Decision Tree Learning

Rahul Dass

Dr. László Egri

CS 695 – Computer Science Research

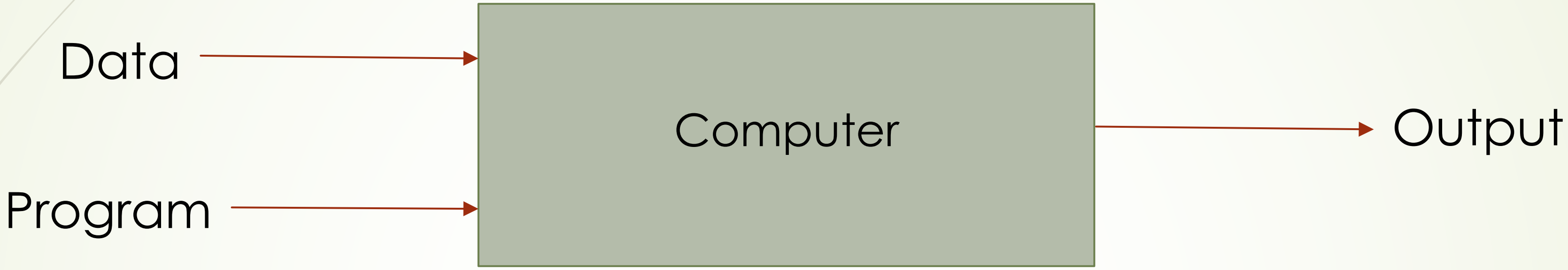
Department of Computer Science – Indiana State University

Contents

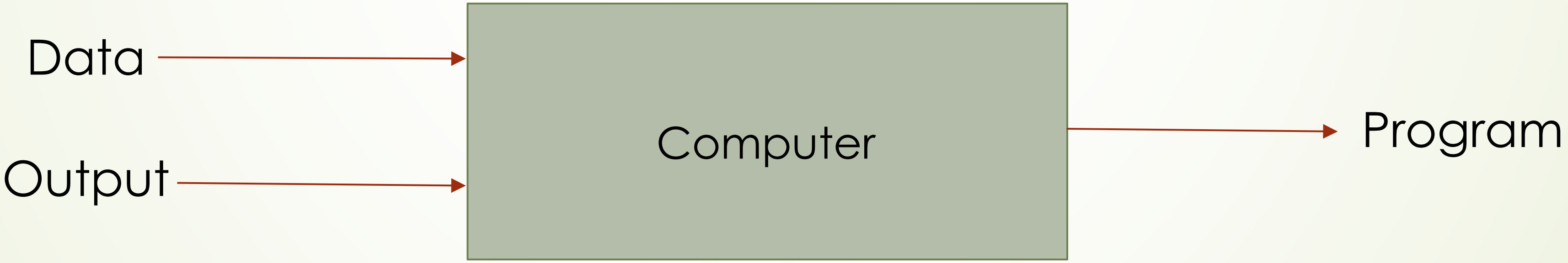
- Machine Learning – magic?
- Decision Tree – representing a ML algorithm
- ID3 algorithm
- Shannon's Entropy – purity measure
- Information gain – finding the "best" attribute
- Gain ratio
- Random Forest approach

Is Machine Learning magic?

- Traditional Programming:



- Machine Learning:



NB: Output of a ML algorithm, is another algorithm.

Think of ML as the "inverse" of traditional programming

Sounds too good to be true?

- No, not magic; ML is more like farming…
  - Seeds = Learning algorithms
    - A simple seed can grow into a whole tree
    - Simple learning algorithms can yield powerful, complex results
  - Fertilizer + water = Data
    - Fertilizer + water helps crops grow
    - In ML, data helps learning algorithms grow
  - Farmer = Programmer (you!)
  - Crops = Programs

Decision Tree – represent a ML program

- Problem: Predict (classify) if Roger plays tennis at any given day?
  - Collected some observations – for 2 weeks
  - He played 9 times and didn't play 5 times
  - The attributes in the dataset contribute to whether he plays tennis, on a given day

- Goal: Build a mechanism such that on Day 15 (unseen example), it would be able to <u>classify</u> if Roger plays or not?

  NB: Want to understand how individual attributes affects Roger's ability to play tennis.

