Aspects of Machine Learning Approach

Machine learning approaches

- Rule-based models
- AI models

Rule based models

- To program a computer to play chess, we can give it the rules of the game, some basic strategic information along the lines of “it's good to occupy the center,” and data regarding typical chess openings
- We can also define rules for evaluating every position on the board for every move. The machine can then calculate all possible moves from a given position and select those that lead to some advantage
- It can also calculate all possible responses by the opponent. This brute force approach evaluated billions of possible moves, but it was successful in defeating expert chess players

Artificial Intelligence models

- Machine learning algorithms, on the other hand, can develop strategically best lines of play based upon the experiences of learning from many other games played previously
- What most machine learning programs have in common is the need for large amounts of known examples as the data with which to train them
- The program uses data to create an internal model or representation that encapsulates the properties of those data. When it receives an unknown sample, it determines whether it fits that model to “train” them
- A neural network is one of the best approaches toward generating such internal models, but in a certain sense, every machine learning algorithm does this. For example, a program might be designed to distinguish between cats and dogs

Artificial Intelligence models

- The training set consists of many examples of dogs and cats. Essentially, the program starts by making guesses, and the guesses are refined through multiple known examples. At some point, the program can classify an unknown example correctly as a dog or a cat. It has generated an internal model of “cat-ness” and “dog-ness.” The dataset composed of known examples is known as the “training set.”
- Before using this model on unlabeled, unknown examples, one must test the program to see how well it works. For this, one also needs a set of known examples that are NOT used for training, but rather to test whether the machine has learned to identify examples that it has never seen before. The algorithm is analyzed for its ability to characterize this “test set.”

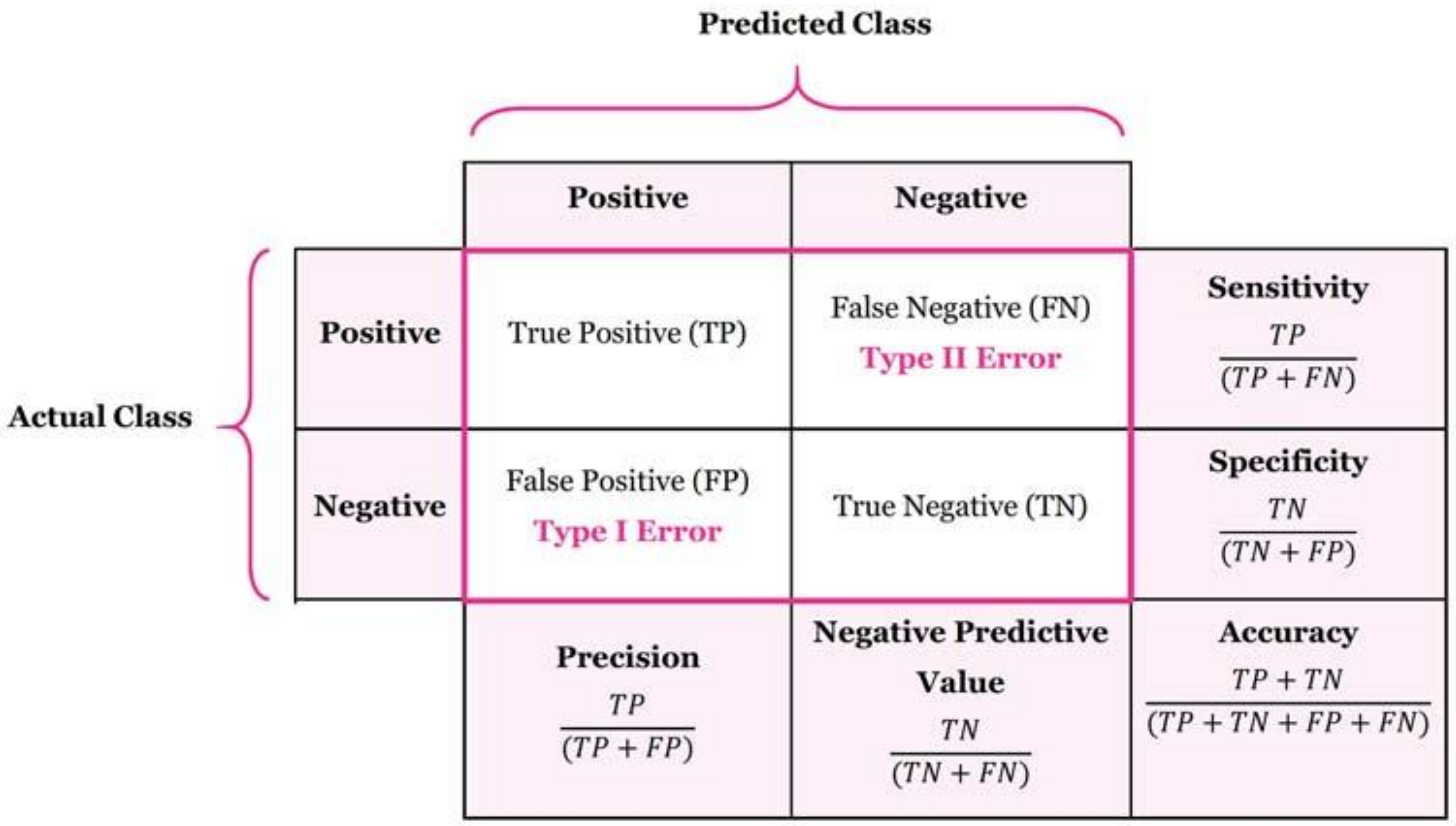
Metrics used for model evaluation

- Accuracy (it is an incomplete measure of model performance when the dataset is imbalanced. For example, in cancer diagnosis where out of say 100000 suspects only 500 people have cancer)
- There are four possible results for an output from the model: true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN)

