Disclaimer

The following slides contain mainly examples for the chapters 12-24 of the lecture „Artificial Intelligence 2“. They do not repeat the stuff in the lecture videos resp. slides.

Often they start with a task and present the solution in subsequent videos. You probably profit the most from the slides, if you try to solve the task before looking at the solution.

The language of the slides is a mix of German and English.

Chapter 12: Quantifying Uncertainty

Wahrscheinlichkeiten in der Wumpus-Welt

- In der Vorlesung wurden Wahrscheinlichkeiten in der Wumpus-Welt berechnet.
  - Wenn der Agent in Feld [1,1] keine Wahrnehmungen und in den Feldern [1,2] und [2,1] jeweils einen Luftzug (Breeze, B) spürt, wurden die Wahrscheinlichkeiten eines Lochs (Pit) für die Felder [1,3], [2,2] und [3,1] wie nebenstehend berechnet:

	1	2	3	4
4				
3	31%			
2	B	86%		
1	o.k.	B	31%	

  - Als nächstes geht der Agent auf das Feld [3,1], fällt nicht in ein Loch, aber verspürt einen Luftzug. Dadurch kann er sicher herleiten, dass das Feld [2,2] ein Loch enthält. Die Frage ist, was er als nächstes tun sollte? Berechnen Sie dazu die Wahrscheinlichkeiten eines Lochs für die drei Felder mit „??“

	1	2	3	4
4				
3	??			
2	B	Pit	??	
1	o.k.	B	B	??

Lösung für Wumpus-Welt

- Für das Feld [1,3] ist die Wahrscheinlichkeit einfach die Apriori-Wahrscheinlichkeit von Löchern (20%), da die Wahrnehmung Luftzug in [1,2] bereits durch das Loch in [2,2] erklärt ist.
- Für die Felder [2,3] und [1,4] ist die Wahrscheinlichkeit höher als 50%, da auch auf beiden Feldern Löcher sein können. Daher:
  - Wahrscheinlichkeit von Loch in [2,3] mit 2 Möglichkeiten:
    - in [4,1] kann ein Loch sein oder nicht:
    - Apriori (Loch) * Wahrscheinlichkeit der 2 Möglichkeiten:
    - 0.2 * (0.2 + 0.8) = 0.2
  - Wahrscheinlich von kein Loch in [2,3] mit 1 Möglichkeit:
    - in [4,1] muss ein Loch sein:
    - 0.8 * 0.2 = 0,16
  - Normierung von <0.2, 0.16> = <0.56, 0.44>
  - ➢Wahrscheinlichkeit für Loch in [2,3]: 56%

	1	2	3	4
4				
3	20%			
2	B	Pit	??	
1	o.k.	B	B	??

	1	2	3	4
4				
3	20%			
2	B	Pit	56%	
1	o.k.	B	B	56%

Wetten in der Wumpus-Welt

- Wie wahrscheinlich ist es, dass unser Agent das Gold findet? Wir wollen darauf eine Wette machen und unseren Einsatz kalkulieren!
- Unser Agent start wieder in Feld [1,1] ohne Wahrnehmung und spürt in den Feldern [1,2] und [2,1] jeweils einen Luftzug (s. rechts)

	1	2	3	4
4				
3				
2	B			
1	o.k.	B		

- Wir wissen außerdem, dass, wenn er auf die weiteren Felder [3,1], [1,3], [1,4] und [2,3] gehen würde, er folgende Wahrnehmungen bekommt. Allerdings weiß das unser Agent anfangs noch nicht (s. rechts; potentielle Wahrnehmung in Klammern)
- Der Agent verhält sich immer wahrscheinlichkeitstheoretisch optimal (Wegkosten spielen hier keine Rolle).

