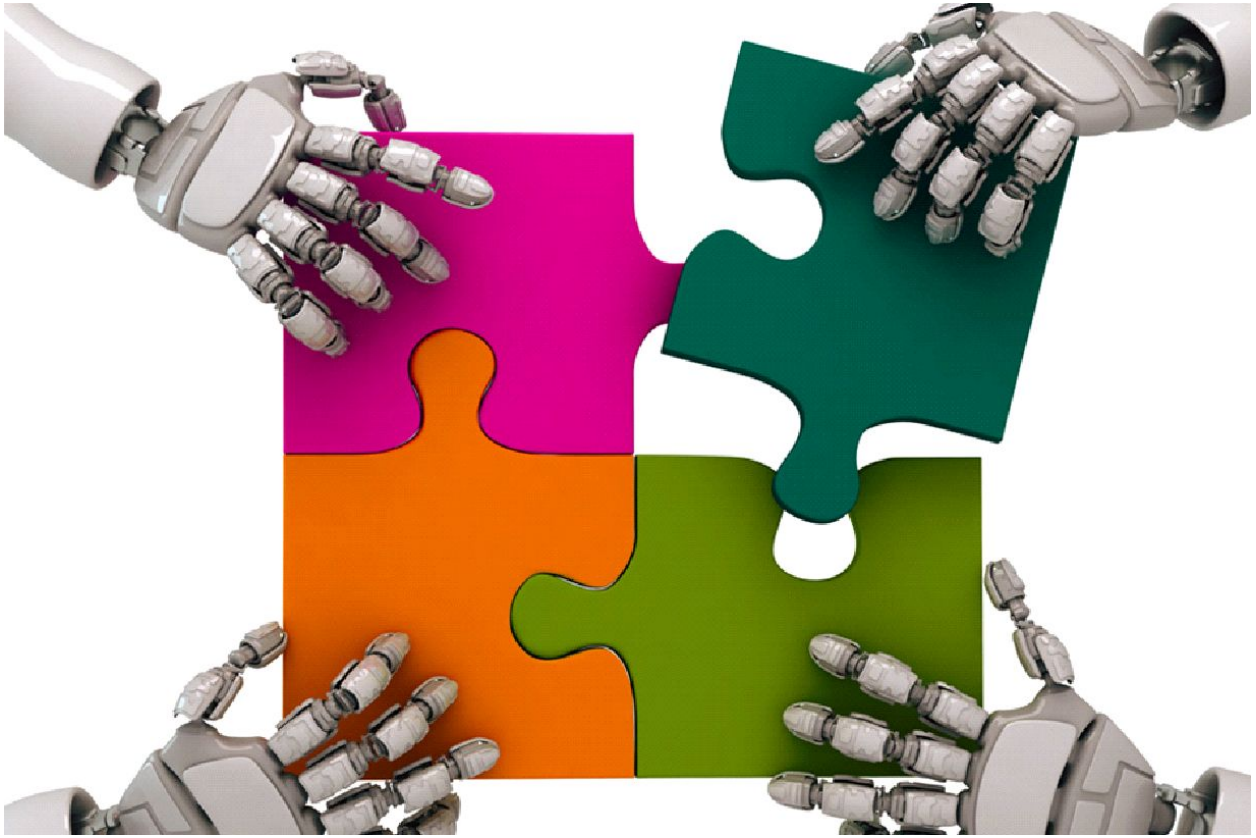
Artificial Intelligence
CC-721

Learning Deterministic Models



Supervised Learning

Supervised learning involves learning a function from a training set.

The function maps a variable *x* (which may be a vector) to a variable *y*

The **training set** is a set of known values of (*x*,*y*) pairs.