Confusion matrix

- A **confusion matrix** is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known

Confusion matrix



		Predicted Class		
		Positive	**Negative**	
Actual Class	**Positive**	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

Metrics used for classification

- Precision (positive predictive value)- positives out of predicted yes/positive
- Sensitivity (recall)- positives out of actual yes/positive
- Specificity- negatives out of actual negatives
- Negative predictive value- negative out of predicted negative
- Accuracy- True cases out of total cases
- F1- score- Harmonic mean of precision and recall

Calculation of classification metrics in case of multiclass classification

Machine learning models

- Machine learning models are akin to mathematical functions -- they take a request in the form of input data, make a prediction on that input data, and then serve a response
- A model is a distilled representation of what a machine learning system has learned
- The output from model training may be used for inference, which means making predictions on new data
- In supervised and unsupervised machine learning, the model describes the signal in the noise or the pattern detected from the training data
- In reinforcement learning, the model describes the best possible course of action given a specific situation

Machine learning models

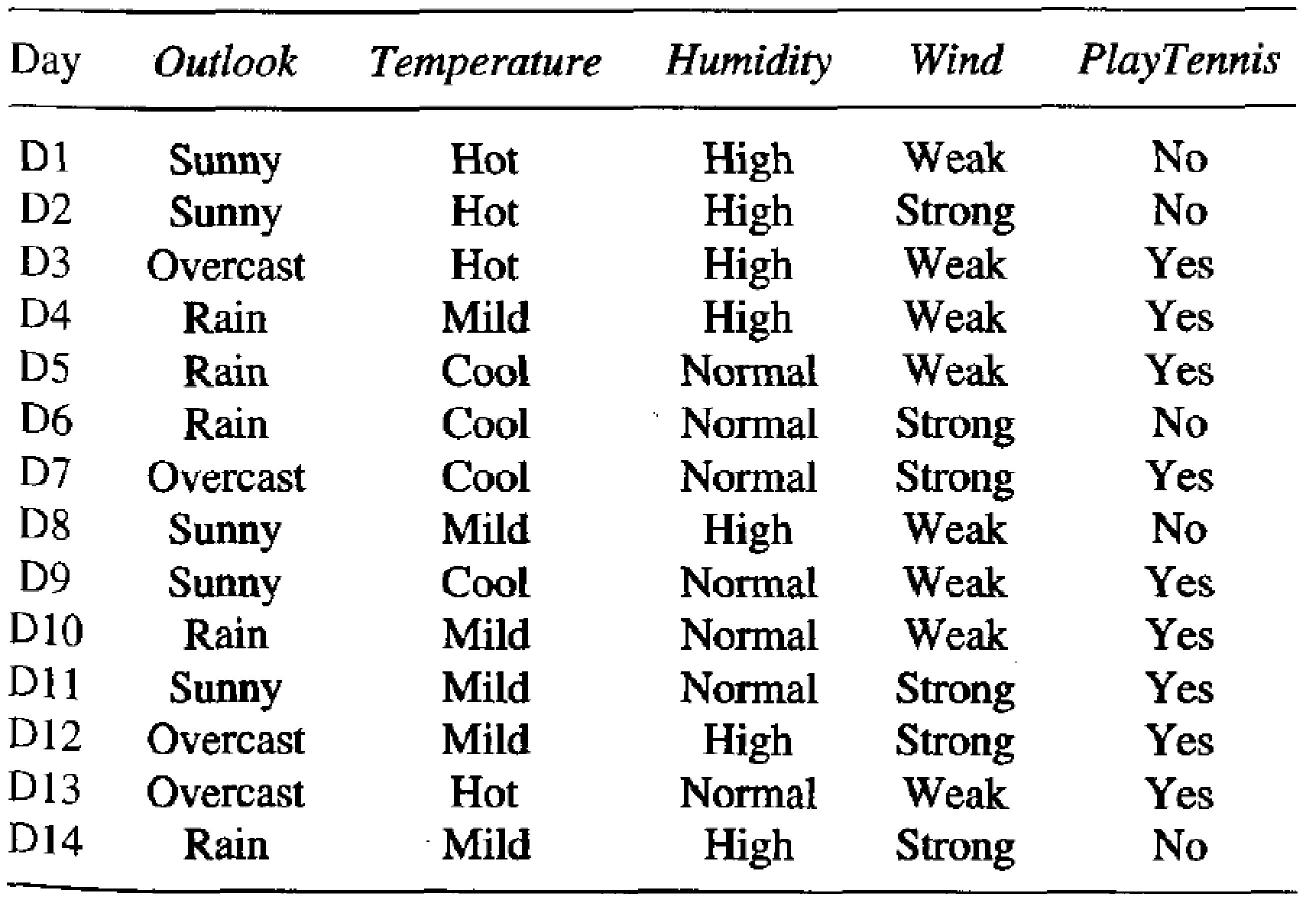
- The final set of trainable parameters (the information the model contains) depends on the specific type of model
- In deep neural networks, a model is the final state of the trained weights of the network. In regression it contains coefficients, and in decision trees it contains the split locations