4	(B)			
3	(G)	Gold		
2	(B)			
1	Start	(B)	(B)	
	1	2	3	4

Lösung für Wette

- Zwei Optimale Sequenzen nach [1,1], [1,2] und [2,1], die sich aus den bisherigen Berechnungen ergeben:
  - [3,1], ([4,1]), [3,2] : Risiko für Loch: 31%
  - [1,3], [3,1], ([4,1]), [3,2]: 0.31 + (0.69 * 0.2) = 0.448 = 44,8%
- Mittelwert beider Sequenzen, da diese initial gleich wahrscheinlich sind: 37,9% Risiko, in ein Loch zu fallen
- Wir können daher eine Wette von 4 € zu 6 € (mindestens 3,79 € zu 6,21 €) anbieten, dass unser Agent das Gold findet (d.h. wir bekommen 4 € bzw. zahlen 6 €)

4	(B)			
3	(G)	Gold		
2	(B)			
1	Start	(B)	(B)	
	1	2	3	4

Naive Bayes mit numerischen Daten

- Sie haben für 14 Fälle aufgezeichnet, ob Sie abhängig von Temperatur und Regenprognose einen Ausflug machen (s. rechts).
- Berechnen Sie daraus mit Naive Bayes die Wahrscheinlichkeit für zwei neue Fälle (unten rechts).

	Temperatur	Regenprognose			Ausflug	
	Celsius	kein	wenig	viel	ja	nein
Fall1	10		x		x	
Fall2	20			x		x
Fall3	15	x			x	
Fall4	5	x			x	
Fall5	3		x			x
Fall6	8		x			x
Fall7	25		x		x	
Fall8	12	x			x	
Fall9	22			x		x
Fall10	11		x			x
Fall11	0	x				x
Fall12	1			x		x
Fall13	6		x			x
Fall14	8	x				x
NeuerFall1	12		x		??	??
NeuerFall2	8	x			??	??

Bedingte Wahrscheinlichkeiten und Mittelwerte

Regenprognose; Temperatur

Temp/A=ja	13,40	6,7
Temp/A=nein	8,80	7,3
R=k/A=ja	0,60	3/5
R=w/A=ja	0,40	2/5
R=v/A=ja	0,00	0/5
R=k/A=nein	0,22	2/9
R=w/A=nein	0,44	4/9
R=v/A=nein	0,33	3/9
A=ja	0,36	5/14
A=nein	0,64	9/14

Ausflug		
ja	nein	
10		
	20	
15		
5		
	3	
	8	
25		
12		
	22	
	11	
	0	
	1	
	6	
	8	
13,4	8,8	Mittelwert
6,7	7,3	Standardabweichung

Berechnung der Lösung

$$\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

DF(Temp=12/A=ja)	0,05999466	1,02239907	0,05868028
DF(Temp=12/A=nein)	0,05432778	1,10100594	0,04934377
DF(Temp=5/A=ja)	0,05999466	1,39036458	0,04315031
DF(Temp=5/A=nein)	0,05432778	1,00562215	0,05402405
	$\frac{1}{\sqrt{2\pi}\sigma}$	$e^{\frac{(x-\mu)^2}{2\sigma^2}}$	

Dichtefunktion

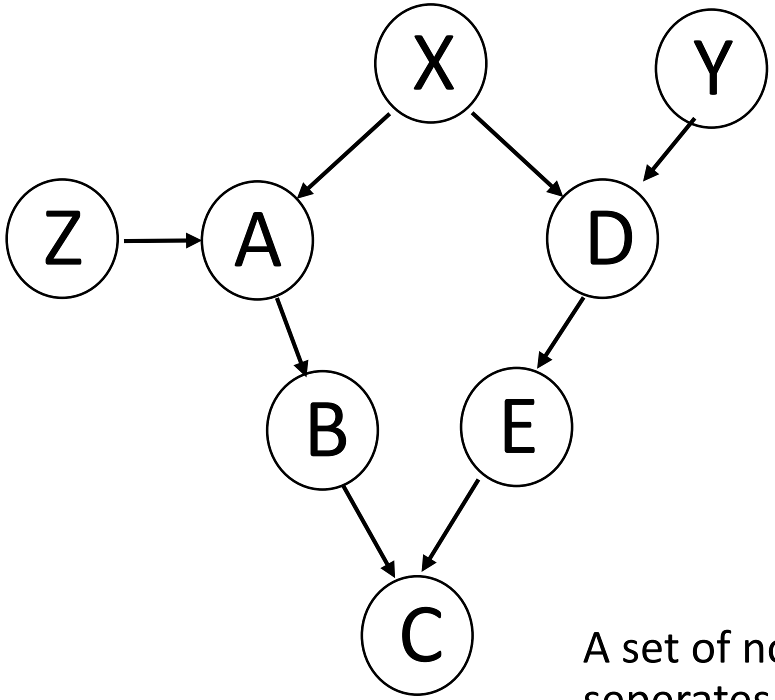
$$\frac{1}{\sqrt{2\cdot\pi}\cdot 6{,}7} \cdot \frac{1}{e^{\frac{(12-13{,}4)^2}{2\cdot(6{,}7)^2}}}$$

