CS4386 AI Game Programming

Lecture 05
Decision Tree Learning

Semester B, 2020-2021
Department of Computer Science
City University of Hong Kong

Decision Tree Learning

- Decision trees can be efficiently learned
  - Constructed dynamically from sets of observations and actions provided through strong supervision
  - The constructed trees can then be used in the normal way to make decisions during gameplay
- There are a range of different decision tree learning algorithms used for classification, prediction, and statistical analysis
  - Those used in game AI are typically based on Quinlan's ID3 algorithm

ID3

- Depending on whom you believe, ID3 stands for “Inductive Decision tree algorithm 3” or “Iterative Dichotomizer 3”
- It is a simple to implement, relatively efficient decision tree learning algorithm
- It has been largely replaced in industrial AI use by optimized versions of the algorithm such as C4, C4.5, and C5

ID3 Algorithm - Outline

- The basic ID3 algorithm uses the set of observation–action examples
  - Observations in ID3 are usually called “attributes”
- The algorithm starts with a single leaf node in a decision tree and assigns a set of examples to the leaf node
- It then splits its current node (initially the single start node) so that it divides the examples into two groups
  - The division is chosen based on an attribute, and the division chosen is the one that is likely to produce the most efficient tree
  - When the division is made, each of the two new nodes is given the subset of examples that applies to them, and the algorithm repeats for each of them

ID3 Algorithm - Details

- This algorithm is recursive: starting from a single node it replaces the nodes with decisions until the whole decision tree has been created
- At each branch creation it divides up the set of examples among its children, until all the examples agree on the same action
  - At that point the action can be carried out; there is no need for further branches
- The split process looks at each attribute in turn (i.e., each possible way to make a decision) and calculates the information gain for each possible division
  - The division with the highest information gain is chosen as the decision for this node

Entropy and Information Gain

- In order to work out which attribute should be considered at each step, ID3 uses the entropy of the actions in the set
- Entropy is a measure of the information in a set of examples
  - In ID3, it measures the degree to which the actions in an example set agree with each other
  - If all the examples have the same action, the entropy will be 0
  - If the actions are distributed evenly, then the entropy will be 1
- Information gain is simply the reduction in overall entropy

Example - Actions and Attributes

- Actions
  1. Attack
  2. Defend
- Attributes (assumed binary)
  1. Health: healthy / hurt
  2. Cover: in cover / exposed
  3. Ammo: with ammo / empty gun
- Examples

Healthy	In Cover	With Ammo	Attack
Hurt	In Cover	With Ammo	Attack
Healthy	In Cover	Empty	Defend
Hurt	In Cover	Empty	Defend
Hurt	Exposed	With Ammo	Defend

Example - Entropy

Healthy	In Cover	With Ammo	Attack
Hurt	In Cover	With Ammo	Attack
Healthy	In Cover	Empty	Defend
Hurt	In Cover	Empty	Defend
Hurt	Exposed	With Ammo	Defend

- For two possible outcomes, attack and defend, the entropy of a set of actions is given by:

$$E = -p_A \log_2(p_A) - p_D \log_2(p_D)$$

where p_A is the proportion of attack actions in the example set, and
p_D is the proportion of defend actions

- For the example, the entropy for the whole set is 0.971

$p_A = 2/5 = 0.4$ $p_D = 3/5 = 0.6$

$E = -0.4\log_2(0.4) - 0.6\log_2(0.6)$

$= -0.4(-1.322) - 0.6(-0.737)$

$= 0.971$

Note that
$\log_2(x) = \log(x)/\log(2)$

Example – Information Gain (1)

- After the entropy of a set E_S is determined, we would like to determine which attribute should be used to split the node into children nodes
  - For a binary attribute, denote the 2 possible outcomes as A and B
  - For each attribute we calculate the information gain G by

    $G = E_S - p_\top E_\top - p_\bot E_\bot$

    where $p_\top / p_\bot$ is the proportion of examples with true/false attributes

    $E_\top / E_\bot$ is the entropy of the examples with true/false attributes
  - The attribute that maximizes the information gain is selected as a decision node for splitting

Example – Information Gain (2)

- We can split the parent node (whole set) into children nodes according to health, cover or ammo. In all these cases $E_s = 0.971$ which is the entropy of the whole set
  1. If we split the parent node according to health

$G = E_S - p_\top E_\top - p_\perp E_\perp$

Here $_\top$ = healthy, $_\perp$ = hurt

$p_\top = 2/5 \quad p_\perp = 3/5$

$E_\top = -p_A \log_2(p_A) - p_D \log_2(p_D)$

$= -(1/2)\log_2(1/2) - (1/2)\log_2(1/2)$

$= 1$

$E_\perp = -p_A \log_2(p_A) - p_D \log_2(p_D)$

$= -(1/3)\log_2(1/3) - (2/3)\log_2(2/3)$

$= 0.918$

$G_{health} = 0.971 - (2/5)(1) - (3/5)(0.918) = 0.020$

Healthy	In Cover	With Ammo	Attack
Hurt	In Cover	With Ammo	Attack
Healthy	In Cover	Empty	Defend
Hurt	In Cover	Empty	Defend
Hurt	Exposed	With Ammo	Defend

Example – Information Gain (3)

2. If we split the parent node according to cover

$G = E_S - p_\top E_\top - p_\perp E_\perp$

Here $_\top$ = in cover, $_\perp$ = exposed

$p_\top = 4/5 \quad p_\perp = 1/5$

$E_\top = -p_A \log_2(p_A) - p_D \log_2(p_D)$

$= -(2/4)\log_2(2/4) - (2/4)\log_2(2/4)$

$= 1$

$E_\perp = -p_A \log_2(p_A) - p_D \log_2(p_D)$

$= -(1)\log_2(1)$

$= 0$

$G_{cover} = 0.971 - (4/5)(1) - (1/5)(0) = 0.171$

Healthy	In Cover	With Ammo	Attack
Hurt	In Cover	With Ammo	Attack
Healthy	In Cover	Empty	Defend
Hurt	In Cover	Empty	Defend
Hurt	Exposed	With Ammo	Defend

