26:198:722 Expert Systems

- Machine learning
- Rule-based Expert Systems
- CLIPS

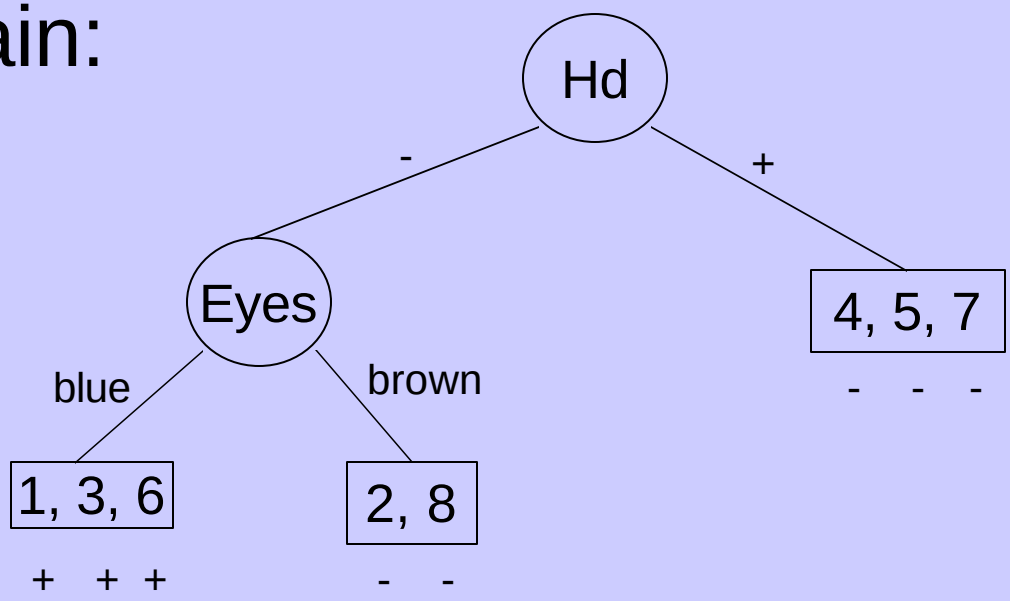
Machine Learning

- ASSISTANT uses Binarization

	----------Attributes----------					---Decision---
	Height	Hb	Hr	Hd	Eyes	Attractiveness
1	short	+	-	-	blue	+
2	tall	+	-	-	brown	-
3	tall	-	+	-	blue	+
4	short	-	-	+	blue	-
5	tall	-	-	+	blue	-
6	tall	+	-	-	blue	+
7	tall	-	-	+	brown	-
8	short	+	-	-	brown	-

Machine Learning

- Applying Information Gain criterion, we obtain:



Machine Learning

- This gives us rules:
  - (Hair, dark) --> (Attractiveness, -)
  - (Hair, ~dark) & (Eyes, blue) --> (Attractiveness, +)
  - (Hair, ~dark) & (Eyes, brown) --> (Attractiveness, -)
- and after dropping we obtain
  - (Hair, dark) --> (Attractiveness, -)
  - (Hair, ~dark) & (Eyes, blue) --> (Attractiveness, +)
  - (Eyes, brown) --> (Attractiveness, -)
- *Note that rules refer to original attributes, not the binarized forms*

Machine Learning

- ID4 is an incremental algorithm developed by Schlimmer &Fisher
  - Based on first case, no attributes needed - all examples are attractive
  - Based on first two cases, select best attribute using ID3 (or C4.5 algorithm): Height
  - Based on first three cases, select best attribute: Eyes
  - Etc. - ID4 will give same final result as ID3, but take much longer!

