Scribe Notes from Tuesday January 20 and Thursday January 22, 2004

By:

Maryam Bavarian

Ladan Mahabadi

Irina Pekerskaya

On Tuesday January 20, 2004, a few criticisms of the FIND-S algorithm were discussed:

1) Lack of Convergence of the learner to all possible target concepts

Criticism:

- FIND-S is confined to determining only one consistent hypothesis. In other words, it is unable to find all the hypotheses that are consistent with the training data when applicable.

- Hard to determine when the algorithm has learned the concept in order to stop

As Mitchell claims, it is desired to have a learning algorithm that "… could determine whether it had converged and, if not, at least characterize its uncertainty regarding the true identity of the target concept.

Plausible justification:

- ✓ At times only one hypothesis is adequat
- ✓ One intrinsic property of every learner is that it is hard to determine certainly when the task of learning has been accomplished. Furthermore, future data can prove any learner wrong!

2) Unclear Advantage of choosing the most specific hypothesis.

Criticism:

- The advantage of the most specific hypothesis, which FIND-S finds, over some other consistent hypothesis such as the most general hypothesis is unclear

Justification:

- ✓ Specific hypotheses are useful

3) Lack of contingency plan in case of noisy data

Criticism:

- FIND-S may be severely misled in case of inconsistent sets of training examples for it ignores negative examples.

Mitchell adds that "… an algorithm that could at least detect when the training data is inconsistent and …accommodate such errors" is preferred.

Plausible justification:

- ✓ Our assumption is consistency. In other words, it is assumed that the given training data is consistent and the target hypothesis can indeed be induced from it

4) Lack of detecting Multiple maximally specific hypothesis

- If in the hypotheses space, there are several maximally specific hypotheses, FIND-S , as alluded in the first criticism, fails to detect them.

The above criticism leads to the following brief introduction of evaluation criteria of algorithms which in turn can clarify: "What constitutes a "good learning algorithm?"

Valid for given assumptions

Prove Performance:

- Convergence
- Mistake bound

- Confidence Interval (even though this criterion is more considered a property of the hypothesis and justifies the estimate)
- Unbiased and consisten Estimation:
  "unbiased estimator" (prove that you are not too far from the truth), "consistent estimator" (prove that it converges). The last two estimators are proved by statisticians.
- Computation Efficiency

Time and Space considerations

- User friendliness
- Scalability
- Training time
- Success on some problems ("Empirical Studying")
-

Even though the above factors are very useful guidelines in proving a learner effective, there is always the issue of incompleteness. In other words, success on some training data does not guarantee future success unless the structure of the training data is conserved in the test data. Furthermore, if a learner is very successful on some problems, it does not imply that will be as effective on all problems. Simply put, there is not one ideal algorithm. This encourages one to look for patterns among problems instead of looking for pattern among data.
One common factor among all learners is that they are based on a plausible principle.
e.g.

- Occam's Razor

"`***Pluralitas non est ponenda sine necessitas.'***

(Plurality should not be posited without necessity.)

A logical principle by William of Occam who stated that one need not make more assumptions than the minimum needed. Simply put, the shortest interpretation is the best. In our applications, the shortest hypothesis and later on the shortest decision tree are preferred based on this principle.
Jacob Eliosoff[2] lists the following machine learning related interpretations of Occam's Razor:

1. **When deciding between two models which make equivalent predictions, choose the simpler one.**
2. **If two decision rules classify the existing data equally well, the simpler one is more likely to classify future data correctly.**
3. **Given a simple decision rule A, and a much more complex rule B which classifies the existing data only slightly better than A, A is likely to classify future data better than B.**
4. **Simpler classifiers are more likely to be correct.**

Evaluating Learners

- S : Simulation studies

a. Randomly generate target concept (probability distribution) = c
b. Randomly generate data according to concept

c. Run learner on data ➔ output *H*

d. Compare c with *H*

A Standard but Dubious Way of Designing a Machine Learning System

1. Choose a syntactic representation for hypotheses (e.g., decision tree, Bayesian Net, neural net)

2. Given evidence, solve the computational problem of finding hypothesis with "best fit"

Problem:

- No performance guarantee (convergence, generalization)

- The syntax implicitly introduces a bias – perhaps not the one the user wanted.