In computing the entropy, when a probability is 0, the corresponding term is also 0, meaning that you do not need to calculate those terms with probability of 0

Example – Information Gain (4)

3. If we split the parent node according to ammo

$G = E_S - p_\top E_\top - p_\perp E_\perp$

Here $_\top$ = with ammo, $_\perp$ = empty

$p_\top = 3/5 \quad p_\perp = 2/5$

$E_\top = -p_A \log_2(p_A) - p_D \log_2(p_D)$

$= -(2/3)\log_2(2/3) - (1/3)\log_2(1/3)$

$= 0.918$

$E_\perp = -p_A \log_2(p_A) - p_D \log_2(p_D)$

$= -(1)\log_2(1)$

$= 0$

$G_{ammo} = 0.971 - (3/5)(0.918) - (2/5)(0) = 0.420$

Healthy	In Cover	With Ammo	Attack
Hurt	In Cover	With Ammo	Attack
Healthy	In Cover	Empty	Defend
Hurt	In Cover	Empty	Defend
Hurt	Exposed	With Ammo	Defend

In computing the entropy, when a probability is 0, the corresponding term is also 0, meaning that you do not need to calculate those terms with probability of 0

Example – Information Gain (4)

- The information gain computed by splitting the whole set according to the 3 attributes are shown below:

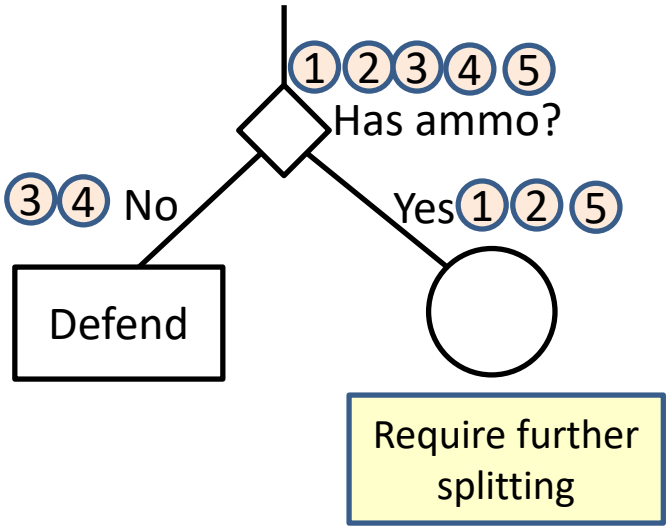
$G_{health} = 0.020$

$G_{cover} = 0.171$

$G_{ammo} = 0.420$

- Since G_{ammo} has the largest value, it is considered as the best indicator of what action we need to take
- By the principle of learning the most general stuff first, we use ammo as our first branch in the decision tree

1	Healthy	In Cover	With Ammo	Attack
2	Hurt	In Cover	With Ammo	Attack
3	Healthy	In Cover	Empty	Defend
4	Hurt	In Cover	Empty	Defend
5	Hurt	Exposed	With Ammo	Defend

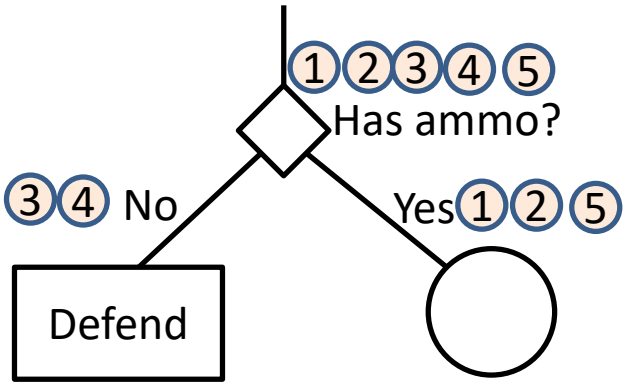


Example – Further Splitting (1)

- The algorithm can be continued for the node that requires further splitting
  - Again, we can split this node into children nodes according to health, cover or ammo
  - However, since the previous level was split using the attribute ammo, there is no need to consider this attribute for splitting (you can still do it if you want, but you will obtain G=0 in this case)

1. If we split the parent node according to health

$$G = E_S - p_\top E_\top - p_\perp E_\perp$$
$$p_\top = 1/3 \quad p_\perp = 2/3$$
$$E_\top = -(1)\log_2(1) = 0$$
$$E_\perp = -(1/2)\log_2(1/2) - (1/2)\log_2(1/2) = 1$$
$$G_{health} = 0.918 - (1/3)(0) - (2/3)(1) = 0.251$$



1	Healthy	In Cover	With Ammo	Attack
2	Hurt	In Cover	With Ammo	Attack
3	Healthy	In Cover	Empty	Defend
4	Hurt	In Cover	Empty	Defend
5	Hurt	Exposed	With Ammo	Defend

Example – Further Splitting (2)

