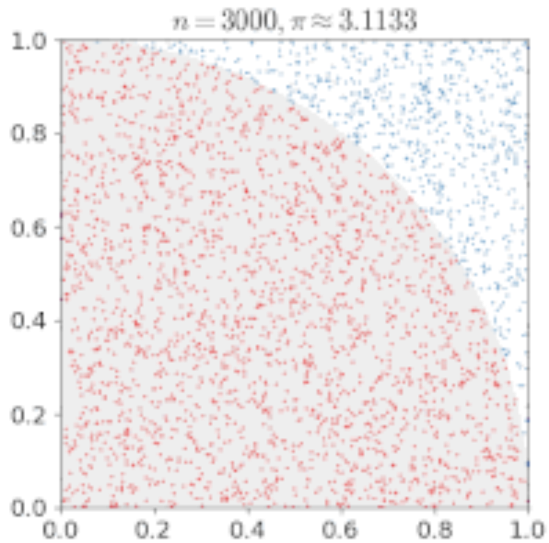
Monte Carlo methods in Machine learning

Based partially on

https://machinelearningmastery.com/monte-carlo-samplig-or-probability

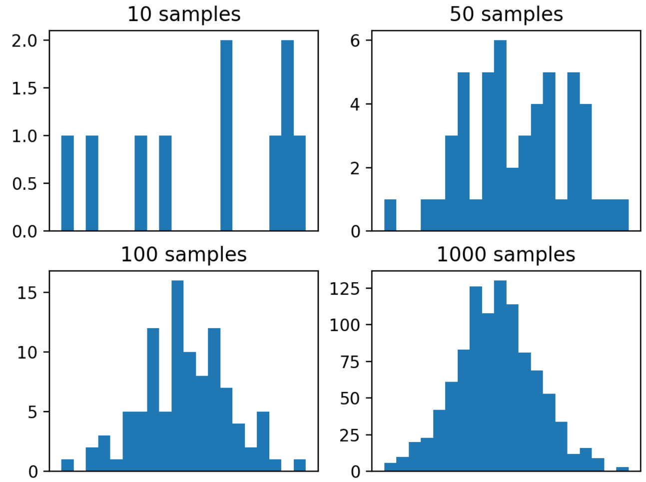
Main idea

- Monte Carlo methods are a class of techniques for randomly sampling a probability distribution
- Example: compute the area under a curve (pic. From Wikipedia).
- Generate points, i.e. couples (x,y) randomly, count the number of points that are under the curve (n) vs. all points generated (N). The area is approximated with n/N



Example & Convergence

- randomly draw samples from a Gaussian distribution with the specified mean (*mu*), standard deviation (*sigma*), and sample size



- Convergence to zero error = approx. 1/sqrt(#trials), not too fast

Monte Carlo in Machine learning

Resampling algorithms.

Monte Carlo methods provide the basis for resampling techniques like the bootstrap method for estimating a quantity, such as the accuracy of a model on a limited dataset.

Random hyperparameter tuning.

Random sampling of model hyperparameters when tuning a model is a Monte Carlo method

Monte Carlo methods also provide the basis for randomized or stochastic optimization algorithms, such as the popular Simulated Annealing optimization technique.

Theory of Machine learning

Peter Flach Book pp.124-126, Tom Mitchell, Machine Learning Chapter 7

We seek theory to relate:

- Probability of successful learning
- Number of training examples
- Complexity of hypothesis space
- Accuracy to which target concept is approximated
- Manner in which training examples presented