More Systematic Approach

1. Determine first what performance you want, or what "the best fit" is.

2. Then choose a syntactic representation that makes it easy to implement this performance.

One major short coming of the FIND-S algorithm is its inability to find all the hypotheses consistent with the training data. CANDIDATE-ELIMINIATION algorithm addresses this shortcoming and "… outputs a description of the set of all hypotheses consistent with the training examples." The beauty of this algorithm is that it does not list all of such hypotheses because that would be impractical. It would only keep track of two hypotheses from which all the other consistent hypotheses may be generated if needed. The two representative hypotheses are the Specific boundary S and the general boundary G.

Definition:

- Specific boundary S:
  The set of maximally specific members of the hypothesis space consistent with the training data

- General boundary G:
  The set of maximally general members of the hypothesis space consistent with the training data

As long as the sets G and S are well defined, the version space (The subset of hypotheses from the hypotheses space consistent with the training data) can be completely specified.

⇒ **<u>Version Space Representation Theorem:</u>**

Version Space = S U G U {consistent hypotheses that lie Between S and G in the partially ordered hypothesis space}

⇒ Proof:

- Show that every consistent hypothesis, h ,that has the following property

$s \leq h \leq g$ (for some s in S and g in G where $\leq$ denotes a partial order)

will be in the version space

By definition of S all of its members will be consistent with all the positive examples in the training data. Since h is more general than s and g is more general than h then they must also be satisfied by all the positive training examples. Similarly, no member g of G can be satisfied by any negative training examples and since g is more general than h and in turn more general than s then h and s can not possible satisfy any of the negative examples of the training data.
Thus
S U G U R is a subset of the Version Space

where R denotes {consistent hypotheses that lie
Between S and G in the partially ordered hypothesis space}

- Show that
  Version Space is a subset of S U G U R

  Proof by contradiction:
  Let 's assume that there is a positive element j of the version space such that
  j is included in S U G U R

  Then since G contains all the maximally general hypotheses in the version space, it must be that
  $j \geq g$ (for all g in G). Thus by the definition of G, j must be a negative training datum. This contradicts our assumption and thus, version space is a subset of S U G U R

- The above two steps prove that
  - Version Space = S U G U R

Candidate-Elimination Learning Algorithm

This algorithm computes the version space containing all hypotheses from H that are consistent with an observed sequence of training examples. The algorithm consists of the following steps:

1. Initialize G to the set of maximally general hypotheses in H

$G_0 \leftarrow \{<?,?,?,?,?,?>\}$

2. Initialize S to the set of maximally specific hypotheses in H

$S_0 \leftarrow \{<\phi,\phi,\phi,\phi,\phi,\phi>\}$

3. For each training example d, do

- *If d is a positive example*
  - *Remove from G any hypothesis inconsistent with d*

- *For each hypothesis in S that is not consistent with d*
  - *Remove s form S*
  - *Add to S all minimal generalizations h of s such that*
    - *h is consistent with d, and some member of G is more general than h*
  - *Remove from S any hypothesis that is more general than another hypothesis in S*

- *If d is a negative example*
  - *Remove form S any hypothesis inconsistent with d*
  - *For each hypothesis g in G that is not consistent with d*
    - *Remove g from G*
    - *Add to G all minimal specializations h of g such that*
      - *h is consistent with d, and some member of S is more specific than h*
    - *Remove form G any hypothesis that is less general than another hypothesis in G*

We can see that as each training example is considered, the S and G boundary sets are generalized and specialized, respectively to eliminate from the version space any hypotheses found inconsistent with the new training example. After all examples have been processed, the computed version space contains all the hypotheses consistent with these examples and only these hypotheses. This algorithm can be applied to any concept learning task and hypotheses space for which these operations are well-defined.

