Solutions for the Exercises

11

11.1 Introduction

Exercise 1.2 A machine can well be intelligent, even if its behavior is very different from that of a human being.

Exercise 1.3 If a problem is NP-complete or can be described as "hard", that means that there are instances in which the problem cannot be solved in an acceptable amount of time. This is the so-called *worst case*. In some applications we have to live with the fact that in the worst case an efficient solution is impossible. This means that even in the future there will be practically relevant problems which in certain special cases are unsolvable.

AI will therefore neither find a universal formula, nor build a super machine with which all problems become solvable. It gives itself rather the task of building systems with a higher probability of finding a solution, or with a higher probability of finding fast, optimal solutions. We humans in everyday life deal with suboptimal solutions quite well. The reason is, quite simply, the excessive cost of finding the optimal solution. For example, it only takes me seven minutes to find my way from point A to point B with a map in an unfamiliar city. The shortest path would have taken only six minutes. Finding the shortest path, however, would have taken perhaps an hour. The proof of the optimality of the path might be even costlier.

Exercise 1.4

(a) The output depends not only on the input, but also on the contents of the memory. For an input x, depending on the contents of the memory, the output could be y_1 or y_2. It is thus not unique and therefore not a function.
(b) If one considers the contents of the memory as a further input, then the output is unique (because the agent is deterministic) and the agent represents a function.

W. Ertel, *Introduction to Artificial Intelligence*, Undergraduate Topics in Computer Science, https://doi.org/10.1007/978-3-658-43102-0_11

Exercise 1.5

(a) Velocity $v_x(t) = \frac{\partial x}{\partial t} = \lim_{\Delta t \to 0} \frac{x(t)-x(t-\Delta t)}{\Delta t} \approx \frac{x(t)-x(t-\Delta t)}{\Delta t}$. v_y is calculated analogously.

(b) Acceleration $a_x(t) = \frac{\partial^2 x}{\partial t^2} = \frac{\partial}{\partial t} v_x(t) = \lim_{\Delta t \to 0} \frac{v_x(t)-v_x(t-\Delta t)}{\Delta t} = \lim_{\Delta t \to 0} \left[\frac{x(t)-x(t-\Delta t)}{(\Delta t)^2} - \frac{x(t-\Delta t)-x(t-2\Delta t)}{(\Delta t)^2} \right] = \frac{x(t)-2x(t-\Delta t)+x(t-2\Delta t)}{(\Delta t)^2}$. a_y is calculated analogously. One also needs the position at the three times $t - 2\Delta t$, $t - \Delta t$, t.

Exercise 1.6

(a) Costs for agent 1 = 11 · 100 cents + 1 · 1 cent = 1,101 cents.
Costs for 2 = 0 · 100 cents + 38 · 1 cent = 38 cents.
Therefore agent 2 saves 1,101 cents − 38 cents = 1,063 cents.

(b) Profit for agent 1 = 189 · 100 cents + 799 · 1 cent = 19,699 cents.
Profit for agent 2 = 200 · 100 cents + 762 · 1 cent = 20,762 cents.
Agent 2 therefore has 20,762 cents − 19,699 cents = 1,063 cents higher profit.
If one assesses the lost profits due to errors, the utility-based agent makes the same decisions as a cost-based agent.

11.2 Propositional Logic

Exercise 2.1 With the signature $\Sigma = \{ \sigma_1, \sigma_2, \ldots, \sigma_n \}$ and the grammar variables ⟨ *formula* ⟩, the syntax of propositional logic can be defined as follows:

$$\begin{aligned} \langle formula \rangle :: = & \sigma_1 \mid \sigma_2 \mid \cdots \mid \sigma_n \mid w \mid f \\ & \mid \neg \langle formula \rangle \mid (\langle formula \rangle) \mid \langle formula \rangle \wedge \langle formula \rangle \\ & \mid \langle formula \rangle \vee \langle formula \rangle \mid \langle formula \rangle \Rightarrow \langle formula \rangle \\ & \mid \langle formula \rangle \Leftrightarrow \langle formula \rangle \end{aligned}$$

Exercise 2.2 Proof by the truth table method.

Exercise 2.3 (a) $(\neg A \vee B) \wedge (\neg B \vee A)$ (b) $(\neg A \vee B) \wedge (\neg B \vee A)$ (c) t

Exercise 2.4 (a) satisfiable (b) true (c) unsatisfiable

Exercise 2.6

(a) In Exercise 2.3(c) it was already shown that $A \wedge (A \Rightarrow B) \Rightarrow B$ is a tautology. The deduction theorem thus ensures the correctness of the inference rule.

(b) We show by the truth table method that $(A \vee B) \wedge (\neg B \vee C) \Rightarrow (A \vee C)$ is a tautology.

Exercise 2.7 Application of the resolution rule to the clause $(f \vee B)$ and $(\neg B \vee f)$ yields the resolvent $(f \vee f) \equiv (f)$. Now we apply the resolution rule to the clauses B and $\neg B$ and obtain the empty clause as the resolvent. Because $(f \vee B) \equiv B$ and $(\neg B \vee f) \equiv \neg B$, $(f) \equiv ()$. It is important in practice that, whenever the empty clause is derived, it is due to a contradiction.

Exercise 2.8 If KB contains a contradiction, then there are two clauses A and $\neg A$, which allow the empty clause to be derived. The contradiction in KB is clearly still in $KB \wedge \neg Q$. Therefore it also allows the empty clause to be derived.

Exercise 2.9 (a) $(A \vee B) \wedge (\neg A \vee \neg B)$ (b) $(A \vee B) \wedge (B \vee C) \wedge (A \vee C)$

Exercise 2.10 *Formalization*: Accomplice: A, Car: C, Key: K

$$WB \equiv (A \Rightarrow C) \wedge [(\neg A \wedge \neg K) \vee (A \wedge K)] \wedge K$$

Transformation into CNF: $(\neg A \wedge \neg K) \vee (A \wedge K) \equiv (\neg K \vee A) \wedge (\neg A \vee K)$

Try to prove C and add $\neg C$ to the set of clauses. The CNF clause set is

$$(\neg A \vee C)_1 \wedge (\neg K \vee A)_2 \wedge (\neg A \vee K)_3 \wedge (K)_4 \wedge (\neg C)_5.$$

Resolution proof: Res(2, 4) : $(A)_6$
Res(1, 6) : $(C)_7$
Res(7, 5) : $()_8$

Thus C has been shown.

Exercise 2.11

(a) $KB \equiv (A \vee B) \wedge (\neg B \vee C)$, $Q \equiv (A \vee C)$
$KB \wedge \neg Q \equiv (A \vee B)_1 \wedge (\neg B \vee C)_2 \wedge (\neg A)_3 \wedge (\neg C)_4$
Resolution proof: Res(1, 3) : $(B)_5$
Res(2, 4) : $(\neg B)_6$
Res(5, 6) : $()$

(b) $\neg(\neg B \wedge (B \vee \neg A) \Rightarrow \neg A) \equiv (\neg B)_1 \wedge (B \vee \neg A)_2 \wedge (A)_3$
Resolution proof: Res(1, 2) : $(\neg A)_4$
Res(3, 4) : $()$

Exercise 2.12 By application of the equivalences from Theorem 2.1, we can immediately prove the claims.

Exercise 2.13

$$\begin{aligned}
&\text{Res}(8,\ 9): \quad (C \wedge F \wedge E \Rightarrow f)_{10}\\
&\text{Res}(3,\ 10): \quad (F \wedge E \Rightarrow f)_{11}\\
&\text{Res}(6,\ 11): \quad (A \wedge B \wedge C \wedge E \Rightarrow f)_{12}\\
&\text{Res}(1,\ 12): \quad (B \wedge C \wedge E \Rightarrow f)_{13}\\
&\text{Res}(2,\ 13): \quad (C \wedge E \Rightarrow f)_{14}\\
&\text{Res}(3,\ 14): \quad (E \Rightarrow f)_{15}\\
&\text{Res}(5,\ 15): \quad ()
\end{aligned}$$

Exercise 2.14 Even the 3-sat problem, i.e., the test for the satisfiability of a CNF formula with three literals per clause is NP-complete. Thus, the satisfiability of any CNF formula is also NP-complete and belongs to the class of particularly difficult decision problems. To date, no algorithm has been found that can solve NP-complete problems in polynomial time. Scientists assume that there is no such algorithm exists. However, this has not yet been proven. This open question, the so-called P-NP problem, is still a major challenge for theoretical computer science [CLR90].