Two roles for Bayesian methods

Tom Mitchell, Machine Learning Chapter 6

- Provides practical learning algorithm
- Provides conceptual frameworks

  gold standard for evaluationg other learning algorithms

  insight to Occam;s razor

Brute Force MAP Hypothesis Learner

1. For each hypothesis h in H, calculate the posterior probability

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}$$

2. Output the hypothesis h_{MAP} with the highest posterior probability

$$h_{MAP} = \underset{h \in H}{\operatorname{argmax}} P(h|D)$$

H … hypotheses
D … learning data
h_{MAP} … maximum a posteriori hypothesis

VC-dimension

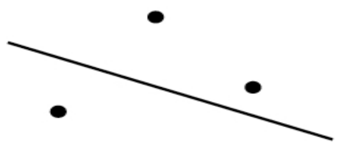
Leede Lee, Quora,Updated March 7, 2017

- In binary classification, for a given dataset with *m* points in a *n*-dimensional real number space, there are 2^m labeling scheme for it. For instance, for a dataset with 2 fixed points whose Cartesian coordinates are (0,0) and (1,0), these points can be labeled {+,+}, {+,-}, {-,+}, {-, -}, so there are 2x2=4 labeling schemes for the dataset.
- We say a hypothesis class ***H*** can "shatter" a GIVEN dataset ***S***, if for ANY labeling scheme of ***S***, there are always at least a hypothesis *h* within ***H*** that can correctly predict every point's label.
- Finally, we say the VC-dimension of the hypothesis class ***H*** is *d*, if the highest cardinality (i.e. the number of points) of dataset which ***H*** can shatter is *d*.

- VC-dimension refers to Vapnik–Chervonenkis dimension, originally put forward by Vladimir Vapnik and Alexey Chervonenkis. Roughly, it measures a hypothesis class's capacity, or expressive power.

VC-dimension: Examples

- The VC-dimension is the size of the largest set of instances that can be shattered by a particular hypothesis language or model class.
- VC-dimension of a linear classifier in d dimensions is d +1: a threshold on the real line can shatter two points but not three (since the middle point cannot be separated from the other two by a single threshold)



- The VC-dimension of 1NN (kNN for k=1) is infinite.
- the VC-dimension of conjunctive concepts over *d* Boolean literals is *d*.

  (= concepts or binary classifiers)
- The VC-dimension can be used to bound the difference between sample error and true error of a hypothesis (more in Flach p.126)

Learnability

Peter Flach Book pp.124-126, Tom Mitchell, Machine Learning Chapter 7

We seek theory to relate:

- Probability of successful learning
- Number of training examples
- Complexity of hypothesis space
- Accuracy to which target concept is approximated
- Manner in which training examples presented

To start things up we need a learning model

One of the most common (although rather pessimistic)

Probably approximately correct (PAC) learning

(L. Valiant. *A theory of the learnable.* Communications of the ACM, 27, 1984.)

PAC Learning

Consider a class C of possible target concepts defined over a set of instances X of length n, and a learner L using hypothesis space H.

Definition: C is **PAC-learnable** by L using H if for all $c \in C$, distributions $\mathcal{D}$ over X, ϵ such that $0 < \epsilon < 1/2$, and δ such that $0 < \delta < 1/2$,

learner L will with probability at least $(1 - \delta)$ output a hypothesis $h \in H$ such that $error_{\mathcal{D}}(h) \leq \epsilon$, in time that is polynomial in $1/\epsilon$, $1/\delta$, n and $size(c)$.

Theorem:

The concept class *C* is PAC learnable **iff** the VC dimension of *C* is finite.

The concept class *C* is PAC learnable by *L* => *L* can generalize.

Kolmogorov complexity and Inductive inference

Kolmogorov complexity of an object, such as a piece of text, is the length of a shortest computer program (in a predetermined programming language) that produces the object as output. It is a measure of the computational resources needed to specify the object.

MDL and learning decision tree

code the decision tree and exceptions to reach the minimum message length

Example

No.	Attributes				Class
	Outlook	*Temperature*	*Humidity*	*Windy*	
1	overcast	hot	high	not	N
2	overcast	hot	high	very	N
3	overcast	hot	high	medium	N
4	sunny	hot	high	not	P
5	sunny	hot	high	medium	P
6	rain	mild	high	not	N
7	rain	mild	high	medium	N
8	rain	hot	normal	not	P
9	rain	cool	normal	medium	N
10	rain	hot	normal	very	N
11	sunny	cool	normal	very	P
12	sunny	cool	normal	medium	P
13	overcast	mild	high	not	N
14	overcast	mild	high	medium	N
15	overcast	cool	normal	not	P
16	overcast	cool	normal	medium	P
17	rain	mild	normal	not	N
18	rain	mild	normal	medium	N
19	overcast	mild	normal	medium	P
20	overcast	mild	normal	very	P
21	sunny	mild	high	very	P
22	sunny	mild	high	medium	P
23	sunny	hot	normal	not	P
24	rain	mild	high	very	N

