AI2 :Metoder i kunstig intelligens

Uncertain knowledge and reasoning

- **Chapter 13:** Quantifying uncertianty (Probability theory)
- **Chapter 14:** Probalistic reasoning (Bayesian network)
- **Chapter 15:** Probalistic reasoning over time (Markov process)
- **Chapter 16:** Making Simple Decisions (Decision networks)
- **Chapter 17:** Making Complex Decisions (Markov Decision Processes)

Learning

- **Chapter 18:** Learning from Examples (Supervised learning & Decision trees & Neural network)
- **Chapter 21:** Reinforcement Learning

Communication, perceiving, and acting

- (**Chapter 22:** Natural language processing)
- **Chapter 23:** Natural language for Communication
- (**Chapter 25:** Robotics)

Conclusions

- **Chapter 26:** Philosophical Fundations
- **Chapter 27:** AI: The Present and Future

Extra

- **Case-based reasoning**

<u>Uncertain knowledge and reasoning</u>

Probability theory

- Different variables i probability theory

- <u>Boolean (propositional) random variable</u>
    - Rain (is it raining?)
    - Rain = true
- <u>Discrete random variable (finite og infinite)</u>
    - e.g weather is one of <rain, sunny, cloudy, snow> (infinite)
    - Wheater = rain (finite)
- <u>Continuous random variable (bounded or unbounded)</u>
    - Temp = 11.6 (bounded)
    - Temp < 12.0 (unbounded)

- Unconditional (prior) probability

- P(Cavity=true) = 0.2 and P(Weather = rain) = 0.72
- P(Weather) = <0.72, 0.1, 0.08, 0.1>
- Joint probability distributions

- Conditional (posterior) probability

- P(cavity|toothache) = 0.8
- <u>Definition of conditional probability:</u> P(a|b) = $\frac{P(a \wedge b)}{P(b)}$
- <u>Product rule:</u> P(a∩b) = P(a|b)P(a) = P(b|a)P(a)
- <u>Independence vs. dependence</u>
    - Independence: P(X|Y) = P(X)
- <u>Bayesian rule</u>

Bayesian Networks

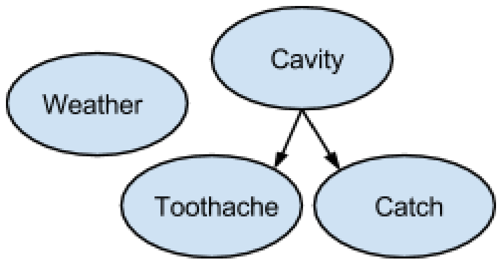
- **Bayesian network:** is a simple, graphical notation for conditional independence assertions and hence for compact specifications of full joint distributions

- Syntax for bayesian networks:

- A set of nodes, one per variable (descrete)
- A directed, acyclic graph (link ≈ "directly influences")
- A conditional distribution for each node given its parents: P(X_i | Parents(X_i))

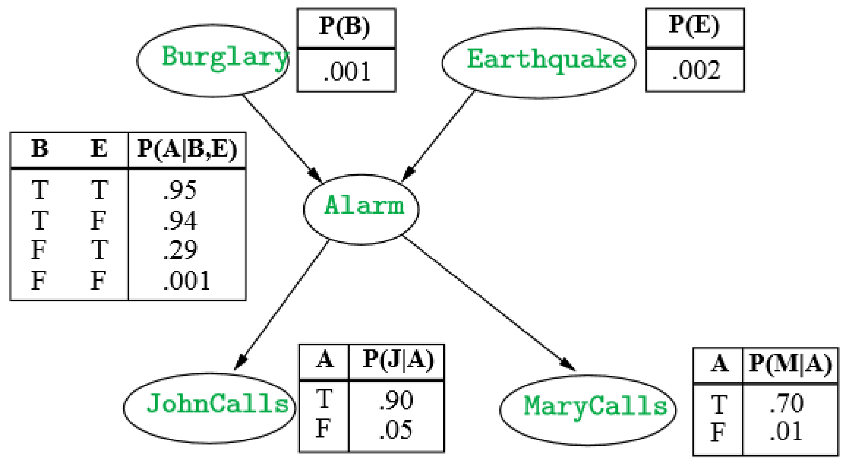
- All conditional distribution represented as a conditional probability table (CPT) given the distribution over X_i for each combination of parent values

- Example of a bayesian network:



- *Weather* is independent of the other variables as it does not influence anything, and is not influenced by anything
- *Toothache and Catch* are conditionally independent given *Cavity* becuase any variable is conditionally independent of all its non-descendants given its parents **(local semantics)**

- Example of a bayesian network with conditional probability table (CPT)



P(B)
.001

P(E)
.002

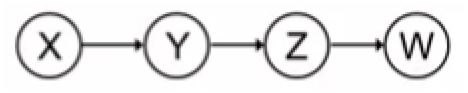
B	E	P(A|B,E)
T	T	.95
T	F	.94
F	T	.29
F	F	.001

A	P(J|A)
T	.90
F	.05

A	P(M|A)
T	.70
F	.01

- **Markov blanket:** each node is conditionally independent of all others given its markov blanket (parents, children, children's parents)

- Example:



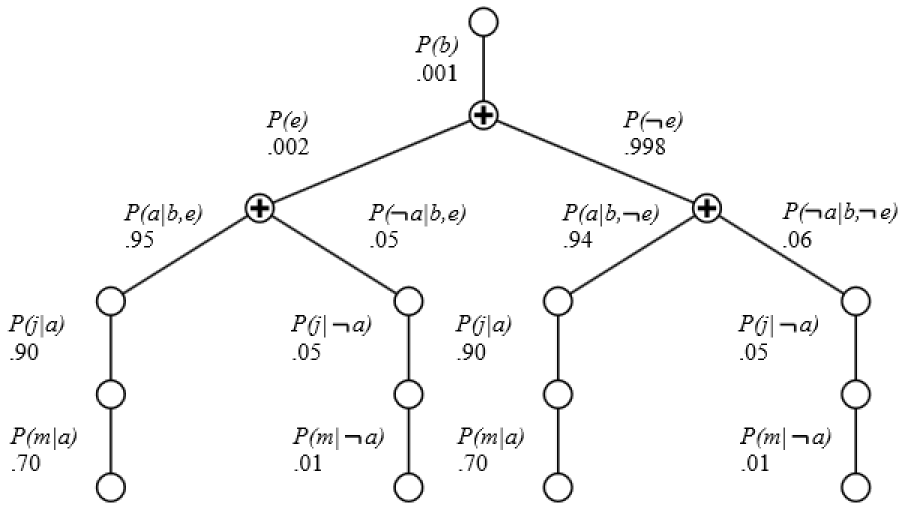
- The chain rule: P(X) P(Y|X) P(Z|X,Y) P(W|X,Y,Z)
- Bayes net: P(X) P(Y|X) P(Z|Y) P(W|Z)
- Z is cond. independent from X given Y
- W is cond. independent from X and Y given Z

- The phases of the model building process

- <u>Step 0:</u> Decide what to model
- <u>Step 1:</u> Defining the variables
- <u>Step 2:</u> The qualitive part (the graphical structure)
- <u>Step 3:</u> The quantitative part (make a CPT for each variable)
- <u>Step 4:</u> Verification of the model

- Inference

- Evaluation tree

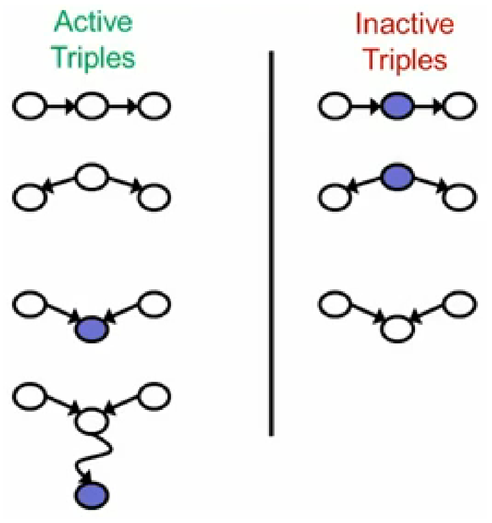


- When to use/not use bayesian networks as a modeling framework?