2. If we split the parent node according to cover

$G = E_S - p_\top E_\top - p_\bot E_\bot$

$p_\top = 2/3 \quad p_\bot = 1/3$

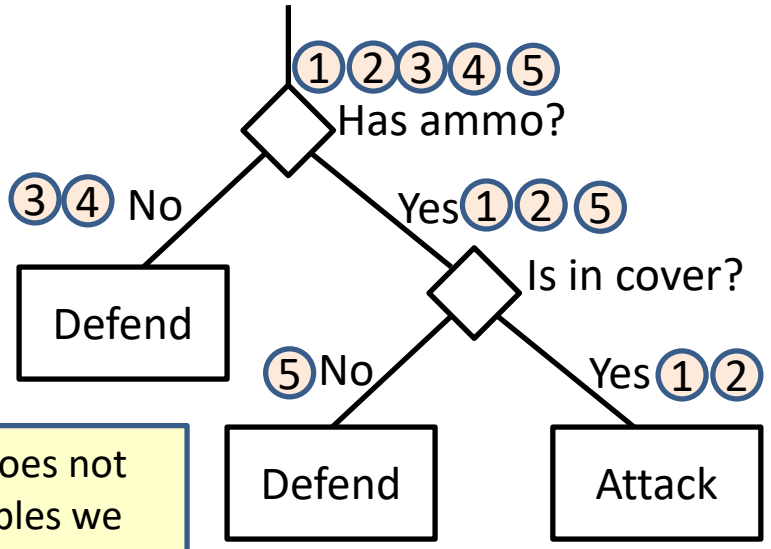
$E_\top = -(1)\log_2(1) = 0$

$E_\bot = -(1)\log_2(1) = 0$

$G_{cover} = 0.918 - (2/3)(0) - (1/3)(0) = 0.918$

- Since $G_{cover} > G_{health}$, the attribute cover should be used for splitting the node

1	Healthy	In Cover	With Ammo	Attack
2	Hurt	In Cover	With Ammo	Attack
3	Healthy	In Cover	Empty	Defend
4	Hurt	In Cover	Empty	Defend
5	Hurt	Exposed	With Ammo	Defend



Notice that the health of the character does not feature at all in this tree; from the examples we were given, it simply is not relevant to the decision

Extension – More than 2 actions

- In the previous example, the entropy is computed by

$$E = -p_{\mathrm{A}} \log_2(p_{\mathrm{A}}) - p_{\mathrm{D}} \log_2(p_{\mathrm{D}})$$

where p_{A} is the proportion of attack actions in the example set,
and p_{D} is the proportion of defend actions

- When there are more than 2 actions, the entropy is generalized to

$$E = -\sum_{i=1}^{n} p_i \log_2 p_i$$

where p_i is the proportion of each action in the example set,
and n is the number of actions

Extension – Non-Binary Attributes

- In the previous example, the information gain is computed by

$$G = E_S - p_\top E_\top - p_\bot E_\bot$$

where $p_\top / p_\bot$ is the proportion of examples with true/false attributes

$E_\top / E_\bot$ is the entropy of the examples with true/false attributes

- When there are more than 2 categories for an attribute, there can be more than 2 children nodes for a decision, and information gain is generalized to

$$G = E_S - \sum_{i=1}^{n} \left(\frac{|S_i|}{|S|} \right) E_{S_i}$$

where S_i is the set of examples for each of the n values of the attribute

ID3 with Continuous Attributes

- ID3-based algorithms cannot operate directly with continuous attributes, and they are impractical when there are many possible values for each attribute
- In either case the attribute values must be divided into a small number of discrete categories (usually two)
- This division can be performed automatically as an independent process, and with the categories in place the rest of the decision tree learning algorithm remains identical

Single Splits

- Continuous attributes can be used as the basis of binary decisions by selecting a threshold level
  - Values below the level are in one category, and values above the level are in another category
  - We can dynamically calculate the best threshold value to use with a process similar to that used to determine which attribute to use in a branch
- We sort the examples using the attribute we are interested in
  - We place the first element from the ordered list into category A and the remaining elements into category B
  - We now have a division, so we can perform the split and calculate information gained, as before
  - We repeat the process by moving the lowest valued example from category B into category A and calculating the information gained in the same way
  - Whichever division gave the greatest information gained is used as the division
  - The numeric threshold value is calculated by finding the average of the highest value in category A and the lowest value in category B

Example

- Consider a set of examples with a single attribute health that can take any value between 0 and 200
- The examples are first ordered and placed into the two categories, and the information gained is calculated in each case

50	Defend
25	Defend
39	Attack
17	Defend

Category	Attribute Value	Action	Information Gain
A	17	Defend	
B	25	Defend	
	39	Attack	
	50	Defend	0.12

Category	Attribute Value	Action	Information Gain
A	17	Defend	
	25	Defend	
B	39	Attack	
	50	Defend	0.31

Category	Attribute Value	Action	Information Gain
A	17	Defend	
	25	Defend	
	39	Attack	
B	50	Defend	0.12

The most information is gained if we put the threshold between 25 and 39
The midpoint between these values is 32, so 32 becomes our threshold value

Problem with Offline Learning

- The approach in previous slides learns a decision tree in a single process
  - A complete set of examples is provided, and the algorithm returns a complete decision tree ready for use
  - This is fine for offline learning, where a large number of observation–action examples can be provided in one go
- When used online, new examples will be generated while the game is running, and the decision tree should change over time to accommodate them
  - With a small number of examples, only broad brush sweeps can be seen, and the tree will typically need to be quite flat
  - With hundreds or thousands of examples, subtle interactions between attributes and actions can be detected by the algorithm, and the tree is likely to be more complex
  - The simplest way to support this scaling is to re-run the algorithm each time a new example is provided
    - This guarantees that the decision tree will be the best possible at each moment
    - Unfortunately, with large databases of examples, this would be very time consuming

Simple Update Procedure (1)

- Incremental algorithms update the decision tree based on the new information, without requiring the whole tree to be rebuilt
- The simplest approach would be to take the new example and use its observations to walk through the decision tree
- When we reach a terminal node of the tree, we compare the action there with the action in our example
- If they match, then no update is required, and the new example can simply be added to the example set at that node. If the actions do not match, then the node is converted into a decision node in the normal way
- Let's use the example in the following slides to try this approach

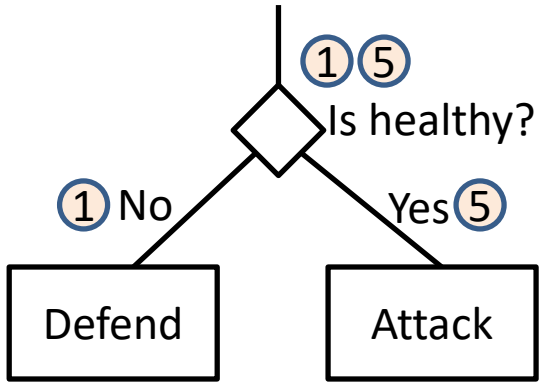
Simple Update Procedure (2)