Decision tree

Introduction

- Decision tree learning is one of the most widely used and practical methods for inductive inference
- It is a method for approximating discrete-valued functions that is robust to noisy data
- Learned function is represented by a decision tree
- Learned trees can also be re-represented as sets of if-then rules to improve human readability
- It is among the most popular of inductive inference algorithms
- Successfully applied to a broad range of tasks from learning to diagnose medical cases to learning to assess credit risk of loan applicants

Information System



Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

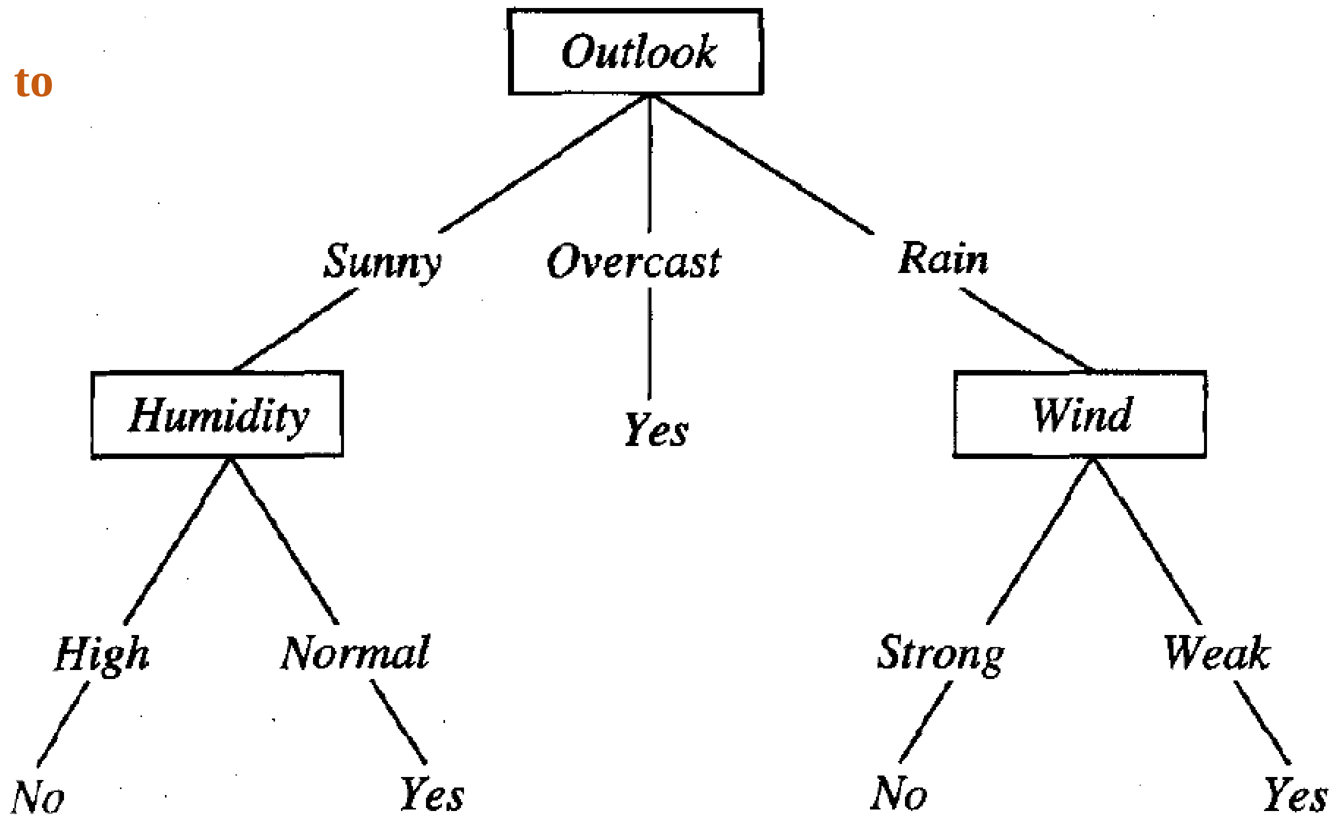
Decision tree representation

- 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 from that node corresponds to one of the possible values for this attribute
- 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 repeated for the subtree rooted at the new node

A typical learned decision tree for the concept PlayTennis

This tree classifies Saturday mornings according to whether or not they are suitable for playing tennis.
For example, the instance
(Outlook = Sunny, Temperature = Hot, Humidity = High, Wind = Strong)

This tree illustrate the ID3 learning algorithm are adapted from (Quinlan ***1986)***