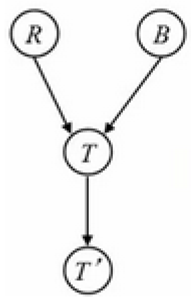
- Hvis ingen av varianlene i domenet er avhengige av noen andre variabler i domenet så vil det gi liten nytte å bruke et bayesisk nettverk
-

D-separation

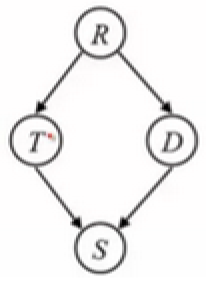
If we can find a inactive triple along a path, we can guaranteed independence.
If we have two paths, we need to find only inactive paths.
Blue nodes indicated that they are observed.



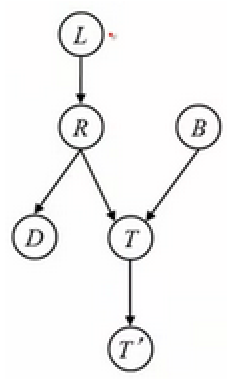
Examples:



$R \perp\!\!\!\perp B$	Yes
$R \perp\!\!\!\perp B \mid T$	Not guaranteed
$R \perp\!\!\!\perp B \mid T'$	" "



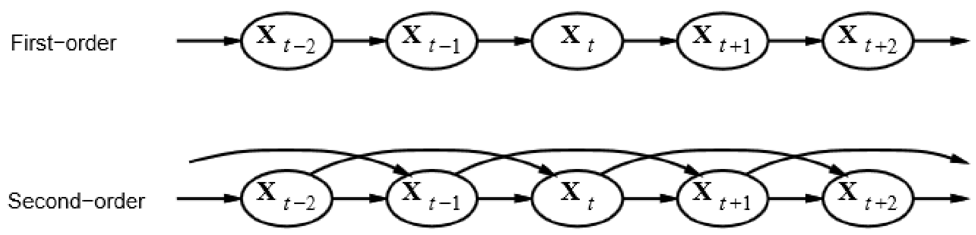
$T \perp\!\!\!\perp D$	Not guaranteed
$T \perp\!\!\!\perp D \mid R$	Yes
$T \perp\!\!\!\perp D \mid R, S$	Not guaranteed



$L \perp\!\!\!\perp T' \mid T$	Yes
$L \perp\!\!\!\perp B$	Yes
$L \perp\!\!\!\perp B \mid T$	Not guaranteed
$L \perp\!\!\!\perp B \mid T'$	Not guaranteed
$L \perp\!\!\!\perp B \mid T, R$	Yes

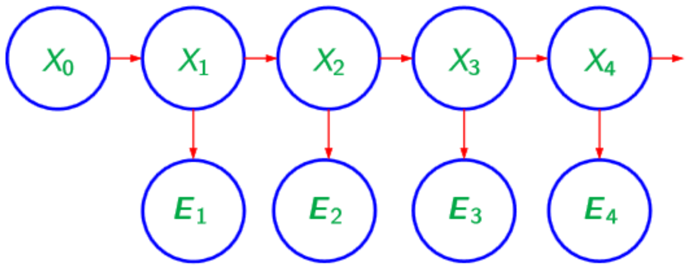
Markov process/chains

- **Motivation:** the world changes; we need to track and predict it
- **Basic idea:** copy state and evidence variables for each time step
- **Markov chain = bayesian network = true**
- **Order of markov process:**
    - **First order markov process:** future is conditional independent of past given present. P(X_t | $X_{0:t-1}$) = P(X_t | X_{t-1})
    - **Second order markov process:** P(X_t | $X_{0:t-1}$) = P(X_t | X_{t-2} , X_{t-1})
    - Det sier noe om hvor langt bak i historien man vil gå. Første orden tar med 1 ledd bak, 2 orden tar med 2 ledd bak osv.



- **Assumptions:**
    - Stationary process
    - k'th-order Markov process
    - Sensor Markov assumption

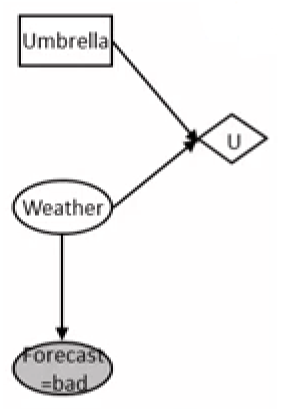
- **Hidden Markov models**
    - X_t = set of *unobservable state variables* at time t
    - E_t = set of *observable evidence variables* at time t (observed by sensors)



- **Inference task:** Filtering, Predicting, Smoothing, Most likely explanation
- **Forward-Backward algorithm**
- **Kalman filters**

Decision networks (Influence diagram)

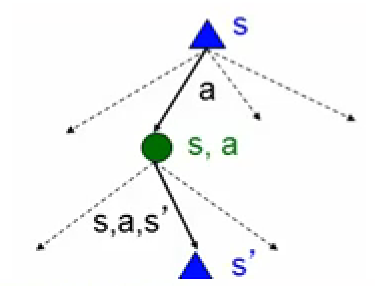
- The nodes:
    - Umbrella = action variable
    - Weather and forecast = chance nodes (forecast--> evidence)
    - U = utility nodes
- The maximum expected value
- Value of perfect information

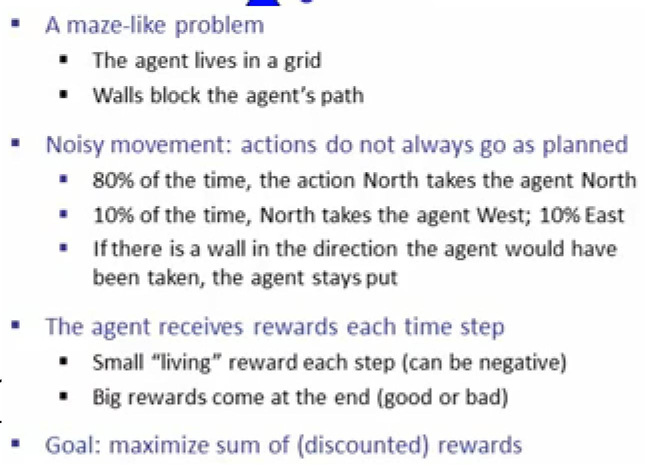


Markov Decision Process (MDP)

- A MDP consists of

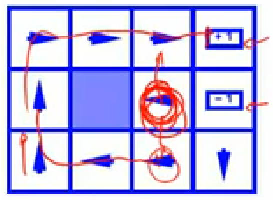
- A set of states S
- A set of actions A
- A transition function P(s’ | s,a) = T(s,a,s’)
    - probability that a from s leads to s’
- A reward function R(s,a, s’)
- A start state
- Maybe a terminal state (bounded vs. infinite)