- Let's apply the approach on previous slide to the case before, but with the examples come at different stages
  - Let's say at the beginning we only receive 2 examples:

1	Healthy	In Cover	With Ammo	Attack

5	Hurt	Exposed	With Ammo	Defend

  - An initial decision tree can be built



Simple Update Procedure (3)

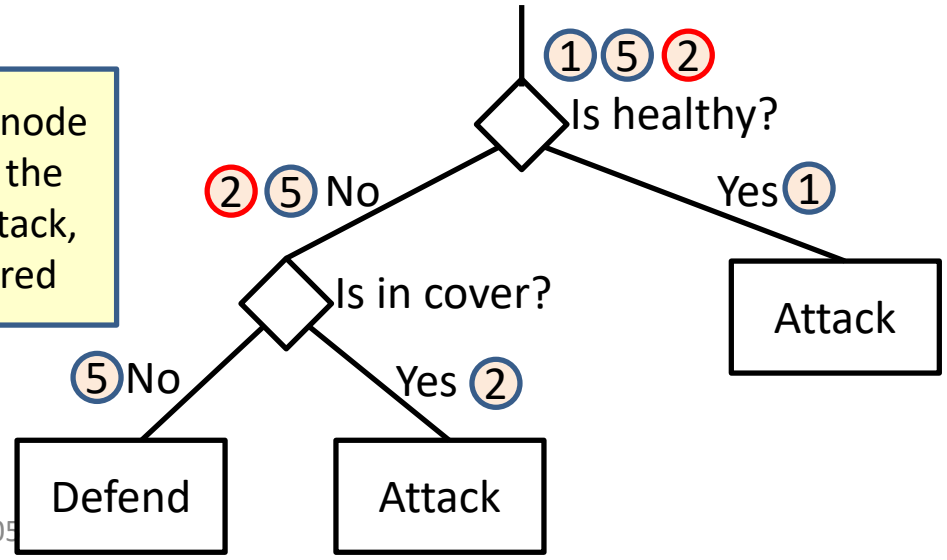
- We have just received 1 new example:

1	Healthy	In Cover	With Ammo	Attack
2	Hurt	In Cover	With Ammo	Attack

5	Hurt	Exposed	With Ammo	Defend

- Walk through the decision tree with this new example

The action in the terminal node is defend but the action in the new example should be attack, so further splitting is required

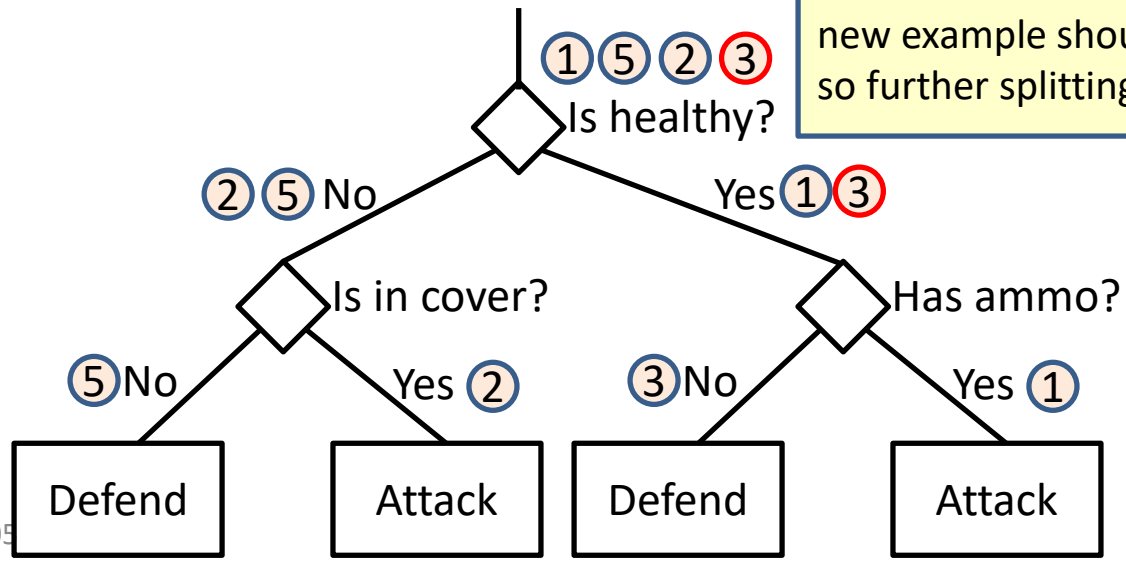


Simple Update Procedure (4)

– We have just received another new example:

1	Healthy	In Cover	With Ammo	Attack
2	Hurt	In Cover	With Ammo	Attack
3	Healthy	In Cover	Empty	Defend
5	Hurt	Exposed	With Ammo	Defend