11.3 First-Order Predicate Logic

Exercise 3.1

(a) $\forall x\ \text{male}(x) \iff \neg\ \text{female}(x)$
(b) $\forall x\ \forall y\ \exists z\ \text{father}(x, y) \iff \text{male}(x) \wedge \text{child}(y, x, z)$
(c) $\forall x\ \forall y\ \text{siblings}(x, y) \iff [(\exists z\ \text{father}(z, x) \wedge \text{father}(z, y)) \vee (\exists z\ \text{mother}(z, x) \wedge \text{mother}(z, y))]$
(d) $\forall x\ \forall y\ \forall z\ \text{parents}(x, y, z) \iff \text{father}(x, z) \wedge \text{mother}(y, z)$
(e) $\forall x\ \forall y\ \text{uncle}(x, y) \iff \exists z\ \exists u\ \text{child}(y, z, u) \wedge \text{siblings}(z, x) \wedge \text{male}(x)$
(f) $\forall x\ \forall y\ \text{ancestor}(x, y) \Leftrightarrow \exists z\ \text{child}(y, x, z) \vee \exists u\ \exists v\ \text{child}(u, x, v) \wedge \text{ancestor}(u, y))$

Exercise 3.2

(a) $\forall x\ \exists y\ \exists z\ \text{father}(y, x) \wedge \text{mother}(z, x)$
(b) $\exists x\ \exists y\ \text{child}(y, x, z)$
(c) $\forall x\ \text{bird}(x) \Rightarrow \text{flies}(x)$
(d) $\exists x\ \exists\ y\ \exists z\ \text{animal}(x) \wedge \text{animal}(y) \wedge \text{eats}(x, y) \wedge \text{eats}(y, z) \wedge \text{grain}(z)$
(e) $\forall x\, \text{animal}(x) \Rightarrow (\exists y\, (\text{eats}(x, y) \wedge (\text{plant}(y) \vee (\text{animal}(y) \wedge \exists z\, \text{plant}(z) \wedge \text{eats}(y, z) \wedge \text{much\_smaller}(y, x)))$

Exercise 3.3 $\forall x\ \forall y\ \exists z\ \ x = \text{father}(y) \iff \text{male}(x) \wedge \text{c hild}(y, x, z)$

Exercise 3.4
$\forall x\ \forall y\ \ x < y \lor y < x \lor x = y,$
$\forall x\ \forall y\ \ x < y \Rightarrow \neg y < x,$
$\forall x\ \forall y\ \forall z\ \ x < y \land y < z \Rightarrow x < z$

Exercise 3.5

(a) MGU: $x/f(z)$, $u/f(y)$, term: $p(f(z), f(y))$
(b) not unifiable
(c) MGU: $x/\cos y$, $z/4 - 7 \cdot \cos y$, term: $\cos y = 4 - 7 \cdot \cos y$
(d) not unifiable
(e) MGU: $u/f(g(w,w), g(g(w,w), g(w,w)), g(g(g(w,w), g(w,w)), g(g(w,w), g(w,w))))$ $x/g(w,w)$, $y/g(g(w,w), g(w,w))z/g(g(g(w,w), g(w,w)), g(g(w,w), g(w,w)))$ term: $q(f(g(w,w),\ g(g(w,w), g(w,w)),\ g(g(g(w,w), g(w,w)), g(g(w,w), g(w,w)))),$ $f(g(w,w),\ g(g(w,w), g(w,w)),\ g(g(g(w,w), g(w,w)), g(g(w,w), g(w,w))))$

Exercise 3.7

(a) Let the unsatisfiable formula $p(x) \land \neg p(x) \land r(x)$ be given. We choose the clause $r(x)$ as the SOS, so no contradiction can be derived.
(b) If the SOS is already unsatisfiable, then no contradiction can be derived. If not, then resolution steps between clauses from SOS and $(KB \land \neg Q)$ SOS are necessary.
(c) If there is no complement to a literal L in a clause K, then the literal L will remain in every clause that is derived using resolution from clause K. Thus the empty clause cannot be derived from K or its resolvent, nor any future resolvent.

Exercise 3.8 $\neg Q \land KB \equiv (e = n)_1 \land (n \cdot x = n)_2 \land (e \cdot x = x)_3 \land (\neg a = b)_4$
Proof : Dem(1, 2) : $(e \cdot x = e)_5$
Tra,Sym(3, 5) : $(x = e)_6$
Dem(4, 6) : $(\neg e = b)_7$
Dem(7, 6) : $(\neg e = e)_8$
$()$
Here "Dem" stands for demodulation. Clause number 6 was derived by application of transitivity and symmetry of the equality in clauses 3 and 5. The empty clause is obtained by applying the reflexive property of equality to e, which yields $(e = e)$, and, via resolution with clause 8, results in the empty clause.

Exercise 3.9 The LOP input files are:

(a)
```
a;b<-.
<-a,b.
b;c<-.
<-b,c.
c;a<-.
<-c,a.
```

(b)
```
<- shaves(barb,X),
   shaves(X,X).
   shaves(barb,X);
   shaves(X,X) <-.
```

(c)
```
e = n.
   <- a = b.
   m(m(X,Y),Z)=m(X,m(Y,Z)).
   m(e,X) = X.
   m(n,X) = n.
```

11.4 Limitations of Logic

Exercise 4.1

(a) Correctly: *We take a complete proof calculus for PL1. With it we find a proof for every true formula in finite time. For all unsatisfiable formulas I proceed as follows*: *I apply the calculus to* $\neg\phi$ *and show that* $\neg\phi$ *is true. Thus* ϕ *is false. Thus we can prove every true formula from PL1 and refute every false formula.* Unfortunately, this process is unsuitable for satisfiable formulas.

Exercise 4.2

(a) He shaves himself if and only if he does not shave. (contradiction)
(b) The set of all sets which do not contain themselves. It contains itself if and only if it does not contain itself.

11.5 PROLOG

Exercise 5.1 PROLOG signals a stack overflow. The reason is PROLOG's depth-first search, which always chooses the first unifiable predicate in the input file. With recursive predicates such as equality, this causes a non-terminating recursion.

Exercise 5.2

```
write_path( [H1,H2|T] ) :- write_path([H2|T]), write_move(H2,H1).
write_path( [_X] ).
write_move( state(X,W,Z,K), state(Y,W,Z,K) ) :-
    write('Farmer from '), write(X), write(' to '), write(Y), nl.
write_move( state(X,X,Z,K), state(Y,Y,Z,K) ) :-
    write('Farmer and wolf from '), write(X), write(' to '), write(Y),nl.
write_move( state(X,W,X,K), state(Y,W,Y,K) ) :-
    write('Farmer and goat from '),write(X),write(' to '), write(Y), nl.
write_move( state(X,W,Z,X), state(Y,W,Z,Y) ) :-
    write('Farmer and cabbage from '),write(X),write(' to '), write(Y), nl.
```

Exercise 5.3

(a) It is needed to output the solutions found. The unification in the fact `plan(Goal,Goal,Path,Path).` takes care of this.

(b) The conditions for entry into the clauses of the predicate `safe` overlap each other. The backtracking caused by the `fail` leads to the execution of the second or third alternative of `safe`, where the same solution will clearly be found again. A cut at the end of the first two `safe` clauses solves the problem. Alternatively, all safe states can be explicitly given, as for example `safe(state(left,left,left,right)).`

Exercise 5.4

(a) The prover E finds a proof here because it uses (in contrast to Prolog) a complete search strategy.
(b) Prolog outputs, as shown in Sec. 5.2, the first solution `X = oscar` and `Y = karen`. The E–prover finds a proof and returns the answer "# Proof found!". However, it does not output a solution in the form of a variable instantiation.

Exercise 5.5

```
?- one(10).
one(0) :- write(1).
one(N) :- N1 is N-1, one(N1), one(N1).
```

Exercise 5.6

(a) With $n_1 = |L1|$, $n_2 = |L2|$ it is true that $T_{\text{append}}(n_1, n_2) = \Theta(n_1)$.
(b) With n=$|L|$ it is true that $T_{\text{nrev}}(n) = \Theta(n^2)$.
(c) With n=$|L|$ it is true that $T_{\text{accrev}}(n) = \Theta(n)$.

Exercise 5.7 For the trees with symbols on the inner nodes, we can use the terms `a(b,c)` and `a(b(e,f,g),c(h),d)`, for example. Trees without symbols: `tree(b,c)`, or `tree(tree(e,f,g),tree(h),d)`.

Exercise 5.8 SWI-PROLOG programs:

(a)
```
fib(0,1). fib(1,1).
fib(N, R) :- N1 is N − 1, fib(N1,
R1), N2 is N − 2, fib(N2,R2), R is R1 + R2.
```
(b) $T(n) = \Theta(2^n)$.
(c)
```
:- dynamic fib/2. fib(0,1). fib(1,1).
fib(N,R) :- N1
is N - 1, fib(N1, R1), N2 is N - 2, fib(N2,R2), R is
R1 + R2, asserta(fib(N,R)).
```
(d) If the facts `fib(0,1)` to `fib(k,`*fib*(k)`)` were already calculated, then for the call `fib(n,X)` it is true that

$$T(n) = \begin{cases} \Theta(1) & \text{if } n \leq k, \\ \Theta(n-k) & \text{if } n > k. \end{cases}$$

(e) Because after the first n calls to the `fib` predicate, all further calls have direct access to facts.

