CS367: AI
Tutorial 9

Machine Learning Part 2
Decision Trees & Neural Networks

Feedback to: goo.gl/auRHe

Decision Trees

- Like concept learning, but learning a discrete-valued output instead of just a boolean one
- DTs are more robust to errors in classifications and attributes, and missing information
- Main algorithm used was ID3
  - Builds a DT by placing the highest information-gain attributes nearest to the root
  - As soon as all examples on a branch are of a single category, it becomes a leaf
  - If there are no examples on a branch, it becomes a leaf of the most common category

Aside: Information-Gain

- Information-gain is a measure of entropy (randomness) of an attribute with respect to an output value
- Given a set of examples S and an attribute A
  - A can take values $v_1 \ldots v_m$
  - Let S_v = {examples which take value v for attribute A}
- Calculate Gain(S,A)
  - Estimates the reduction in entropy we get if we know the value of attribute A for the examples in S

$$Entropy(S) \equiv \sum_{i=1}^{c} -p_i \log_2 p_i$$

Where p_i is the proportion of S in class i

$$Gain(S,A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v)$$

Decision Trees Example

Weekend	Weather	Parents	Money	Decision (Category)
W1	Sunny	Yes	Rich	Cinema
W2	Sunny	No	Rich	Tennis
W3	Windy	Yes	Rich	Cinema
W4	Rainy	Yes	Poor	Cinema
W5	Rainy	No	Rich	Stay in
W6	Rainy	Yes	Poor	Cinema
W7	Windy	No	Poor	Cinema
W8	Windy	No	Rich	Shopping
W9	Windy	Yes	Rich	Cinema
W10	Sunny	No	Rich	Tennis

Which attribute has the highest information gain (by inspection)?
ie. Which best classifies the cases (by output class)

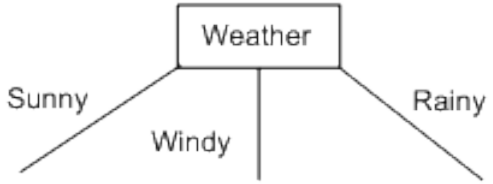
Taken from 2010 slides

Information Gain

- S = {W1,W2,…,W10}
- Firstly, we need to calculate:
  - Entropy(S) = … = 1.571
- Next, we need to calculate information gain for all the attributes we currently have available (which is all of them at the moment)
  - Gain(S, weather) = … = 0.7
  - Gain(S, parents) = … = 0.61
  - Gain(S, money) = … = 0.2816
- Hence, the weather is the first attribute to split on because this gives us the biggest information gain

Top of the Tree

- So, this is the top of our tree:



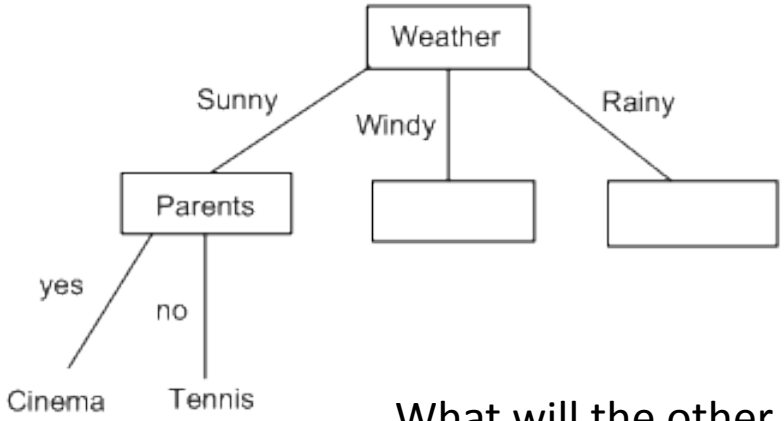
- Now, we look at each branch in turn
  - In particular, we look at the examples with the attribute prescribed by the branch
- S_{sunny} = {W1,W2,W10}
  - Categorisations are cinema, tennis and tennis for W1,W2 and W10
  - What does the algorithm say?
    - Set is neither empty, nor a single category
    - So we have to replace S by S_{sunny} and start again

Working with S_{sunny}

- Need to choose a new attribute to split on
  - Cannot be weather, of course – we've already had that
- So, calculate information gain again:
  - Gain(S_{sunny}, parents) = … = 0.918
  - Gain(S_{sunny}, money) = … = 0
- Hence we choose to split on parents

Getting to the leaf nodes

- If it's sunny and the parents have turned up
  - Then, looking at the table in previous slide
    - There's only one answer: go to cinema
- If it's sunny and the parents haven't turned up
  - Then, again, there's only one answer: play tennis
- Hence our decision tree looks like this:



What will the other branches look like?