Remarks on Candidate-Elimination algorithm:

1. The target concept is exactly learned when S and G boundary sets converge to a single, identical hypothesis.
2. The version space learned by this algorithm will converge toward the hypothesis that correctly describes the target concept, if:
   1) There are no errors in the training examples;
   Ex.1: If a positive example is incorrectly presented as negative, the algorithm will remove the correct target concept from a version space.
   2) There is some hypothesis in H that correctly describes the target concept
   Ex.2

	Sky	…	Enjoy Sport
1	Sunny	…	Yes
2	Cloudy	…	Yes
3	Rainy	…	No

   If we use only conjunctions, we are unable to represent disjunctive target concept such as "Sky=Sunny or Sky=Cloudy"

3. The optimal query strategy is to generate instances that satisfy exactly half of the hypotheses in the current version space.
Why?
If the trainer classifies this instance as positive – S will be generalized
If the trainer classifies it as negative – G will be specialized

So, the size of the version space will be reduced by half with each new example, and the correct target concept can be found in $\log_2$[VS] steps.

4. Even if the version space still contains multiple hypotheses, indicating that the target concept has not been learned, it is possible to classify certain examples with the same degree of confidence as if target concept had been uniquely determined.

1) Instance classified as positive by every hypothesis in the current version space. Then, regardless of which hypotheses will be found to be the correct target concept, it is already clear the example will be classified as positive.
2) Instance classified negative by every hypothesis - it can be safely considered negative
3) If half of hypotheses classify an example as positive, and another half as negative, we can't say anything about its actual class.
4) More hypotheses classified as positive than as negative (or vice versa) – case is still quite ambiguous, but one approach it to output the majority vote. The proportion of hypotheses voting for the certain class can be interpreted as the probability that instance belong to this class.

Supervised Learning

- Many forms of machine learning ~ function approximation

Given: Set of **training examples**:
$\{(x_1, f(x_1)), \ldots, (x_m, f(x_m))\}$
Space of **hypotheses** H

Find: Hypothesis $h \in H$ that is good approx'n to f
(ie, s.t. $h(x) \approx f(x)$ for most x in space)

- Note $f: X \rightarrow R$ not known (f need not be in H)
  Want h that works well throughout "instance space" X …. Training examples only small subset

  Typically x_i represented by vector of **feature values:**

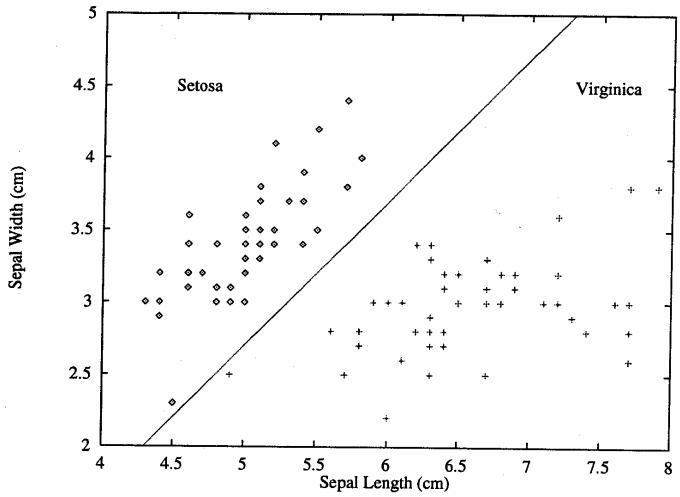
$$x_i = \langle x_i^{(1)}, \ldots, x_n^{(n)} \rangle \in X$$

….perhaps x_i in R^n, or discrete, pr combination, or …

- Kinds of functions
  - Real-Valued functions: $f(x)$ in R
  - Probability Distibution: $P_f(y \mid x) = P(f(x)|x)$
  - Classifiers: f(x) in $\{c_1,\ldots,c_k\}$