Exercise 5.9

(a) The solution is not disclosed here.
(b)

```
start :-
  fd_domain([Briton, Swede, Dane, Norwegian, German],1,5),
  fd_all_different([Briton, Swede, Dane, Norwegian, German]),
  fd_domain([Tea, Coffee, Water, Beer, Milk],1,5),
  fd_all_different([Tea, Coffee, Water, Beer, Milk]),
  fd_domain([Red, White, Green, Yellow, Blue],1,5),
  fd_all_different([Red, White, Green, Yellow, Blue]),
  fd_domain([Dog, Bird, Cat, Horse, Fish],1,5),
  fd_all_different([Dog, Bird, Cat, Horse, Fish]),
  fd_domain([Pallmall, Dunhill, Marlboro, Winfield, Rothmanns],1,5),
  fd_all_different([Pallmall, Dunhill, Marlboro, Winfield, Rothmanns]),
  fd_labeling([Briton, Swede, Dane, Norwegian, German]),
  Briton #= Red,              % The Briton lives in the red house
  Swede #= Dog,               % The Swede doesn't have a dog
  Dane #= Tea,                % The Dane likes to drink tea
  Green #= White - 1,         % The green house is to the left of the white house
  Green #= Coffee,            % The owner of the green house drinks coffee
  Pallmall #= Bird,           % The person who smokes Pall Mall has a bird
  Milk #= 3,                  % The man in the middle house drinks milk
  Yellow #= Dunhill,          % The owner of the yellow house smokes Dunhill
  Norwegian #= 1,             % The Norwegian lives in the first house
  dist(Marlboro,Cat)#= 1,     % The Marlboro smoker lives next to the cat
  dist(Horse,Dunhill) #= 1,   % The man with the horse lives next to the Dunhill smoker
  Winfield #= Beer,           % The Winfield smoker likes to drink beer
  dist(Norwegian,Blue) #= 1, % The Norwegian lives next to the blue house
  German #= Rothmanns,        % The German smokes Rothmanns
  dist(Marlboro,Water)#=1,    % The Marlboro smoker's neighbor drinks water.
  write([Briton, Swede, Dane, Norwegian, German]), nl,
  write([Dog, Bird, Cat, Horse, Fish]), nl.
```

11.6 Search, Games and Problem-Solving

Exercise 6.1

(a) On the last level there are b^d nodes. All previous levels together have

$$N_b(d_{max}) = \sum_{i=0}^{d-1} b^i = \frac{b^d - 1}{b - 1} \approx b^{d-1}$$

nodes, if b becomes large. Because $b^d / b^{d-1} = b$ there are about b times as many nodes on the last level as on all other levels together.

(b) For a non-constant branching factor this statement is no longer true, as the following counter-example shows: A tree that branches heavily up to the level before last, followed by a level with a constant branching factor of 1, has exactly as many nodes on the last level as on the level before last.

Exercise 6.2

(a) In Fig. 6.3 the structure of the tree at the second level with its eight nodes repeats. Thus we can calculate the average branching factor b_m from $b_m^2 = 8$ to $b_m = \sqrt{8}$.

(b) Here the calculation is not so simple because the root node of the tree branches more heavily than all others. We can say, however, that in the interior of the tree the branching factor is exactly 1 smaller than it would be without the cycle check. Thus $b_m \approx \sqrt{8} - 1 \approx 1.8$.

Exercise 6.3

(a) For the average branching factor the number of leaf nodes is fixed. For the effective branching factor, in contrast, the number of nodes in the whole tree is fixed.
(b) Because the number of all nodes in the tree is usually a better measurement of the computation time for searching an entire tree than the number of leaf nodes.
(c) For a large b according to (6.1) we have $n \approx \bar{b}^{d+1}/\bar{b} = \bar{b}^d$, yielding $\bar{b} = \sqrt[d]{n}$.

Exercise 6.4

(a) 3-puzzle: 4! = 24 states, 8-puzzle: 9! = 362 880 states, 15-puzzle: 16! = 20 922 789 888 000 states.
(b) After moving the empty square 12 times in the clockwise direction we reach the starting state again and thus create a cyclical sub-space with 12 states.

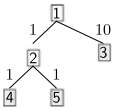
Exercise 6.6 (a) Mathematica program:

```
0 BreadthFirstSearch[Node_, Goal_, Depth_] := Module[{i, NewNodes={}},
1   For[i=1, i<= Length[Node], i++,
2        NewNodes = Join[ NewNodes, Successors[ Node[[i]] ] ];
3        If[MemberQ[Successors[ Node[[i]] ], Goal],
4            Return[{"Solution found", Depth+1}], 0
5          ]
6      ];
7   If[Length[NewNodes] > 0,
8       BreadthFirstSearch[NewNodes, Goal, Depth+1],
9       Return[{"Fail", Depth}]
10    ]
11 ]
```

(b) Because the depth of the search space is not limited.

Exercise 6.7

(a) For a constant cost 1 the cost of all paths at depth d are smaller than the costs of all paths at depth $d + 1$. Since all nodes at depth d are tested before the first node at depth $d + 1$, a solution of length d is guaranteed to be found before a solution of length $d + 1$.
(b) For the neighboring search tree, node number 2 and the solution node number 3 of cost 10 are generated first. The search terminates. The nodes 4 and 5 with path costs of 2 each are not generated. The optimal solution is therefore not found.



Exercise 6.8

(a) It is appropriate to store the predecessor nodes of the current search path in a linked list. This has the advantage that during expansion only one node must be appended and during backtracking only one node must be deleted.

(b) A look at Fig. 6.9 shows that without the cycle check, and thus without storing the predecessor nodes, we must store $\Theta((b-1)\cdot d)$ nodes. When storing all predecessors in the tree structure, $\Theta(b\cdot d)$ storage space is necessary because at each level all nodes must be stored.

(c)
$$\sum_{k=1}^{d} k\cdot b^k = b\frac{d}{db}\sum_{k=1}^{d} b^k = b\frac{d}{db}\frac{b^{d+1}-b}{b-1} = \frac{b}{(b-1)^2}(db^{d+1}-(d+1)b^d+1)$$
$$= \Theta\left(\frac{b}{b^2}db^{d+1}\right) = \Theta(db^d)$$

Exercise 6.11

(a) Let $d^\star(x,y)$ be the shortest distance from x to y. Then a path from x over z to y cannot be even shorter. Then $d^\star(x,y) \le d^\star(x,z) + d^\star(z,y)$.

(b) If the only direct connection between x and y is a curvy mountain pass, then it can be shorter to drive from x to y through z. Use a sketch to illustrate this.

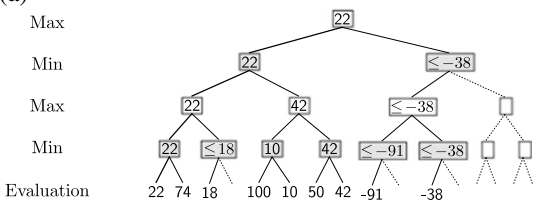
Exercise 6.13 Depth-first search: the new nodes must have a lower rating than all open nodes. Breadth-first search: the new nodes must have a higher rating than all open nodes.

Exercise 6.14 Just like an admissible heuristic, the wife underestimates the distance to the goal. This could result in the two of them finding the quickest way to the goal–though with great effort. This is only true, however, if the lady always underestimates the distance.

Exercise 6.15 $h1$ calculates a cost of 1 for each tile that is in the wrong place, and for all others a cost of 0. This corresponds to the assumption that one can bring each tile to its destination in one step. This is obviously less than the actual number of steps required in most game states. An overestimation of the number of steps is therefore not possible. So h_1 is admissible. h_2 assumes that one can bring each tile to the goal on the shortest path. Since there is no shorter path than this Manhattan distance, h_2 is also admissible.

Exercise 6.16

(a)



11.7 Reasoning with Uncertainty

Exercise 7.1

1. $P(\Omega) = \frac{|\Omega|}{|\Omega|} = 1$.
2. $P(\emptyset) = 1 - P(\Omega) = 0$.
3. For pairwise exclusive events A and B: $P(A \vee B) = \frac{|A \vee B|}{|\Omega|} = \frac{|A|+|B|}{|\Omega|} = P(A) + P(B)$.
4. For two complementary events A and $\neg A$: $P(A) + P(\neg A) = P(A) + P(\Omega - A) = \frac{|A|}{|\Omega|} + \frac{|\Omega - A|}{|\Omega|} = \frac{|\Omega|}{|\Omega|} = 1$.
5. $P(A \vee B) = \frac{|A \vee B|}{|\Omega|} = \frac{|A|+|B|-|A \wedge B|}{|\Omega|} = P(A) + P(B) - P(A \wedge B)$.
6. For $A \subseteq B$: $P(A) = \frac{|A|}{|\Omega|} \leq \frac{|B|}{|\Omega|} = P(B)$.
7. According to Definition 7.1: $A_1 \vee \cdots \vee A_n = \Omega$ and $\sum_{i=1}^{n} P(A_i) = \sum_{i=1}^{n} \frac{|A_i|}{|\Omega|} = \frac{\sum_{i=1}^{n} |A_i|}{|\Omega|} = \frac{|A_1 \vee \cdots \vee A_n|}{|\Omega|} = \frac{|\Omega|}{|\Omega|} = 1$.

Exercise 7.2

(a)

$$P(y > 3 | Class = \text{good}) = 7/9 \quad P(Class = \text{good}) = 9/13 \quad P(y > 3) = 7/13,$$

$$P(Class = \text{good} | y > 3) = \frac{P(y > 3 | Class = \text{good}) \cdot P(Class = \text{good})}{P(y > 3)} = \frac{7/9 \cdot 9/13}{7/13} = 1,$$

$$P(Class = \text{good} | y \leq 3) = \frac{P(y \leq 3 | Class = \text{good}) \cdot P(Class = \text{good})}{P(y \leq 3)} = \frac{2/9 \cdot 9/13}{6/13} = \frac{1}{3}.$$

(b) $P(y > 3 | Class = \text{good}) = 7/9$.

Exercise 7.3

(a) eight events.
(b)

$$P(Prec = \text{dry}|\, Sky = \text{clear}, Bar = \text{rising}) = \frac{P(Prec = \text{dry}, Sky = \text{clear}, Bar = \text{rising})}{P(Sky = \text{clear}, Bar = \text{rising})} = \frac{0.4}{0.47} = 0.85.$$

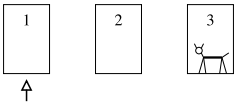
(c) Missing row in the table:

Cloudy	Falling	Raining	0.12

$$P\,(Prec = \text{raining}|\, Sky = \text{cloudy}) = \frac{P\,(Prec = \text{raining}, Sky = \text{cloudy})}{P\,(Sky = \text{cloudy})} = \frac{0.23}{0.35} = 0.66.$$

(d) The missing probability measure is 0.15. The indifference principle (Definition 7.5) now requires that both missing rows be symmetrically allocated the value 0.075.