Bayes Formel mit Dichtefunktion

0,0083829	37,29%		α P(A=ja) * DF(Temp=12/A=ja) P(R=w/A=ja)
0,01409822	62,71%		α P(A=nein) * DF(Temp=12/A=nein) P(R=w/A=nein)
0,0092465	54,51%		α P(A=ja) * DF(Temp=8/A=ja) P(R=k/A=ja)
0,00771772	45,49%		α P(A=nein) * DF(Temp=8/A=nein) P(R=k/A=nein)

Chapter 13: Probabilistic Reasoning

Conditional Inpedendance

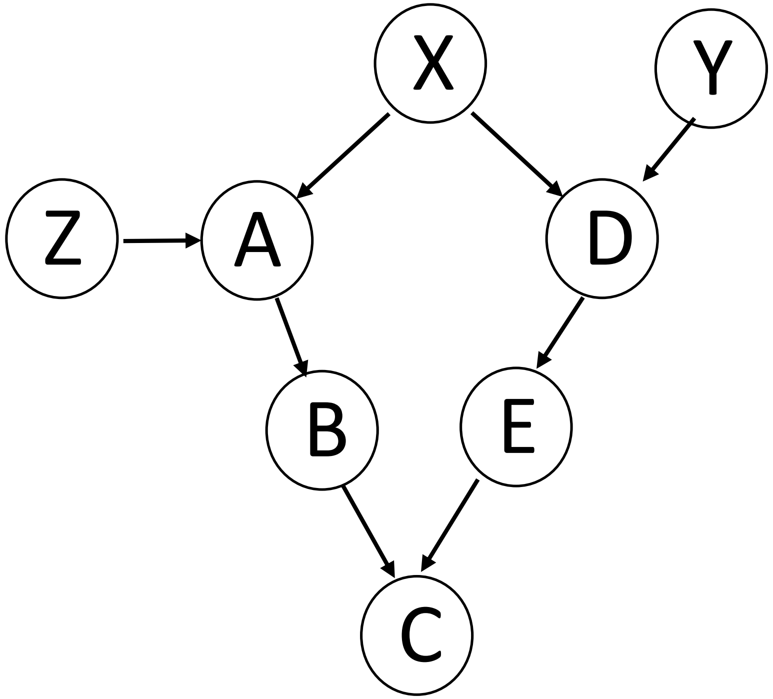


- Sind X und Y unabhängig, wenn Z gegeben ist?
- Sind B und E unabhängig, wenn Z gegeben ist?
- Sind A und C unabhängig, wenn B gegeben ist?
- Sind A und D unabhängig, wenn X gegeben ist?

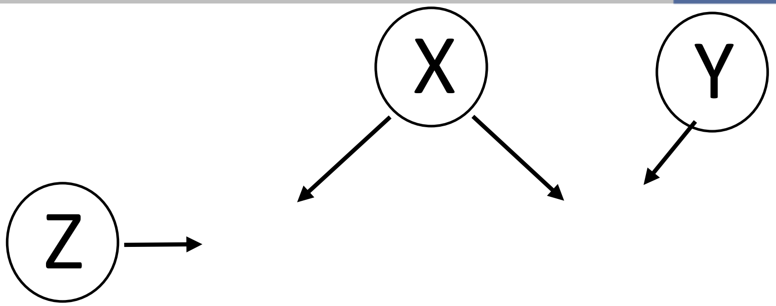
A set of nodes X is conditional independant to a set of nodes Y, given a third set Z, if Z d-seperates X and Y.