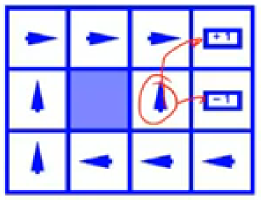


- A maze-like problem
    - The agent lives in a grid
    - Walls block the agent’s path
- Noisy movement: actions do not always go as planned
    - 80% of the time, the action North takes the agent North
    - 10% of the time, North takes the agent West; 10% East
    - If there is a wall in the direction the agent would have been taken, the agent stays put
- The agent receives rewards each time step
    - Small “living” reward each step (can be negative)
    - Big rewards come at the end (good or bad)
- Goal: maximize sum of (discounted) rewards

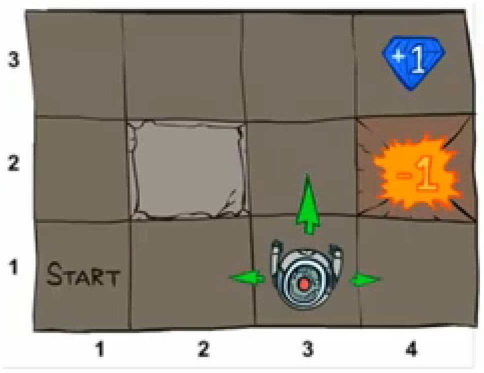
- Hvorfor heter alt Markov somthing? “Markov” generally means that given the present state, the future and the past are independent
- **Policies (π):** we want an optimal plan, or sequence of actions, from start to a goal.
- **Optimal policies:** hva som er optimalt bestemmes av hva rewarden er. Hvis reward er mius 0.01 så vil det være mer optimalt å ta en omvei for å unngå å havne i - exit state helt til høyre. Det er i dette tilfellet spesifisert a selv om man vil opp så kan man med en % havne til høyre eller venstre pga støy. Så hvis man går opp kan man havne i en tilsand som gir -1 i stedet for å gå en trygg vei som er 6 ganger minus 0.01. den siste der rewarden er minus 2 så vil vi havne fortest mulig i exit state -1!

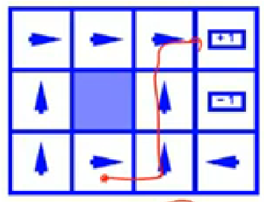


R(s) = -0.01

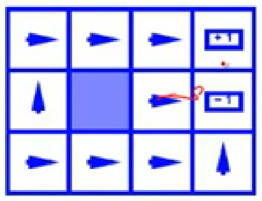


R(s) = -0.03



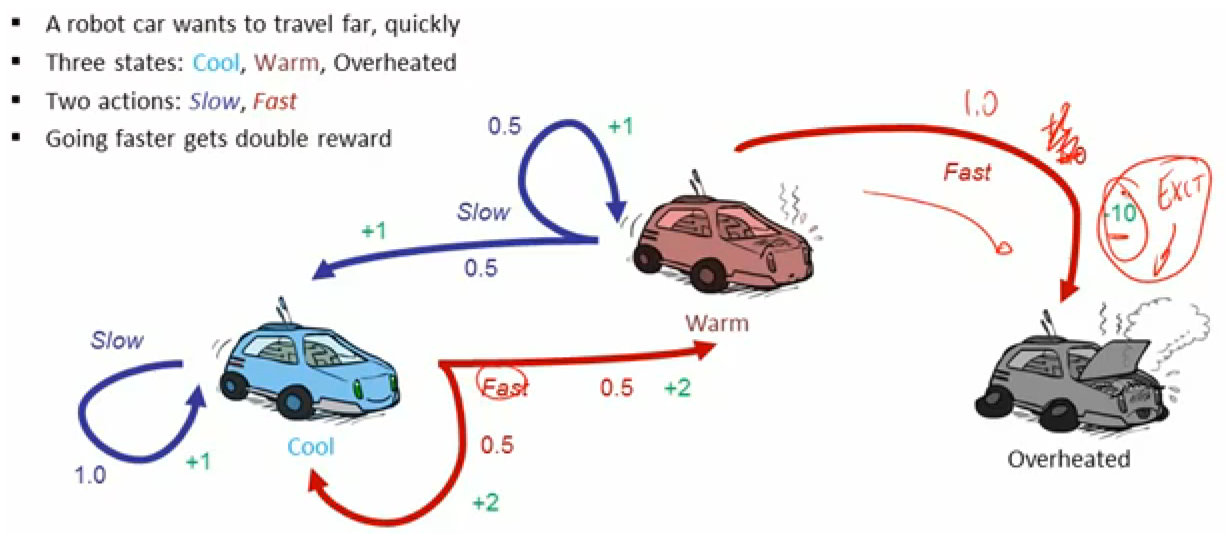


R(s) = 0.4



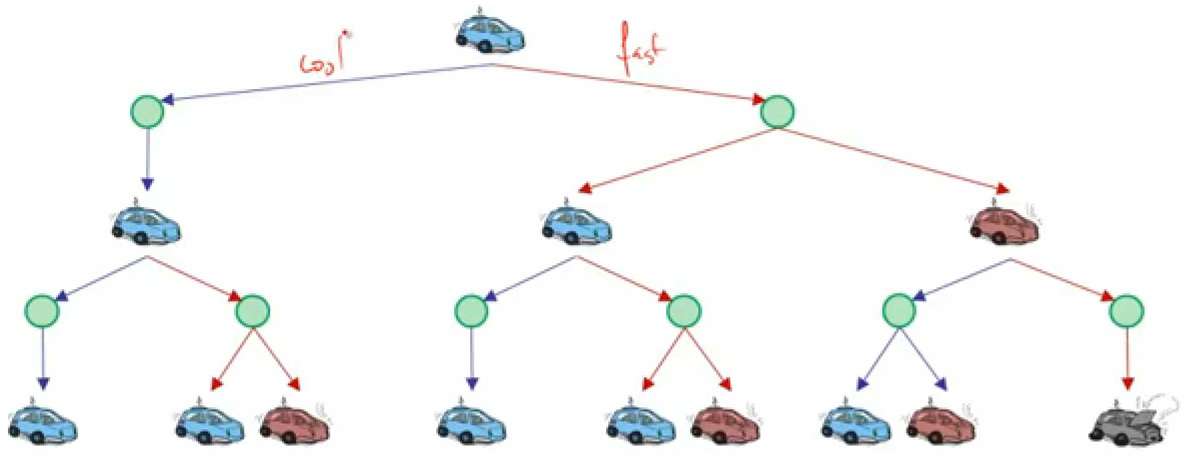
R(s) = -2.0

- Example: Racing car



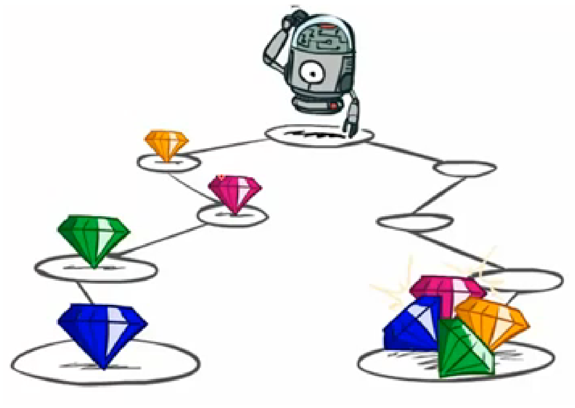
- A robot car wants to travel far, quickly
- Three states: Cool, Warm, Overheated
- Two actions: *Slow*, *Fast*
- Going faster gets double reward