Exercise 7.4



The candidate first chose door number 1. Then the host opened door number 3. We introduce the abbreviation A_i for "Car is behind door number i (before the experiment begins) and M_i for "Host (moderator) opens door number i". Then $P(A_1) = P(A_2) = P(A_3) = 1/3$ and we compute

$$\begin{aligned} P\,(A_1\,|M_3\,) &= \frac{P\,(M_3\,|A_1\,)\,P\,(A_1)}{P\,(M_3)} \\ &= \frac{P\,(M_3\,|A_1\,)\,P\,(A_1)}{P\,(M_3\,|A_1\,)\,P\,(A_1) + P\,(M_3\,|A_2\,)\,P\,(A_2) + P\,(M_3\,|A_3\,)\,P\,(A_3)} \\ &= \frac{1/2 \cdot 1/3}{1/2 \cdot 1/3 + 1 \cdot 1/3 + 0 \cdot 1/3} = \frac{1/2 \cdot 1/3}{1/2} = 1/3, \\ P\,(A_2\,|M_3\,) &= \frac{P\,(M_3\,|A_2\,)\,P\,(A_2)}{P\,(M_3)} = \frac{1 \cdot 1/3}{1/2} = 2/3. \end{aligned}$$

Thus it is clear that it is better to switch to the other door.

Exercise 7.5 The Lagrange function reads $L = -\sum_{i=1}^{n} p_i \ln p_i + \lambda(\sum_{i=1}^{n} p_i - 1)$. Setting the partial derivatives with respect to p_i and p_j equal to zero yields

$$\frac{\partial L}{\partial p_i} = -\ln p_i - 1 + \lambda = 0 \quad \text{and} \quad \frac{\partial L}{\partial p_j} = -\ln p_j - 1 + \lambda = 0.$$

Subtraction of these two equations results in $p_i = p_j$ for all $i, j \in \{1,\ldots, n\}$. Thus $p_1 = p_2 = \cdots = p_n = 1/n$.

Exercise 7.6

PIT input data:

```
var A{t,f}, B{t,f};
P([A=t]) = 0.5;
P([B=t] | [A=t]) = 0.94;
QP([B=t]);
```

PIT output:

```
Reporting State of Queries

Nr Truthvalue Probability Query
1 UNSPECIFIED 7.200e-01 QP([B=t]);
```

Because PIT operates numerically, it cannot compute symbolically with the parameters α and β, rather only with concrete numbers.

Exercise 7.7
If: $p_1 = P(A, B)$, $p_2 = P(A, \neg B)$, $p_3 = P(\neg A, B)$, $p_4 = P(\neg A, \neg B)$

$$\text{The constraints are : } p_1 + p_2 = \alpha \tag{11.1}$$

$$p_4 = 1 - \beta \tag{11.2}$$

$$p_1 + p_2 + p_3 + p_4 = 1 \tag{11.3}$$

It follows that $p_3 = \beta - \alpha$. From (11.1) we infer, based on indifference, that $p_1 = p_2 = \alpha/2$. Thus $P(B) = p_1 + p_3 = \alpha/2 + \beta - \alpha = \beta - \alpha/2 = P(A \vee B) - 1/2P(A)$. The

```
var A{t,f}, B{t,f};
P([A=t]) = 0.6;
P([B=t] OR [A=t]) = 0.94;
```

corresponding PIT program reads `QP([B=t]).`

Exercise 7.8 The Lagrange function reads

$$L = -\sum_{i=1}^{4} p_i \ln p_i + \lambda_1(p_1 + p_2 - \alpha) + \lambda_2(p_1 + p_3 - \gamma) + \lambda_3(p_1 + p_2 + p_3 + p_4 - 1). \tag{11.4}$$

Taking partial derivatives for p_1, p_2, p_3, p_4 we obtain

$$\frac{\partial L}{\partial p_1} = -\ln p_1 - 1 + \lambda_1 + \lambda_2 + \lambda_3 = 0, \tag{11.5}$$

$$\frac{\partial L}{\partial p_2} = -\ln p_2 - 1 + \lambda_1 + \lambda_3 = 0, \tag{11.6}$$

$$\frac{\partial L}{\partial p_3} = -\ln p_3 - 1 + \lambda_2 + \lambda_3 = 0, \tag{11.7}$$

$$\frac{\partial L}{\partial p_4} = -\ln p_4 - 1 + \lambda_3 = 0 \tag{11.8}$$

and compute

$$(11.5)-(11.6): \ \ln p_2 - \ln p_1 + \lambda_2 = 0, \tag{11.9}$$

$$(11.7)-(11.8): \ \ln p_4 - \ln p_3 + \lambda_2 = 0, \tag{11.10}$$

from which it follows that $p_3/p_4 = p_1/p_2$, which we immediately substitute into (7.12):

$$\begin{aligned} &\frac{p_2 p_3}{p_4} + p_2 + p_3 + p_4 = 1 \\ \Leftrightarrow \quad &p_2\left(1 + \frac{p_3}{p_4}\right) + p_3 + p_4 = 1 \\ (7.10)-(7.11): \quad &p_2 = p_3 + \alpha - \gamma \end{aligned} \tag{11.11}$$

which, substituted for p_2, yields $(p_3 + \alpha - \gamma)(1 + \frac{p_3}{p_4}) + p_3 + p_4 = 1$

$$(7.10) \text{ in } (7.12): \quad \alpha + p_3 + p_4 = 1. \tag{11.12}$$

Thus we eliminate p_4 in (11.8), which results in

$$(p_3 + \alpha - \gamma)\left(1 + \frac{p_3}{1 - \alpha - p_3}\right) + 1 - \alpha = 1 \tag{11.13}$$

$$\Leftrightarrow (p_3 + \alpha - \gamma)(1 - \alpha - p_3 + p_3) = \alpha(1 - \alpha - p_3) \tag{11.14}$$

$$\Leftrightarrow (p_3 + \alpha - \gamma)(1 - \alpha) = \alpha(1 - \alpha - p_3) \tag{11.15}$$

$$\Leftrightarrow p_3 + \alpha - \gamma - \alpha p_3 - \alpha^2 + \alpha\gamma = \alpha - \alpha^2 - \alpha p_3 \tag{11.16}$$

$$\Leftrightarrow p_3 = \gamma(1 - \alpha). \tag{11.17}$$

With (11.12) it follows that $p_4 = (1 - \alpha)(1 - \gamma)$ and with (11.11) we obtain $p_2 = \alpha(1 - \gamma)$ and $p_1 = \alpha\gamma$.

Exercise 7.9

(a)

$$M = \begin{pmatrix} 0 & 1000 \\ 10 & 0 \end{pmatrix}.$$

(b)

$$M \cdot \begin{pmatrix} p \\ 1-p \end{pmatrix} = \begin{pmatrix} 1000 \cdot (1-p) \\ 10 \cdot p \end{pmatrix}.$$

Because the decision between *delete* or *read* is found by establishing the minimum of the two values, it is sufficient to compare the two values: $1000(1–p) < 10p$, which results in $p > 0.99$. In general, for a matrix $M = \begin{pmatrix} 0 & k \\ 1 & 0 \end{pmatrix}$ we compute the threshold $k/(k + 1)$.

Exercise 7.10

(a) Because A and B are independent, $P(A|B) = P(A) = 0.2$.
(b) By applying the conditioning rule (page 176) we obtain

$$P(C|A) = P(C|A, B)P(B) + P(C|A, \neg B)P(\neg B) = 0.2 \cdot 0.9 + 0.1 \cdot 0.1 = 0.19.$$

Exercise 7.11

(a)

$$\begin{aligned}
P\,(Al) &= P\,(Al,\ Bur,\ Ear) + P\,(Al,\ \neg Bur,\ Ear) \\
&\quad + P\,(Al,\ Bur,\ \neg Ear) + P\,(Al,\ \neg Bur,\ \neg Ear) \\
&= P\,(Al|Bur,\ Ear)\ P\,(Bur,\ Ear) + P\,(Al|\neg Bur,\ Ear)\ P\,(\neg Bur,\ Ear) \\
&\quad + P\,(Al|Bur,\ \neg Ear)\ P\,(Bur,\ Ear) \\
&\quad + P\,(Al|\neg Bur,\ \neg Ear)\ P\,(\neg Bur,\ \neg Ear) \\
&= 0.95 \cdot 0.001 \cdot 0.002 + 0.29 \cdot 0.999 \cdot 0.002 + 0.94 \cdot 0.001 \cdot 0.998 \\
&\quad + 0.001 \cdot 0.999 \cdot 0.998 \\
&= 0.00252, \\
P\,(J) &= P\,(J,\ Al) + P\,(J,\ \neg Al) = P\,(J|Al)\ P\,(Al) + P\,(J|\neg Al)\ P\,(\neg Al) \\
&= 0.9 \cdot 0.0025 + 0.05 \cdot (1 - 0.0025) = 0.052, \\
P\,(M) &= P\,(M,\ Al) + P\,(M,\ \neg Al) = P\,(M|Al)\ P\,(Al) + P\,(M|\neg Al)\ P\,(\neg Al) \\
&= 0.7 \cdot 0.0025 + 0.01 \cdot (1 - 0.0025) = 0.0117.
\end{aligned}$$