– Walk through the decision tree with this new example



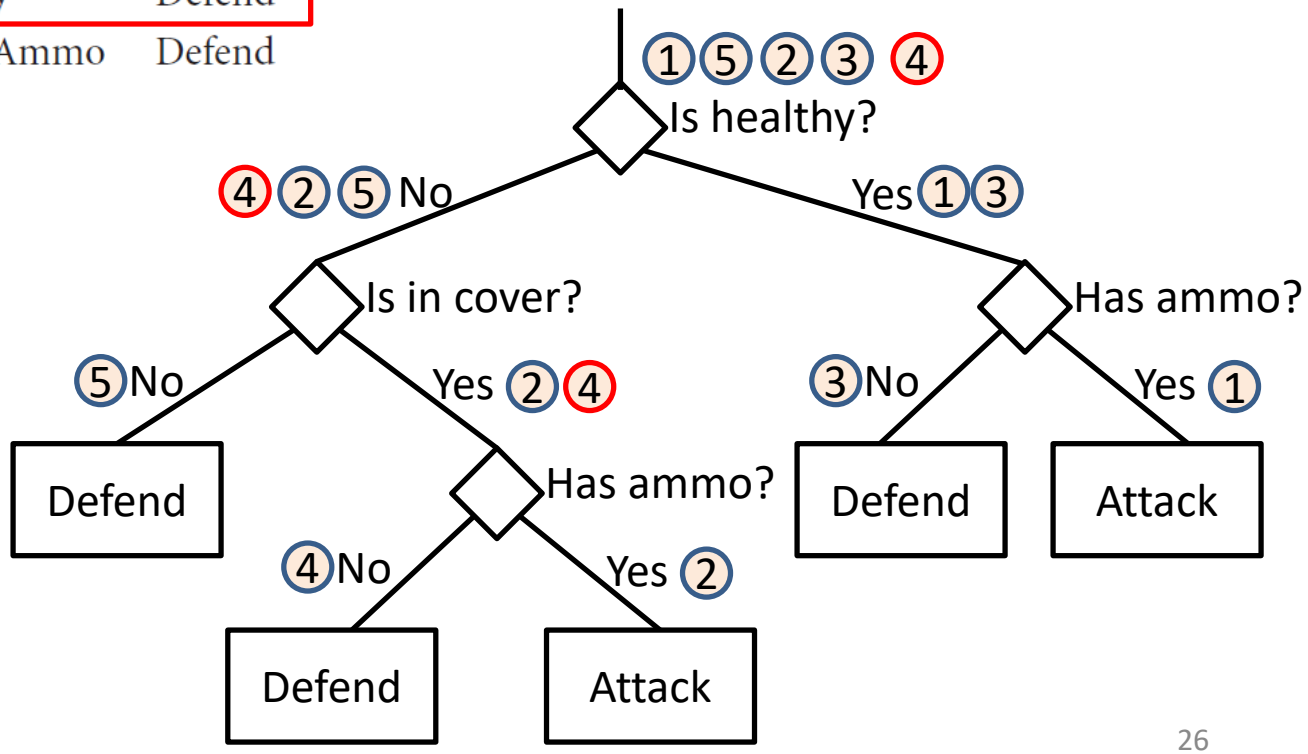
The action in the terminal node is attack but the action in the new example should be defend, so further splitting is required

Simple Update Procedure (5)

- We have just received the last new example:

1	Healthy	In Cover	With Ammo	Attack
2	Hurt	In Cover	With Ammo	Attack
3	Healthy	In Cover	Empty	Defend
4	Hurt	In Cover	Empty	Defend
5	Hurt	Exposed	With Ammo	Defend

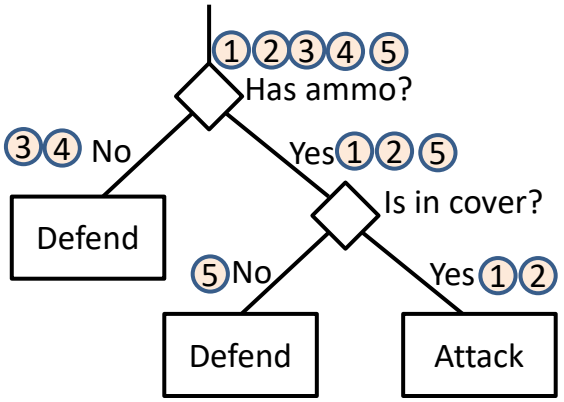
- Walk through the decision tree with this new example

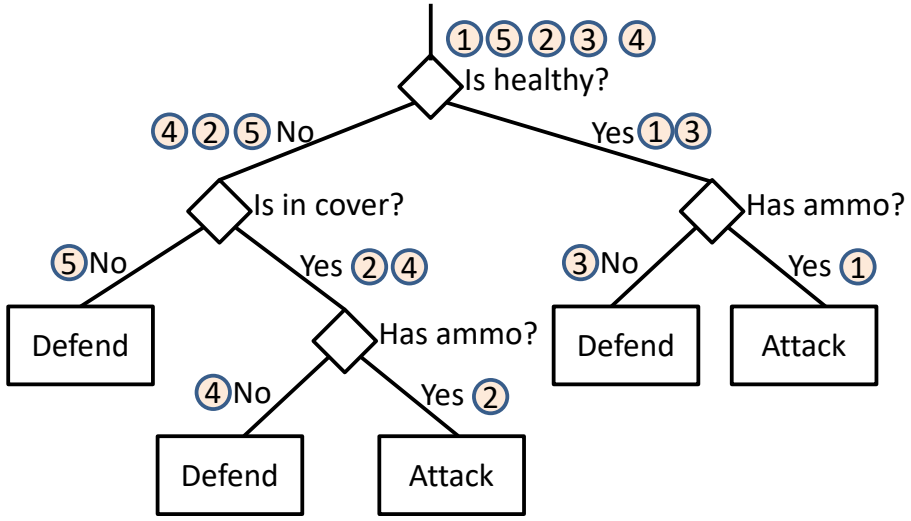


The action in the terminal node is attack but the action in the new example should be defend, so further splitting is required

Simple Update Procedure (6)

- From the previous example, you can always update the decision tree with new examples
- However, it always adds further examples to the end of a tree and can generate huge trees with many sequential branches





- Ideally we would like to create trees that are as flat as possible, where the action to carry out can be determined as quickly as possible
- So we will explore another incremental tree learning algorithm instead

Incremental Algorithm - ID4 (1)

- As its name suggests, it is related to the basic ID3 algorithm
- We start with a decision tree, as created by the basic ID3 algorithm. Each node in the decision tree also keeps a record of all the examples that reach that node
- To support incremental learning, we can ask any node in the tree to update itself given a new example

Incremental Algorithm - ID4 (2)