How do Decision trees work?

- Using a divide and conquer approach + recursion:
  - Split the dataset into disjoint subsets based on all values per attribute
  - Check if those subsets are pure or not?
    [i.e. see if target value are all “yes” or all “no”]
  - If yes, STOP! Don’t need to make any further decisions
  - If not, repeat process (minus the attribute that we just considered)

Example…

- Consider the "Outlook" attribute – 3 distinct values (Sunny, Overcast, Rain)
- Split the dataset into 3-disjoint subsets
- Look at the target attribute to see if Roger plays consistently or not?
  - If yes, we conclude for that value of the attribute leads to Roger playing or not
  - If no, we repeat (recursively) the process of splitting those disjoint subsets further, but removing the "Outlook" attribute from the subset
  - Continue doing so, until we are left with pure sets

ID3 algorithm – building the decision tree

1. **Split (node, {examples})***
2. **A ← the "best" attribute for splitting the {examples}**
3. **Decision attribute for this node ← A**
4. **For each value of A, create a new child_node**
5. **Split training {examples} to child nodes**
6. **For each child_node:**

   If subset{examples} is pure: STOP

   else Split(child_node, {subset})

*Each node in the tree contains the decision node (attribute), and children (subset of training examples based on the values of the decision node)

- Computer Science: Ross Quinlan (ID3 – 1986), (C4.5 – 1993)
- Statistics: Breimanetal (CaRT – 1984)

Finding the “best” attribute

- Choose any attribute to split on => different partitioning of the dataset
- Crux: How to decide if one partitioning is “better” than the other?
- Consider splitting dataset based on:

Outlook (9 yes / 5 no)

- Sunny (2 yes / 3 no)
- Outlook (4 yes / 0 no)
- Rain (3 yes / 2 no)

Wind (9 yes / 5 no)

- Weak (6 yes / 2 no)
- Strong (3 yes / 3 no)

Shannon's entropy – "purity" measure

- **Def^n:** $\boldsymbol{H(S) = -p_+ log_2\, p_+ - p_- log_2\, p_-}$

  where: S: subset of {examples}
  p_+/p_-: proportion of $S_{\{examples\}}$ that have +ve/-ve target value

  log_2 => number of bits needed to overcome uncertainty in determining if an item is +ve or –ve.

- Q1: $S_{\{examples\}}$ is completely +ve – how many bits do we need if we pick one element from it?

- Q2: $S_{\{examples\}}$ is perfectly impure (3 p_+/3 p_-), – how many bits do we need if we pick one element from it?

Information gain

- Each node could have several children => multiple subsets
- So, take average of the entropy for all subsets per node
- **Def^n:** $\boldsymbol{Gain(S, A) = H(S) - \sum_{V \in Values(A)} \frac{|S_V|}{|S|} H(S_V)}$

  NB: $\frac{|S_V|}{|S|}$ is the weight of S_V compared with S

  Give more weight to larger subsets

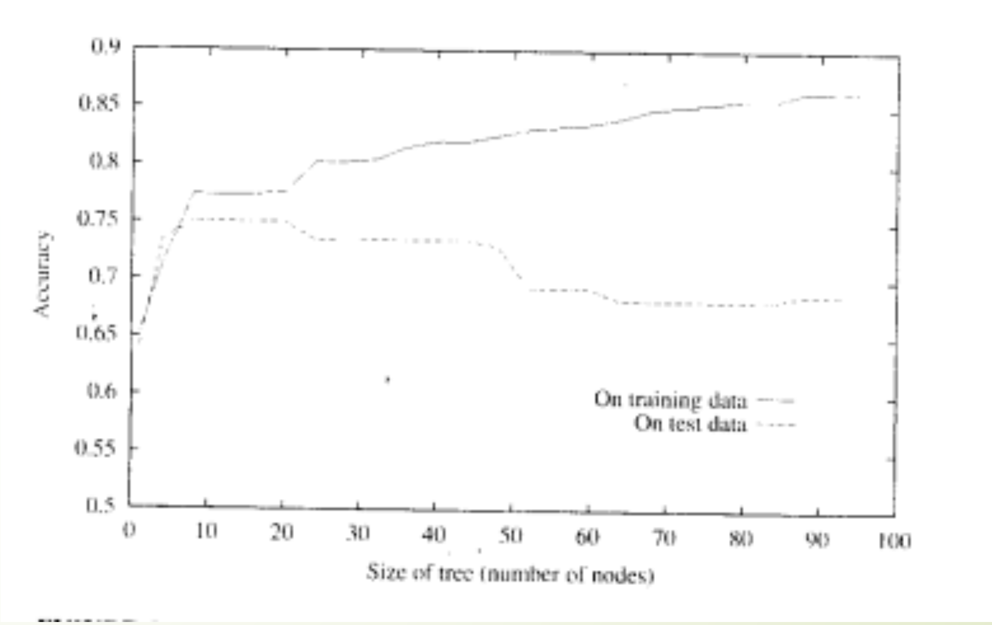
- Link between information gain and "best" attribute?
  => Choose the attribute with maximum gain after splitting S