- 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 tests, and the tree itself to a disjunction of these conjunctions

$$\begin{aligned} & (Outlook = Sunny \ \wedge \ Humidity = Normal) \\ \vee \quad & (Outlook = Overcast) \\ \vee \quad & (Outlook = Rain \ \wedge \ Wind = Weak) \end{aligned}$$

Appropriate problems for decision tree learning

- **Instances are represented by attribute-value pairs**
- **The target function has discrete output values**
- **Disjunctive descriptions may be required.** As noted above, decision trees naturally represent disjunctive expressions
- **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 (e.g., if the **Humidity** of the day is known for only some of the training examples)

The basic decision tree learning algorithm

- Most algorithms that have been developed for learning decision trees are variations on a core algorithm that employs a top-down, greedy search through the space of possible decision trees. This approach is exemplified by the ID3 algorithm (Quinlan 1986) and its successor C4.5 (Quinlan 1993)
- ID3 algorithm, stands for Iterative Dichotomiser 3, learns decision trees by constructing them top down, beginning with the question "which attribute should be tested at the root of the tree?
- To answer this question, each instance attribute is evaluated using a statistical test to determine how well it alone classifies the training examples
- The best attribute is selected and used as the test at the root node of the tree. A descendant of the root node is then created for each possible value of this attribute, and the training examples are sorted to the appropriate descendant node
- The entire process is then repeated using the training examples associated with each descendant node to select the best attribute to test at that point in the tree
- This forms a greedy search for an acceptable decision tree, in which the algorithm never backtracks to reconsider earlier choices

Which Attribute Is the Best Classifier?

- What is a good quantitative measure of the worth of an attribute?
- ID3 uses **information gain** measure to select among the candidate attributes at each step while growing the tree
- To define information gain precisely, we begin by defining a measure commonly used in information theory, called ***entropy,*** that characterizes the (im)purity of an arbitrary collection of examples
- Given a collection S, containing positive and negative examples of some target concept, the entropy of S relative to this boolean classification is

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

where

$p_{\oplus}$, is the proportion of positive examples in S

$p_{\ominus}$, is the proportion of negative examples in S.

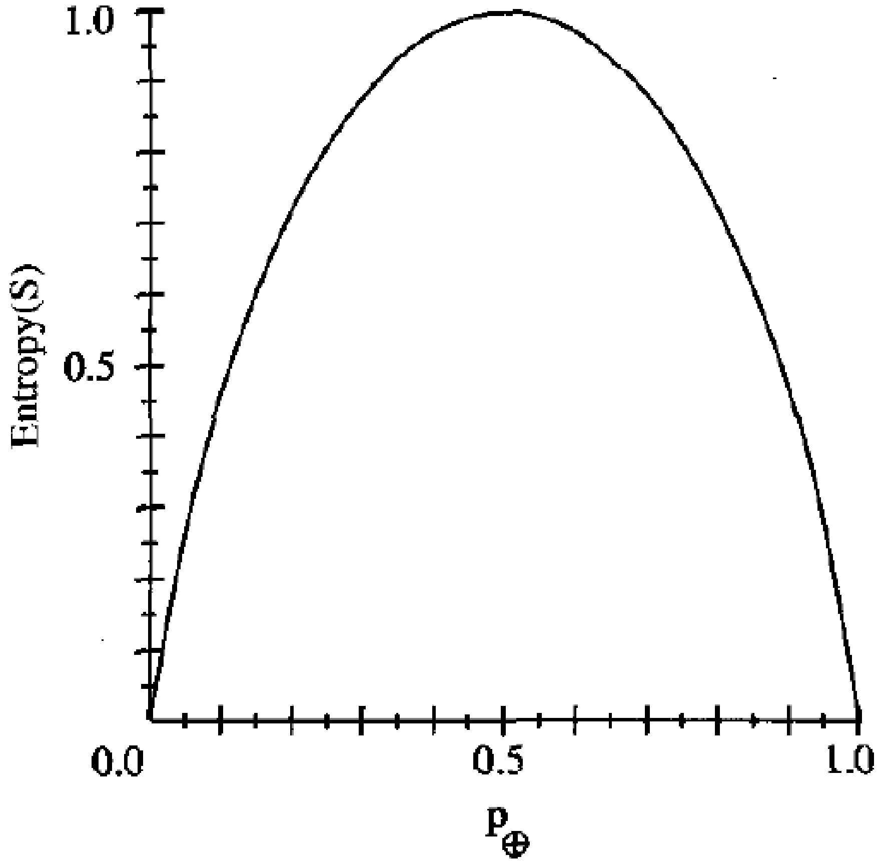
- In all calculations involving entropy we define 0 log 0 to be 0

Illustration of entropy

- Suppose S is a collection of 14 examples of some Boolean concept, including 9 positive and 5 negative examples (we adopt the notation [9+, 5-] to summarize such a sample of data). Then the entropy of S relative to this boolean classification is

$$\begin{aligned} Entropy([9+, 5-]) &= -(9/14)\log_2(9/14) - (5/14)\log_2(5/14) \\ &= 0.940 \end{aligned}$$