Decision Trees: Question

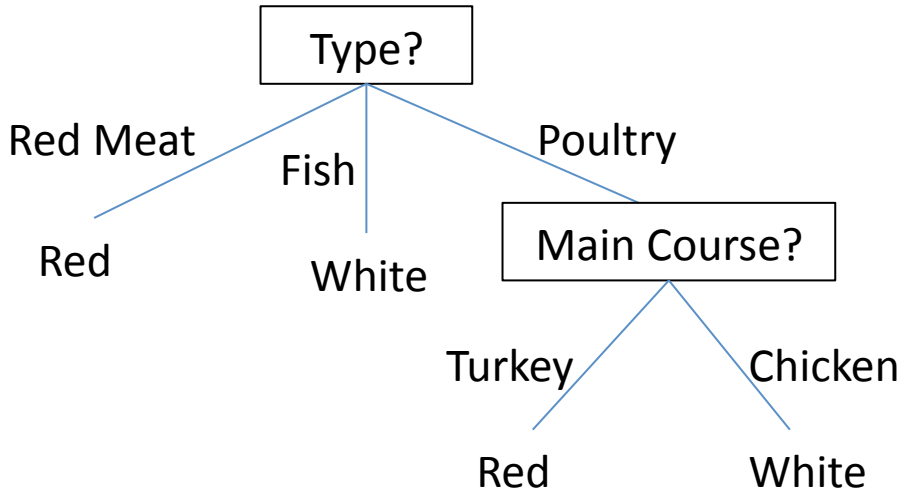
- Below are heuristics for choosing which wine to have with a meal:

Main Course	Type	Wine Choice
steak	red meat	red
turkey	poultry	red
dory	fish	white
chicken	poultry	white
beef	red meat	red
snapper	fish	white

- Construct a decision tree to represent these heuristics.

Decision Trees: Question

- What is the highest information-gain attribute (just by inspection)?
  - Type
- So we have:
- Now what?
  - Add leaves on red meat and fish:
  - Split poultry by main course:

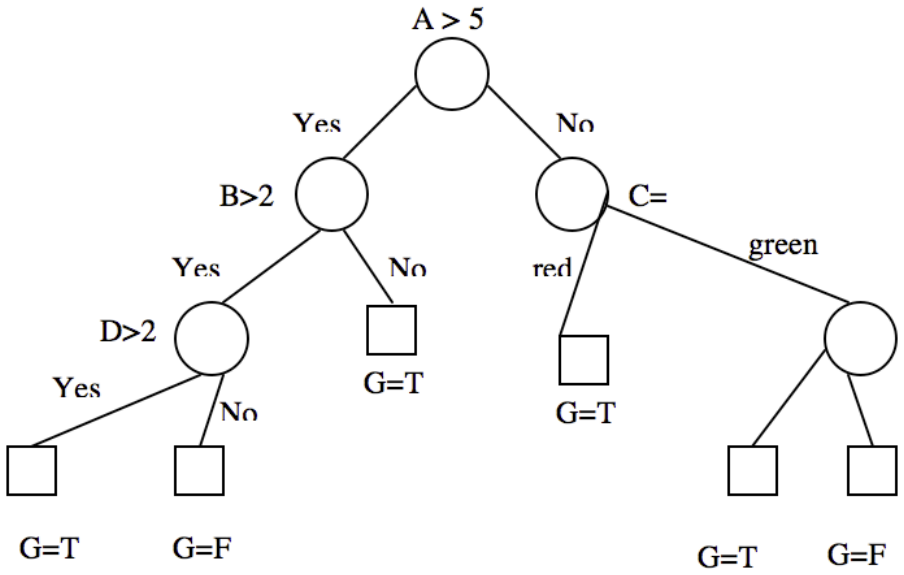


Remarks on Decision Trees / ID3

- Decision tree construction is effectively searching a space of all possible trees to find one that fits the data
- They have an inductive bias toward shorter trees with high information gain close to the root
- A potential problem with DTs is overfitting, which can be due to too few or too noisy training instances
  - This can be reduced by stopping growing the tree before it completely fits the training data, pruning the tree after growing, or validating against other test data
- Many modifications to DTs are possible (listed in lecture)

Decision Trees: Question

- Decision trees can be converted into rules. How many rules does the decision tree below have? Write out 2 of the rules in propositional logic.
- There are 6 rules.
- A>5 ∧ B>2 ∧ D>2 ⇒ G=
- A>5 ∧ B>2 ∧ D≤2 ⇒ G=
- There are 4 other correct rules….