(b)

$$\begin{aligned}
P\,(Al|Bur) &= P\,(Al|Bur, Ear)\ P\,(Ear) + P\,(Al|Bur, \neg Ear)\ P\,(\neg Ear) \\
&= 0.95 \cdot 0.002 + 0.94 \cdot 0.998 = 0.94002, \\
P\,(M|Bur) &= P\,(M,\ Bur)\,/P\,(Bur) = [P\,(M,\ Al,\ Bur) + P\,(M,\ \neg Al,\ Bur)] \\
&\quad /P\,(Bur) \\
&= [P\,(M|Al)\ P\,(Al|Bur)\ P\,(Bur) + P\,(M|\neg Al)\ P\,(\neg Al|Bur)\ P\,(Bur)] \\
&\quad /P\,(Bur) \\
&= P\,(M|Al)\ P\,(Al|Bur) + P\,(M|\neg Al)\ P\,(\neg Al|Bur) \\
&= 0.7 \cdot 0.94002 + 0.01 \cdot 0.05998 = 0.659.
\end{aligned}$$

(c)

$$P(Bur|M) = \frac{P(M|Bur)\,P(Bur)}{P(M)} = \frac{0.659 \cdot 0.001}{0.0117} = 0.056.$$

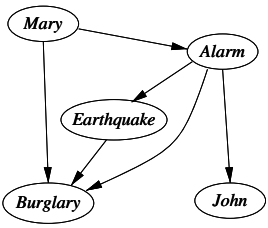
(d)

$$\begin{aligned}
P(Al|J,M) &= \frac{P(Al,J,M)}{P(J,M)} = \frac{P(Al,J,M)}{P(Al,J,M) + P(\neg Al,J,M)} \\
&= \frac{1}{1 + \frac{P(\neg Al,J,M)}{P(Al,J,M)}} = \frac{1}{1 + \frac{P(J|\neg Al)\,P(M|\neg Al)\,P(\neg Al)}{P(J|Al)\,P(M|Al)\,P(Al)}} \\
&= \frac{1}{1 + \frac{0.05 \cdot 0.01 \cdot 0.9975}{0.9 \cdot 0.7 \cdot 0.0025}} = 0.761, \\
P(J,M|Bur) &= P(J,M|Al)\,P(Al|Bur) + P(J,M|\neg Al)\,P(\neg Al|Bur)) \\
&= P(J|Al)\,P(M|Al)\,P(Al|Bur) \\
&\quad + P(J|\neg Al)\,P(M|\neg Al)\,P(\neg Al|Bur)) \\
&= 0.9 \cdot 0.7 \cdot 0.94 + 0.05 \cdot 0.01 \cdot 0.06 = 0.5922, \\
P(Al|\neg Bur) &= P(Al|\neg Bur, Ear)\,P(Ear) + P(Al|\neg Bur, \neg Ear)\,P(\neg Ear) \\
&= 0.29 \cdot 0.002 + 0.001 \cdot 0.998 = 0.00158, \\
P(J,M|\neg Bur) &= P(J,M|Al)\,P(Al|\neg Bur) + P(J,M|\neg Al)\,P(\neg Al|\neg Bur)) \\
&= P(J|Al)\,P(M|Al)\,P(Al|\neg Bur) \\
&\quad + P(J|\neg Al)\,P(M|\neg Al)\,P(\neg Al|\neg Bur)) \\
&= 0.9 \cdot 0.7 \cdot 0.00158 + 0.05 \cdot 0.01 \cdot 0.998 = 0.00149, \\
P(J,M) &= P(J,M|Bur)\,P(Bur) + P(J,M|\neg Bur)\,P(\neg Bur) \\
&= 0.5922 \cdot 0.001 + 0.00149 \cdot 0.999 = 0.00208, \\
P(Bur|J,M) &= \frac{P(J,M|Bur)\,P(Bur)}{P(J,M)} = 0.5922 \cdot 0.001/0.00208 = 0.284.
\end{aligned}$$

(e) $P(J)P(M) = 0.052 \cdot 0.0117 = 0.00061$, but $P(J, M) = 0.00208$ (see above). Therefore $P(J)P(M) \neq P(J, M)$.

(f) See Sect. 7.4.5.

(g)



Other solutions are possible due to the counter-productive variable order. Thus we learn: Always use a variable order that respects causality!

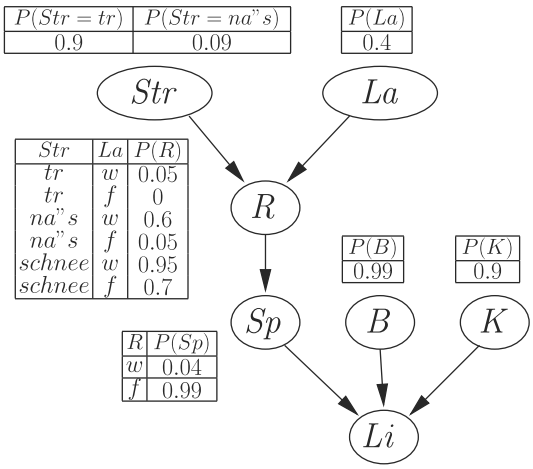


Fig. 11.1 Bayesian network for the bicycle light. The CPT for Li is given in the problem statement

(h) Only the CPT of the alarm node changes. All of the other nodes are independent of earthquake or independent of earthquake given alarm.

(i)

Bur	P(Al)
t	0.94
f	0.0016

Exercise 7.12

(a), (b) See Fig. 11.1.

(c) The edge (*Str*, *Li*) is missing because *Li* and *Str* are conditionally independent given *V*, i.e., $P(Li|V, Str) = P(Li|V)$ because the street condition has no direct influence on the light, rather only over the dynamo and the voltage it generates. (The conditional independence *Li* and *Str* given *V* cannot be shown here by calculation based on available data for $P(Li|V, Str)$ and $P(Li|V)$, rather it can only be asserted.)

(d) For the given CPTs:

$$\begin{aligned} P\,(R|Str) &= P\,(R|Str, Flw)\;P\,(Flw) + P\,(R|Str, \neg Flw)\;P\,(\neg Flw) \\ &= 0.95 \cdot 0.4 + 0.7 \cdot 0.6 = 0.8, \\ P\,(V|Str) &= P\,(V|R)\;P\,(R|Str) + P\,(V|\neg R)\;P\,(\neg R|Str) \\ &= 0.04 \cdot 0.8 + 0.99 \cdot 0.2 = 0.23. \end{aligned}$$

11.8 Machine Learning and Data Mining

Exercise 8.1

(a) The agent is a function A, which maps the vector $(t_1, t_2, t_3, d_1, d_2, d_3, f_1, f_2, f_3)$ consisting of three values each for temperature, pressure, and humidity to a class value $k \in$ sunny, cloudy, rainy.

(b) The training data file consists of a line of the form

$$t_{i1}, t_{i2}, t_{i3}, d_{i1}, d_{i2}, d_{i3}, f_{i1}, f_{i2}, f_{i3}, k_i$$

for every single training data item. The index i runs over all training data. A concrete file could for example begin:

```
17,17,17,980,983,986,63,61,50,sunny
20,22,25,990,1014,1053,65,66,69,sunny
20,22,18,990,979,929,52,60,61,rainy
```

Exercise 8.2 We can see the symmetry directly using the definition of correlation coefficients, or of covariance. Swapping i and j does not change the value. For the diagonal elements we calculate

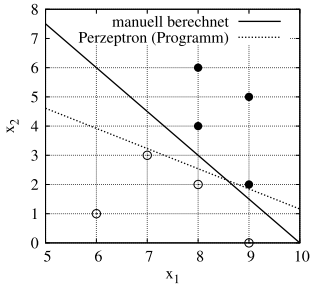
$$K_{ii} = \frac{\sigma_{ii}}{s_i \cdot s_i} = \frac{\sum_{p=1}^{N}(x_i^p - \bar{x}_i)(x_i^p - \bar{x}_i)}{\sum_{p=1}^{N}(x_i^p - \bar{x}_i)^2} = 1.$$

Exercise 8.3 The sequence of values for $\boldsymbol{w}$ is

$$(1, 1),\ (2.2, -0.4),\ (1.8, 0.6),\ (1.4, 1.6),\ (2.6, 0.2),\ (2.2, 1.2)$$

Exercise 8.4

(a)



(b) Drawing a straight line in the graph yields

$$1.5x_1 + x_2 - 15 > 0.$$

After 442 iterations, the perceptron learning algorithms with the start vector $\boldsymbol{w} = (1, 1, 1)$ returns:

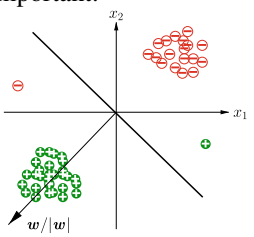
$$\boldsymbol{w} = (11, 16, -129).$$

This corresponds to: $0.69x_1 + x_2 - 8.06 > 0$

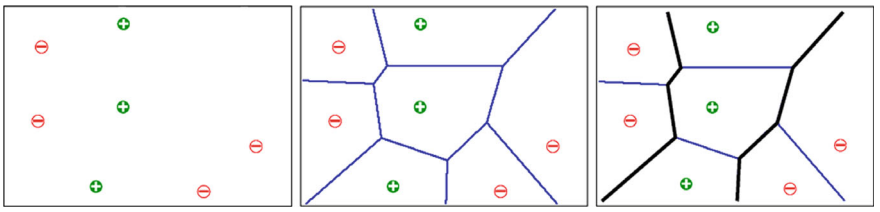
Exercise 8.5

(a) The vector $\sum_{x \in M_+} x$ points in an "average" direction of all positive points and $\sum_{x \in M_-} x$ points in an "average" direction of all negative points. The difference of these vectors points from the negative points toward the positive points. The dividing line is then perpendicular to this difference vector.