The variables in *x* are called the **predictors**

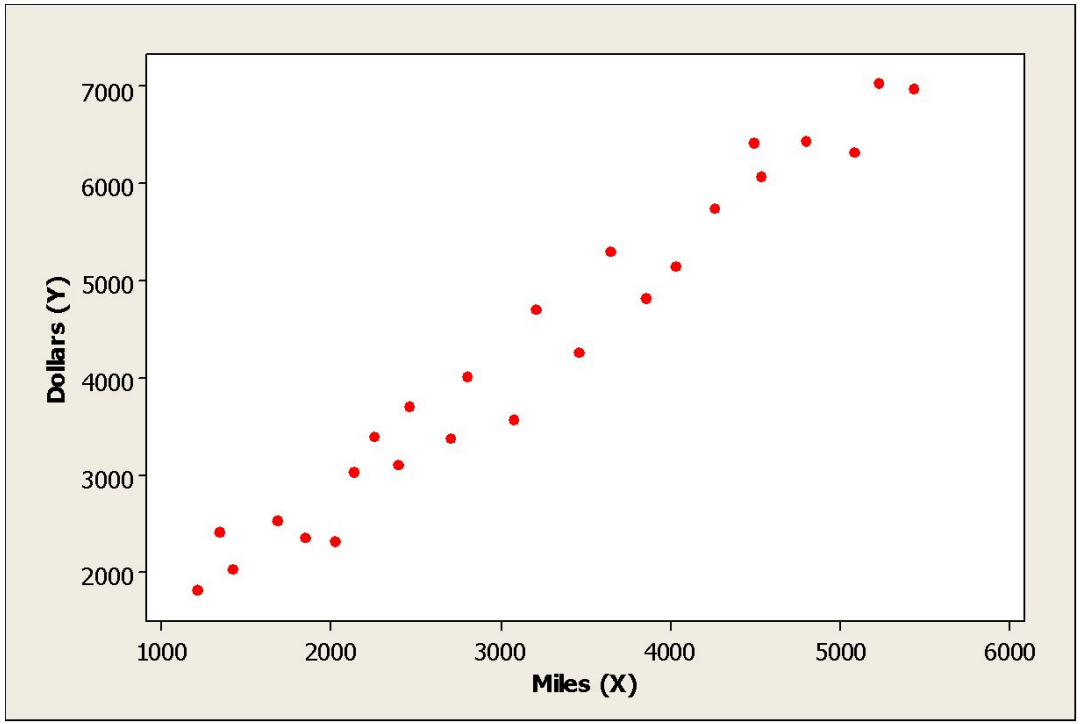
Variable *y* is called the **target**.

American Express suspected that charges on American Express cards increased with the number of miles traveled by the card holder. To investigate this matter, a research firm randomly selected 25 card holders and obtained the data shown in the following table.

Miles and Dollars for 25 American Express Card Holders

Passenger	Miles (X)	Dollars (Y)
1	1211	1802
2	1345	2405
3	1422	2005
4	1687	2511
5	1849	2332
6	2026	2305
7	2133	3016
8	2253	3385
9	2400	3090
10	2468	3694
11	2699	3371
12	2806	3998
13	3082	3555
14	3209	4692
15	3466	4244
16	3643	5298
17	3852	4801
18	4033	5147
19	4267	5738
20	4498	6420
21	4533	6059
22	4804	6426
23	5090	6321
24	5233	7026
25	5439	6964

The following is a scatterplot of the data in the table:



It looks like there is an approximate linear relationship between Dollars and Miles.

Linear regression endeavors to find such a linear relationship.

In **simple linear regression**, we assume we have an independent random variable X and a dependent random variable Y such that

$$y = \beta_0 + \beta_1 x + \varepsilon_x,$$

where ε_x is a random variable, which depends on the value x of X, with the following properties:

1) For every value x of X, ε_x is normally distributed with 0 mean
2) For every value x of X, ε_x has the same standard deviation σ
3) The random variables ε_x for all x are mutually independent

Note that these assumptions entail that the expected value of Y given a value x of X is given by

$$E(Y \mid X = x) = \beta_0 + \beta_1 x.$$

The idea is that the expected value of Y is a deterministic linear function of x.

However, the actual value y of Y is not uniquely determined by the value of X because of a random error term ε_x.

To estimate the values of β_0 and β_1, we find the values of b_0 and b_1 that minimize the **Mean Square Error (MSE)**, which is

$$\frac{\sum_{i=1}^{n}[y_i - (b_0 + b_1 x_i)]^2}{n}$$

where n is the size of the sample, and x_i and y_i are the values of X and Y for the ith item.