Example

No.	Attributes				Class
	Outlook	*Temperature*	*Humidity*	*Windy*	
1	overcast	hot	high	not	N
2	overcast	hot	high	very	N
3	overcast	hot	high	medium	N
4	sunny	hot	high	not	P
5	sunny	hot	high	medium	P
6	rain	mild	high	not	N
7	rain	mild	high	medium	N
8	rain	hot	normal	not	P
9	rain	cool	normal	medium	N
10	rain	hot	normal	very	N
11	sunny	cool	normal	very	P
12	sunny	cool	normal	medium	P
13	overcast	mild	high	not	N
14	overcast	mild	high	medium	N
15	overcast	cool	normal	not	P
16	overcast	cool	normal	medium	P
17	rain	mild	normal	not	N
18	rain	mild	normal	medium	N
19	overcast	mild	normal	medium	P
20	overcast	mild	normal	very	P
21	sunny	mild	high	very	P
22	sunny	mild	high	medium	P
23	sunny	hot	normal	not	P
24	rain	mild	high	very	N

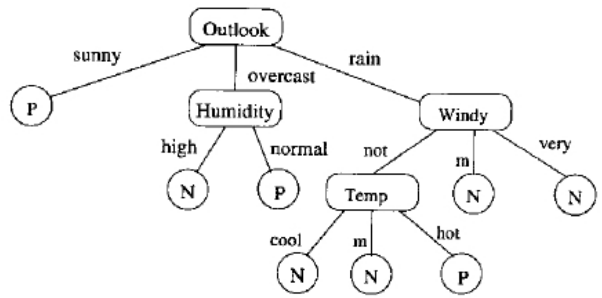


FIGURE 5.5. Perfect decision tree

Example

No.	Attributes				Class
	Outlook	*Temperature*	*Humidity*	*Windy*	
1	overcast	hot	high	not	N
2	overcast	hot	high	very	N
3	overcast	hot	high	medium	N
4	sunny	hot	high	not	P
5	sunny	hot	high	medium	P
6	rain	mild	high	not	N
7	rain	mild	high	medium	N
8	rain	hot	normal	not	P
9	rain	cool	normal	medium	N
10	rain	hot	normal	very	N
11	sunny	cool	normal	very	P
12	sunny	cool	normal	medium	P
13	overcast	mild	high	not	N
14	overcast	mild	high	medium	N
15	overcast	cool	normal	not	P
16	overcast	cool	normal	medium	P
17	rain	mild	normal	not	N
18	rain	mild	normal	medium	N
19	overcast	mild	normal	medium	P
20	overcast	mild	normal	very	P
21	sunny	mild	high	very	P
22	sunny	mild	high	medium	P
23	sunny	hot	normal	not	P
24	rain	mild	high	very	N

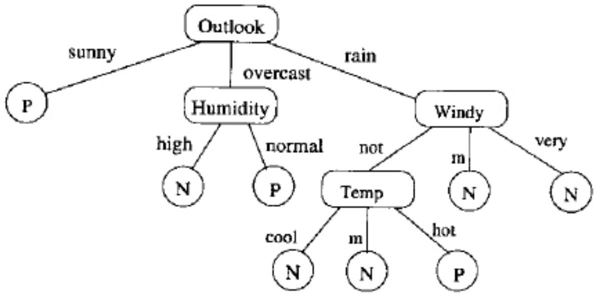
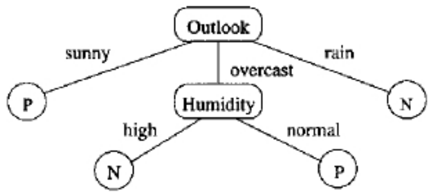


FIGURE 5.5. Perfect decision tree



Example

No.	Attributes				Class
	Outlook	*Temperature*	*Humidity*	*Windy*	
1	overcast	hot	high	not	N
2	overcast	hot	high	very	N
3	overcast	hot	high	medium	N
4	sunny	hot	high	not	P
5	sunny	hot	high	medium	P
6	rain	mild	high	not	N
7	rain	mild	high	medium	N
8	rain	hot	normal	not	P
9	rain	cool	normal	medium	N
10	rain	hot	normal	very	N
11	sunny	cool	normal	very	P
12	sunny	cool	normal	medium	P
13	overcast	mild	high	not	N
14	overcast	mild	high	medium	N
15	overcast	cool	normal	not	P
16	overcast	cool	normal	medium	P
17	rain	mild	normal	not	N
18	rain	mild	normal	medium	N
19	overcast	mild	normal	medium	P
20	overcast	mild	normal	very	P
21	sunny	mild	high	very	P
22	sunny	mild	high	medium	P
23	sunny	hot	normal	not	P
24	rain	mild	high	very	N