(b) The point cloud of the positive and negative points dominates during the calculation of $\boldsymbol{w}$. The two outliers hardly play a role here. For determining the dividing line, however, they are important.



Exercise 8.6



Exercise 8.7

(a) Nearest neighbor is (8, 4), thus class 1.

(b) $k = 2$: class undefined since one instance of class 1 and one instance of class 0. $k = 3$: decision 2:1 for class 0. $k = 5$: decision 2:3 for class 1.

Exercise 8.9

(a) For $x = \min$ we have $\tilde{x} = 0$ and for $x = \max$ we have $\tilde{x} = 1$. Between the two points must be a straight line because x only appears linearly in the formula.

(b) We are looking for a function $f(x) = mx + b$ which passes through the points $(\min, -1)$ and $(\max, 1)$. By substituting the points into the equation of the straight line we get

$$m \cdot \min + b = -1 \quad \text{und} \quad m \cdot \max + b = 1.$$

Subtracting the first equation from the second leads to $m = \frac{2}{\max - \min}$ und $b = 1 - \frac{2\max}{\max - \min}$. The desired formula is given by

$$f(x) = \frac{2(x - \max)}{\max - \min} + 1$$

(c) Let $\tilde{x} = \frac{x-\mu}{\sigma}$ and let $x_1, \ldots, x_n$ be the values of the variable x in the training data. Then $\mu = \frac{1}{n}\sum_{j=1}^{n} x_j$. To find the average of the normalized variable $\tilde{x}$ we calculate

$$\tilde{\mu} = \frac{1}{n}\sum_{j=1}^{n} \tilde{x}_j = \frac{1}{n\sigma}\sum_{j=1}^{n}(x_j - \mu) = \frac{1}{\sigma}\left(\frac{1}{n}\sum_{j=1}^{n} x_j - \frac{1}{n}\sum_{j=1}^{n}\mu\right)$$
$$= \frac{1}{\sigma}(\mu - \mu) = 0.$$

With $\sigma := \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x-\mu)^2}$ we calculate

$$\tilde{\sigma} := \sqrt{\frac{1}{n-1}\sum_{i=j}^{n}\left(\frac{x_j - \mu}{\sigma} - 0\right)^2} = \frac{1}{\sigma}\cdot\sigma = 1.$$

Exercise 8.10

(a)

$$P(\text{classified as positive} \mid \text{positive}) = \frac{|\text{positive and classified as positive}|}{|\text{positive}|} = \frac{TP}{TP + FN}$$

The first equation follows from the definition in Eq. 7.16 and the second from the confusion matrix in Table 8.4. Similarly we obtain

$$P(\text{classified as negative} \mid \text{negative}) = \frac{|\text{negative and classified as negative}|}{|\text{negative}|} = \frac{TN}{FP + TN}$$

(b)

$$1 - \text{specificity} = 1 - \frac{TN}{FP + TN} = \frac{FP}{FP + TN}$$

Exercise 8.11

$$\text{sensitivity} = \frac{0.23}{0.23 + 0.05} = 0.82 \qquad \text{specificity} = \frac{0.41}{0.31 + 0.41} = 0.57$$
$$\text{precision} = \frac{0.23}{0.31 + 0.23} = 0.43 \qquad \text{F-score} = \frac{2 \cdot 0.43 \cdot 0.82}{0.43 + 0.82} = 0.56$$

Exercise 8.12 $\lim_{x\to 0} x \log_2 x = \lim_{x\to 0} \frac{\log_2 x}{1/x} = \lim_{x\to 0} \frac{1/(x \ln 2)}{-1/x^2} = \lim_{x\to 0} \frac{-x}{\ln 2} = 0$, where for the second equation l'Hospital's rule was used.

Exercise 8.13 (a) 0 (b) 1 (c) 1.5 (d) 1.875 (e) 2.32 (f) 2

Exercise 8.14

(a) From $\log_2 x = \ln x / \ln 2$, it follows that $c = 1/\ln 2 \approx 1.44$.
(b) Since both entropy functions only differ by a constant factor, the position of the entropy maximum does not change. Extrema under constraints also maintains the same position. Thus the factor c does not cause a problem for the MaxEnt method. For learning of decision trees, the information gains of various attributes are compared. Because here only the ordering matters, but not the absolute value, the factor c does not cause a problem here either.

Exercise 8.15

(a) For the first attribute we calculate

$$\begin{aligned} G(D, x_1) &= H(D) - \sum_{i=6}^{9} \frac{|D_{x_1=i}|}{|D|} H\left(D_{x_1=i}\right) \\ &= 1 - \left(\frac{1}{8} H\left(D_{x_1=6}\right) + \frac{1}{8} H\left(D_{x_1=7}\right) + \frac{3}{8} H\left(D_{x_1=8}\right) + \frac{3}{8} H\left(D_{x_1=9}\right) \right) \\ &= 1 - \left(0 + 0 + \frac{3}{8} \cdot 0.918 + \frac{3}{8} \cdot 0.918 \right) = 0.311 \end{aligned}$$

$G(D, x_2) = 0.75$, thus x_2 is selected. For $x_2 = 0, 1, 3, 4, 5, 6$ the decision is clear. For $x_2 = 2$, x_1 is selected. The tree then has the form

```
x2 = 0: 0 (1/0)
x2 = 1: 0 (1/0)
x2 = 2:
| x1 = 6: 0 (0/0)
| x1 = 7: 0 (0/0)
| x1 = 8: 0 (1/0)
| x1 = 9: 1 (1/0)
x2 = 3: 0 (1/0)
x2 = 4: 1 (1/0)
x2 = 5: 1 (1/0)
x2 = 6: 1 (1/0)
```

(b) Information gain for the continuous attribute x_2 as the root node:

Threshold Θ	0	1	2	3	4	5
$G(D, x_2 \leq \Theta)$	0.138	0.311	0.189	0.549	0.311	0.138

Since $G(D, x_2 \leq 3) = 0.549 > 0.311 = G(D, x_1)$, $x_2 \leq 3$ is selected. For $x_2 \leq 3$ the classification is not unique. We calculate $G(D_{x_2 \leq 3}, x_1) = 0.322$, $G(D_{x_2 \leq 3}, x_2 \leq 0) = 0.073$, $G(D_{x_2 \leq 3}, x_2 \leq 1) = 0.17$, $G(D_{x_2 \leq 3}, x_2 \leq 2) = 0.073$. Thus x_1 is selected. For $x_1 = 9$ the classification is not unique. Here the decision is clearly $G(D_{(x_2 \leq 3, x_1 = 9)}, x_1) = 0$, $G(D_{(x_2 \leq 3, x_1 = 9)}, x_2 \leq 1) = 1$, $x_2 \leq 1$ is chosen, and

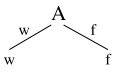
the tree once again has 100% correctness.

```
Tree:
x2 <= 3:
| x1 = 6: 0 (1/0)
| x1 = 7: 0 (0/0)
| x1 = 8: 0 (1/0)
| x1 = 9:
| | x2 <= 1: 0 (1/0)
| | x2 > 1: 1 (1/0)
x2 > 3: 1 (3/0)
```

Exercise 8.16

(a) 100% for the training data, 75% for the test data, 90% total correctness.
(b) $(A \wedge \neg C) \vee (\neg A \wedge B)$

(c)



66.6% correctness for the training data
100% correctness for the test data
80% total correctness

Exercise 8.17

(a) Equation (8.7) for calculating the information gain of an attribute A reads

$$\text{InfoGain}(D,\ A) = H(D) - \sum_{i=1}^{n} \frac{|D_i|}{|D|} H(D_i),$$

where n is the number of the values of the currently observed attribute. If the attribute A is tested somewhere in the subtree as a successor of the value a_j, then only the value $A = a_j$ will occur in the dataset D_j and each of its subsets D'. Thus $D' = D'_j$ and for all $k \neq j$, $|D'_k| = 0$ and we obtain

$$\text{InfoGain}(\text{D}', \text{A}) = H(D') - \sum_{i=1}^{n} \frac{|D'_i|}{|D'|} H(D'_i) = H(D'_j) - \frac{|D'_j|}{|D'_j|} H(D'_j) = 0.$$

The repeated attribute thus has an information gain of zero, and thus it is no longer used.

(b) From every continuous attribute A a binary attribute $A > \Theta_{D,A}$ is generated. If, further down in the tree, the attribute A is discretized again with a different threshold $\Theta_{D',A}$, then the attribute $A > \Theta_{D',A}$ is different from $A > \Theta_{D,A}$. If it then has a higher information gain than all other attributes, it will be used in the tree. However, this also means that for multiple occurrences of a continuous attribute, the thresholds must be different.

Exercise 8.18 (a)

$P(Sky = clear) = 0.65$
$P(Bar = rising) = 0.67$

Sky	*Bar*	$P(Prec = dry \mid Sky, Bar)$
Clear	Rising	0.85
Clear	Falling	0.44
Cloudy	Falling	0.45
Cloudy	Falling	0.2

(b) $P(Sky = clear) = 0.65$

Bar	$P(Prec = dry \mid Bar)$
Rising	0.73
Falling	0.33

Sky	$P(Bar = rising \mid Sky)$
Clear	0.72
Cloudy	0.57

(c) The necessary CPTs for $P(Prec|Sky, Bar)$ and $P(Bar|Sky)$, as well as $P(Sky)$ are already known.