In the case of the American Express example we obtain the following:

$$y = b_0 + b_1 x$$
$$= 274.8 + 1.26x.$$

A statistical package usually provides the following information when doing linear regression:

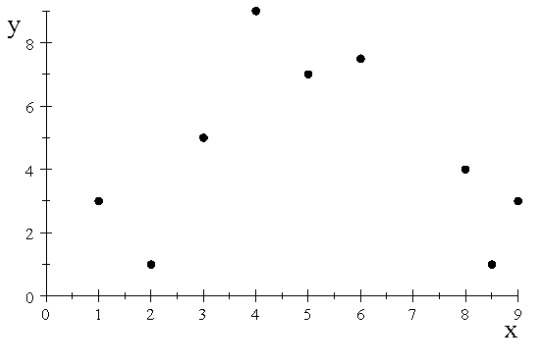
Predictor	Coefficient	SE Coefficient	T	P
Constant	$b_0 = 274.8$	170.3	1.61	.12
x	$b_1 = 1.255$	.0497	25.25	0

The far right value is the p-value.

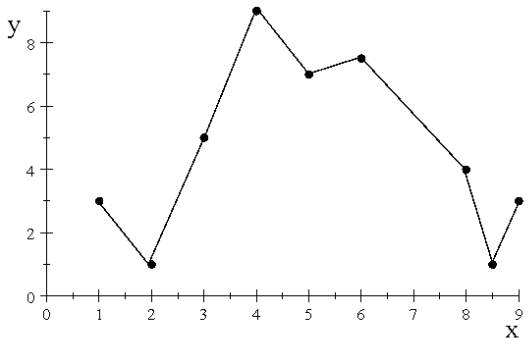
So, we can be highly confident in the linear relationship between *Y* and *X*.

However, we can't be so confident in the constant.

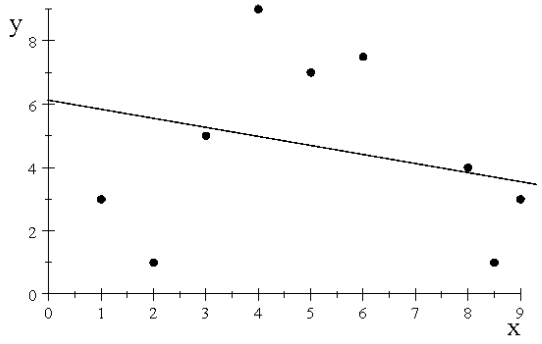
Over Fitting the Data



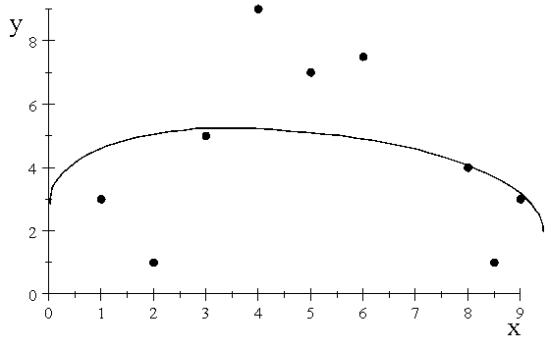
(a) 9 data points



(b) conect the dots



(c) linear regression



(d) quadratic regression

(b), (c), and (d) show 3 models we could learn from the data in (a).

(b) predicts the data in the sample perfectly.

However, (b) might over fit the data and perform poorly on out-of-sample data.

In order to avoid overfitting, we use techniques that evaluate the fit of the model to the underlying system.

In the **test set method**, we partition the data into a **training set** and a **test set**.

We then learn a model from the training set and estimate the performance using the test set.

Allocating 70% of the data previous figure to the training set, and 30% to the test set, we obtain the results in the first row of the following table (MSE denotes mean square error):

Table 5.3 MSE for Several Evaluation Methods and Several Model Learning Techniques

Method	Connect the Dots	Linear Regression	Quadratic Regression
Test Set	2.20	2.40	0.90
LOOCV	3.32	2.12	0.96
3-FoldCV	2.93	2.05	1.11

LOOCV and 3-Fold CV are other techniques discussed next.

All 3 techniques found quadratic regression to be the best model.

In **Leave-One-Out Cross Validation (LOOCV)**, we remove one of the *n* data items and train using the remaining *n*-1 data items.

We then compute the error for the removed item relative to the model learned.

After this process is repeated for all *n* data items, the MSE is computed.

In ***k*-Fold Cross Validation**, we divide the data into *k* partitions of the same size.

For each partition *j* we train using the data items in the remaining *k*-1 partitions, and we compute the error for each data item in partition *j* relative to the model learned.

After this process is repeated for all *k* partitions, the MSE for all data items is computed.