- Notice that the entropy is 0 if all members of S belong to the same class. For example, if all members are positive ($p_\oplus$ = 1), then $p_\ominus$, is 0, and Entropy(S) = -1 . $\log_2$ (1) - 0 . $\log_2$ 0 = -1 . 0 - 0 . $\log_2$ 0 = 0
- Note the entropy is 1 when the collection contains an equal number of positive and negative examples
- If the collection contains unequal numbers of positive and negative examples, the entropy is between **0** and 1
- In general, $Entropy(S) \equiv \sum_{i=1}^{c} -p_i \log_2 p_i$


1.0
0.5
Entropy(S)
0.0
0.5
1.0
$p_\oplus$

Information gain measures the expected reduction in entropy

- It is simply the expected reduction in entropy caused by partitioning the examples according to a particular attribute
- The information gain, ***Gain(S, A)*** of ***an*** attribute **A,** relative to a collection of examples ***S,*** is defined as

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

- Note the first term in above equation is just the entropy of the original collection ***S,*** and the second term is the expected value of the entropy after ***S*** is partitioned using attribute ***A***
- ***Gain(S, A)*** is therefore the expected reduction in entropy caused by knowing the value of attribute A. Put another way, ***Gain(S, A)*** is the information provided about the ***target function value,*** given the value of some other attribute ***A***

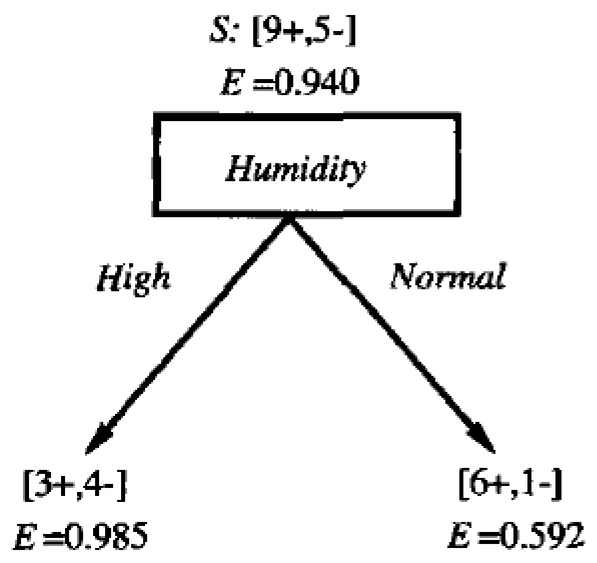
Calculating Information Gain

- Suppose ***S*** is a collection of training-example days described by attributes including ***Wind,*** which can have the values ***Weak*** or ***Strong.*** As before, assume ***S*** is a collection containing ***14*** examples, [9+, 5-1. Of these 14 examples, suppose 6 of the positive and 2 of the negative examples have ***Wind = Weak,*** and the remainder have ***Wind = Strong.*** The information gain due to sorting the original ***14*** examples by the attribute ***Wind*** may then be calculated as

$$
\begin{aligned}
Values(Wind) &= Weak, Strong \\
S &= [9+, 5-] \\
S_{Weak} &\leftarrow [6+, 2-] \\
S_{Strong} &\leftarrow [3+, 3-] \\
Gain(S, Wind) &= Entropy(S) - \sum_{v \in \{Weak, Strong\}} \frac{|S_v|}{|S|} Entropy(S_v) \\
&= Entropy(S) - (8/14)Entropy(S_{Weak}) \\
&\quad - (6/14)Entropy(S_{Strong}) \\
&= 0.940 - (8/14)0.811 - (6/14)1.00 \\
&= 0.048
\end{aligned}
$$

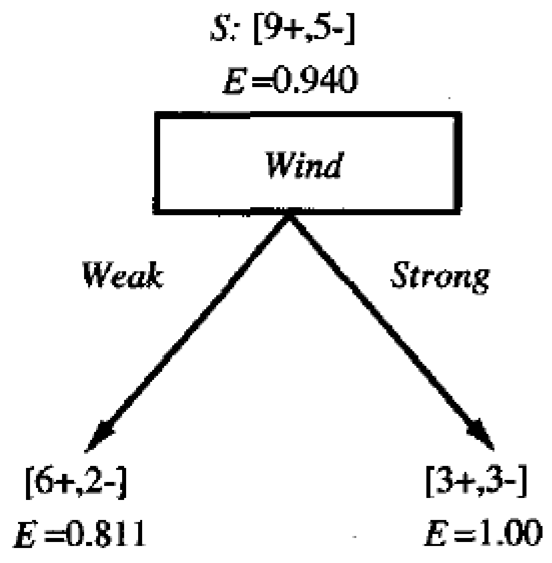
Which attribute is best classifier

ID3 selects the attribute value with highest Information Gain



Gain (S, Humidity)