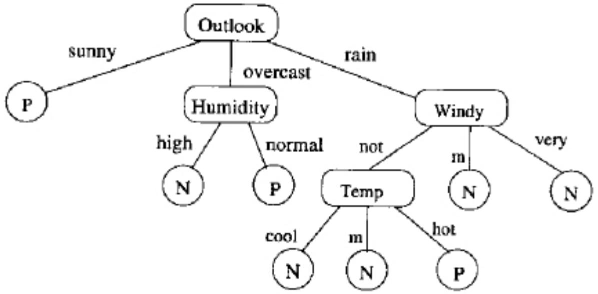
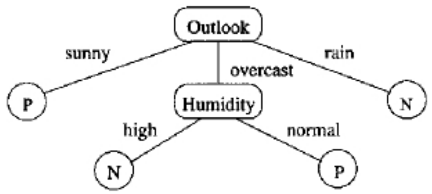


FIGURE 5.5. Perfect decision tree



+ single exception

Example

No.	Attributes				Class
	Outlook	*Temperature*	*Humidity*	*Windy*	
1	overcast	hot	high	not	N
2	overcast	hot	high	very	N
3	overcast	hot	high	medium	N
4	sunny	hot	high	not	P
5	sunny	hot	high	medium	P
6	rain	mild	high	not	N
7	rain	mild	high	medium	N
8	rain	hot	normal	not	P
9	rain	cool	normal	medium	N
10	rain	hot	normal	very	N
11	sunny	cool	normal	very	P
12	sunny	cool	normal	medium	P
13	overcast	mild	high	not	N
14	overcast	mild	high	medium	N
15	overcast	cool	normal	not	P
16	overcast	cool	normal	medium	P
17	rain	mild	normal	not	N
18	rain	mild	normal	medium	N
19	overcast	mild	normal	medium	P
20	overcast	mild	normal	very	P
21	sunny	mild	high	very	P
22	sunny	mild	high	medium	P
23	sunny	hot	normal	not	P
24	rain	mild	high	very	N

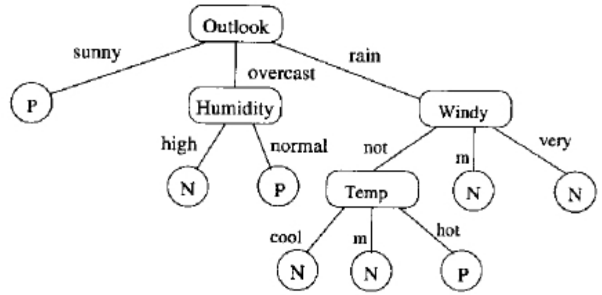
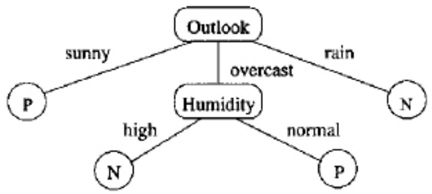


FIGURE 5.5. Perfect decision tree



+ single exception

How to code a decision tree and exceptions

Perfect tree

For the tree in Figure 5.5, she writes down

1 Outlook 0 P 1 Humidity 0 N 0 P 1 Windy 0 N 0 N
1 Temperature 0 N 0 N 1 P.

Imperfect tree

1 Outlook 0 P 1 Humidity 0 N 0 P 0 N.

Coding the Exceptions Since the decision tree in Figure 5.4 is not perfect, we need to indicate where the exceptions are. In this case there is a single exception. The most straightforward way is to indicate its *position* among all 54 possible combinations of attributes. This costs $\log 54 \approx 5.75$ extra bits.

Summary

Thus, the encoding using the decision tree in Figure 5.4 uses 19.335 bits; the encoding using the decision tree in Figure 5.5 uses 25.585 bits. The MDL principle tells us to use the former method, which is also much shorter than the 54-bit plain encoding.

Information sources

- Tom Mitchell, Machine Learning
- Peter Flach Book
- http://work.caltech.edu/lectures.html
- Li & Vitanyi, An Introduction to Komogorov Complexity and Its Applications