Learning a Decision Tree

Information Theory

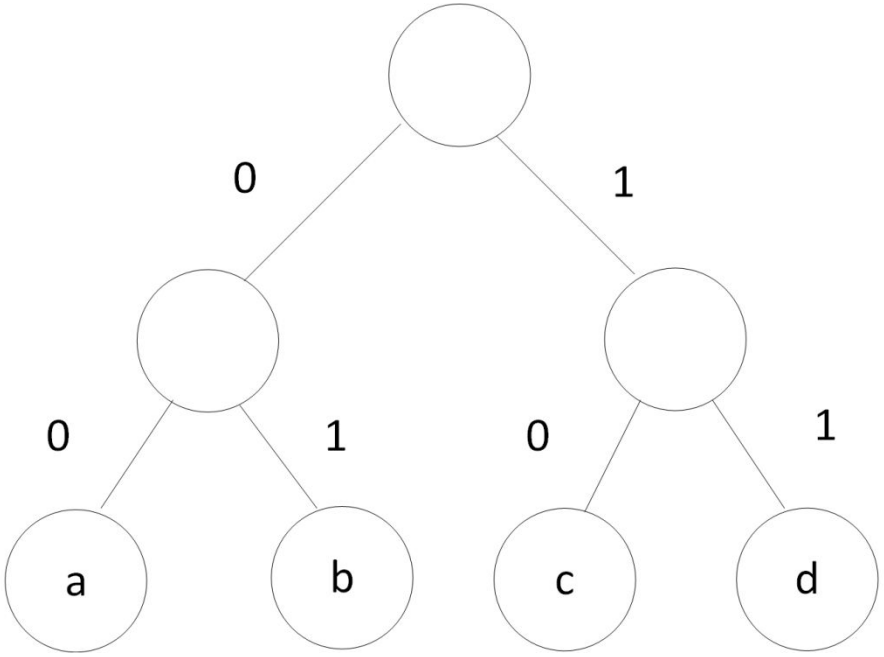
Huffman Code

Suppose we are going to transmit to a colleague a sequence as follows:

aaabbbccaaddcccc

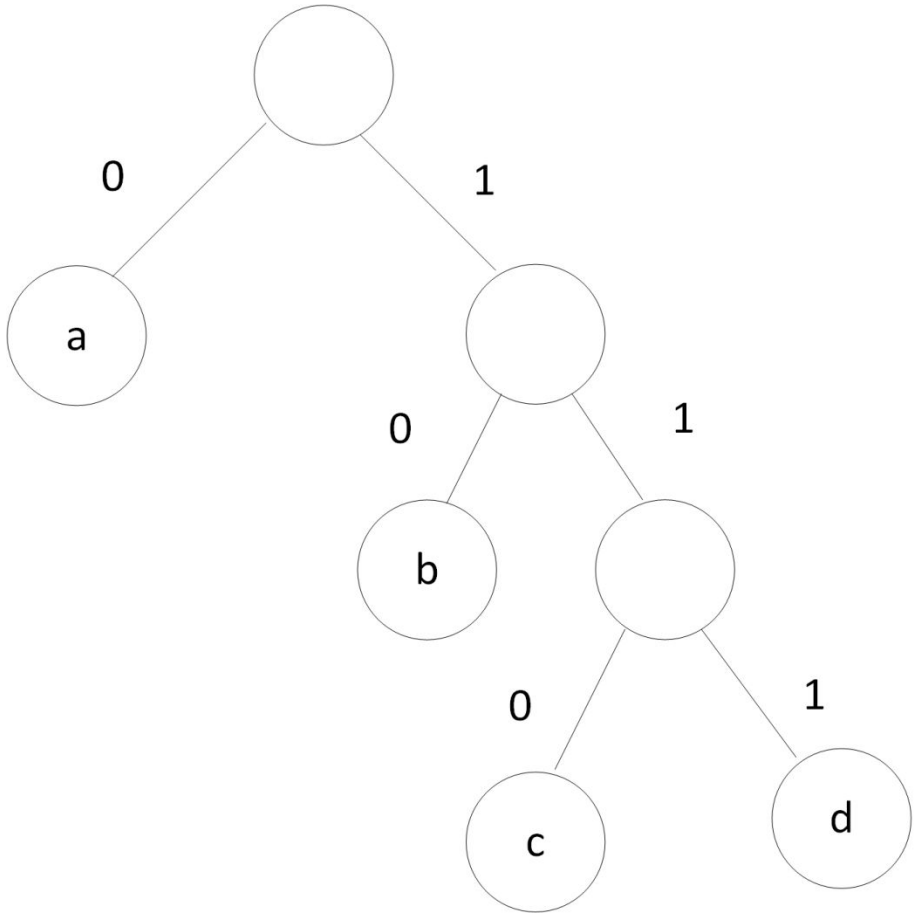
Fixed length binary code:

a: 00 b: 01 c: 10 d: 11



variable length binary code:

a: 0 b: 10 c: 110 d: 111



The variable length code has more total bits than the fixed length code.