(d) $\boldsymbol{P}_a = (0.37, 0.065, 0.095, 0.12, 0.11, 0.13, 0.023, 0.092)$ $\boldsymbol{P}_b = (0.34, 0.13, 0.06, 0.12, 0.15, 0.054, 0.05, 0.1)$ $\boldsymbol{P} = (0.4, 0.07, 0.08, 0.1, 0.09, 0.11, 0.03, 0.12)$ (original distribution) Quadratic distance: $d_q(\boldsymbol{P}_a, \boldsymbol{P}) = 0.0029$, $d_q(\boldsymbol{P}_b, \boldsymbol{P}) = 0.014$ Kullback–Leibler dist.: $d_k(\boldsymbol{P}_a, \boldsymbol{P}) = 0.017$, $d_k(\boldsymbol{P}_b, \boldsymbol{P}) = 0.09$
Both distance metrics show that the network (a) approximates the distribution better than network (b). This means that the assumption that *Prec* and *Sky* are conditionally independent given *Bar* is less likely true than the assumption that *Sky* and *Bar* are independent.

(e) $textbi\,P_c = (0.4, 0.07, 0.08, 0.1, 0.09, 0.11, 0.03, 0.12)$. This distribution is exactly equal to the original distribution $\boldsymbol{P}$. This is not surprising because there are no missing edges in the network. This means that no independencies were assumed.

Exercise 8.19 We can immediately see that scores and perceptrons are equivalent by comparing their definitions. Now for the equivalence to Naive Bayes: First we establish that $P(K \mid S_1, \ldots, S_n) > 1/2$ is equivalent to $P(K \mid S_1, \ldots, S_n) > P(\neg K \mid S_1, \ldots, S_n)$ because $P(\neg K \mid S_1, \ldots, S_n) > 1 - P(K \mid S_1, \ldots, S_n)$. We are in fact dealing with a binary Naive Bayes classifier here.

We apply the logarithm to the Naive Bayes formula

$$P(K|S_1, \ldots, S_n) = \frac{P(S_1|K) \cdot \cdots \cdot P(S_n|K) \cdot P(K)}{P(S_1, \ldots, S_n)},$$

and obtain

$$\log P\,(K|S_1, \ldots, S_n) = \log P\,(S_1|K) + \cdots + \log P\,(S_n|K) + \log P\,(K) - \log P\,(S_1, \ldots, S_n)\,. \tag{11.18}$$

To obtain a score, we must interpret the variables S_1, ..., S_n as numeric variables with the values 1 and 0. We can easily see that

$$\log P(S_i|K) = (\log P(S_i = 1|K) - \log P(S_i = 0|K))S_i + \log P(S_i = 0|K).$$

It follows that

$$\sum_{i=1}^{n} \log P\,(S_i|K) = \sum_{i=1}^{n} (\log P\,(S_i = 1|K) - \log P\,(S_i = 0|K))S_i + \sum_{i=1}^{n} \log P\,(S_i = 0|K)\,.$$

we define $w_i = \log P(S_i = 1|K) - \log P(S_i = 0|K)$ and $c = \sum_{i=1}^{n} \log P(S_i = 0|K)$ and simplify

$$\sum_{i=1}^{n} \log P(S_i|K) = \sum_{i=1}^{n} w_i S_i + c.$$

Substituted in (11.18) we obtain

$$\log P(K|S_1, \ldots, S_n) = \sum_{i=1}^{n} w_i S_i + c + \log P(K) - \log P(S_1, \ldots, S_n).$$

For the decision K it must be the case, according to the definition of the Bayes classifier, that $\log P(K|S_1, \ldots, S_n) > \log(1/2)$. Thus it must either be the case that

$$\sum_{i=1}^{n} w_i S_i + c + \log P(K) - \log P(S_1, \ldots, S_n) > \log(1/2)$$

or that

$$\sum_{i=1}^{n} w_i S_i > \log 1/2 - c - \log P(K) + \log P(S_1, \ldots, S_n),$$

with which we have defined a score with the threshold $\Theta = \log 1/2 - c - \log P(K) + \log P(S_1, \ldots, S_n)$. Because all of the transformations can be reversed, we can also transform any score into a Bayesian classifier. With that, the equivalence has been shown.

Exercise 8.20 Taking the logarithm of (8.8) results in

$$\log P(I|s_1, \ldots, s_n) = \log c + \log P(I) + \sum_{i=1}^{l} n_i \log P(w_i|I).$$

Thereby very small negative numbers become moderate negative numbers. Since the logarithm function grows monotonically, to determine the class we maximize according to the rule

$$I_{Naive-Bayes} = \underset{I \in \{w,f\}}{\operatorname{argmax}} \log P(I|s_1, \ldots, s_n).$$

The disadvantage of this method is the somewhat longer computation time in the learning phase for large texts. During classification the time does not increase, because the values $\log P(I|s_1, \ldots, s_n)$ can be saved during learning.

Exercise 8.23 Let f be strictly monotonically increasing, that is, $\forall x, y\ x < y \Rightarrow f(x) < f(y)$. If now $d_1(s,t) < d_1(u,v)$, then clearly $d_2(s,t) = f(d_1(s,t)) < f(d_1(u,v)) = d_2(u,v)$. Because the inverse of f is also strictly monotonic, the reverse is true, that is, $d_2(s,t) < d_2(u,v) \Rightarrow d_1(s,t) < d_1(u,v)$. Thus it has been shown that $d_2(s,t) < d_2(u,v) \iff d_1(s,t) < d_1(u,v)$.

Exercise 8.24

$$\begin{aligned} x_1x_2 = 4, &\quad \text{and thus} \quad d_s(x_1,x_2) = \tfrac{\sqrt{23\cdot 26}}{4} = 6.11 \\ x_2x_3 = 2, &\quad \text{and thus} \quad d_s(x_2,x_3) = \tfrac{\sqrt{26\cdot 20}}{2} = 11.4 \\ x_1x_3 = 1, &\quad \text{and thus} \quad d_s(x_1,x_3) = \tfrac{\sqrt{23\cdot 20}}{1} = 21.4 \end{aligned}$$

Sentences 1 and 2 are most similar w.r.t. the distance metric d_s.

Exercise 8.25 The number of subsets of an n element set is 2^n. If we partition the set of our n data points into each subset and its complement, we get 2^n partitions. Since the complement of a subset is also a subset, each of these partitions has an identical copy. So there are 2^{n-1} disjoint partitions of the set in two parts. If we now add all other partitions, we get the so-called Bell number $B(n)$ with $B(n) \geq 2^{n-1}$. Therefore a clustering algorithm that enumerates all partitions has at least an exponential time complexity.

Exercise 8.26 Help for problems with KNIME: www.knime.org/forum.

11.9 Neural Networks

Exercise 9.1 We want to show that $f(\Theta + x) + f(\Theta - x) = 1$.

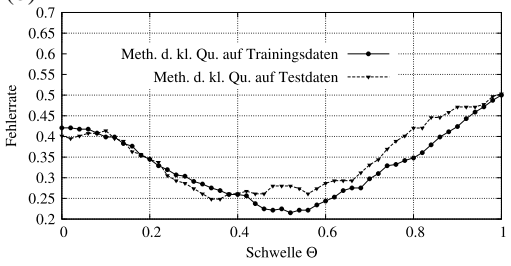
$$f(\Theta + x) = \frac{1}{1 + e^{-\frac{x}{T}}} = \frac{e^{\frac{x}{T}}}{1 + e^{\frac{x}{T}}}, \qquad f(\Theta - x) = \frac{1}{1 + e^{\frac{x}{T}}},$$
$$f(\Theta + x) + f(\Theta - x) = 1.$$

Exercise 9.3 Each pattern saved in the network has a size of n bits. The network has a total of $n(n-1)/2$ weights. If we reserve 16 bits per weight and define binary storage of size $16n(n-1)/2$ as equally large, then this can clearly store $N = 8n(n-1)/n = 4(n-1)$ patterns n bits in size. For large n we obtain $N = 8n$ as the limit. If we take the quotient α of the number of storable bits and the number of available storage cells, as in (9.11), then we obtain the value 1 for the list memory and the value $\alpha = 0.146n^2/(16n(n-1)/2) \approx 0.018$ for the Hopfield network. The classical storage thus has (for 16 bits per weight), a capacity roughly 55 times higher.

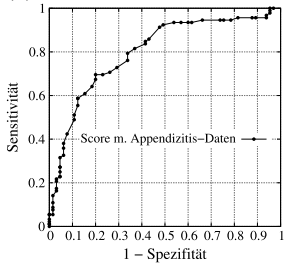
Exercise 9.4 (a) Mathematica program for the least square method.

```
LeastSq[q_,a_] := Module[{Nq,Na,m,A,b,w},
        Nq = Length[q]; m = Length[q[[1]]]; Na = Length[a];
        If[Nq != Na, Print["Length[q] != Length[a]"]; Exit, 0];
        A = Table[N[Sum[q[[p,i]] q[[p,j]], {p,1,Nq}]], {i,1,m}, {j,1,m}];
        b = Table[N[Sum[a[[p]] q[[p,j]], {p,1,Nq}]], {j,1,m}];
        w = LinearSolve[A,b]
]
LeastSq::usage = "LeastSq[x,y,f] computes from the query vectors q[[1]],...,
   q[[m]] a table of coefficients w[[i]] for a linear mapping f[x] =
        Sum[w[[i]] x[[i]], {i,1,m}] with f[q[[p]]] = a[[p]]."
```

(b)



(c)



Exercise 9.5

$$\sigma'(x) = \frac{d}{dx}\frac{1}{1+e^{-x}} = \frac{e^{-x}}{(1+e^{-x})^2} = \sigma(x)\frac{e^{-x}}{1+e^{-x}} = \sigma(x)\frac{1+e^{-x}-1}{1+e^{-x}}$$
$$= \sigma(x)(1-\sigma(x)).$$

Exercise 9.7 Three hidden neurons are sufficient to learn the patterns without error. Learning is not possible without error for less than three neurons. Explanation: eight different patterns can be encoded with 3 bits, but not with 2 bits. At least not binary.