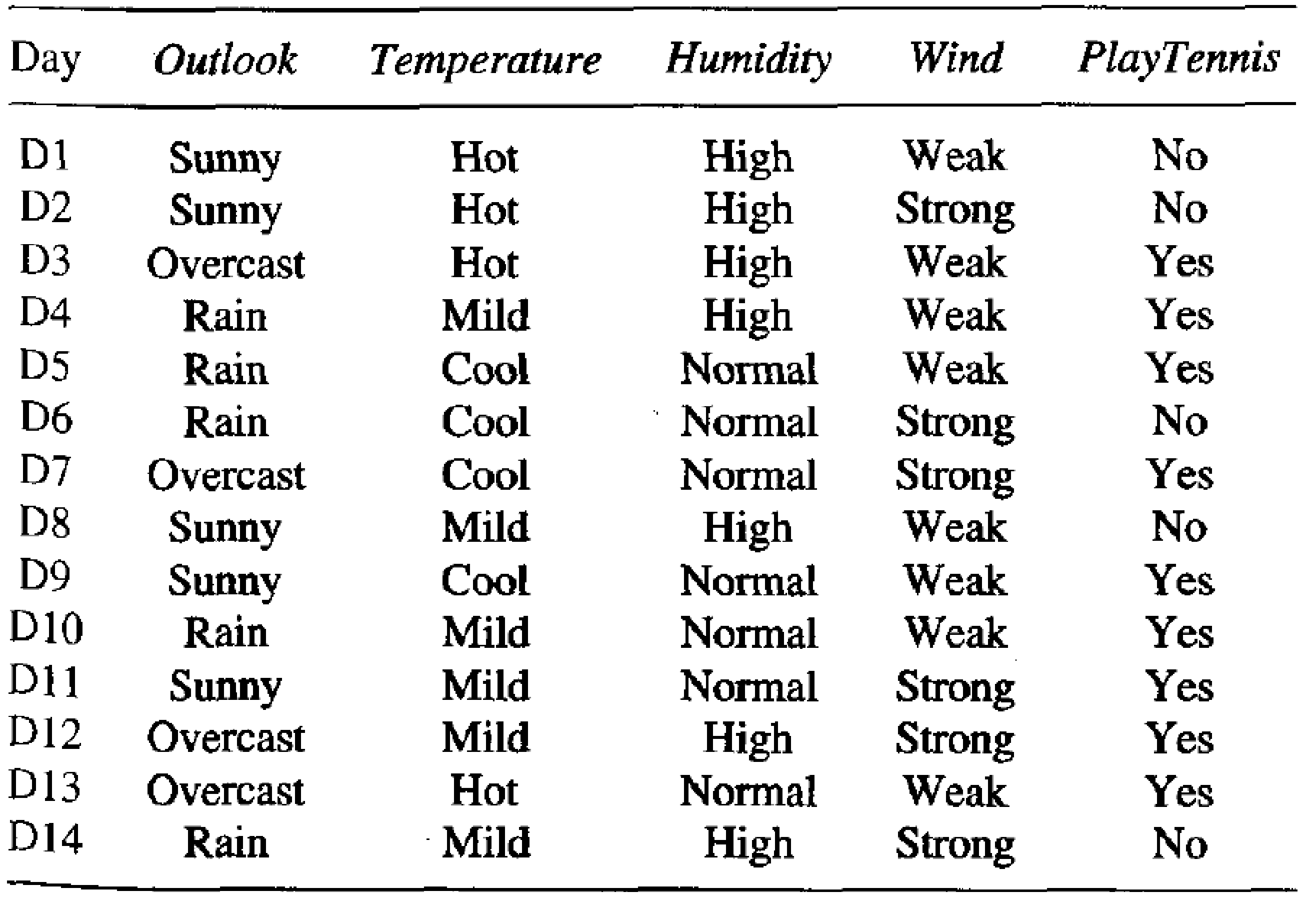
= .940 - (7/14).985 - (7/14).592
= .151



Gain (S, Wind)

= .940 - (8/14).811 - (6/14)1.0
= .048

Information System



Day	*Outlook*	*Temperature*	*Humidity*	*Wind*	*PlayTennis*
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

