Lesson 1: Decision Trees

Classification and regression

In supervised learning, you are presented with instances (e.g. images of individuals) with labels (e.g. "BOY" or "GIRL") as training data. The task is to label new unlabeled instances (assuming the instance has the sufficient information).

- **Classification** – labels are discrete values (often finite, often true or false).
  - Better definition [Shashir]: labels (codomain of the target function) have no meaningful order.
- **Regression** – labels are reals.
  - Better definition [Shashir]: labels have a meaningful order.

Classification learning

- **Instances** – inputs (vectors of features).
- **Concept** – a function which maps instances to labels (there many concepts: |labels|^|instance space|).
  - [Shashir] Concepts are the set of functions mapping from the instance space to the labels space.
- **Target concept** – the function which maps instances to the **correct** labels.
  - [Shashir] The target concept is a specific concept which we wish to model.
- **Hypothesis** – set of concepts which we are willing to search for the best approximation of the target concept.
  - [Shashir] Subset of the concepts set. Easier space to search through, but introduces *inductive bias*.
- **Sample (training set)** – set of inputs with correct labels.
- **Candidate** – the "best" concept chosen from the hypothesis by the learning algorithm using the sample.
  - [Shashir] Element of hypothesis which best approximates the target concept according to our learning algorithm.
- **Testing set** – set of instances with correct labels, similar to the training set, but used to measure how well the candidate performs on novel data.

The testing set should contain many examples not found in the training set.

Decision trees

Sequence of tests (path of nodes starting from a root node) applied to every instance in order to arrive at its label (leaf).

Decision trees: learning

1. Pick the best attribute to split the data.
2. Asked test every possible value of the attribute.
3. Follow the correct answer path.
4. Go to 1 until the possibilities has been narrowed to one answer.

How can this intuition be used to build a "tree" for all possible instances of the problem?

Decision trees: expressiveness

Consider the inclusive disjunction: **OR(a1, a2, . . . , an)** (any). Note that the tree is "linear"

height: O(n)

nodes: O(n).

```
--->a1--T-->a2--T-->...--T-->an--T-->TRUE
        |       |                |
        F       F                F
        |       |                |
        v       v                v
      FALSE   FALSE            FALSE
```

Next, consider the exclusive disjunction: **XOR(a1, a2, . . . , an)** (odd parity). Note that the tree is balanced.

height: O(n)

nodes: O(2ˆn)

Number of nodes is exponential to n.

```
--->a1--T-->a2--T-->a3--T-->TRUE
     |       |       |
     |       |       F----->FALSE
     |       |
     |       F----->a3--T-->FALSE
     |               |
     |               F----->TRUE
     |
     F----->a2--T-->a3--T-->FALSE
             |       |
             |        F---->TRUE
             |
             F----->a3--T-->FALSE
                     |
F----->TRUE
```

It's better just to add in integers mod 2.

Decision trees: expressiveness (search space)

Given n binary attributes, how many possible decision trees are there? $2^{(2}$n)

Intuition: * There are 2ˆn possible configurations of the attributes. * Each "unique" decision tree maps these 2ˆn configurations into a 2ˆn-sized bit vector. * Each bit has two possibilities (True or False). * So, There are $2^{(2}$n) possible bit vectors of size 2ˆn. * Therefore, there must are $2^{(2}$n) possible unique classifiers.(Note: $O(2^{2^n})$ is very large number.) * Note that more than one tree may map to a single classifier, so the hypothesis space is even larger (thanks to inductive bias, we can cut down the problem significantly).

ID3 Algorithm

```
A <- best attribute from remaining attributes (initially, all attributes)
Assign A as decision attribute for Node
For each value of A, create a new descendant of Node
Sort training examples to leaves
If examples are perfectly classified, STOP.
Else if we ran out of attributes, STOP
Else, start over for each leaf (with corresponding set of training examples)
```

The **best attribute** is that one with the greatest information gain.

$$GAIN(S, A) = ENTROPY(S) - \sum_v \frac{|S_{A=v}|}{|S|} ENTROPY(S_{A=v})$$

- Where S is the collection of training dataset, and A is a particular attribute.
- Means the total entropy minus the average entropy over the sets of values of the attribute A.

Where **ENTROPY** is defined as:

$$ENTROPY(S) = -\sum_v p(v) log(p(v))$$

The **best attribute** is the one that splits the data into subsets whose entropies' weighted sum is the least (maximizing the information gain).

$$A^* = argmin_A \sum_v |S_v| \cdot ENTROPY(S_v)$$

The **inductive bias** of the ID3 algorithm: * The best splitters appear earlier (closer to the root). * Produces shorter trees. * Prefers correct classifiers over incorrect trees (thanks for that)

Bias

- **restriction bias**:
  - The hypothesis set your care about.
  - For example, in this case, it's all decision trees. We're not considering the $f(x) = x^2$ or anything else.
- **preference bias**:
  - From the hypothesis set, what do we prefer?
  - This is the heart of the inductive bias.

Decision trees: other considerations

- How do handle continuous attributes?
  - Use intervals
    - Split age range 0-90 into 0-40 and 40-90
    - Perhaps even use a modified ID3 to find the best splitting age.
- Does it make sense to repeat an attribute **along a path (appear as a child of the former oone)** in the tree?
  - No for finite-valued attributes.
  - **However**, *continuous attributes* can be tested with different questions
  - The same question doesn't need to be asked twice.
- When do we stop?
  - Everything classified correctly (or nearly correct – we do not want to **overfit**).
  - No more attributes.
  - Do not **overfit**.
    - Try not to have a tree which is too big.
    - Try many trees and cross-validation.
    - Variant of cross-validation where you hold out a subset of the data and build a tree breadth-first on the remaining data. Stop when error is "low enough."
    - Build the whole tree and prune (vote if the classification is not perfect).
- Regression
  - Splitting criteria: variance?
  - Model output and group them (round off or cluster).
  - On leaves: Report average, or vote, or locally fit a line (hybrid).

Decision trees

We learned: * Representation (tree… set of questions) * ID3: a top down learning algorithm * Expressiveness of DTs * Bias of ID3 * “Best” attribute Gain(S, A) * Dealing with overfitting.