  ... perhaps $\{c_1,c_2\} = \{T, F\}$

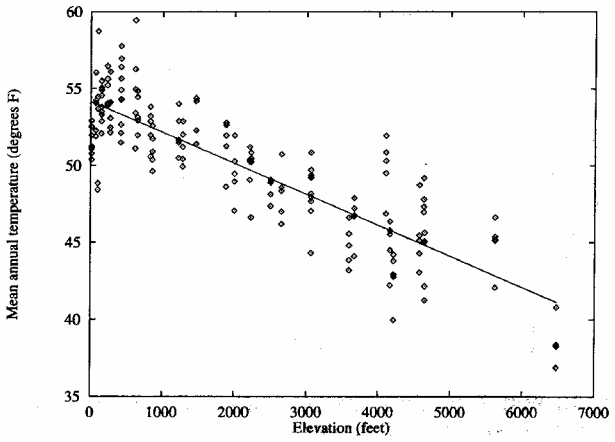
Discrete - Valued Functions: Classification



- **Unknown function:** maps from flower measurements to species of flower
- **Examples:** 100 flowers measured and classified by R.A. Fisher
- **Hypothesis space:** All linear discrimiators of form

$h(x) = \begin{cases} setosa & \text{if } w_0 + w_1.x.SepalWidth + w_2.x.SepalLength > 0 \\ Virginica & \text{otherwise} \end{cases}$

Real – Valued Functions



- **Unknown function:** maps elevation to mean annual temperature
- **Examples:** 175 weather stations with known elevation and temperature
- **Hypothesis space:** All functions of form

$$Temp = w_0 + w_1.elev$$

Inductive bias

1. An unbiased learner

As we have seen, the hypotheses space for Candidate-Elimination algorithm is biased – we consider only conjunctive hypotheses, the number for which is 973 for the example from the homework.
If we wish to assure that the hypothesis space contains the unknown target concept, we can consider **unbiased learner**, that will be capable to represent any possible subset of instances. For our examples there will be 2^{96} distinct target concepts.
Now we can reformulate EnjoySport task by defining new hypotheses space allowing arbitrary disjunctions, conjunctions and negations of our earlier hypotheses.

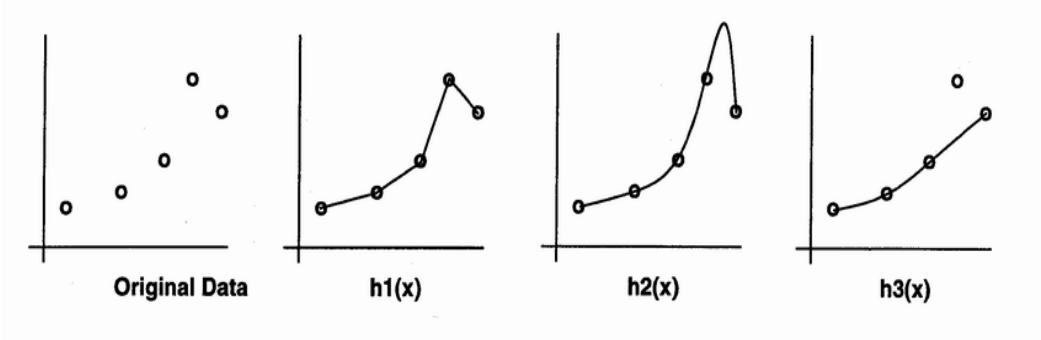
- ✓ We can express any target concept

? Our algorithm is unable to generalize beyond the observed instances.
Why?
S boundary – always disjunctions of observed positive examples
G boundary – negated disjunction of observed negative examples
Now, in order to converge to a single concept, we need to have every single instance in X as a training example.

The fundamental property of inductive inference: a learner that makes no a priory assumptions regarding the identity of the target concept has no rational basis for classifying any unseen instances.

2. Finding an appropriate hypothesis.

Our goal is **to generalize**, that is to build hypotheses according to original data



Which of the hypotheses is the best?
It depends:

- If data is noisy or not?
- What is known about the function? (Ex. Piece-wise linear, smooth…)

3. Bias

Learning algorithms embodies some "bias" to prefer one hypothesis over another

There are two types of bias:

- **Restriction bias or language bias** – specifies what hypothesis space is searched
- **Preference bias or search bias** – specifies how hypothesis space is explored

So, the tradeoff is
If choosing more expressive language with bigger hypothesis space, it's harder to find good hypothesis.