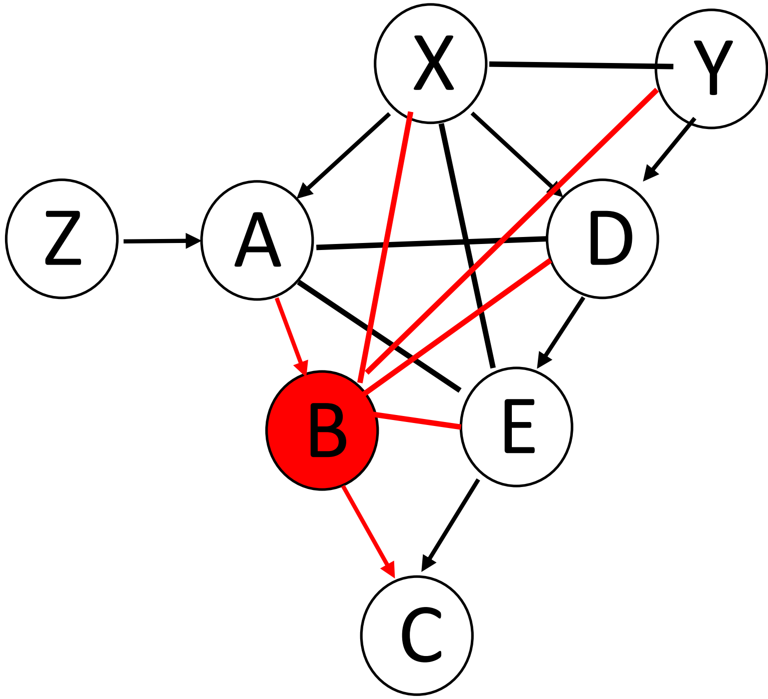
1. Consider just the ancestral subgraph consisting of X, Y, Z and their ancestors.
2. Add links between an unlinked pair of nodes that share a common child („marry them“), now we have the so-called moral graph („Moralize“)
3. Replace all directed links by undirected links („Disorient“)
4. If Z blocks all paths between X and Y, then Z d-separates X and Y („Delete givens“)

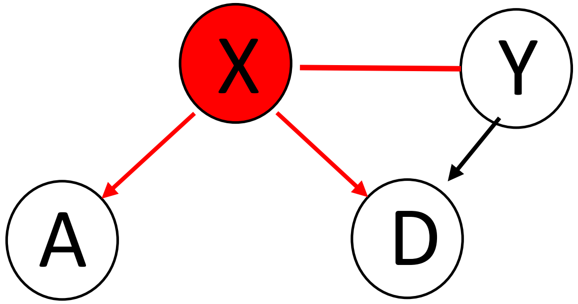
Conditional Inpedendance



- Sind X und Y unabhängig, wenn Z gegeben ist?
  - ja
- Sind B und E unabhängig, wenn Z gegeben ist?
  - nein (hat nichts mit Z zu tun)
- Sind A und C unabhängig, wenn B gegeben ist?
  - nein
- Sind A und D unabhängig, wenn X gegeben ist?
  - ja