Racing search tree → får veldig dypt tre med maaange duplikater

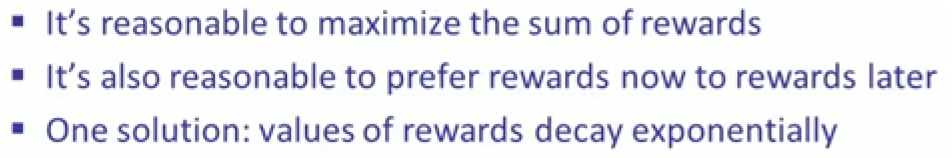


Utility of sequences:

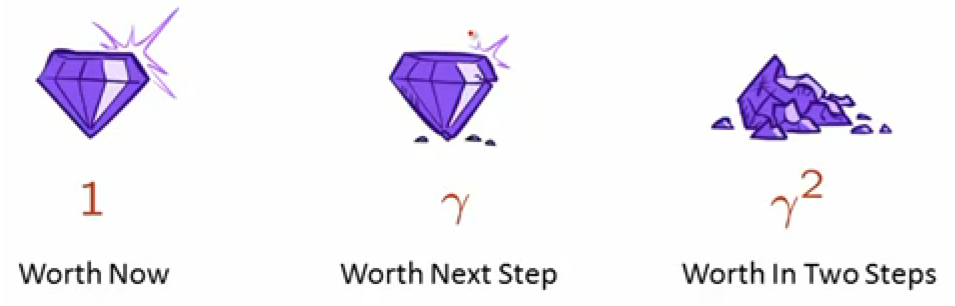
- More or less? Now or later?

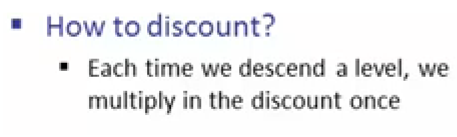


- Discounting:

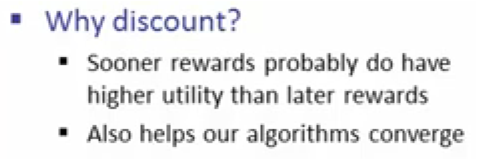


- It's reasonable to maximize the sum of rewards
- It's also reasonable to prefer rewards now to rewards later
- One solution: values of rewards decay exponentially

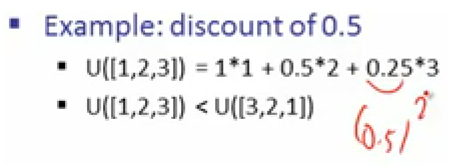




- How to discount?
  - Each time we descend a level, we multiply in the discount once



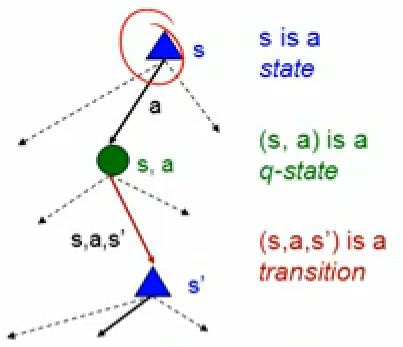
- Why discount?
  - Sooner rewards probably do have higher utility than later rewards
  - Also helps our algorithms converge



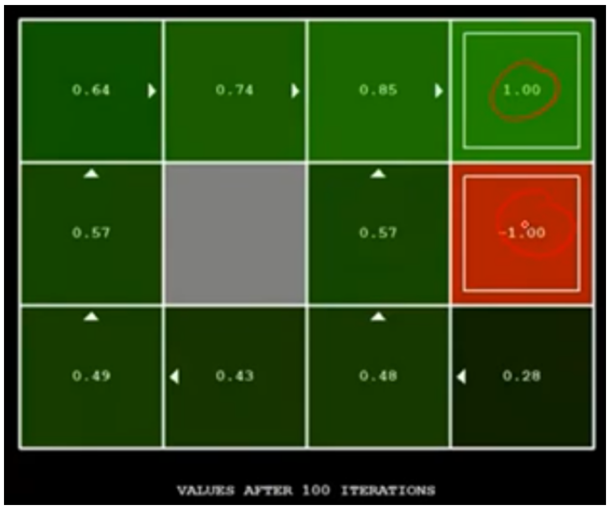
- Example: discount of 0.5
  - U([1,2,3]) = 1*1 + 0.5*2 + $\underbrace{0.25}_{(0.5)^2}$*3
  - U([1,2,3]) < U([3,2,1])

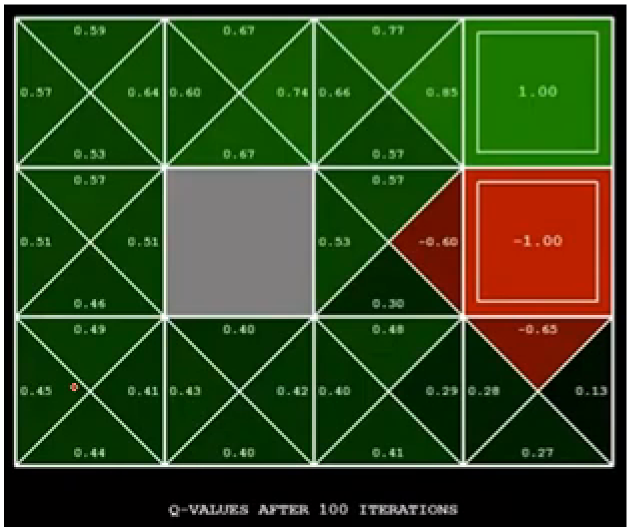
Optimal quantities:

- The value (utility) of a state s:
  - $V^*(s)$ = expected utility starting in s and acting optimally
- The value (utility) of a q-state (s,a):
  - $Q^*(s,a)$ = expected utility starting out having taken action a from state s and (thereafter) acting optimally
- The optimal policy:
  - $\pi^*(s)$ = optimal action from state s $= \arg\max_a Q^*(s,a)$