Def: Consider a concept learning algorithm L for the set of instances X. Let c be an arbitrary concept defined over X, and let $D_c = \{(x,c(x))\}$ be an arbitrary set of training examples of c. Let $L(x_{i,} D_{c)}$ denote the classification assigned to the instance x_i by L after training on the data $D_{c.}$ **The inductive bias** of L is any minimal set of assertions B such that for any target concept c and corresponding training examples D_c
$(\forall x_i \in X)\ [(B \wedge D_c \wedge X_i) \Rightarrow L(x_{i,} D_c)]$
where the notation $y \Rightarrow z$ indicates that z follows deductively from y (i.e. that z is provable from y

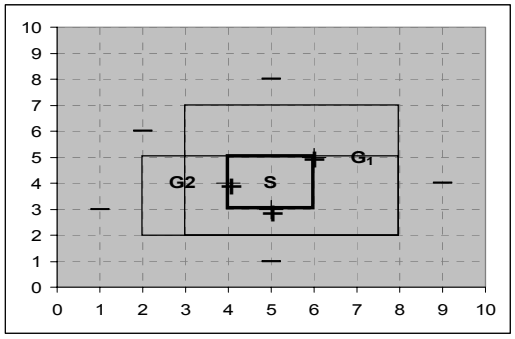
Inductive bias of Candidate-Elimination algorithm – the target concept c is contained in the given hypothesis space H.

4. Three learners with different biases:

Algorithm	**Principles of classification**	**Bias**	**Effectiveness of work**	**Correctness**
Role learner	Stores all observed examples. Can classify x if and only if it matches previously unseen examples.	Has no inductive bias, as no additional assumptions required	Cannot classify any unseen examples	Those that were classified done correctly
Candidate Elimination algorithm	New instances are classified only in the case where all member of the current version space agree on the classification	The target concept is represented in its hypothesis space	Can classify some instances that Role-learner will not	Correctness depends on the correctness of the inductive bias
Find-S	Finds the most specific hypothesis consistent with the training examples	The target concept is represented in its hypothesis space, All instances are negative unless the opposite is entailed by its other knowledge	Can classify even more unseen instances	Correctness depends on the correctness of the inductive bias

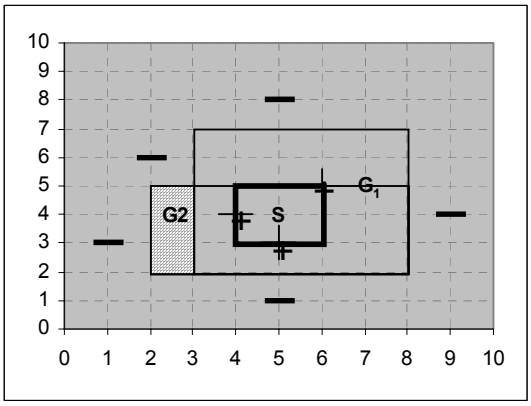
So, if we do more assumptions about hypothesis space (i.e. the bias is stronger) the algorithm can classify more unseen examples, but the correctness of classification strongly depends on the correctness of the inductive bias.

Class discussions on the homework:
2.4. b)



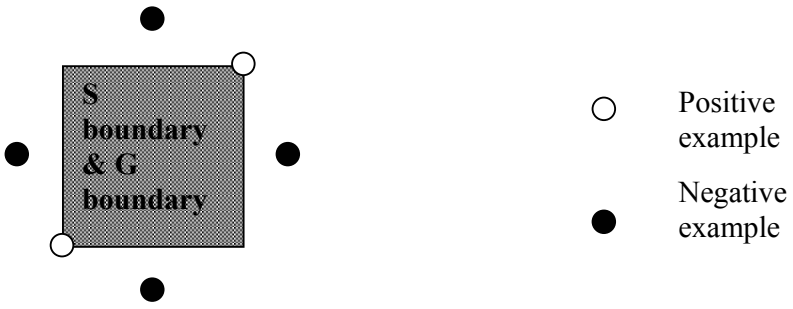
G: 2≤x≤8, 2≤y≤5
And 3≤x≤8 2≤y≤7

When we are performing Candidate-Elimination algorithm, we add the hypothesis h to G only if some member of S is more specific than h. So, when S-boundary contains only one hypothesis, G boundary has necessarily include S boundary, that's why the following answer is not correct:

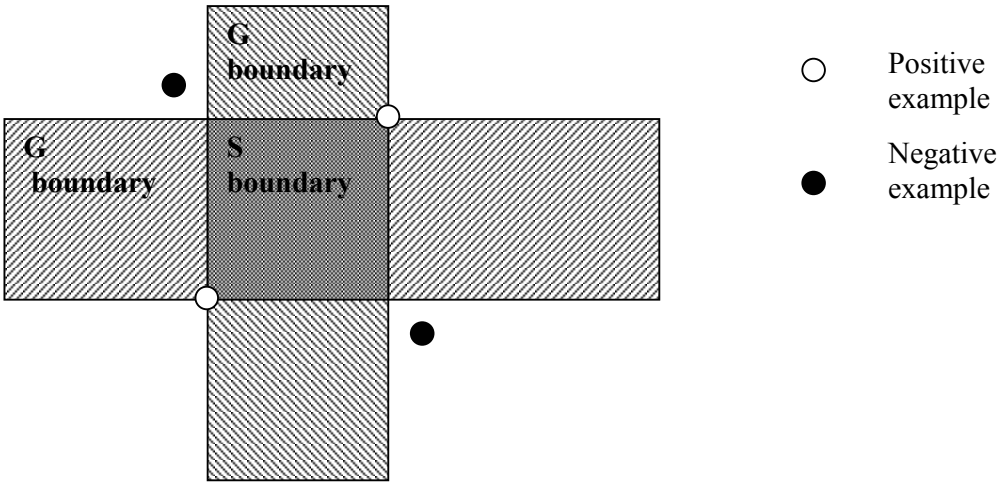


2≤x≤3 2≤y≤5
and 3≤x≤8 2≤y≤7

2.4.d) The smallest number of training examples is 6 that is 2 positive and 4 negative.



We cannot take less negative instances, because otherwise we will receive G-boundary much bigger than S-boundary. But for the target concept to be exactly learned S and G boundary sets have to converge to a single, identical, hypothesis that is for our example a single rectangle.



Actually, even if we take 4 negative examples, but accommodate them in the corners, the resulted version space won't converge to a single rectangle.

Decision Tree Learning

- Framework
  - Classification Learning
  - Definition: Decision tree learning is a method for approximating discrete-valued target functions, in which the learned function is represented by a decision tree. One of the most popular inductive inference algorithms that has been successfully applied to a broad range of tasks from learning to diagnose medical cases to learning access credit risks of loan applicants.
- Algorithm for learning decision tree
  - Entropy: a measure commonly used in information theory that characterizes the (im)purity of an arbitrary collection of examples. It will be discussed more later.
  - Inductive Bias (Occam's Razor): the inductive bias is the set of assumptions that together with training data deductively justify the classifications assigned by the learner to future instances.
- Evaluation
  - Cross Validation: One of the methods used for comparison of learning algorithms.

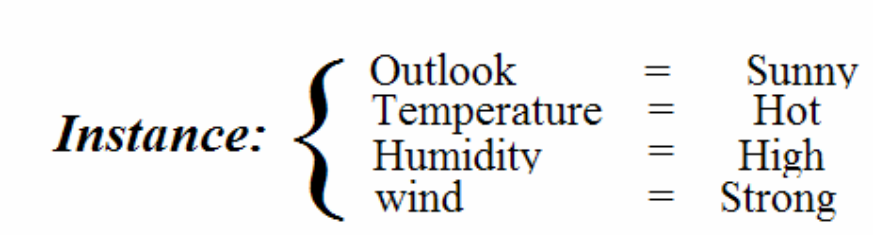
- Overfitting
  - Given a hypothesis space H, a hypothesis $h \varepsilon H$ is said to overfit the training data if there exists some alternative hypothesis $h' \varepsilon H$, such that h has smaller error over h', but h' has a smaller error than h over the entire distribution of instances. Overfitting is a significant practical difficulty for decision tree learning and many other learning methods.
  - Post Pruning: one of the approaches to avoid overfitting in decision tree learning.