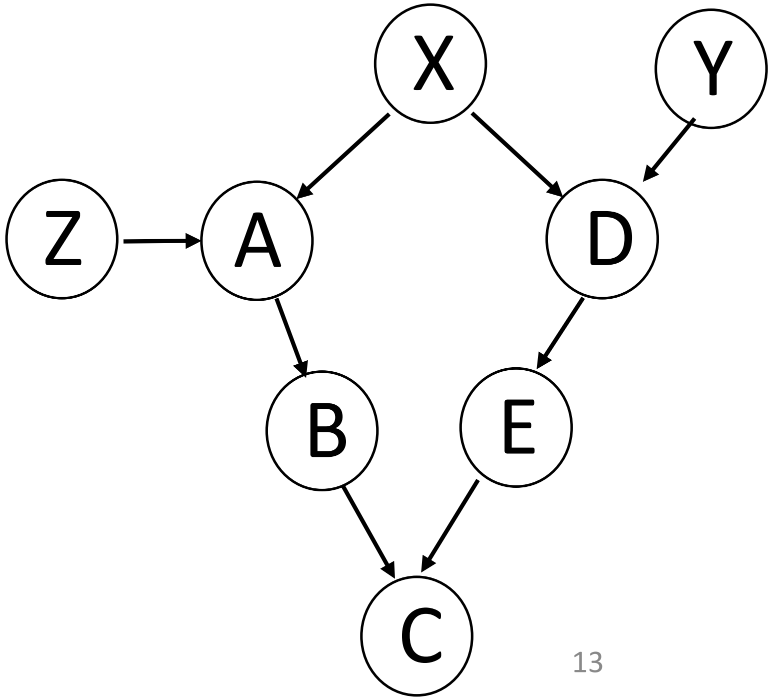






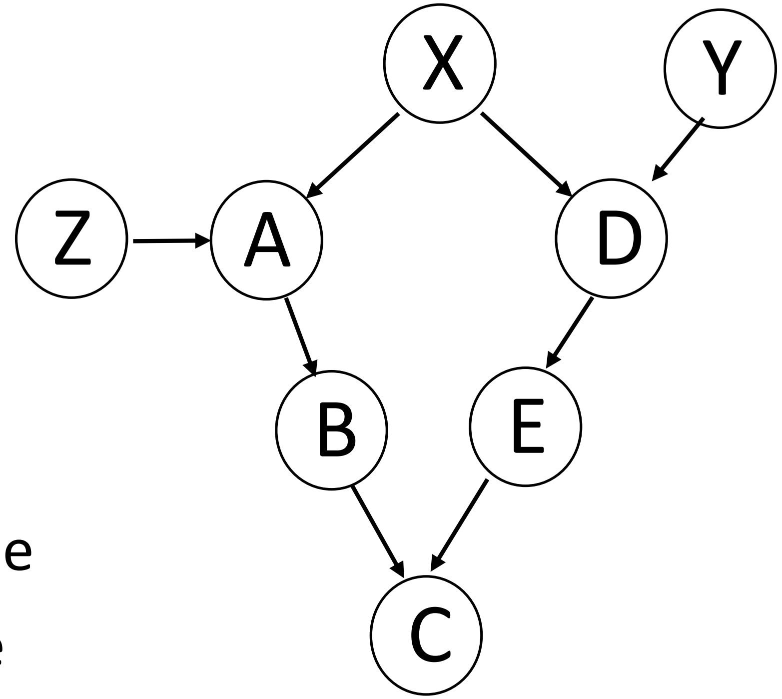
Inferenztypen in Bayesschen Netzen

- Enumerate-Joint-Ask-Algorithmus:
  - Berechnung bedingter Wahrscheinlichkeiten aus vollständiger Wahrscheinlichkeitsverteilung
  - Aufwand: Exponentiell zur Anzahl boolscher Variablen (n) im Netz: $O(n*2^n)$
    - Verbesserung bei (fast) einfach verbundenen Netzen auf O(n)
- Stochastische Simulationsmethoden
  - Direct Sampling (DS)
  - Likelihood Weighting (LW)
  - Markov Chain Monte Carlo (MCMC)
- Gesucht sei P(B|X,C)
  - Geben Sie einen Simulationslauf für LW an.
  - Geben Sie einen Simulationslauf für MCMC an.



Ein Simulationslauf von Likelihood Weighing

- Anfrage: P(B|X,C)
- Vorgesetzte Variablen: X=True, C=True
- Freie Variablen: Y, Z, A, D, B, E
- Setze X= True und werte den Lauf mit P(X=True)
- Wähle zufällig Wert für Y gemäß P(Y), z.B. True
- Wähle zufällig Wert für Z gemäß P(Z), z.B. False
- Wähle zufällig Wert für A gemäß P(A|Z=False,X=True), z.B. False
- Wähle zufällig Wert für D gemäß P(D|X=True,Y=True), z.B. True
- Wähle zufällig Wert für B gemäß P(B|A=False), z.B. True
- Wähle zufällig Wert für E gemäß P(E|D=True), z.B. False
- Setze C = True und werte den Lauf mit P(C|B=True,E=False)
- Ergebnis dieses Laufes: B=True mit Gewicht: P(X=True)*P(C|B=True,E=False)



Ein Simulationslauf von MCMC (Gibbs Sampling)

- Anfrage: P(B|X,C)
- Vorgesetzte Variablen: X=True, C=True
- Freie Variablen: Y, Z, A, D, B, E
- Initiale Variablenbelegung sei (zufällig) Y=True, Z=True, A=True, D=True, B=True, E=True
- Wähle zufällig Variable (gemäß Dichteverteilung) und ändere sie zufällig (gemäß ihrer aktuellen Markov Blanket)
- Wähle Y mit MB = P(Y|D=True,X=True); Sei Y = False
- Wähle D mit MB = P(D|X=True,Y=False,E=True); Sei D = True
- Wähle B mit MB = P(B|A=True,C=True,E=True); Sei B = False
- Wähle A mit MB = P(A|X=True,Z=True,B=True); Sei A = False
- Wähle B mit MB = P(B|A=False,C=True,E=True); Sei B = False
- usw. (N Wahlvorgänge)