Neural Networks

- Neural networks are simply a large number of weighted connections between simple decision-making elements
  - Learn by adjusting these weights

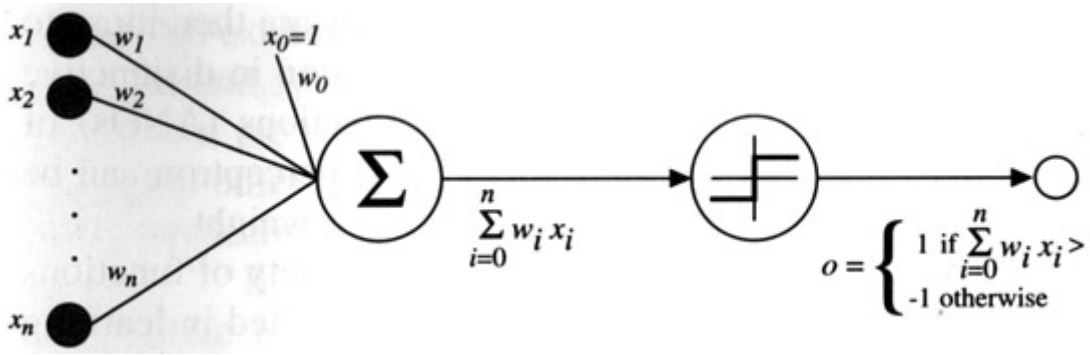


Diagram taken from lecture slides

- They can be trained to infer a continuous-valued function from input/output examples
  - Recall concept learning (infers boolean function) and decision trees (infers discrete function)

Neural Networks: Question

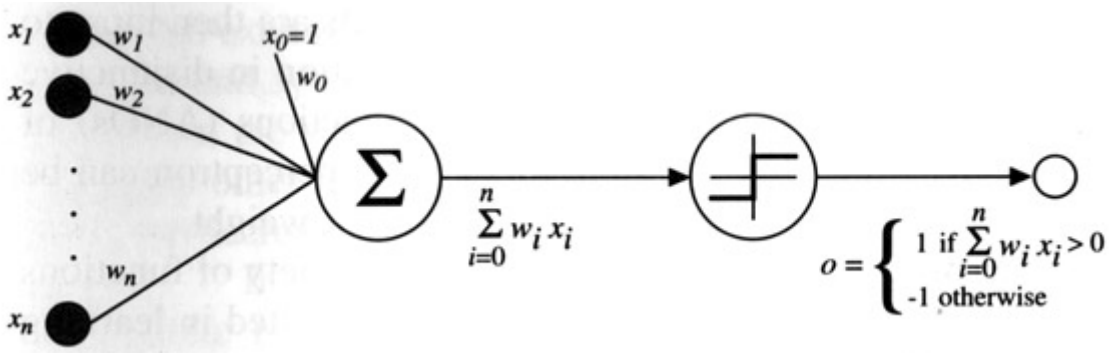
- In ANNs, gradient descent is used to update each weight associated with each edge in the neural network. So what is the hypothesis space that the ANN searches through?
  - Gradient descent is trying to find the combination of weight assignments which best approximate the target function.
  - So, it is searching through the space of all possible weight assignments to edges.

Neural Networks: Why?

- Neurons in animal brains have some nice properties we would like to copy:
  - Parallel / distributed processing
  - Powerful outcomes with few computation steps
  - Fault-tolerant of failing / misbehaving neurons
  - Fuzzy (approximating / probabilistic) logic