- Topics:
  - k-ary attribute values
  - Real attribute values

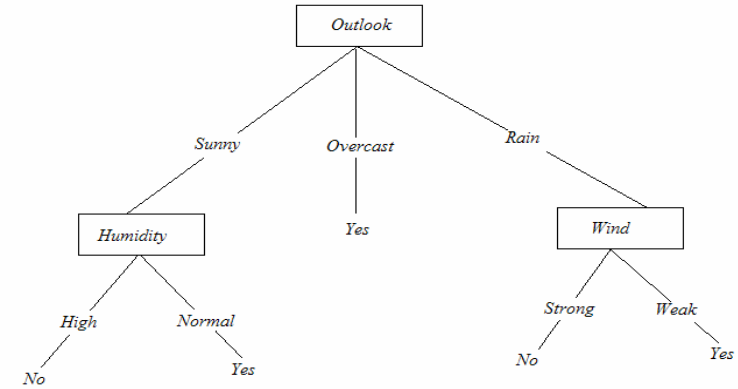
- Other splitting criteria
- Atrribute cost
- Missing values
- …

Decision Tree Representation

Decision tree classify instances by sorting them down the tree from the root to some leaf node, which provides the classification of the instance. Each node in the tree specifies a test of some *attribute* of the instance, and each branch descending form that node corresponds to one of the possible values for this attribute. In summary i*nternal nodes* test value of feature x_j and branch according to result of test and *leaf nodes* specify class *h(x)*. An instance is classified by starting at the root node of the tree, testing the attribute specified by this node, then moving down the tree branch corresponding to the value of the attribute in the given example. This process is then represented for the subtree rooted at the new node.



classified as "No"



In the figure above you can see a typical learned decision tree. In general, decision trees represent a disjunction of conjunctions of constraints on the attribute values of instances. Each path from the tree root to a leaf corresponds to a conjunction of attribute test and the tree itself to a disjunction of conjunctions. For example, the decision tree shown in the figure above corresponds to the expression

(Outlook = Sunny and Humidity = Normal)
or (Outlook = Overcast)
or (Outlook = Rain and Wind = Weak)

Decision Trees

Decision Tree Hypothesis Space is …

- Variable Size: it can represent any Boolean function

- Deterministic
- Discrete and Continuous Parameters

Learning algorithm is …

- Constructive Search: Build tree by adding nodes.
- Eager

- Batch: we take the algorithm and keep feeding the data
  - we update based on 1-1 in online algorithms which is incremental learning
  - or we update on the whole batch

Using Decision Trees

Decision tree learning is generally best suited to problems with the following characteristics:

- *Instances are represented by attribute-value pairs.* The easiest situation for decision tree learning is when each attribute takes on a small number of disjoint possible values. However extensions to the basic algorithm allow handling real-valued attributes as well. "Bar = Yes", "Size = Large", "Type = French", "Temp = 82.6", …(Boolean, discrete, nominal, continuous)
- *The target function has discrete output values.* Our focus is on such target functions. A more substantial extension allows learning target function with real-valued outputs, though the application of decision trees in this setting is less common.
- Can handle:
  - *Disjunctive descriptions may be required.*
  - *The training data may contain errors.* Decision tree learning methods are robust to errors, both errors in classifications of the training examples and errors in the attribute values that describe these examples.
  - *The training data may contain missing attribute values.* Decision tree methods can be used even when some training examples have unknown values.