Machine Learning

- ID4-hat
  - Modifies ID4 by only re-building tree at each stage if existing tree fails to classify new item correctly - if the tree is not re-built,then it may no longer be the *best* tree
  - Now final tree may differ from ID3
  - For both ID4 and ID4-hat, either stop when entropy is 0 or maintain counts of the decision outcomes e.g., (+, -) and stop when all except one are 0

Machine Learning

- ID5 is an improved incremental algorithm by Utgoff
  - Start as for ID4
  - As each new item is added, if it is not correctly classified by the the existing tree, add the best next attribute using information gain as usual (otherwise keep the existing tree)
  - BUT, at each stage, if the bottom attribute has lower conditional entropy (for all items so far) than the attribute above it, rebuild the tree by splitting, inverting, merging and simplifying

Machine Learning

- ID5-hat
  - In ID5, conditional entropies are re-checked (and the tree re-structured if necessary) whenever an item is added
  - ID5-hat is the same as ID5, except that conditional entropies are only re-considered when an attribute has been added to a tree that failed to classify the new item correctly

Machine Learning

- After rules are generated via machine learning, we should check
  - Are they complete?
  - Are they consistent?
- As part of validation, we can count errors
  - If items are not classified
  - If items are classified incorrectly

Machine Learning

- Of course, if the original table is inconsistent, ID3-style algorithms cannot possibly classify without error
- Hence, Quinlan decided to simplify the tree by pruning: this still produces errors, but at lower cost
- From C4 onwards, Quinlan decided to prune even when no errors - this introduces error, but makes the tree more general and less dependent on the training set

Machine Learning

- Quinlan experimented with a variety of pruning algorithms, e.g.
  - cost-complexity pruning
  - χ^2 test
- C4.5 pruning is quite sophisticated, based on the Binomial distribution

Machine Learning

- Suppose there are N examples, and x is some number of classification errors (i.e cases not classified or mis-classified), based on an error rate of π
- Then $P(x,E)=\binom{N}{x}\pi^{x}(1-\pi)^{N-x}$
- Now let $U_{25\%}(E,N)=_{df}\ \pi\left|\sum_{x=1}^{x=E}P(x,N)=0.25\right.$

Machine Learning

- In other words, $U_{25\%}(E, N)$ is the predicted *error rate* that gives a total probability of 25% that the actual number of errors in N examples is E or less
- 25% is the default value in C4.5, although it can be changed

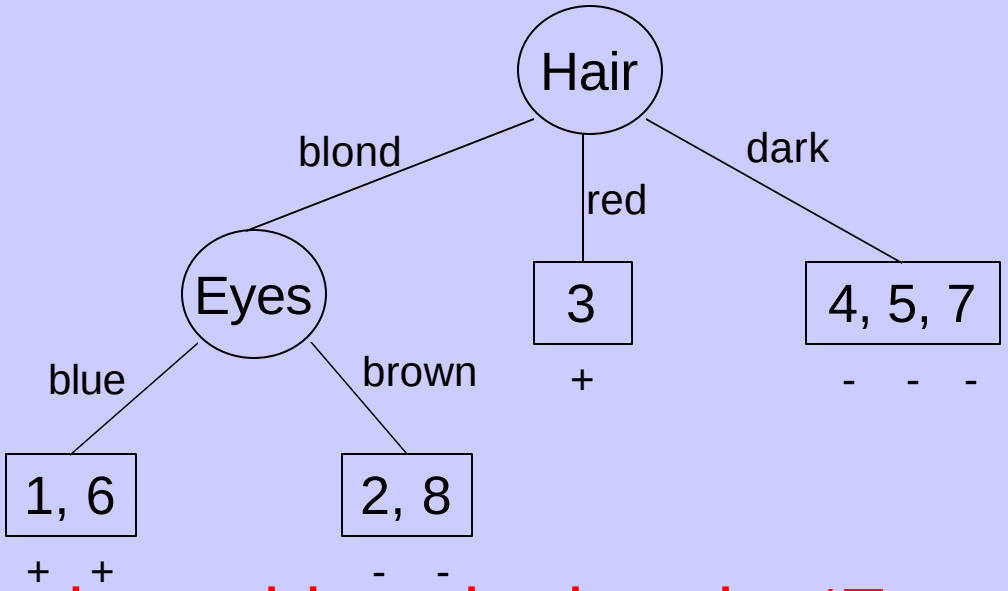
Machine Learning

- Some useful values of $U_{25\%}(E, N)$:

N	E	U
1	0	.750
2	0	.500
2	1	.866
3	0	.370
3	1	.674
3	2	.909
4	0	.293
4	1	.544
4	2	.757
4	3	.931

Machine Learning

- Recall the final ID3 tree from Class 4:



- and consider whether the ‘Eyes’ node should be pruned

Machine Learning

- Before pruning
  - Eyes: blue has two examples (1 and 6), neither incorrectly classified
    - so predicted *error rate* is .500
    - hence predicted *number* of errors is 2 x .500 = 1
  - Eyes: brown has two examples (2 and 8), neither incorrectly classified
    - so predicted *error rate* is .500
    - hence predicted *number* of errors is 2 x .500 = 1
  - Hence total predicted number of erros is 1 + 1 = 2

Machine Learning

- If Eyes pruned
  - Four examples (1, 2 , 6, 8) must all be treated as *the same*
  - Select whether + or - by *voting*: in this case, it is a 2-2 tie, so select + (occurs first)
  - Now there are 2 errors in four cases (two '-' that will be treated as '+') - so the predicted *error rate* is .757
  - Hence the predicted *number* of errors is 4 x .757 = 3.028

Machine Learning

- Pruning takes place only if it does not increase the *predicted number of errors*
  - In this case, it makes no sense to prune
- C4.5 pruning is based on applying this method recursively to the leaves of the tree
  - Once pruning of a branch stops, it obviously does not continue upwards
  - To avoid branches with single values, heuristically C4.5 does not even try to separate classes that are already reduced to only two elements

Machine Learning

- ID3-style algorithms also have problems with missing attributes. There are several possible approaches to this:
  - Ignore examples with missing data
  - Treat “missing” as a special value
  - Probabilistic approaches (e.g., C4.5)
  - Switch attribute and decision
  - Replicate example for all possible values

Machine Learning

- C4.5 chooses an attribute by considering entropies of each for those cases for which data is given
- *Thus H(decision) may now vary for each attribute*
- Information gain (ratio) is reduced pro-rata to proportion of complete cases to all cases
- When the best attribute is chosen, for purposes of later choices, missing cases are allocated fractionally to values of the attribute in proportion to cases for which data is available, and the algorithm continues

Machine Learning

- C4.5 also includes techniques to discretize “continuous” data:
  - Choose the best attribute using entropy
    - C4.5 use of information gain ratio will be important here to counteract bias of ID3 in favor of attributes with many different values
  - Make a binary split of the chosen attribute using minimum class entropy
  - Divide the examples into two sub-tables
  - Recurse

Rule-based Expert Systems

- Production Rules
  - Derived from *rewrite* rules in grammars
  - Emil Post (1943) studied canonical systems
  - Newell and Simon (1972) used for psychological modeling
  - Buchanan and Feigenbaum (1978) used for Expert Systems

Rule-based Expert Systems

- Canonical Systems
  - An alphabet *A* for making strings
  - Some strings that are taken as *axioms*
  - A set of productions of the form

    $\alpha_1 \$_1 ... \alpha_m \$_m \rightarrow \beta_1 \$'_1 ... \beta_n \$'_n$

    - each α_i and β_i is a fixed string
    - α_1 and α_m are often null
    - some or all of the α_i or β_i may be null
    - each $\$_i$ is a variable string which can be null
    - each $\$_i$ is replaced by a certain $\$'_i$

Rule-based Expert Systems

- Canonical Systems
  - Rewrite strings of symbols
  - Any formal system can be realized as a canonical system

Rule-based Expert Systems

- Production Rules
  - A set *N* of objects in the domain
  - A set *P* of property names that impute attributes to objects
  - A set *V* of values that these attributes can take
  - Object-attribute-value triples
    - (Pastis, alcohol-content, high))

Rule-based Expert Systems

- Production Systems
  - Rule set
    - Production memory
  - Rule interpreter
    - Decides when to apply which rules
  - Working memory
    - Holds data, goal statements, and intermediate results