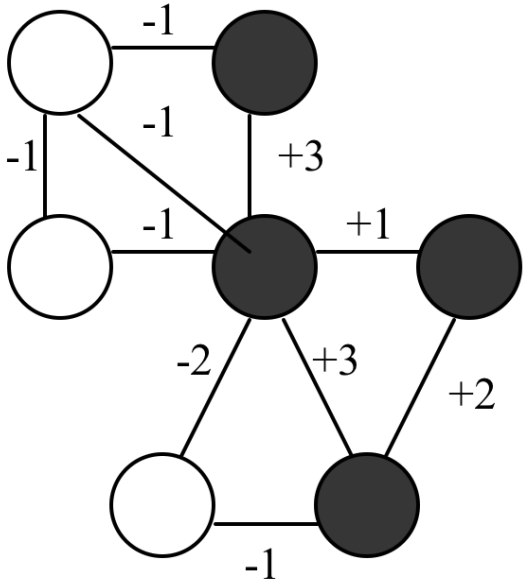
Neural Networks: Perceptrons

- Perceptrons are the artificial neurons in ANNs
  - Take inputs from the environment or other perceptrons, multiply by weights and sum up.
  - If the sum is over a threshold value, output 1, otherwise, output -1.
- Might seem simple and limited, but can layer to be very powerful and parallelisable
- Slow to learn (compared with CL / DTs)
- Weights adjusted by minimising errors (like checkers example)



Hopfield Networks

- A type of ANN which can be used for representation of information by a group of interacting elements
  - Distributed, asynchronous
- Can store a number of patterns and find the closest match to an input pattern.
- Some elements can be faulty and the system will still work.
  Input may be a partial pattern to match.
  - Fault tolerant

