribute Based on Information Theory cont.*

$$Gain(A) = \left(I\frac{p}{p+n},\frac{n}{p+n}\right) - remainder(A)$$

Prior to splitting - after splitting

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

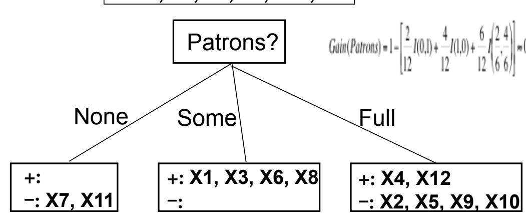
$$Gain(Type) = 1 - \left[\frac{2}{12}I\left(\frac{1}{2},\frac{1}{2}\right) + \frac{2}{12}I\left(\frac{2}{4},\frac{2}{4}\right) + \frac{4}{12}I\left(\frac{2}{4},\frac{2}{4}\right) + \frac{4}{12}I\left(\frac{2}{4},\frac{2}{4}\right)\right] = 0b$$

Choose attribute with largest gain which is Patrons

Splitting the Examples

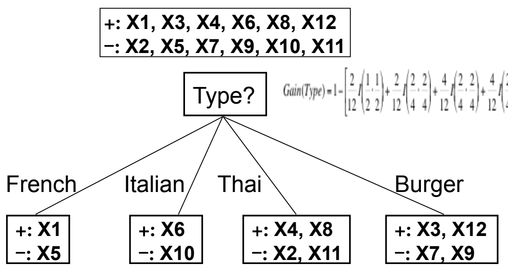
- A perfect attribute divides the examples into sets that are all positive and negative

+: X1, X3, X4, X6, X8, X12
−: X2, X5, X7, X9, X10, X11



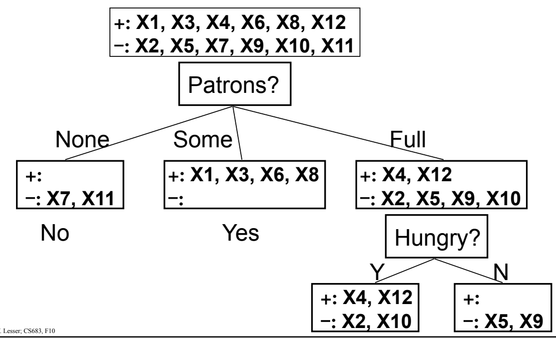
$$Gain(Patrons) = 1 - \left[\frac{2}{12}I(0,1) + \frac{4}{12}I(1,0) + \frac{6}{12}I\left(\frac{2}{6},\frac{4}{6}\right)\right] = 0.541 bits$$

Splitting Examples cont.



$$Gain(Type) = 1 - \left[\frac{2}{12} I\left(\frac{1}{2}, \frac{1}{2}\right) + \frac{2}{12} I\left(\frac{2}{4}, \frac{2}{4}\right) + \frac{4}{12} I\left(\frac{2}{4}, \frac{2}{4}\right) + \frac{4}{12} I\left(\frac{2}{4}, \frac{2}{4}\right)\right] = 0b$$

Splitting Examples cont.



Example *(Quinlan '83)*

Class	Height	Hair	Eyes
-	Short	Blond	Brown
-	Tall	Dark	Brown
+	Tall	Blond	Blue
-	Tall	Dark	Blue
-	Short	Dark	Blue
+	Tall	Red	Blue
-	Tall	Blond	Brown
+	Short	Blond	Blue

HEIGHT

	SHORT	TALL
+	1	2
–	2	3

HAIR

	BLOND	DARK	RED
+	2	0	1
–	2	3	0

EYES

	BROWN	BLUE
+	0	3
–	3	2

Partition on <u>hair</u> has most information gain

Example *(Quinlan '83) cont.*

BLOND

Class	Height	Eyes
-	Short	Brown
+	Tall	Blue
-	Tall	Brown
+	Short	Blue

DARK

Class	Height	Eyes
-	Tall	Brown
-	Tall	Blue
-	Short	Blue

RED

Class	Height	Eyes
-	Tall	Blue

HEIGHT

	SHORT	TALL
+	1	1
-	1	1

$\frac{2}{4}\mathrm{I}(1/2,1/2) + \frac{2}{4}\mathrm{I}(1/2,1/2)$

1.0(Blond)- 1.0(Height)

EYES

	BROWN	BLUE
+	0	2
-	2	0

$\frac{2}{4}\mathrm{I}(0,2) + \frac{2}{4}\mathrm{I}(2,0)$

1.0(Blond)- 0.0(Eyes)

Dividing Blond by attribute Height or Eyes

EYES ARE BETTER ATTRIBUTE (Gain of 1 vs. 0)