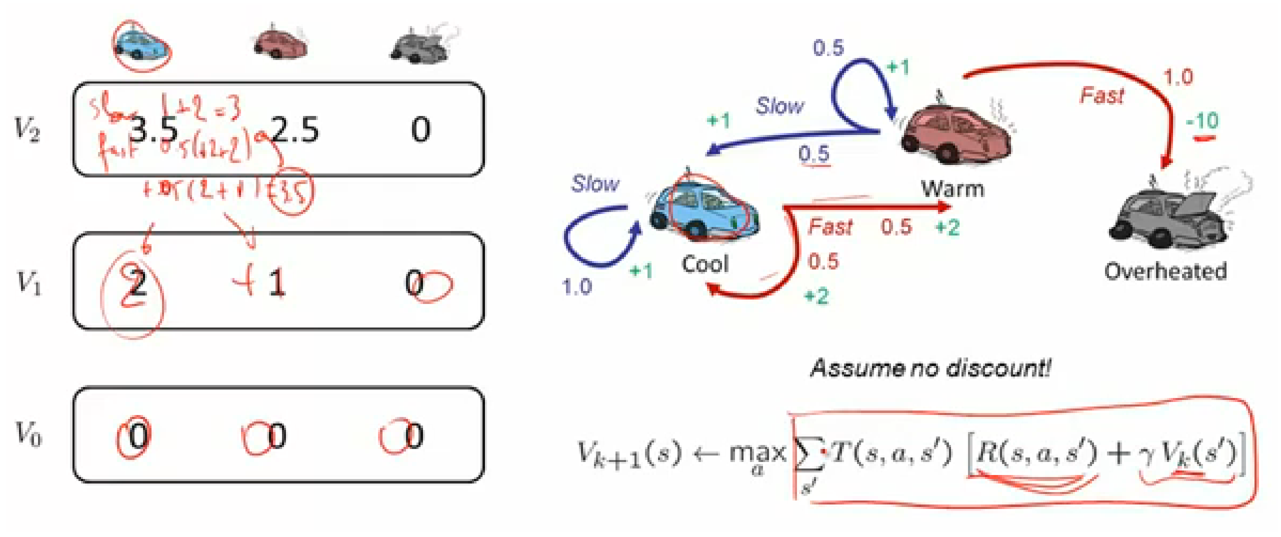
- Quantities:
  - Policy = map of states to actions
  - Utility = sum of discounted rewards
  - Values = expected future utility from a state (max node)
  - Q-Values = expected future utility from a q-state (chance node)





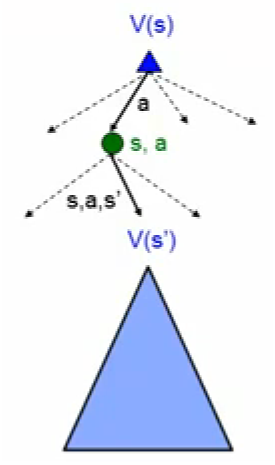
Figurer: V* (values) and Q* (Q-values)

Value iteration



Figur: her regnes value iteration (Ligningen er bellman ligningen)

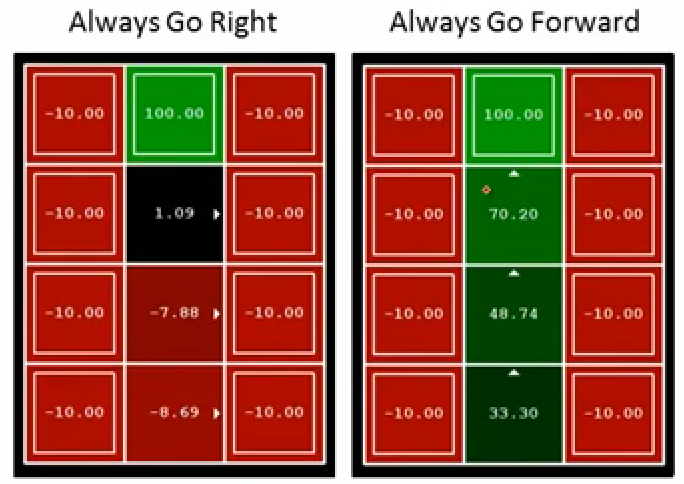
$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V_k(s')\right]$$



Policy iteration

$$V^{\pi}(s) = \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V^{\pi}(s')]$$

$$V_{k+1}^{\pi}(s) \leftarrow \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V_{k}^{\pi}(s')]$$

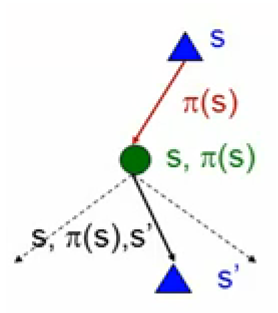


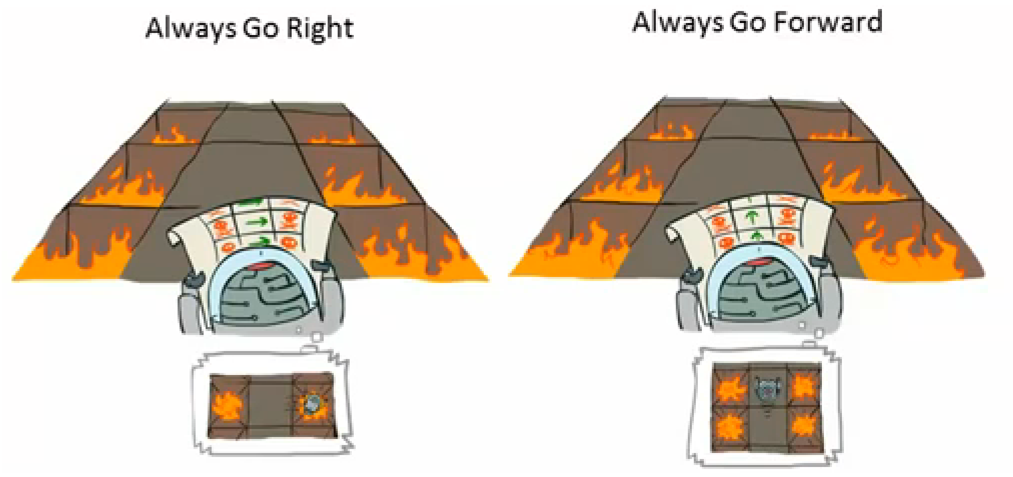
Always Go Right

-10.00	100.00	-10.00
-10.00	1.09 ▸	-10.00
-10.00	-7.88 ▸	-10.00
-10.00	-8.69 ▸	-10.00

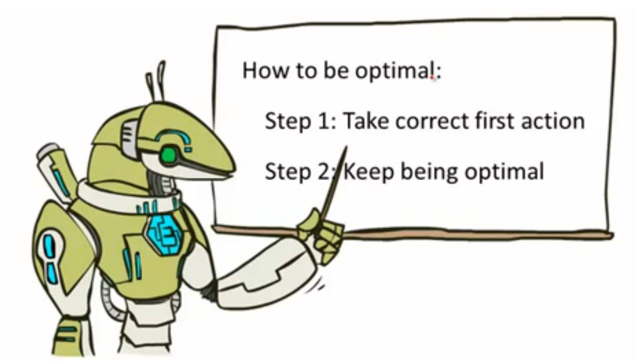
Always Go Forward

-10.00	100.00	-10.00
-10.00	▴ 70.20	-10.00
-10.00	▴ 48.74	-10.00
-10.00	▴ 33.30	-10.00





The Bellaman Equation



How to be optimal:

Step 1: Take correct first action

Step 2: Keep being optimal

$$V^*(s) = \max_a \sum_{s'} T(s,a,s') \left[R(s,a,s') + \gamma V^*(s') \right]$$

Learning

Three types of learning:

- Unsupervised learning: no examples, just do by learning
- Reinforcement learning: learns from a series of reinforcements (rewards or punishments)
- Supervised learning: learning from observed input-output pairs and learns a function that maps from input to output.