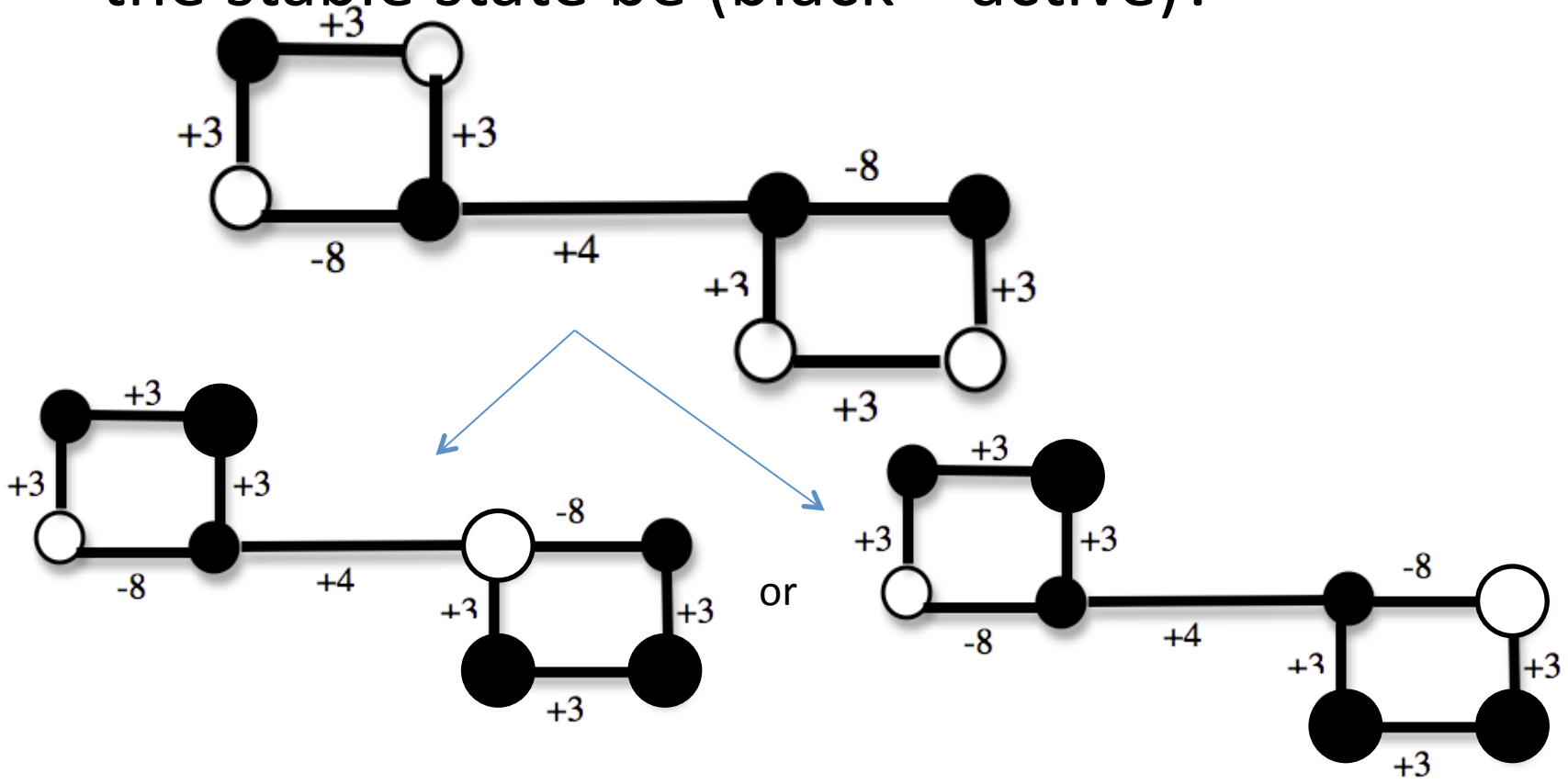


HNs: Parallel Relaxation

- Hopfield networks work by a process called parallel relaxation.
  - A random unit is chosen
  - The unit computes the sum of the weights on the connections to all of its active neighbors
  - If the sum is positive, the unit becomes active, otherwise it becomes inactive
  - This repeats until the network reaches a steady state
- **Parallel** relaxation will always reach a stable state

Hopfield Networks: Question

- Given the Hopfield network below what would the stable state be (black = active)?

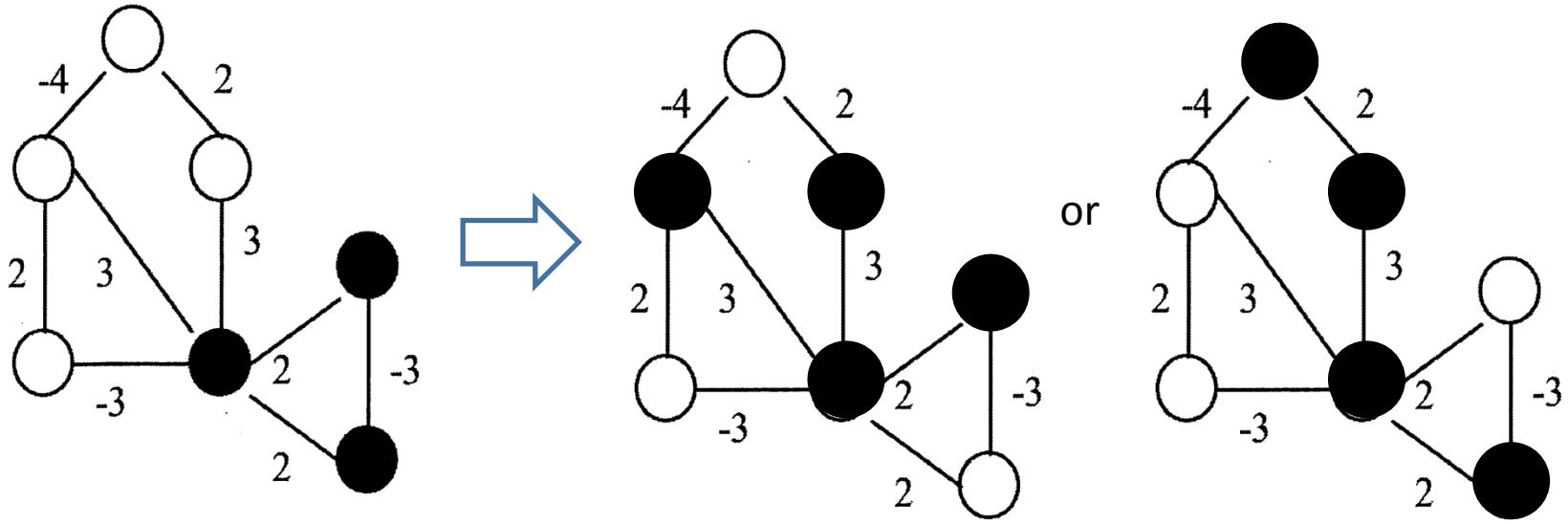


Hopfield Networks: Questions

- The parallel relaxation algorithm from Hopfield networks is a search algorithm. What does the hypothesis space look like?
  - When given an input, the Hopfield network tries to settle into a steady state that matches the input.
  - So it is searching through a space of all possible active/inactive allocations for nodes in the network, trying to find an allocation which is reachable from the input state and does not require any more updates to the network.

Hopfield Networks: Questions

- What will be the final stable Hopfield Network (black = active)?



Or a combination of the two