Rule-based Expert Systems

- Production Rules

  $$P_1,\ldots,P_m \rightarrow Q_1,\ldots,Q_n$$

  read as

  - if *premises* P_1 and . . . and P_m are true
    then perform actions Q_1 and . . . and Q_n

- Premises are often known as conditions and actions as conclusions

Rule-based Expert Systems

- Premises are usually represented by object-attribute-value vectors
- Premises are patterns that are meant to match vectors in working memory
- Actions modify working memory
- Variables are used as placeholders for matching

Rule-based Expert Systems

- If a condition contains no variables, it is satisfied when an identical expression is present in working memory
- If a condition contains one or more variables, it is satisfied when there is an expression in working memory with an attribute-value pair which matches it in a way that is consistent with the way in which other conditions in the same rule have already been matched

Rule-based Expert Systems

- Recognize-Act Cycle
  - *Match* premise patterns of rules against elements in working memory
  - If there is more than one rule that could fire, then *choose* on to apply; this step is called *conflict resolution*
  - *Apply* the rule, perhaps adding a new item to working memory or deleting and old one, and then iterate

Rule-based Expert Systems

- Recognize-Act Cycle
  - Computation halts if there is a cycle in which no rules become active, or if the action of a fired rule contains an explicit command to halt
  - Sets of pairs of rules and variable bindings derived from pattern matching are called *instantiations*

Rule-based Expert Systems

- Deterministic rule sets only ever have one rule eligible to fire
- Global control
  - Domain-independent
  - Hard-coded into the interpreter
- Local control
  - Domain-dependent
  - Meta-rules
  - Soft-coded by the programmer

Rule-based Expert Systems

- Set of rules eligible to fire is called the *conflict set*
  - In CLIPS, it is called the *agenda*
- Good conflict resolution strategies will exhibit *sensitivity* and *stability*

Rule-based Expert Systems

- Conflict Resolution
  - Refractoriness
    - Rules should not be allowed to fire more than once on the same data
  - Recency
    - Rules which use more recent data are preferred
  - Specificity
    - Instantiations derived from more specific rules (having more conditions) are preferred to more general rules
  - Salience
    - Assign priorities explicitly

Rule-based Expert Systems

- Conflict Resolution in CLIPS
  - Depth strategy (recency)
  - Breadth strategy (non-recency)
  - Simplicity strategy
  - Complexity strategy
  - LEX
    - refraction, salience, recency, specificity
  - MEA (Means-End Analysis)
    - for backward reasoning

Rule-based Expert Systems

- Conflict Resolution - Examples
  - Rules
    - R1 A & B --> C
    - R2 A & ~D --> E
    - R3 C & ~D --> E
    - R4 C & D --> F
    - R5 E & F --> L
    - R6 E & H --> ~G
    - R7 E & ~H --> G
    - R8 I --> J
    - R9 J --> K

Rule-based Expert Systems

- Conflict Resolution - Examples
  - Conflict Resolution
    - Rules are ordered according to their names
    - The first applicable rule is selected
    - During each session, each rule may be fired only once

Rule-based Expert Systems

- Conflict Resolution - Examples
  - What are the contents of working memory
    - After forward-chaining if the initial contents are {A, B, ~D, ~H, I}
    - After forward-chaining if the initial contents are {A, B, D, E, I}
  - Is the goal {L} supported
    - After backward-chaining if the initial contents are {A, B, ~D, E}
  - Is the goal {K, L} supported
    - After backward-chaining if the initial contents are {A, ~D, ~H, I}

Rule-based Expert Systems

Conflict Resolution - Examples (1)

- WM: A, B, ~D, ~H, I — CS: R1, R2, R8
  - R1 selected first: C added
- WM: A, B, ~D, ~H, I, C — CS: R2, R3, R8
  - R2 selected first: E added
- WM: A, B, ~D, ~H, I, C, E — CS: R3, R7, R8
  - R3 selected first: no new fact added
- WM: A, B, ~D, ~H, I, C, E — CS: R7, R8
  - R7 selected first: G added
- WM: A, B, ~D, ~H, I, C, E, G — CS: R8
  - R8 selected: J added
- WM: A, B, ~D, ~H, I, C, E, G, J — CS: R9
  - R9 selected: K added
- WM: A, B, ~D, ~H, I, C, E, G, J, K — (empty)
  - forward-chaining halts