If all characters are transmitted the same number of times, the fixed length code is better.

However, if “a” is transmitted a large fraction of the time, the variable length code is better.

Huffman's algorithm finds the optimal binary code.

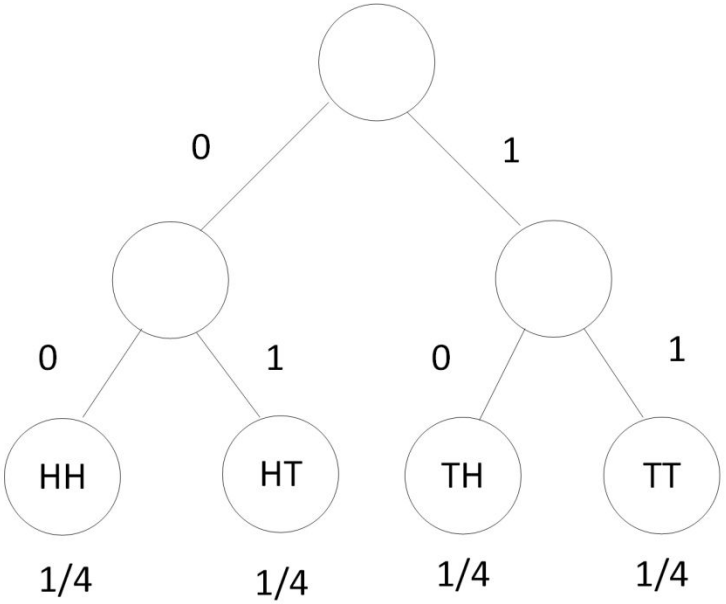
Character	Frequency	Fixed Length	Huffman
a	16	000	00
b	5	001	1110
c	12	010	110
d	17	011	01
e	10	100	1111
f	25	101	10

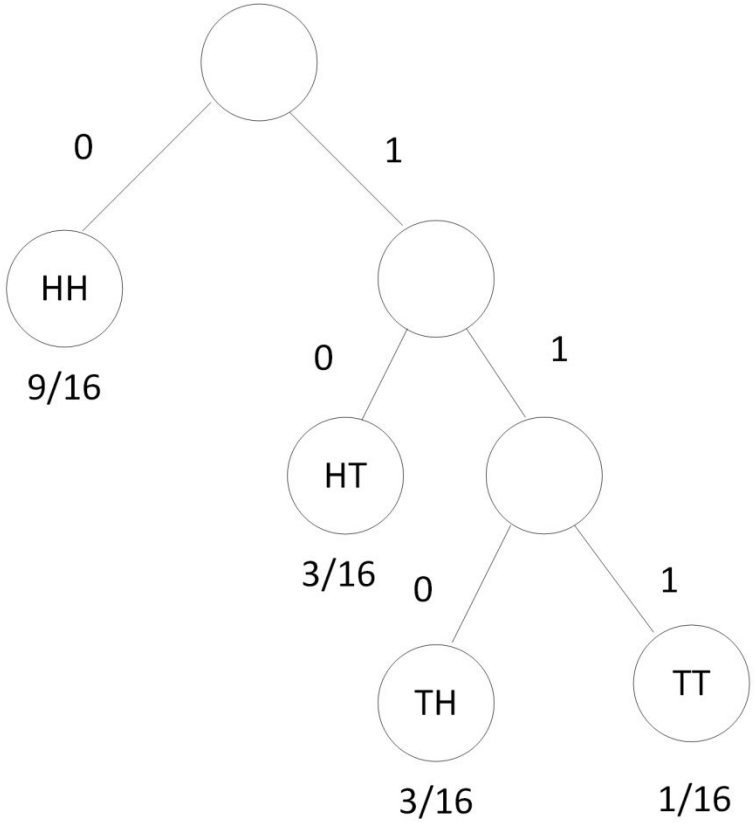
Suppose we toss a coin once.

If P(H)= ½, expected value of number of bits needed to report outcome is 1.