Supervised learning

- **The task of supervised learning is:** given a training set of N example input-output pairs ((x_1, y_1) , (x_2, y_2), …. , (x_N, y_N)) where each y was generated by a unknown function y = f (x), discover a function h that approximates the true function f.
- **x and y (input and output**) can be any value; they need not to be numbers.
- **Function h** is a hypothesis.
- **How good is the hypothesis/function h?** Test on a new example set (test set that is distinct from the training set)
- We say that a hypothesis **generalizes** well if it correctly predicts the value y for novel examples.
- **Different learning problems**:
  - Classification: when output y is one of a finite set of values (rainy, sunny, cloudy)
  - Regression: when y (the output) is a number

Decision trees

- Decision trees → supervised learning!
- **A decision tree** represent a function that takes as input a vector of attribute values and returns a decision (a single output value).
- The input and output can be descrete or continous
- **Internal node** = decision to make
- **Leaf node** = return value → Anwvears the goal predicate (WillWait)
- **Expressiveness of decision tree**: any function in propositional logic can be expressed as a decision tree. Example: Path = (Patrons = Full ∧ WaitEstimate =0-10)

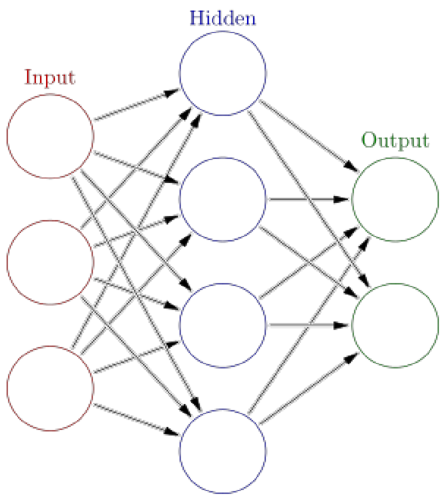
- **The decision tree learning algorithm (DTL):** er en grådig algoritme som tar inn et sett med treningseksempler og returnerer et descision tree basert på input. Målet er å få bygget et tre som har minst mulig dybde (fordi "søk" i dette treet vil gå i dybden). Det er en rekursiv algoritme og man må se opp for følgende hendelser:
1) Det er ingen eksempler eller det er ingen attributter: returner default value
2) Alle attributter har samme klassifikasjon: returner klassifikasjon:
3) alle har forskjellig klassifikasjon: IMPORTANCE funksjon for å finne beste attributt å splitte på.

- **The importance function** is a measure of "really good" or "really useless". Here we use the notation o**f imorfation gain** , which is defined in terms of **entropy.**
- **Measuring performance:** decision tree bygges ved treningsdata. Etter å ha bygget et tre ved DTL kan man putte inn test data for å se hvor mange av input som får rett output fra treet (antall rett/total input).
- **Overfitting:**
- **Decision tree pruning:**

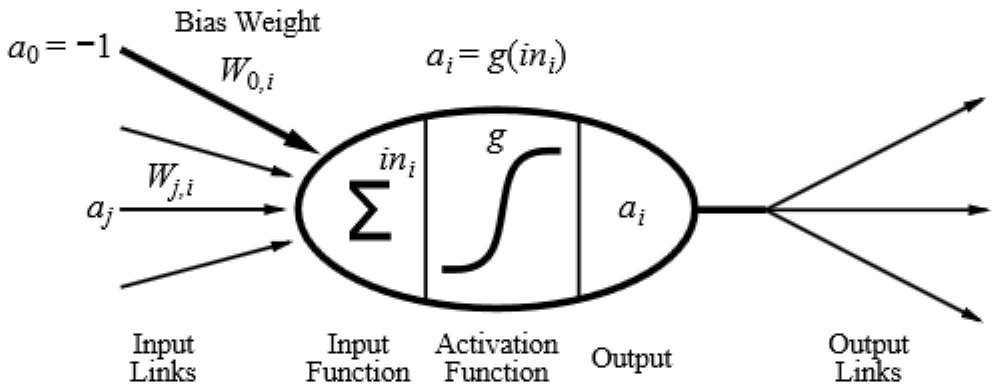
Neural networks

- **A neural network** consists of an interconnected group of artificial neurons, and it processes information using a connectionist approach to computation. In most cases a neural network is an adaptive system changing its structure during a learning phase. Neural networks are used for modeling complex relationships between inputs and outputs or to find patterns in data.

- A neural network is trained by a big set of data.
- A neural network can have many layers
- Can be complex to use in the real world because it needs a BIG set of data to act in the real world.
- Feedforward network is a DAG like this one
- Et neuralt netverk blir lært helt til den har en error som er mindre eller lik x (som velges selv)

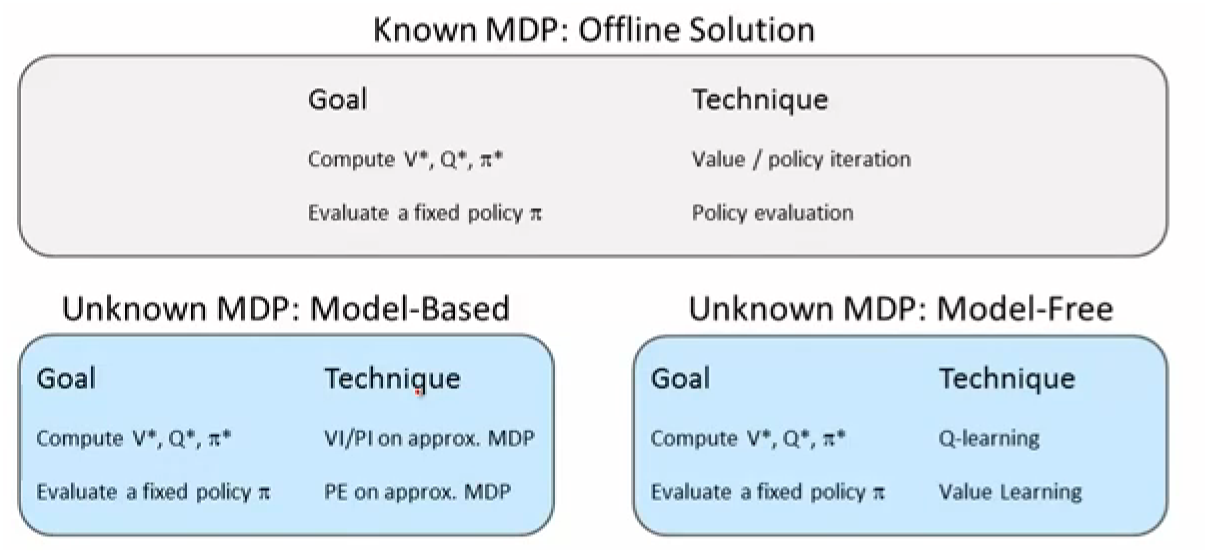


- **Backpropagation:** The goal and motivation for developing the backpropagation algorithm is to find a way to train multi-layered neural networks such that it can learn the appropriate internalrepresentations to allow it to learn any arbitrary mapping of input to output.
the backpropagation learning algorithm can be divided into two phases: propagation (forward and backward) and weight update



Reinforcement learning

- MDP (offline learning) vs Reinforcement Learning (online learning)



Known MDP: Offline Solution

Goal	Technique
Compute V*, Q*, π*	Value / policy iteration
Evaluate a fixed policy π	Policy evaluation

Unknown MDP: Model-Based

Goal	Technique
Compute V*, Q*, π*	VI/PI on approx. MDP
Evaluate a fixed policy π	PE on approx. MDP

Unknown MDP: Model-Free

Goal	Technique
Compute V*, Q*, π*	Q-learning
Evaluate a fixed policy π	Value Learning

Q-learning

$$Q_{t+1}(s_t,a_t) = \underbrace{Q_t(s_t,a_t)}_{\text{old value}} + \underbrace{\alpha_t(s_t,a_t)}_{\text{learning rate}} \times \left[\overbrace{\underbrace{R_{t+1}}_{\text{reward}} + \underbrace{\gamma}_{\text{discount factor}} \underbrace{\max_a Q_t(s_{t+1},a)}_{\text{estimate of optimal future value}}}^{\text{learned value}} - \underbrace{Q_t(s_t,a_t)}_{\text{old value}} \right]$$