Rule-based Expert Systems

Conflict Resolution - Examples (2)

- WM: A, B, D, E, I — CS: R1, R8
  - R1 selected first: C added
- WM: A, B, D, E, I, C — CS: R4, R8
  - R4 selected first: F added
- WM: A, B, D, E, I, C, F — CS: R5, R8
  - R5 selected first: L added
- WM: A, B, D, E, I, C, F, L — CS: R8
  - R8 selected: J added
- WM: A, B, D, E, I, C, F, L, J — CS: R9
  - R9 selected: K added
- WM: A, B, D, E, I, C, F, L, J, K — (empty)
  - forward-chaining halts

Rule-based Expert Systems

Conflict Resolution - Examples (3)

- Goal: L
- WM: A, B, ~D, E
- L not in WM, ~L not in WM CS: R5
- Sub-goals: E, F
  - Goal: E
  - E is in WM
  - Goal F
  - F not in WM, ~F not in WM CS: R4
  - Sub-goals: C, D
    - Goal: C
    - C not in WM, ~C not in WM CS: R1

Rule-based Expert Systems

- Conflict Resolution - Examples (3)
  - Sub-goals: A, B
  - A is in WM
  - B is in WM
  - R1 fires: C added to WM
  - WM: A, B, ~D, E, C
  - Goal: D
  - D is not in WM, ~D is in WM
  - D cannot be supported
  - R4 cannot be fired
  - No other rule can be found to support F
  - R5 cannot be fired

Rule-based Expert Systems

- Conflict Resolution - Examples (3)
  - No other rule can be found to support L
  - The goal L is not supported

Rule-based Expert Systems

Conflict Resolution - Examples (4)

- Goal: K, L
- WM: A, ~D, H, I
- K not in WM, ~K not in WM CS: R9
- Sub-goals: J
  - Goal: J
  - J not in WM, ~J not in WM CS: R8
  - Sub-goal: I
    - Goal: I
    - I is in WM
  - R8 fires: J added to WM
  - WM: A, ~D, H, I, J

Rule-based Expert Systems

Conflict Resolution - Examples (4)

- R9 fires: K is added to WM
- WM: A, ~D, H, I, J, K
- L not in WM, ~L not in WM — CS: R5
- Sub-goals: E, F
  - Goal: E
  - E not in WM, ~E not in WM — CS: R2, R3
  - Sub-goals: A, ~D
  - A is in WM
  - ~D is in WM
  - R2 fires: E is added to WM
  - WM: A, ~D, H, I, J, K, E

Rule-based Expert Systems

Conflict Resolution - Examples (4)

- Goal: F
- F not in WM. ~F not in WM CS: R4
- Sub-goals: C, D
  - Goal: C
  - C not in WM, ~C not in WM CS: R1
  - Sub-goals: A, B
    - Goal: A
    - A is in WM
    - Goal: B
    - B not in WM, ~B not in WM (empty)
    - No other rule can be found to support B

Rule-based Expert Systems

- Conflict Resolution - Examples (4)
  - Rule R1 cannot be fired
  - No other rule can be found to support C
  - Rule R4 cannot be fired
  - No other rule can be found to support F
  - Rule R5 cannot be fired
  - No other rule can be found to support L
  - The goal L cannot be supported
  - The goal {K, L} cannot be supported

Rule-based Expert Systems

- CLIPS
  - Essentially forward-chaining
  - LHS drives reasoning (for MYCIN it was RHS)
  - Note that we can distinguish
    - Directionality of *chaining*
    - Directionality of *reasoning*
  - Even with forward chaining we can force top-down reasoning by manipulating goal tokens

Rule-based Expert Systems

- Meta-Rules
  - Direct reasoning rather than performing reasoning
  - Admit uncertainty
  - Not available in CLIPS
    - But influence over preferential rule selection can be exerted via salience
    - Beware of excessive reliance on salience