Decision tree learning

```
function DTL(examples, attributes, default) returns a decision tree
  if examples is empty then return default
  else if all examples have the same classification then return the classification
  else if attributes is empty then return MODE(examples)
  else
     best ← CHOOSE-ATTRIBUTE(attributes, examples)
     tree ← a new decision tree with root test best
     for each value v_i of best do
        examples_i ← {elements of examples with best = v_i}
        subtree ← DTL(examples_i, attributes − best, MODE(examples)); recur
        add a branch to tree with label v_i and subtree subtree
     return tree
```


Hypothesis Space Search in Decision Tree

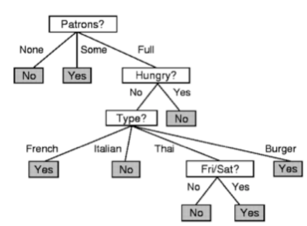
- Complete space of finite discrete-valued functions relative to available attributes
- Maintains only a single current hypothesis (decision tree)
- Performs no backtracking in its search
- Uses all training examples at each step in the search to make statistically-based decisions regarding how to refine current hypothesis

Inductive Bias in Decision Tree Construction

- Selects in favor of shorter trees over longer ones
- Selects trees that place the attributes with highest information gain closest to the root

Full Learned Decision Tree

- How "correct" is this?
  - Can we even judge this idea?
  - Not all attributes used
- How does the number of examples seen relate to the likelihood of "correctness"?



Performance Measurement

- How do we measure how close our hypothesis

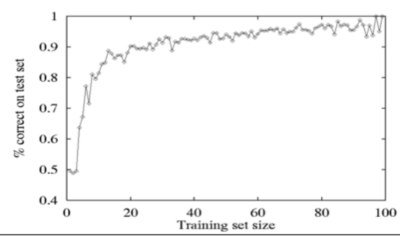
  h(x) is to *f(x)*?

- Try *h(x)* on a *test set (data not trained on)*

- Learning curve: Measure % correct predictions on the test set as a function of the size of the training set.

Assessing Performance of Learning Algorithm

- Randomly divide available examples into test and training set



Could this be lower than 1 with test and training equal ?

Is that necessarily bad?

A learning curve for the decision tree algorithm on 100 randomly generated examples in the restaurant domain. The graph summarizes 20 trials.

Overfitting in Decision Trees

A hypothesis *overfits* the training examples if there is some other hypothesis that fits the *training examples less well, yet actually performs better over the entire distribution of instances*

- Causes of overfitting
  - Lack of Examples — small number of examples associated with leaf
    - Coincidental regularities cause the construction of more detail tree than warranted
  - Noisy Data — construct tree to explain noisy data

Avoiding Overfitting

- Stop growing the tree earlier, before it reaches the point where it perfectly classifies the training data

- Post-prune the tree
  - Use non-training instances (test data) to evaluate based on a statistical test to estimate whether pruning a particular node is likely to produce an improvement beyond the training set

Broadening the applicability - Missing Data

- 1: Add new attribute value - "unknown"
- 2: Estimate missing value based on other examples for which this attribute has a known value
  - Assign value that is most common among training examples at parent node – why parent node?
- 3: Instantiated example with all possible values of missing attribute but assign weights to each instance based on likelihood of missing value being a particular value given the distribution of examples in the parent node
  - Modify decision tree algorithm to take into account weighting

Broadening the applicability - Multi-valued Attributes

- Handling multi-valued (large) attributes and classification
  - Need another measure of information gain
  - Information gain measure gives inappropriate indication of attributed usefulness because of likelihood of singleton values
  - Normalized Gain rather than Absolute Gain
    - Gain ratio--Gain over intrinsic information content

Broadening the Applicability - Continuous-Valued attributes

- Continuous-valued attributes
  - Discretize
    - Example $, $$, $$$
  - Preprocess to find out which ranges give the most useful information for classification purposes

Preprocessing for Continuous-Valued Attributes

- Sort instances based on value of an attribute (e.g. temperature)
- Identify adjacent examples that differ in their target classification
- Generate a set of candidate thresholds midway between corresponding examples
- Use information gain to decide appropriate threshold

Incremental Decision Tree Construction

- Assumed all case available at start of construction of decision tree
  - Exploits knowledge of all cases to make decisions what attributes to use next
- What happens if we are doing the learning on-line
  - Reconstruction decision tree after you acquire a certain number of new cases vs.
  - Approach tree construction as incremental process where as you acquire new information you exploit it

Neural Networks

- Representing functions using networks of simple arithmetic computing elements
- Learning such representations from examples

Biological Inspiration Learning: The Brain

- Approximately 10^{11} neurons, 10^4 synapses (connections) per neuron.
- Neuron "fires" when its inputs exceed a threshold.
- Inputs are weighted and can have excitory or inhibitory effect.
- Individual firing is slow ($\approx$ .001 second) but bandwidth is very high ($\approx 10^{14}$ bits/sec).
- The brain performs many tasks much faster than a computer (Scene recognition time $\approx$ .1 second).
  - *Turning point coming in 2015? – Computational power of computers equal that of Brain– Called the Singularity!*
- Learning and graceful degradation.