If P(H)= ¾, expected value of number of bits need to report outcome is 1.

Suppose we toss the coin twice.





P(H) = 1/2

E(avg. # bits) = (2(1/4)+2(1/4)+2(1/4)+2(1/4))/2
= 1

P(H) = 3/4

E(avg. # bits) = (1(9/16)+2(3/16)+3(3/16)+3(16))/2
= 0.84375

The entropy *H* of a probability distribution is defined as follows:

$$H = \lim_{n \to \infty} E(\text{avg \# bits} \mid \text{optimal code})$$

It is possible to show that for a binary outcome (coin toss):

$$H = -(p_1 \log_2 p_1 + p_2 \log_2 p_2)$$

If P(Heads) = 0.5,

$$\begin{aligned} H &= -(p_1 \log_2 p_1 + p_2 \log_2 p_2) \\ &= -(.5 \log_2 .5 + .5 \log_2 .5) = 1 \end{aligned}$$

If P(Heads) = 0.75,

$$\begin{aligned} H &= -(p_1 \log_2 p_1 + p_2 \log_2 p_2) \\ &= -(.75 \log_2 .75 + .25 \log_2 .25) = 0.81128 \end{aligned}$$

H is minimized when $p = 1$ for some outcome.

$$H = -(1\log_2 1 + 0\log_2 0) = 0$$

H is maximized when $p_1 = p_2 = 1/2$

$$H = -(.5\log_2 .5 + .5\log_2 .5) = 1$$

H is a measure of uncertainty in the outcome.

When there are *m* outcomes, the entropy is as follows:

$$H = -\sum_{i=1}^{m} p_i \log_2 p_i$$

Entropy is minimized when *p* = 1 for some outcome.

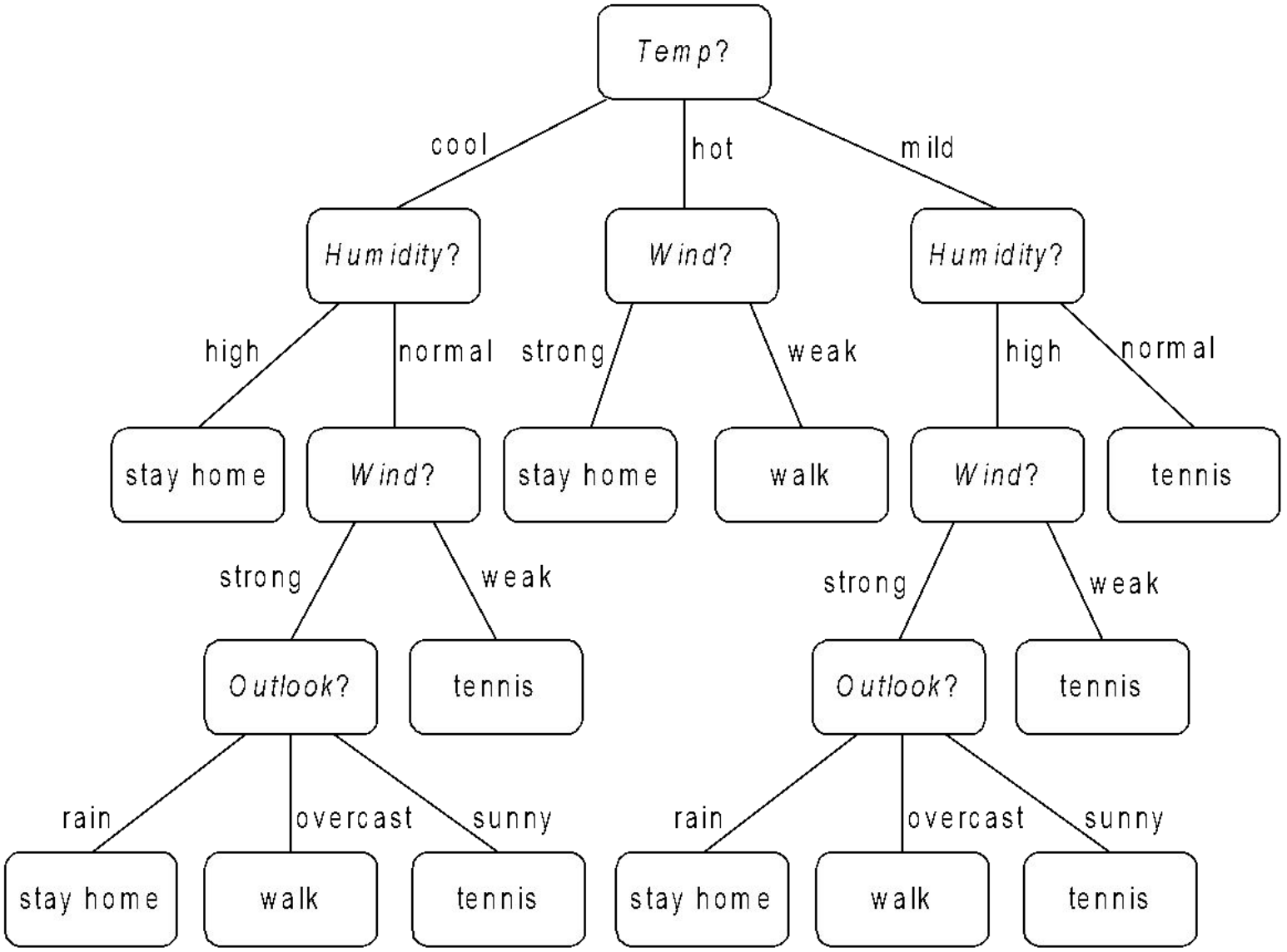
Entropy is maximized when *p* = 1/*m* for all outcomes.