Many practical problems have been found to fit these characteristics. Decision tree learning has therefore been applied to problems such as learning to classify medical patients by their disease, equipment malfunctions by their cause, and loan applicants by their likelihood of defaulting on payment (credit risk analysis). Such problems, in which the task is to classify examples into one of a discrete set of possible categories, are often referred to as *classification problems.*

Important Fact:

⇒ **<u>For every boolean function there is a decision tree representing it</u>**

Proof:

It has been proven in complexity theory that any boolean function can be transformed into disjunctive normal form (a formula is in disjunctive normal form if it is disjunction of clauses which are in turn conjunctions of literals or their negations). Then, the transformed formula is the representation of a decision tree because every decision tree's path from

the root to the leaf is a conjunction of literals (positive or negative), and the disjunction of all such paths will be the initial formula.

Constructing Decision Trees:

Most algorithms , such as ID3, that have been applied in inducing decision trees from training data use a greedy, top down method.

ID3 operates as the following:

1) determines a promising candidate for the root (a top down approach).
2) Use a Statistical test
3) Choose the best attribute for the root
4) Create a descendent for the root for each of its attribute values
5) Partitioned training data based on the descendent nodes
6) Repeat steps 1 - 5 for all the training data remaining and associated with the descendent nodes

Features of the algorithm:
-Greedy
-Never back tracks
-Deterministic (i.e. there is no randomness involved)

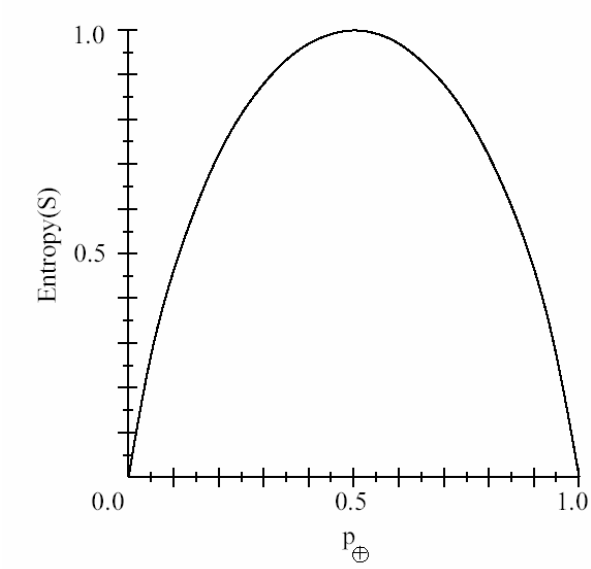
Criteria for choosing an attribute for a node:

-Information Gain

- A statistical property that "… measures how well a given attribute separates the training examples according to their target classification"
- This is used by ID3
- Can be precisely defined through entropy

One of the most important concepts in machine learning is Entropy which:

- "measures homogeneity of examples"
- characterizes the impurity of an arbitrary collection of examples
- is 0 if all members of the collection belong to the same class (the collection is homogeneous) and 1 when the collection has an equal number of positive and negative examples
- is considered as an indicator of the minimum number of bits of information needed to encode the classification of an arbitrary member of a collection in the context of information theory
- is "a measure of the expected encoding length measured in bits. In other words, it is the expected number of bits needed to obtain full information:
    - Entropy (S) = $\Sigma_{i = 1 \text{ to } c} (-p_i \log_2 p_i)$ where p_i is the proportion of collection S belonging to class i
- Entropy as a function:



where

- S is a sample of training examples
- $p_{\oplus}$ is the proportion of positive examples in S
- $p_{\ominus}$ is the proportion of negative examples in S
- Entropy measures the impurity of S

$$Entropy(S) \equiv -p_{\oplus} \log_2 p_{\oplus} - p_{\ominus} \log_2 p_{\ominus}$$

References:

- "Machine Learning," Tom M. Mitchell published 1997 by McGraw-Hill
- http://cgm.cs.mcgill.ca/~soss/cs644/projects/jacob/interpretations.html
- Notes taken from Dr. Schulte's class on the aforementioned dates.