- When asked to update itself, one of three things can happen
  1. If the node is a terminal node (i.e., it represents an action), and if the added example also shares the same action, then the example is added to the list of examples for that node
  2. If the node is a terminal node, but the example's action does not match, then we make the node into a decision and use the ID3 algorithm to determine the best split to make
  3. If the node is not a terminal node, then it is already a decision. We determine the best attribute to make the decision on, adding the new example to the current list. The best attribute is determined using the information gain metric, as we saw in ID3
     a) If the attribute returned is the same as the current attribute for the decision (and it will be most times), then we determine which of the children nodes the new example gets mapped to, and we update that child node with the new example
     b) If the attribute returned is different, then it means the new example makes a different decision optimal. If we change the decision at this point, then all of the tree further down the current branch will be invalid. So we delete the whole tree from the current decision down and perform the basic ID3 algorithm using the current decision's examples plus the new one

Incremental Algorithm - ID4 (3)

- Note that when we reconsider which attribute to make a decision on, several attributes may provide the same information gain
  - If one of them is the attribute we are currently using in the decision, then we favor that one to avoid unnecessary rebuilding of the decision tree
- In summary, at each node in the tree, ID4 checks if the decision still provides the best information gain in light of the new example
  - If it does, then the new example is passed down to the appropriate child node
  - If it does not, then the whole tree is recalculated from that point on
  - This ensures that the tree remains as flat as possible

ID4 Example – Initial Decision Tree (1)

- Let's start with another case with 5 examples
- First we construct the initial decision tree using ID3

$E_S = -\sum_{i=1}^{n} p_i \log_2 p_i = -\left(\frac{1}{5}\right)\log_2\left(\frac{1}{5}\right) - \left(\frac{2}{5}\right)\log_2\left(\frac{2}{5}\right) - \left(\frac{2}{5}\right)\log_2\left(\frac{2}{5}\right) = 1.522$

1. Split by health

$p_\top = 3/5 \quad p_\perp = 2/5$

$E_\top = -(1/3)\log_2(1/3) - (1/3)\log_2(1/3) - (1/3)\log_2(1/3) = 1.585$

$E_\perp = -(1/2)\log_2(1/2) - (1/2)\log_2(1/2) = 1$

$G_{health} = 1.522 - (3/5)(1.585) - (2/5)(1) = 0.171$

2. Split by cover

$p_\top = 4/5 \quad p_\perp = 1/5$

$E_\top = -(2/4)\log_2(2/4) - (2/4)\log_2(2/4) = 1$

$E_\perp = -(1)\log_2(1) = 0$

$G_{cover} = 1.522 - (4/5)(1) - (1/5)(0) = 0.722$

3. Split by ammo

$p_\top = 2/5 \quad p_\perp = 3/5$

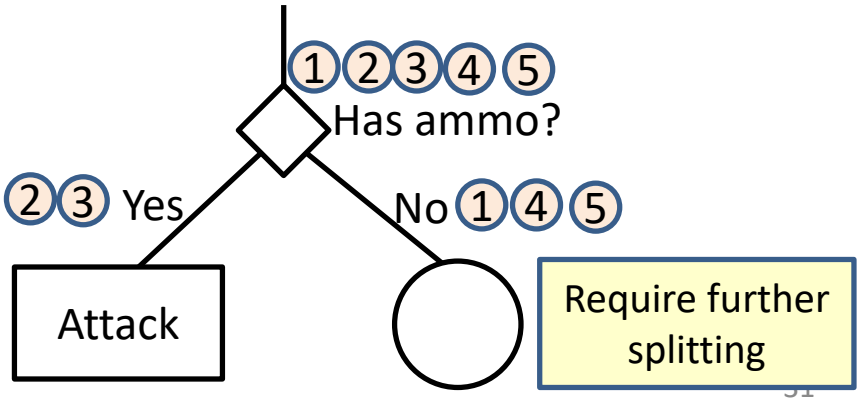
$E_\top = -(1)\log_2(1) = 0$

$E_\perp = -(1/3)\log_2(1/3) - (2/3)\log_2(2/3) = 0.918$

$G_{ammo} = 1.522 - (2/5)(0) - (3/5)(0.918) = 0.971$

- Since G_{ammo} has the largest value, we use ammo as our first branch in the decision tree

1	Healthy	Exposed	Empty	Run
2	Healthy	In Cover	With Ammo	Attack
3	Hurt	In Cover	With Ammo	Attack
4	Healthy	In Cover	Empty	Defend
5	Hurt	In Cover	Empty	Defend

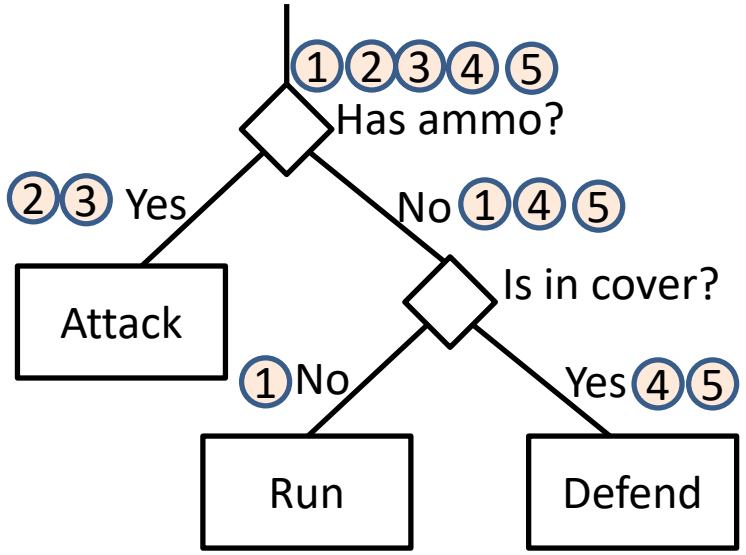


ID4 Example – Initial Decision Tree (2)

- Continue splitting the node
  $G_{health} = 0.251$
  $G_{cover} = 0.918$
  $G_{ammo} = 0$
- Since G_{cover} has the largest value, we use cover as our next branch in the decision tree