What is Connectionist Computation?

Computational architectures and cognitive models that are neurally-inspired:

- Faithful to coarse neural constraints — not neural models
- Large numbers of simple (neuron-like) processing units interconnected through weighted links
- They compute by transmitting symbolically coded messages
  - Inhibitory and excitory signals
- *"program" in the structure of the interconnections*
- "massive parallelism" and no centralized control

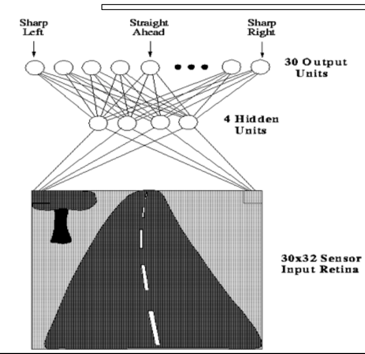
Some Properties of Connectionist Systems

- *Ability to bring large numbers of interacting soft constraints to bear on problem solving*
- *Noise resistance, error tolerance, graceful degradation*
- Ability to do complex multi-layer recognition with a large number of inputs/outputs (quickly)
- Learning with generalization
- Biological plausibility
- Potential for speed of processing through fine-grained parallelism

Applications of neural networks

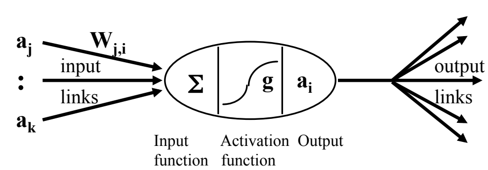
- Automobile automatic guidance systems
- Credit application evaluation, mortgage screening, real estate appraisal
- Object recognition (faces, characters)
- Speech recognition and voice synthesis
- Market forecasting, automatic bond trading
- Robot control, process control
- Breast cancer cell analysis
- Oil and gas exploration
- Image and data compression

ALVINN drives 70 mph on highways

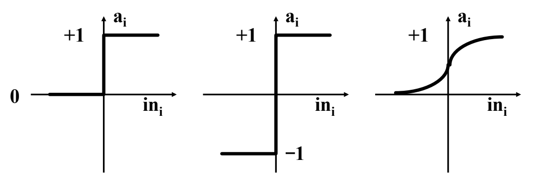


Artificial Neural Networks

- Compose of nodes/units connected by links
- Each link has a numeric weight associated with it
- Processing units compute weighted sum of their inputs, and then apply a threshold function.
  - Linear function combines inputs= sum ($w_{j,i}$• aj …. $w_{k,i}$• a_k); *interacting constraints*
  - Non-linear function **g** transforms combined input to activation value



Sample G's - activation functions



(a) Step function **(b) Sign function** **(c) Sigmoid function**

Can make each of these functions a threshold such that it outputs a 1 when the input is greater than threshold, can also do through dummy link

Representation of Boolean Functions

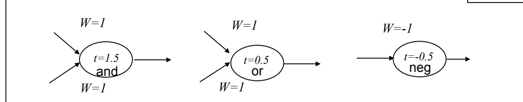
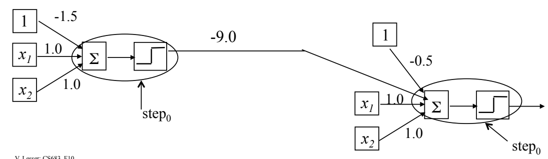


Figure 19.6 Units with a step function for the activation function can act as logic gates, given appropriate thresholds and weights.

XOR requires multi-layer network



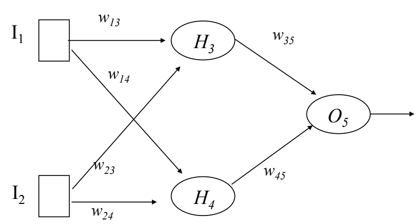
Neural Network Learning

- Robust approach to approximating real-valued, discrete-value and vector-valued target functions
- *Learning the Weights* (and Connectivity)
  - $w_{j,i} = 0$ implies no connectivity (no constraints) among nodes a_j and a_i

Network Structure

- Feed-Forward Networks: unidirectional links
  - No cycles (DAG)
  - No internal state other than weights
  - **Layered feed-forward**
    - Each unit is linked only to units in the next layer
    - Synchronized movement of information from layer to layer
  - Relatively understood

Multi-Layer Network: Hidden Units



A very simple, two-layer, feed-forward network with two inputs, two hidden nodes, and one output node.

Network Structure cont.

- Recurrent Network: arbitrary links
  - Activation is fed back to units that caused it
  - Internal state stored in activation levels
    - Notice no state held in feed-forward network
  - Can be unstable, oscillate etc.
  - *Can represent more complex functions*

Next Lecture

- Continuation of Neural Networks