Rule-based Expert Systems

- Production rule system are non-monotonic (and therefore non-classical) since actions can delete facts from working memory
- 90% of the time in inferencing is spent in matching patterns (and instantiation)
- The RETE algorithm is applied here

Rule-based Expert Systems

- Pattern matching
  - RETE
    - The RETE pattern-matching algorithm used in CLIPS and many other production rule systems is intended to improve the speed of forward-chaining by limiting the effort required to re-compute the conflict set after a rule is fired.
    - Its drawback is that it has high memory space requirements

Rule-based Expert Systems

- Pattern matching
  - RETE
    - RETE takes advantage of two empirical observations
      - temporal redundancy
        - The firing of a rule usually changes only a few facts, and only a few rules are affected by each of these changes
      - structural similarity
        - The same pattern often appears in the LHS of more than one rule

Rule-based Expert Systems

- Pattern matching
  - RETE
    - RETE uses a rooted directed acyclic graph where nodes other than the root represent patterns and paths from the root to the leaves represent the LHS of rules. At each node is stored information about the facts satisfied by the patterns of the nodes in the paths from the root up to and including this node. This information is a relation representing the possible values of the variables occurring in the patterns in the path

Rule-based Expert Systems

- Pattern matching
  - RETE
    - RETE keeps up-to-date the information associated with the nodes in the graph. When a fact is added or removed, a token representing that fact and operation is entered at the root and propagated to the leaves
    - The RETE graph consists of a *root node*, one-input *pattern nodes*, and two-input *join nodes*
    - For each rule and pattern we create a one-input *alpha node*
    - For each rule we construct two-input *beta nodes* from its alpha nodes

Rule-based Expert Systems

- RETE
  - For further details, see the document at
    http://yoda.cis.temple.edu:8080/UGAIWWW/lectures/rete.html

Rule-based Expert Systems

- As a consequence of the design of RETE, it is important to tune the rules so that large numbers of partial matches are not generated
- Patterns may be ordered so as to improve efficiency:
  - Most specific patterns go first
  - Patterns matching volatile facts go last
  - Patterns matching the fewest facts go first
- *but these guidelines may conflict!*

CLIPS

- C Language Integrated Production System
- Developed by NASA in the mid-1980s
- Production rule language
- Procedural language
- Fact base
- Rule base
- Recognize-Act Cycle

CLIPS

- Facts
  - (assert (today is Sunday))
  - (facts)
  - (retract 1)
  - (clear)
  - (load “myfile”)
  - (reset)

CLIPS

- Rules
  - (defrule chores
    “Things to do on Sunday”
    (salience 10)
    (today is Sunday) (weather is warm)
    =>
    (assert (wash car)) (assert (chop wood)))
  - Variables
    - ?day
    - $?

CLIPS

- Commands
  - (run)
  - (refresh)
  - (watch rules)
  - (dribble-on “dribble.clp”)
  - (agenda)
  - (list-defrules)
  - (list-deffacts) etc.

CLIPS

- Templates
  - Resemble simple records
  - (deftemplate student “Student Record”
    (slot name (type STRING))
    (slot age (type NUMBER) (default 18)))
- Functions
  - (deffunction hypotenuse (?a ?b)
    (sqrt (+ (* ?a ?a) (* ?b ?b))))

CLIPS

- COOL
  - CLIPS Object-Oriented Language
  - Objects help manage complexity by keeping rules clean and simple, implementing mechanisms for data update as class message handlers
  - ‘Pistol’ example program

CLIPS

- Advanced Features
  - Contexts
  - Backtracking
  - Incremental system development
  - Handling “reported speech”
  - Computational mechanisms
    - Forward reasoning with conflict resolution
    - Goal-directed reasoning using task tokens
    - Multiple contexts based on different assumption
  - ‘Truth Games’ example program

CLIPS

- Programming Project
  - Car Troubleshooting
    - Interactive system
    - Forward and backward chaining
    - Due October 21