1	Healthy	Exposed	Empty	Run
2	Healthy	In Cover	With Ammo	Attack
3	Hurt	In Cover	With Ammo	Attack
4	Healthy	In Cover	Empty	Defend
5	Hurt	In Cover	Empty	Defend

In simple situation like this, the splitting can be determined by inspection without the need for any computation

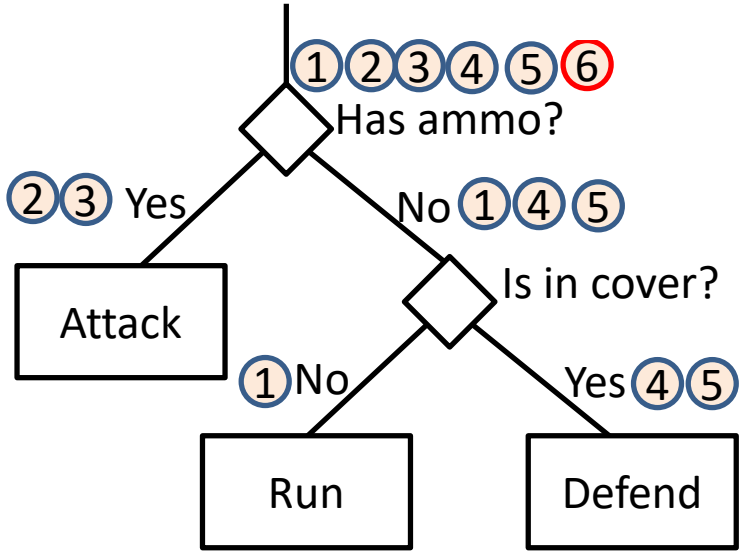


ID4 Example – Incremental Update (1)

1	Healthy	Exposed	Empty	Run
2	Healthy	In Cover	With Ammo	Attack
3	Hurt	In Cover	With Ammo	Attack
4	Healthy	In Cover	Empty	Defend
5	Hurt	In Cover	Empty	Defend

- We will add 2 more examples, one at a time

6	Hurt	Exposed	With Ammo	Defend



$E_S = -(1/6)\log_2(1/6) - (2/6)\log_2(2/6) - (3/6)\log_2(3/6) = 1.459$

1. Split by health

$p_\top = 3/6 \quad p_\bot = 3/6$

$E_\top = -(1/3)\log_2(1/3) - (1/3)\log_2(1/3) - (1/3)\log_2(1/3) = 1.585$

$E_\bot = -(1/3)\log_2(1/3) - (2/3)\log_2(2/3) = 0.918$

$G_{health} = 1.459 - (3/6)(1.585) - (3/6)(0.918) = 0.208$

2. Split by cover

$p_\top = 4/6 \quad p_\bot = 2/6$

$E_\top = -(2/4)\log_2(2/4) - (2/4)\log_2(2/4) = 1$

$E_\bot = -(1/2)\log_2(1/2) - (1/2)\log_2(1/2) = 1$

$G_{cover} = 1.459 - (4/6)(1) - (2/6)(1) = 0.459$

3. Split by ammo

$p_\top = 3/6 \quad p_\bot = 3/6$

$E_\top = -(2/3)\log_2(2/3) - (1/3)\log_2(1/3) = 0.918$

$E_\bot = -(1/3)\log_2(1/3) - (2/3)\log_2(2/3) = 0.918$

$G_{ammo} = 1.459 - (3/6)(0.918) - (3/6)(0.918) = 0.541$

ID4 Example – Incremental Update (2)

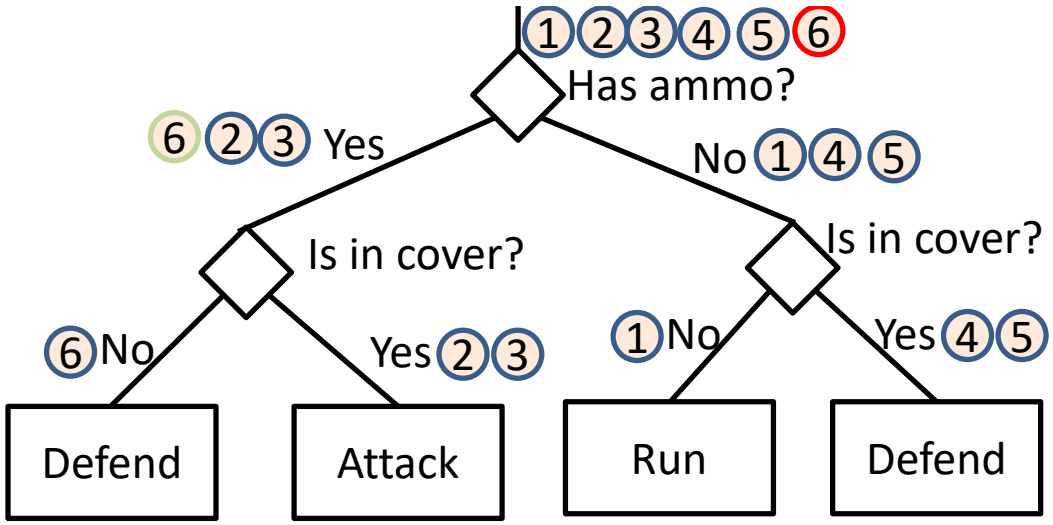
1	Healthy	Exposed	Empty	Run
2	Healthy	In Cover	With Ammo	Attack
3	Hurt	In Cover	With Ammo	Attack
4	Healthy	In Cover	Empty	Defend
5	Hurt	In Cover	Empty	Defend

- We will add 2 more examples, one at a time

6	Hurt	Exposed	With Ammo	Defend

$G_{health} = 0.208$
$G_{cover} = 0.459$
$G_{ammo} = 0.541$

The highest gain is achieved for ammo, which is the same as the original decision
Now this example is sent to the child node



