**CS 4320 Lecture 3 (9/11/2020):**

* Supervised and inductive learning
  + Ex- classify tasty apples based on experience
    - Training set: Apples you have tasted in the past, their features and whether they were tasty
    - A screenshot of a cell phone

      Description automatically generated
* Supervised Learning
  + Instance space: Insatnce space X including fesature representation
    - X=(A,B)x(red, green)x…
  + Target attr:
    - A set Y of labels E.g. Y=(tasty,not tasty) or Y=(Yes,No)
  + Hidden target function
    - An unknown function f: X->Y, how apples feature correlate with its tastiness in real life
  + Training data:
    - A set of labelled pairs of X,Y that we have seen before and have target attr for
  + Hypothesis Space
    - A prefixed set H of function we consider when looking for h: X->Y
  + Ex-
    - All function from X->Y
    - All hypotheses that are AND of feature-values: (color=red) AND (firmness=Crunchy)
    - A screenshot of a cell phone

      Description automatically generated
* Consistency
  + A function h: X->Y is consistent with a set of labeled training examples S if and only if h(x)=y for all (x,y) element of S
  + Ex- (Color=red AND (firmness=Crunchy) is NOT consistent with the table on the last slide as it made a mistake on apple 5
  + (Color=red AND (firmness=Crunchy) is consistent with the table below
  + A screenshot of a cell phone

    Description automatically generated
* Inductive Learning
  + Goal: Given a large enough number of training examples and a hypothesis space H, learn a hypothesis h element of H that approx. f(.)
  + Our hope: There is a h element of H that approx. f even on unseen data
  + Our strategy: Our training examples are representative of the reality
    - We have seen many examples
    - They were selected without any bias
    - A y hypothesis that is consistent with the training data would be accurate on unobserved instances
    - 
* Using consistency in Learning
  + Idea: On a training set S, return all h element of H that is consistent with it
  + Version space: the version space VS(H,S) is a subset of functions from H that is consistent with S.
* Version Space Example
  + For the following set of training examples S and H that is all hypotheses that are AND of feature-values, what is VS(H,S)?
  + A screenshot of a cell phone

    Description automatically generated
  + 1. To classify #10 as yes, any consistent function has to be subsets of its features (Farm=A) ^(Color =Red)^(Size=Small)^(Firmness=Crunchy)
  + 2. To classify #7 as No but #10 as Yes (Firmness =Crunchy) is necessary.
  + 3. To classify #2 as No but #10 as Yes (Color=Red) is necessary.
  + 4. List all functions and check to see that they are all consistent:
    - 𝑓1: (Farm=A)∧(𝐶𝑜𝑙𝑜𝑟=𝑅𝑒𝑑)∧(𝑆𝑖𝑧𝑒=𝑆𝑚𝑎𝑙𝑙)∧(𝐹𝑖𝑟𝑚𝑛𝑒𝑠𝑠=𝐶𝑟𝑢𝑛𝑐ℎ𝑦)
    - 𝑓2: (𝐶𝑜𝑙𝑜𝑟=𝑅𝑒𝑑)∧(𝑆𝑖𝑧𝑒=𝑆𝑚𝑎𝑙𝑙)∧(𝐹𝑖𝑟𝑚𝑛𝑒𝑠𝑠=𝐶𝑟𝑢𝑛𝑐ℎ𝑦)
    - 𝑓3: (Farm=A)∧(𝐶𝑜𝑙𝑜𝑟=𝑅𝑒𝑑)∧(𝐹𝑖𝑟𝑚𝑛𝑒𝑠𝑠=𝐶𝑟𝑢𝑛𝑐ℎ𝑦)
    - 𝑓4:(𝐶𝑜𝑙𝑜𝑟=𝑅𝑒𝑑)∧(𝐹𝑖𝑟𝑚𝑛𝑒𝑠𝑠=𝐶𝑟𝑢𝑛𝑐ℎ𝑦)
* Computing the version space
  + At a high level:
    - List all hypothesis in H,
    - Remove them if they are not consistent with some example in S
    - A screenshot of a cell phone

      Description automatically generated
  + Takeaway: The algorithm works, but
    - Keeping track of the version space is time and space consuming
    - Only need one consistent
    - Build a consistent hypothesis directly
* Decision Tree Definition
  + Functions that are displayed as a tree:
    - Internal nodes [rectangles]: test one feature, e.g. color.
  + Branches from an internal node [arrows]:
    - Possible values of the feature that’s being tested e.g. {red,green}
  + Leaf nodes [ovals]:
    - The label of instances whose features correspond to the root-to-leaf path e.g. Yes or No
    - A close up of text on a white background

      Description automatically generated
* Using a Decision Tree
  + To eval a decision tree f on an instance x
    - Follow the root-to-lead path indicated by feature-values of x.
    - Return the leaf value
  + Let function f be the following decision tree
    - 1. What is f((B, Red, Large, Crunchy))? “Yes”
    - 2. What is f((A, Red, Large, Soft))? “No”
* Checking for Consistency
  + To see if decision tree f is consistent with a data set, one can:
    - 1. Consider root-to-leaf paths as logical ANDs
    - 2. See if instances captured by these ANDs are labeled correctly
  + Is f consistent with the training set in the table
  + A close up of text on a white background

    Description automatically generated
* Making a Tree
  + Ex- What is the decision tree corresponding to functions
  + A close up of text on a white background

    Description automatically generated
* Growing a Consistent Decision Tree
  + Given a data set S, how do we find a decision tree that is consistent with it?
  + Idea:
    - Don’t use List-then-Eliminate
      * It’s too inefficient
      * It does not leverage the structure of decision trees
    - Instead grow a good decision tree to start with
      * We grow from root to leaves
      * Repeatedly take an exiting leaf that does not include a definitive label and replace it with an internal code
* Top-Down Induction of Decision Trees (IDT)
  + A screenshot of a cell phone

    Description automatically generated
  + A picture containing clock

    Description automatically generated
  + IDT ex- Use IDT on the following table, and let feature A be Firmness, Color, Size, Farm, in order
  + A screenshot of a cell phone

    Description automatically generated
  + A screenshot of a cell phone

    Description automatically generated
  + A screenshot of a cell phone

    Description automatically generated
  + A screenshot of a cell phone

    Description automatically generated
  + A screenshot of a cell phone

    Description automatically generated
  + A close up of text on a white background

    Description automatically generated
* Which split?
  + How should we pick feature A for splitting?
    - How does the prediction improve after the split?
    - At each node, take the number of positive or negative instances
    - If we were to stop, best to predict the majority label
    - How do we achieve a more definite prediction?
    - A close up of text on a white background

      Description automatically generated
  + Based on error
    - Before split: Err(S) = size of the minority label
    - After split: Err(S|A) =A picture containing drawing

      Description automatically generated
    - Choose A that maximizes Err(S)-Err(S|A)
  + Based on entropy
    - A screenshot of a cell phone

      Description automatically generated