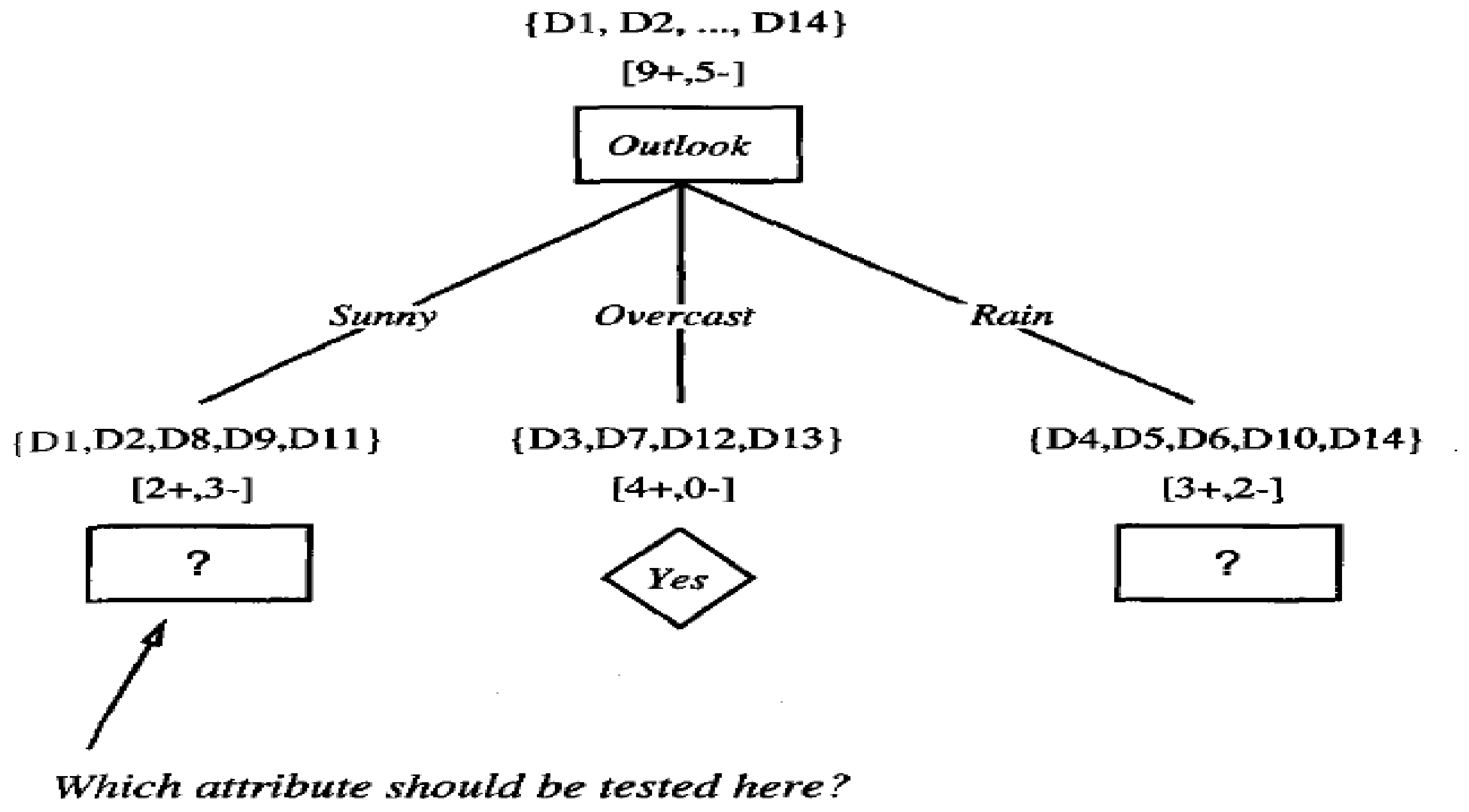
An Illustrative Example

- ID3 determines the information gain for each candidate attribute (i.e., ***Outlook, Temperature, Humidity,*** and ***Wind),*** then selects the one with highest information gain

$$
\begin{aligned}
Gain(S, Outlook) &= 0.246 \\
Gain(S, Humidity) &= 0.151 \\
Gain(S, Wind) &= 0.048 \\
Gain(S, Temperature) &= 0.029
\end{aligned}
$$

- ***Outlook*** is selected as the decision attribute for the root node, and branches are created below the root for each of its possible values (i.e., ***Sunny, Overcast,*** and ***Rain)***
- Note that every example for which ***Outlook = Overcast*** is also a positive example of ***PlayTennis.*** Therefore, this node of the tree becomes a leaf node with the classification ***PlayTennis = Yes***
- In contrast, the descendants corresponding to ***Outlook = Sunny*** and ***Outlook = Rain*** still have nonzero entropy, and the decision tree will be further elaborated below these nodes

- The process of selecting a new attribute and partitioning the training examples is now repeated for each nonterminal descendant node, this time using only the training examples associated with that node
- Attributes that have been incorporated higher in the tree are excluded, so that any given attribute can appear at most once along any path through the tree
- This process continues for each new leaf node until either of two conditions is met: (1) every attribute has already been included along this path through the tree, or (2) the training examples associated with this leaf node all have the same target attribute value (i.e., their entropy is zero)