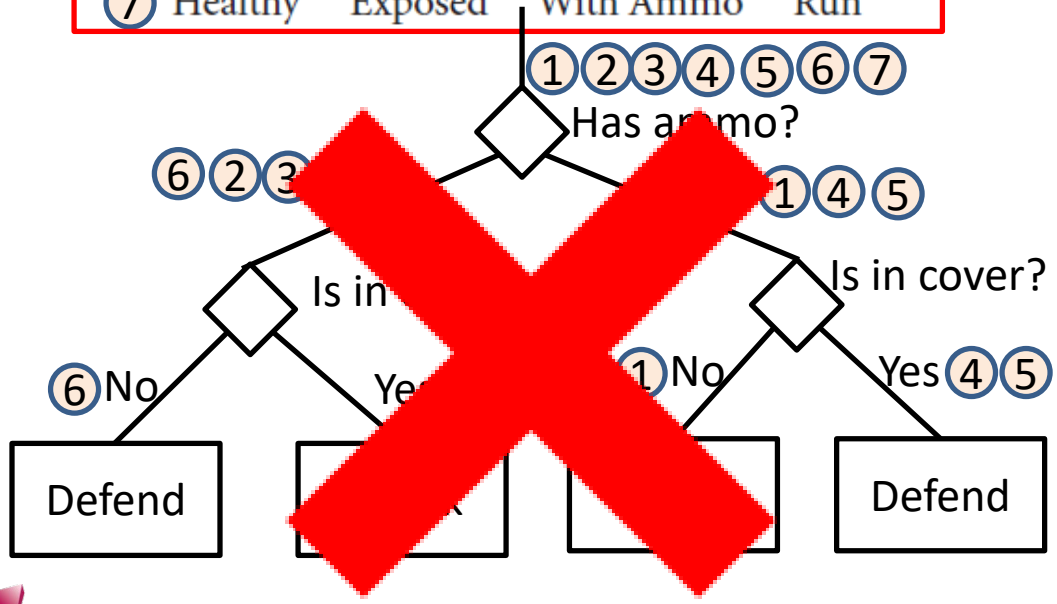
The action in the child node is attack which is different from the action defend of the new example. A new subtree is needed to replace the child node
After applying ID3 (or by inspection for this simple situation), the attribute cover will be used as the decision

ID4 Example – Incremental Update (3)

1	Healthy	Exposed	Empty	Run
2	Healthy	In Cover	With Ammo	Attack
3	Hurt	In Cover	With Ammo	Attack
4	Healthy	In Cover	Empty	Defend
5	Hurt	In Cover	Empty	Defend

- We will add the last example

6	Hurt	Exposed	With Ammo	Defend
7	Healthy	Exposed	With Ammo	Run



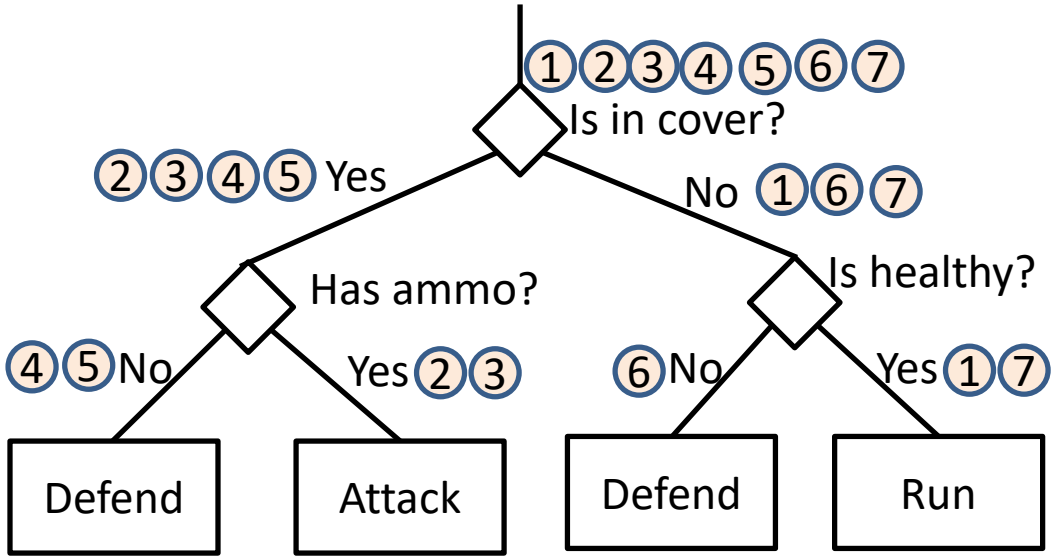
$G_{health} = 0.020$
$G_{cover} = 0.592$
$G_{ammo} = 0.306$

The highest gain is achieved for cover, which is different from the attribute ammo used for decision

Since this is the root node, a completely new tree is needed to replace the existing one, which can be obtained from the ID3 algorithm

ID4 Example – Incremental Update (3)

1	Healthy	Exposed	Empty	Run
2	Healthy	In Cover	With Ammo	Attack
3	Hurt	In Cover	With Ammo	Attack
4	Healthy	In Cover	Empty	Defend
5	Hurt	In Cover	Empty	Defend
6	Hurt	Exposed	With Ammo	Defend
7	Healthy	Exposed	With Ammo	Run



Performance of ID4

- ID4 can be very effective in creating optimal decision trees
  - As the first few examples come in, the tree will be largely rebuilt at each step
  - As the database of examples grows, the changes to the tree often decrease in size, keeping the execution speed high
- The tree generated by ID4 will always be the same as that generated by ID3 for the same input examples
- At worst, ID4 will have to do the same work as ID3 to update the tree
- At best, it is as efficient as the simple update procedure
- In practice, for sensible sets of examples, ID4 is considerably faster than repeatedly calling ID3 each time and will be faster in the long run than the simple update procedure (because it is producing flatter trees)

Reference

- Artificial Intelligence for Games
  - Chapter 7.6