Explotation vs Exploration

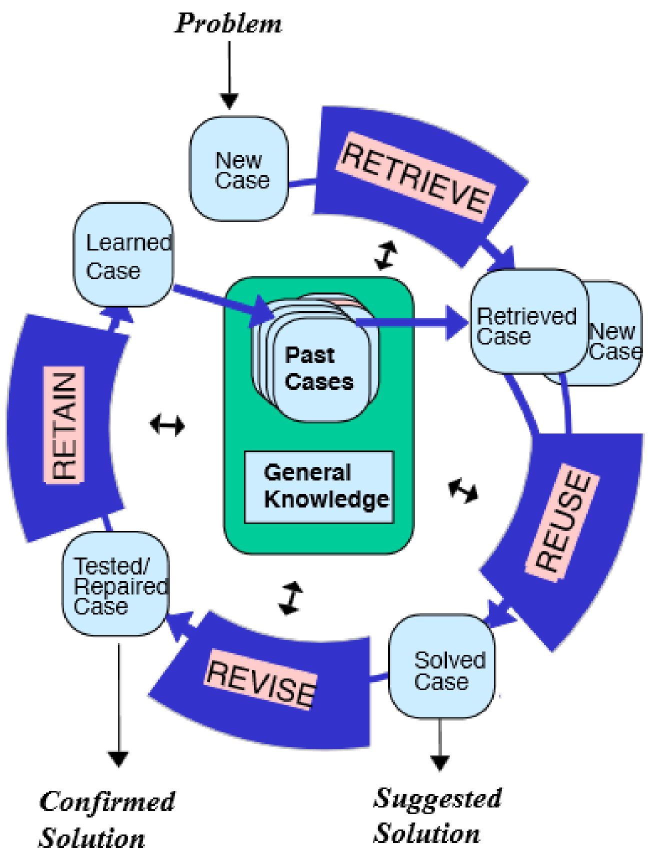
Case-based reasoning (CBR)

- Case-based reasoning: case-based problem solving + case-based learning

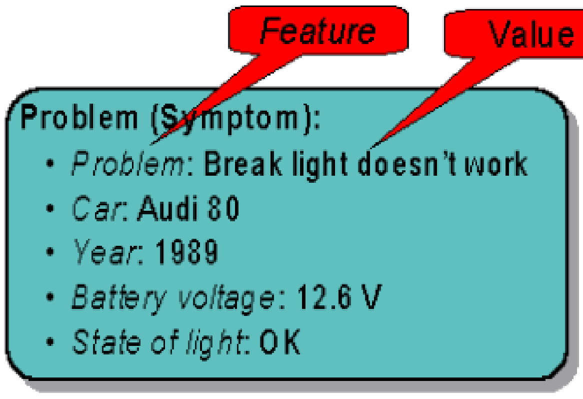
- CBR assumptions:
- Similar problems have similar solutions
- The world are a regular place: what holds true today will probably hold true tomorrow
- Situations repeat: if they dont do, there is no point in remembering them

- CBR main components
- Case base
- Method for retrieval of relevant cases
- Method for adaption of solution
- Method for learning from problem solved

- The CBR cycle



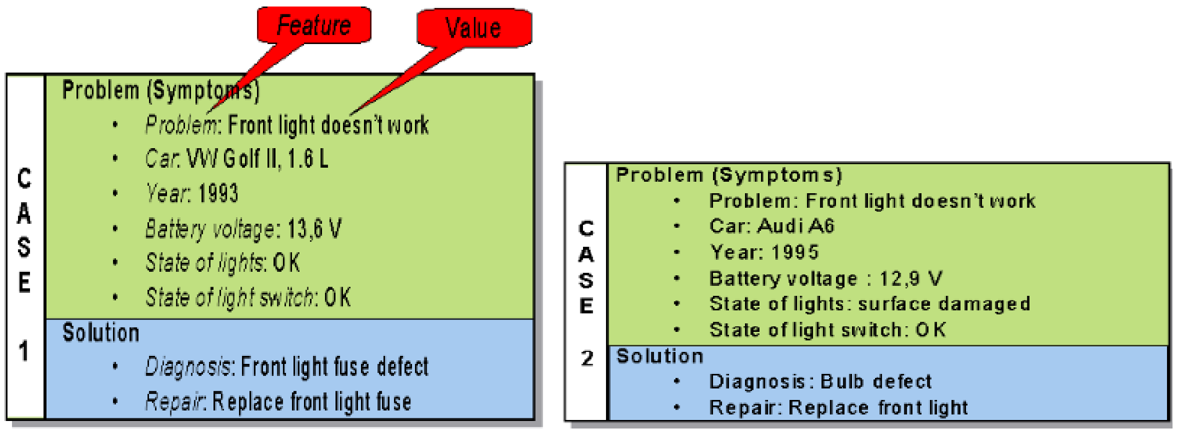
- Solving a new problem: retrieving similar problems → adapting retrived solutions
- New problem to be solved



Problem (Symptom):

- *Problem*: Break light doesn't work
- *Car*: Audi 80
- *Year*: 1989
- *Battery voltage*: 12.6 V
- *State of light*: OK

- Retrive similar solutions for a car problem



CASE 1

Problem (Symptoms)

- *Problem*: Front light doesn't work
- *Car*: VW Golf II, 1.6 L
- *Year*: 1993
- *Battery voltage*: 13,6 V
- *State of lights*: OK
- *State of light switch*: OK

Solution

- *Diagnosis*: Front light fuse defect
- *Repair*: Replace front light fuse

CASE 2

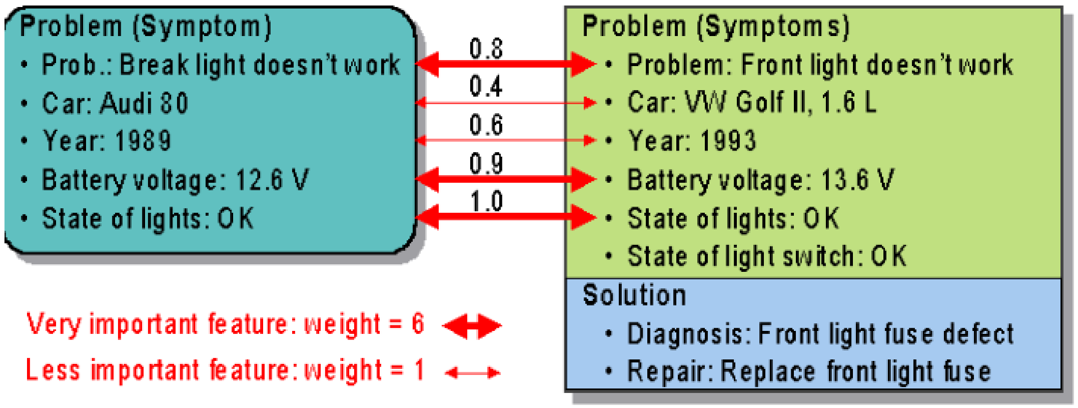
Problem (Symptoms)

- Problem: Front light doesn't work
- Car: Audi A6
- Year: 1995
- Battery voltage : 12,9 V
- State of lights: surface damaged
- State of light switch: OK