Using Information gain

- Take every A that we have in our data
- Compute the information gain for that A
- Select A that has the highest information gain:
  - Because that A will most reduce our uncertainty
  - Out of all $A's$, that A will lead to the purest possible split
- If there are mixed sets as children, then recursively compute the information gain for sets (minus the A we just looked at)

Always give perfect solution? - Yes and no!

- Yes! – Run ID3 to build the decision tree, it will always partition the dataset perfectly – guaranteed (singletons)
- No! – going so deep is maybe something that we don't want
  - Singletons => not a lot of confidence in predictions (for unseen)
  - "Lack of confidence" seen when decision tree accuracy is compared with their size



Information Gain revisited…

- Information gain is a good measure – but also flawed!
- If we included day, itself, as an attribute – then we've found the "holy-grail" of decisions
  - Found an <u>optimal split</u> because by choosing "day" as the attribute, each of its values is perfectly pure!
  - However, this isn't useful at all – when applied to new, unseen examples [Eg: Day 15 – there is no branch?!]
  - In general for very large datasets, this will happen as gain would latch on to such attributes and put them as high nodes in the tree
  - Results to poor predictions for test data, resolve this by using Gain ratio…

Gain ratio

- **Def^n:** $\boldsymbol{SplitEntropy(S,A) = -\sum_{V \in Values(A)} \frac{|S_V|}{|S|} log_2 \frac{|S_V|}{|S|}}$

- **Def^n:** $\boldsymbol{GainRatio(S,A)} = \frac{Gain(S,A)}{SplitEntropy(S,A)}$

- Take the information gain of A and normalize it using the SplitEntropy

- SplitEntropy = does A give a lot of small subsets or not?

- Doesn't look at +ve/-ve values for p, instead it looks at the split itself => high values (lots of small subsets) or low values (otherwise)

Random Forest algorithm

- Simple, (surprisingly) very effective idea…
- Instead of growing a single tree for S, we grow K-different trees, by randomizing the input:
  - Take S and take a random subset: S_r
  - Use S_r to learn/build a decision tree (run ID3, with no pruning):
  - When splitting, pick a subset of the total number of attributes: A_r
  - Compute gain based on S_r instead of the full dataset S
  - Repeat K-times as we're using S_r different subsets of data, and A_r different subsets of attributes
  - So for new example, run it through all K-trees; each tree will give a classification; take the majority vote!

Summary

- Decision trees are fast, compact and easy to understand
- ID3: recursively grows a decision tree from the root down
  - Greedy when it selects the "best" attribute (using entropy/gain)
  - Entropy: measure of how pure/impure is a subset of the dataset
  - Gain: difference in the entropy between before and after splitting on an attribute
- ID3 will continue splitting perfectly, if we don't stop it
- Looked at ways of optimizing the decision tree using:
  - Information Gain ratio
  - Random Forest approach

References

[1] P. Domingos, *Data mining and Machine Learning*, University of Washington - CSEP 546 Lecture Series, 2016.

[2] V. Lavrenko and N. Goddard, *Introductory Applied Machine Learning*, University of Edinburgh – INFR10069 Lecture Series, 2015.

[3] T. Mitchell, *Machine Learning*, McGraw Hill, 1997.

[4] I. H. Witten, E. Frank and M. A. Hall, *Data Mining: Practical Machine Learning Tools and Techniques (3rd Edition)*, Morgan Kaufmann, 2011.