Exercise 9.8 The formulas are the same, because backpropagation with sigmoid function in the last layer works exactly the same as logistic regression. It is easy to see that the formulas are the same if you insert the formula for $\Delta_p w_{ji}$ into the backpropagation learning rule for output neurons $\delta_j^{(p)}$, which yields $\Delta_p w_{ji} = \eta x_j^{(p)}(1 - x_j^{(p)})(t_j^{(p)} - x_j^{(p)})x_i^{(p)}$.

Exercise 9.9

(a) The network learns the task.
(b) The network cannot learn the XOR function, because a two-layer network is not more powerful than a linear one.

Exercise 9.10

(a) A mapping f is called linear if for all x, y, k it is the case that $f(x + y) = f(x) + f(y)$ and $f(kx) = kf(x)$. Now let f and g be linear mappings. Then $f(g(x + y)) = f(g(x) + g(y)) = f(g(x)) + f(g(y))$ and $f(g(kx)) = f(kg(x)) = kf(g(x))$. Thus, successive executions of linear mappings are a linear mapping.
(b) We observe two arbitrary output neurons j and k. Each of the two represents a class. Classification is done by forming the maximum of the two activations. Let $net_j = \sum_i w_{ji}x_i$ and $net_k = \sum_i w_{ki}x_i$ be the weighted sum of values arriving at neurons j and k. Furthermore, let $net_j > net_k$. Without an activation function, class j is output. Now if a strictly monotonic activation function f is applied, nothing changes in the result because, due to the function being strictly monotonic, $f(net_j) > f(net_k)$.

Exercise 9.11

(a) For a stride of one, each input neuron is injectively assigned to an output neuron. Therefore, the number of neurons in both layers (excluding the padding neurons) is the same. Since the ResNet calculates $f_2(f_1(\mathbf{x})) + \mathbf{x}$, the two vectors $f_2(f_1(\mathbf{x}))$ and $\mathbf{x}$ must be the same length. If f_1 and f_2 are both 3×3 convolutions with padding, this is the case.
(b) By a factor of s^2. For a stride of 3, this is a factor of 9.

Exercise 9.12
(a)

Parameter	Value
Learning rate η	0.01–2
First convolution layer num. filters or feature maps	5–20
First convolution layer padding type	Zeroes or copy
First convolution layer stride	1, 2, 3, 4
First convolution layer filter size	3, 4, 5, 6, 7, 8, 9
First pooling layer window size	2, 3, 4, 5, 6
Second convolution layer num. filters or feature maps	5–20
Second convolution layer padding type	zeroes or copy
Second convolution layer stride	1, 2, 3, 4
Second convolution layer filter size	3, 4, 5, 6, 7, 8, 9
Second pooling layer window size	2, 3, 4, 5, 6
Num. hidden neurons in layer before output layer	10–98

(b) If we use the values 0.01, 0.02, 0.05, 0.1, 0.2, …, 2 for the learning rate and a step size of 4 for the neurons in the layer before the output layer, we get

$$23 \cdot 16 \cdot 2 \cdot 4 \cdot 7 \cdot 5 \cdot 16 \cdot 2 \cdot 4 \cdot 7 \cdot 5 \cdot 23 = 10,617,241,600$$

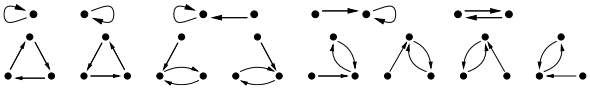
training runs for the network.
(c) Computation time $= \frac{10,617,241,600}{3600 \cdot 24 \cdot 365} \approx 337$ years

Exercise 9.13 The best-known deep learning systems achieve a correctness of 99.97% on the MNIST data.

Exercise 9.14 As can be seen in Fig. 9.32, the best highly optimized systems, which are trained on high-performance GPU clusters, achieve error rates of 2.3% on ImageNet in the ILSVRC competition.

Exercise 9.15 $f_1(x_1, x_2) = x_1^2, \quad f_2(x_1, x_2) = x_2^2$. Then the dividing line in the transformed space has the equation $y_1 + y_2 = 1$.

11.10 Reinforcement Learning

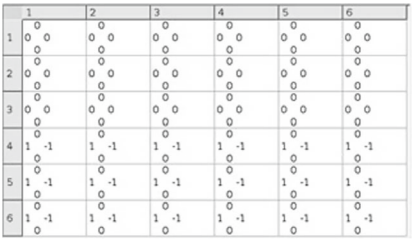
Exercise 10.1 (a) n^n (b) $(n-1)^n$
(c)
(d) 

Exercise 10.2 Value iteration leads to the following value tables. Depending on the order of the updates of the $\hat{V}$ values, the configurations of the intermediate states can vary. The end state, i.e., the last table (limit), remains the same.

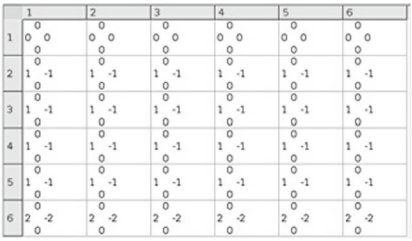
0	0	$\rightarrow$	0.81	0.9	$\rightarrow$	1.35	1.49	$\rightarrow \cdots \rightarrow$	2.36	2.62
0	0		0.73	1		1.21	1.66		2.12	2.91

Exercise 10.3

(c)



6×6 feedback for a completely smooth surface



6×6 feedback for a very heavy survace (like thick snow)

(d) We see that the longer a policy becomes (i.e., the more steps, for example, that a cycle of a cyclical strategy has), the closer the value γ must be to 1 because a higher value for γ makes a longer memory possible. However, value iteration converges that much more slowly.

Exercise 10.4 The value $V^\star(3, 3)$ at bottom right in the state matrix is changed as follows:

$$V^\star(3, 1) = 0.9V^\star(2, 1) = 0.9^2V^\star(2, 2) = 0.9^3V^\star(2, 3) = 0.9^4V^\star(3, 3). \quad (11.19)$$

This chain of equations follows from (10.6) because, for all given state transitions, the maximum immediate reward is $r(s, a) = 0$, and it is the case that

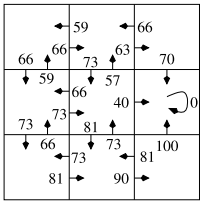
$$V^\star(s) = \max_a[r(s, a) + \gamma V^\star(\delta(s, a))] = \gamma V^\star(\delta(s, a)) = 0.9V^\star(\delta(s, a)).$$

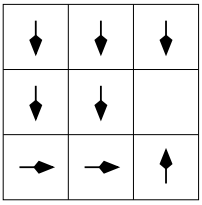
From (10.6) it also follows that $V^\star(3, 2) = 1 + 0.9V^\star(3, 1)$, because $r(s, a) = 1$ is maximal. Analogously it is true that $V^\star(3, 3) = 1 + 0.9V^\star(3, 2)$ and the circle closes. The two last equations together yield $V^\star(3, 3) = 1 + 0.9(1 + 0.9V^\star(3, 1))$. From (11.19) it follows that $V^\star(3, 1) = 0.9^4V^\star(3, 3)$. Substituted in $V^\star(3, 3)$, this yields

$$V^\star(3, 3) = 1 + 0.9(1 + 0.9^5V^\star(3, 3)),$$

from which the claim follows.

Exercise 10.5 Stable Q-values and an optimal policy:





Exercise 10.6

(a) Number of states = ℓ^n. Number of actions per state = ℓ^n. Number of policies = $(\ell^n)^{\ell^n} = \ell^{n\ell^n}$.

(b)

	$l=1$	$l=2$	$l=3$	$l=4$	$l=10$
$n=1$	1	4	27	256	10^{10}
$n=2$	1	256	3.9×10^{8}	1.8×10^{19}	10^{200}
$n=3$	1	1.7×10^{7}	4.4×10^{38}	3.9×10^{115}	10^{3000}
$n=4$	1	1.8×10^{19}	3.9×10^{154}	3.2×10^{616}	10^{40000}
$n=8$	1	3.2×10^{616}	1.4×10^{25043}	6.7×10^{315652}	$10^{800000000}$

(c) Per state there are now $2n$ possible actions. Thus there are $(2n)^{\ell^n}$ policies.

	$l=1$	$l=2$	$l=3$	$l=4$	$l=10$
textitn = 1	2	4	8	16	1024
$n=2$	4	256	2.6×10^{9}	4.3×10^{9}	1.6×10^{60}
$n=3$	6	1.7×10^{6}	1.0×10^{21}	6.3×10^{49}	1.4×10^{778}
$n=4$	8	2.8×10^{14}	1.4×10^{73}	1.6×10^{231}	7.9×10^{9030}
$n=8$	16	1.8×10^{308}	1.7×10^{7900}	1.6×10^{78913}	$1.8 \times 10^{120411998}$

(d) $10^{120411998}$ different policies can never be explored combinatorially, even if all of the available computers in the world were to operate on them in parallel. Thus "intelligent" algorithms are necessary to find an optimal or nearly optimal policy.