Solution

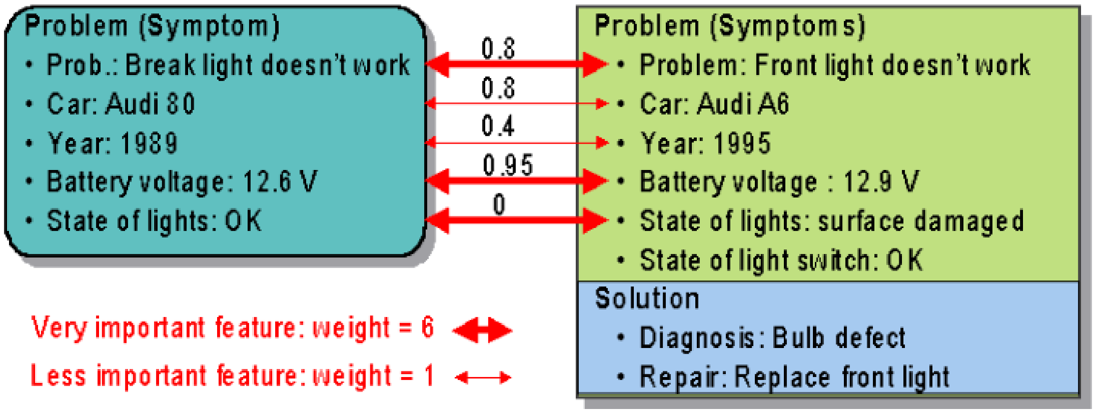
- Diagnosis: Bulb defect
- Repair: Replace front light

- Evaluate the similar solution retrived (case 1 and 2)


Problem (Symptom)
• Prob.: Break light doesn't work
• Car: Audi 80
• Year: 1989
• Battery voltage: 12.6 V
• State of lights: OK
0.8
0.4
0.6
0.9
1.0
Problem (Symptoms)
• Problem: Front light doesn't work
• Car: VW Golf II, 1.6 L
• Year: 1993
• Battery voltage: 13.6 V
• State of lights: OK
• State of light switch: OK
Solution
• Diagnosis: Front light fuse defect
• Repair: Replace front light fuse
Very important feature: weight = 6
Less important feature: weight = 1

Similarity Computation by Weighted Average

similarity(new,case 1) = 1/20 * [6*0.8 + 1*0.4 + 1*0.6 + 6*0.9 + 6* 1.0] = **0.86**


Problem (Symptom)
• Prob.: Break light doesn't work
• Car: Audi 80
• Year: 1989
• Battery voltage: 12.6 V
• State of lights: OK
0.8
0.8
0.4
0.95
0
Problem (Symptoms)
• Problem: Front light doesn't work
• Car: Audi A6
• Year: 1995
• Battery voltage : 12.9 V
• State of lights: surface damaged
• State of light switch: OK
Solution
• Diagnosis: Bulb defect
• Repair: Replace front light
Very important feature: weight = 6
Less important feature: weight = 1

Similarity Computation by Weighted Average

similarity(new,case 2) = 1/20 * [6*0.8 + 1*0.8 + 1*0.4 + 6*0.95 + 6*0] = **0.585**

- Reuse solution from best case

Conclusions

Philosophical foundations

Weak AI

- **Hypothesis:** machines act as if they were intelligent
- Are machines intelligent?
- Turing test
- The argument from disability
  - “a machine can never do X”
- The mathematical objection
- The argument from informality
  - **Qualification problem:** the inability to capture everything in a set of logical rules
  - **GOFAI (Good Old-Fashioned AI):** this critic is supposed to claim that all intelligent behaviour can be captured by a system that reasons logically from a set of facts and rules describing the domain. It therefore correspond to the simplest logical agent.

Strong AI

- **Hypothesis:** if the intelligent machines are actually thinkin
- Can machines think? → Can submarines swim?
- Machine that passes the turing test
  - Is the machine actually thinking?
  - Or is it just a simulation of thinking?
- **The argument from consciousness (bevissthet):** the machine has to be aware of its own mental state and actions.
- **Phenomenology (fenomenologi):** the study of direct experienc → the machine has to actually feel emotions.
- **Intentionality (intensjonalitet):** the question of whether the machine’s purported beliefs, desires and other representations are actually “about” somthig in the real world.
- **Mind- body problem**
  - **Dualist theory** (mind and body are two separate parts)
  - **Monist theory** or physicalism (mind and body are one part and a mental state is a physical state)
- **Mental states and the brain in a vat**
  - What does it means that a person (or machine) is in a mental state?
- **Content of mental state**
  - **Wide content view:** interprets it from the point of view of an omniscient outside observer with access to the whole situation , who can distinguish differences in the world. Under this view, the content of mental states involves both the brain state and the environment history.

    - **Narrow content view:** considers only the brain state.The narrow content of the brain states of a real hamburger-eater and a brain-in-a-vat “hamburger-eater” is the same in both cases.
- **Functionalism and the brain replacement experiment**
    - **The theory of functionalism:** mental state is any intermediate causal condition between input and output.
- **Biological naturalism and the Chinese Room**
    - The theory of biologial naturalism: mental state cannot be dublicated just on the basis of some program having the same functional structure with the same input-output behaviour.

<u>The ethics and risks of developing artificial intelligence</u>

26.3 - The ethics and risk of developing artificial intelligence

- **Ethics and risks:**

- People might loose their jobs to automation
- People might have too much (or too little) leisure time
- People might lose their sense of being unique
- AI systems might be used toward undesirable ends
- The use of AI systems might result in a loss of accountability
- The success of AI might mean the end of the human race.

- **Ultraintelligent machine**: is defined as a machine that can far surpass all the intellectual activities of any man however clever.
- **Technological singularity**: The "intelligence explosion"
- **Transhumanism**: is a new word for the active social movement that looks forward to this future in which humans are merged with or placed by robotic and biotech inventions
- **Friendly A**I: (a desire not to harm humans)

AI: the present and future

27.1 - Agent components

- **Agnet components**
    - Interaction with the environment through sensors and actuators
    - Keeping track of the state of the world
    - Projecting, evaluating, and selecting future courses of action
    - Utility as an expression of preferences
    - Learning

27.2 - Agent architectures

- **Hybrid architecture**: Setter sammen flere agenter til en arkitektur
- **Real-time AI**: the problem when a system is in a complex domain and the problems becomes real-time --> it will take too long time to solve many of the problems
- **Controlling deliberation**:
    - Anytime algorithm: is an algorithm whose output quality improves gradually over time(example of an anytime algorithms include iterative deepening in game-tree search and MCMC in Bayesian networks)
    - Decision-theoretic metareasoning: ?
- **Reflective architecture**:

27.3 - Are we going in the right direction?

- **The goal of AI:** vi startet med å si at målet er å alltid gjøre det rette (perfect rationality), men er det egentlig mulig? Her er 3 forskjellige måter å bygge en agents utgangspunkt ifra:
    - Perfect rationality: a perfectly rational agent acts at every instant in such way as to maximize its expected utility, given the information it has acquired from the environment.
    - Calculative rationality: a calculatively rational agent eventually returns what would have been the rational choice at the beginning of its deliberation
    - Bounded rationality: en agent som kommer med svar som er "gode nok".
    - Bounded optimality: a bounded optimal agent behaves as well as possible, given its computional resources (en agent som gjør så godt den kan).

27.4 - What if AI does succeed?