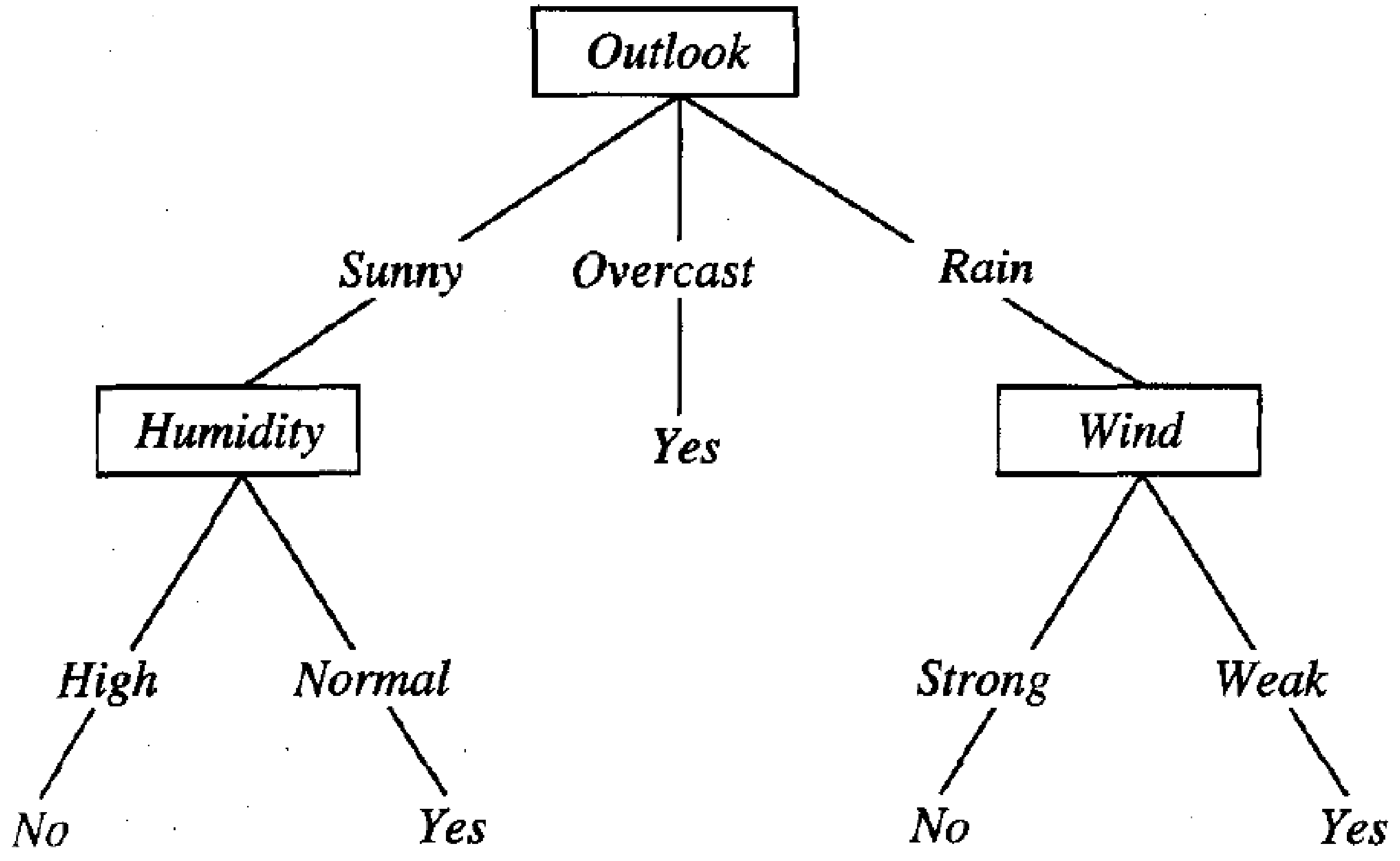
S_{sunny} = {D1,D2,D8,D9,D11}

Gain (S_{sunny}, *Humidity*) = .970 - (3/5) 0.0 - (2/5) 0.0 = .970

Gain (S_{sunny}, *Temperature*) = .970 - (2/5) 0.0 - (2/5) 1.0 - (1/5) 0.0 = .570

Gain (S_{sunny}, *Wind*) = .970 - (2/5) 1.0 - (3/5) .918 = .019

Final decision tree learned by ID3 from the 14 training examples



Confusion matrix
Example

	Predicted class POSITIVE (spam ✉)	**Predicted class NEGATIVE (normal 📧)**
Actual class POSITIVE (spam ✉)	TRUE POSITIVE (TP) ✉ ✉ **320**	FALSE NEGATIVE (FN) ✉ 📧 **43**
Actual class NEGATIVE (normal ✉)	FALSE POSITIVE (FP) ✉ ✉ **20**	TRUE NEGATIVE (TN) ✉ 📧 **538**

Sensitivity and Specificity

	Predicted class POSITIVE (spam ✉)	**Predicted class NEGATIVE (normal ✉)**	
Actual class POSITIVE (spam ✉)	TRUE POSITIVE (TP) ✉ ✉ 320	FALSE NEGATIVE (FN) ✉ ✉ 43	$Sensitivity = \frac{TP}{TP+FN} = \frac{320}{320+43} = 0.882$
Actual class NEGATIVE (normal ✉)	FALSE POSITIVE (FP) ✉ ✉ 20	TRUE NEGATIVE (TN) ✉ ✉ 538	$Specificity = \frac{TN}{FP+TN} = \frac{538}{20+538} = 0.964$

Recall, Precision and F-Measure

	Predicted class POSITIVE (spam ✉)	**Predicted class NEGATIVE (normal ✉)**	
Actual class POSITIVE (spam ✉)	TRUE POSITIVE (TP) ✉ ✉ **320**	FALSE NEGATIVE (FN) ✉ ✉ **43**	*Recall* $= \frac{TP}{TP+FN} = \frac{320}{320+43} = 0.882$
Actual class NEGATIVE (normal ✉)	FALSE POSITIVE (FP) ✉ ✉ **20**	TRUE NEGATIVE (TN) ✉ ✉ **538**	
	Precision $= \frac{TP}{TP+FP} = \frac{320}{320+20} = 0.941$		