Suppose we want a decision that informs us of the activity we should engage in on a given day based on the outlook, temp, humidity, and wind.

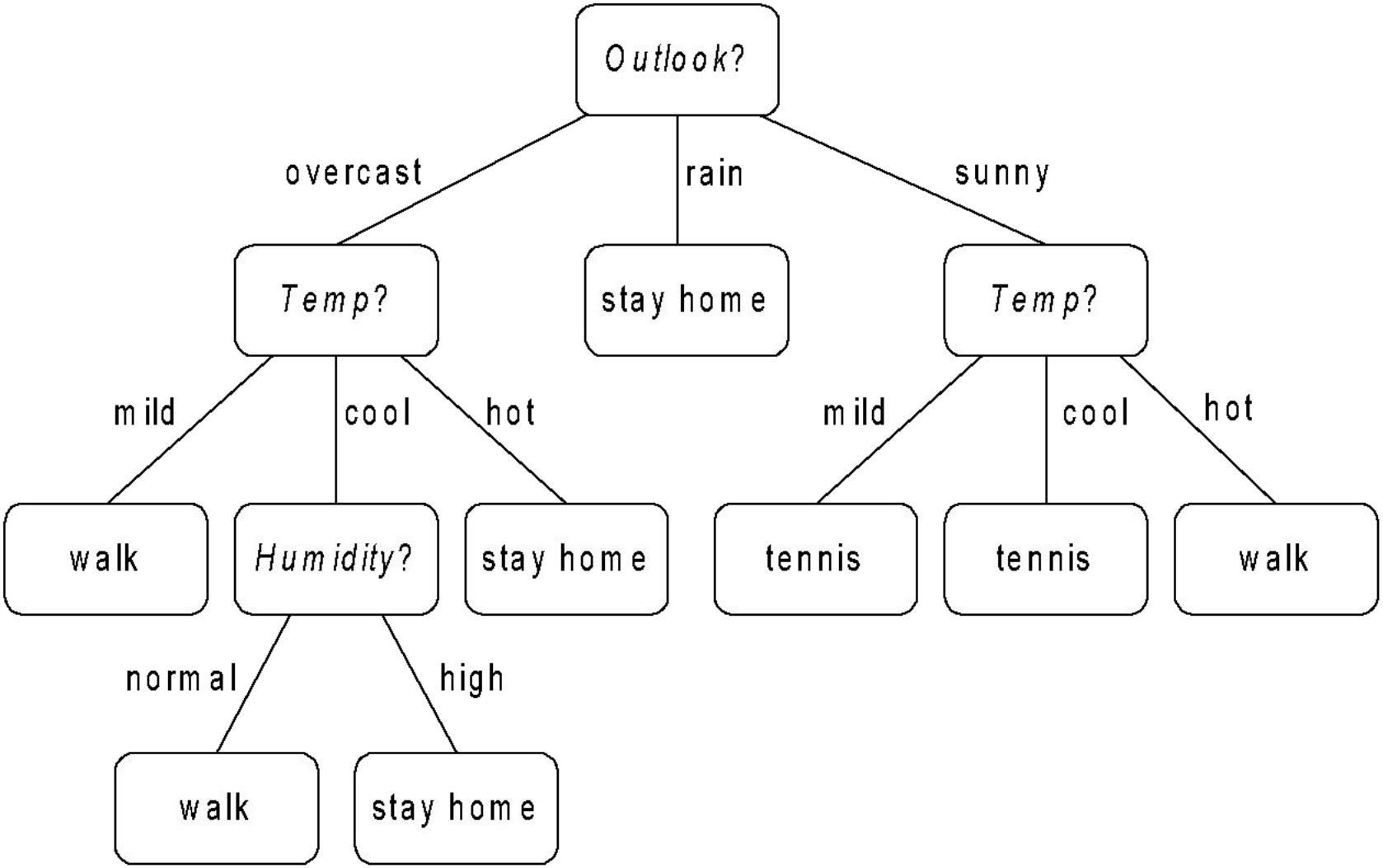
We want the decision tree to classify these data correctly:

Day	Outlook	Temp	Humidity	Wind	Activity
1	rain	hot	high	strong	stay home
2	overcast	cool	high	strong	stay home
3	overcast	cool	normal	strong	walk
4	rain	cool	normal	strong	stay home
5	sunny	cool	normal	strong	tennis
6	sunny	cool	normal	weak	tennis
7	rain	hot	normal	strong	stay home
8	sunny	hot	normal	weak	walk
9	sunny	mild	normal	strong	tennis
10	sunny	mild	high	weak	tennis
11	rain	mild	high	strong	stay home
12	overcast	mild	high	strong	walk
13	sunny	mild	high	strong	tennis
14	overcast	hot	high	strong	stay home

This decision tree classifies the data:



This parsimonious decision tree classifies the data:



Our goal is to learn the most parsimonious decision tree from the data.

Algorithm 5.3 (ID3) uses information theory to learn a parsimonious tree.

Next we illustrate how the algorithm learns the top node.

The entropy of Activity:

$$H(Activity) = -\left(\frac{6}{14}\log_2\frac{6}{14} + \frac{3}{14}\log_2\frac{3}{14} + \frac{5}{14}\log_2\frac{5}{14}\right) = 1.5306$$

The entropy of Activity given rain, sunny, and overcast:

$$H(Activity \mid rain) = -\left(\frac{4}{4}\log_2\frac{4}{4} + \frac{0}{4}\log_2\frac{0}{4} + \frac{0}{4}\log_2\frac{0}{4}\right) = 0$$

$$H(Activity \mid sunny) = -\left(\frac{5}{6}\log_2\frac{5}{6} + \frac{1}{6}\log_2\frac{1}{6} + \frac{0}{6}\log_2\frac{0}{6}\right) = 0.650$$

$$H(Activity \mid overcast) = -\left(\frac{2}{4}\log_2\frac{2}{4} + \frac{2}{4}\log_2\frac{2}{4} + \frac{0}{4}\log_2\frac{0}{4}\right) = 1.0$$

The expected value of entropy of Activity given Outlook:

$$EH(Activity \mid Outlook) = 0\left(\frac{4}{14}\right) + 0.650\left(\frac{6}{14}\right) + 1\left(\frac{4}{14}\right) = 0.564$$

The information gain of Activity relative to Outlook:

$$IG(Activity; Outlook) = H(Activity) - EH(Activity \mid Outlook)$$
$$= 1.5306 - 0.564 = 0.967$$

The information gains of Activity relative to Humidity, Temp, and Wind are all smaller.

So, we choose *Outlook* as the top node in the decision tree.

The ID3 algorithm then recursively learns the remaining nodes in the same fashion.

Formal Definition of Information Gain

Let $P(Z)$ be a probability distribution, where Z has m alternatives, and X be a random variable with alternative x_j. We define conditional entropy of Z given x_j as follows:

$$H(Z \mid x_j) = -\sum_{i=1}^{m} P(z_i \mid x_j) \log_2 P(z_i \mid x_j)$$

The conditional entropy of *Z* given *X* is the expected value of the entropy of *Z* given *X*. It is defined as follows (where *X* has *m* alternatives).

$$EH(Z \mid X) = -\sum_{j=1}^{m} H(Z \mid x_j) P(x_j).$$

The Information Gain of *X* for *Z* is as follows.

$$IG(Z; X) = H(Z) - EH(Z \mid X)$$