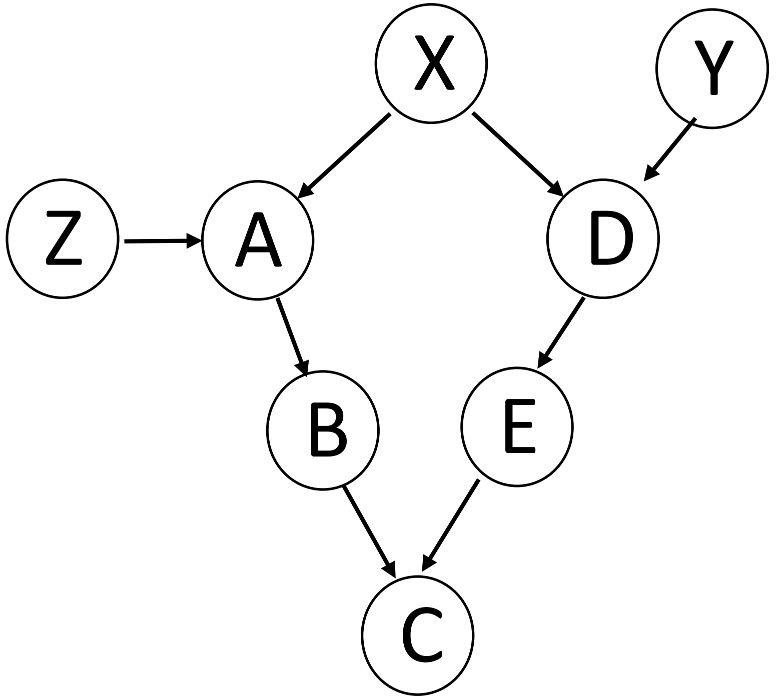
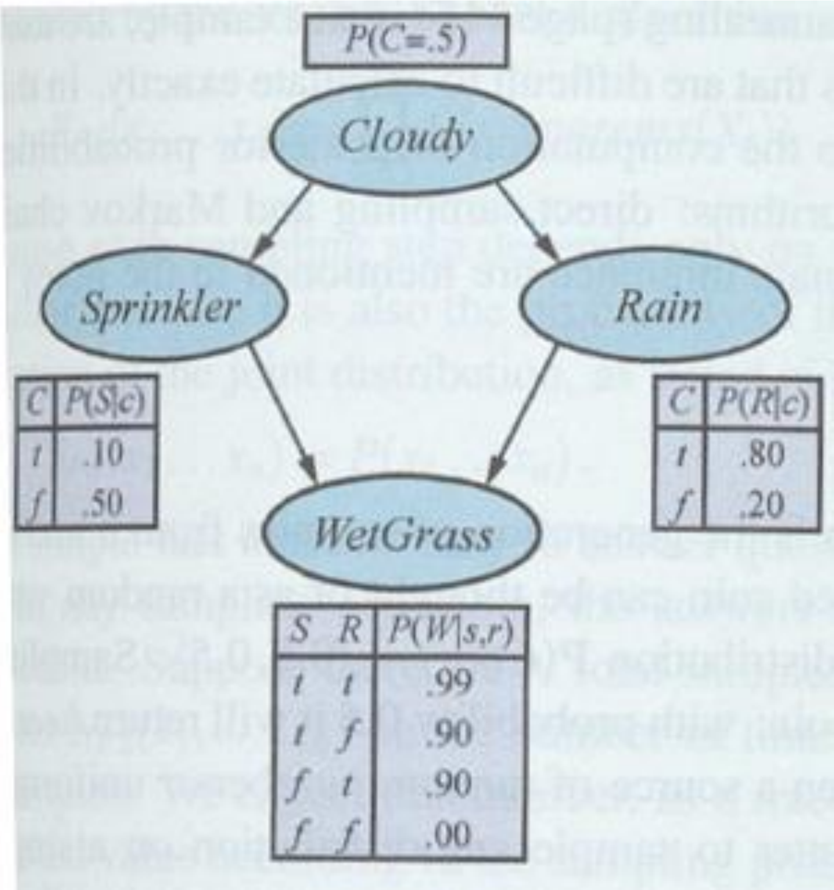
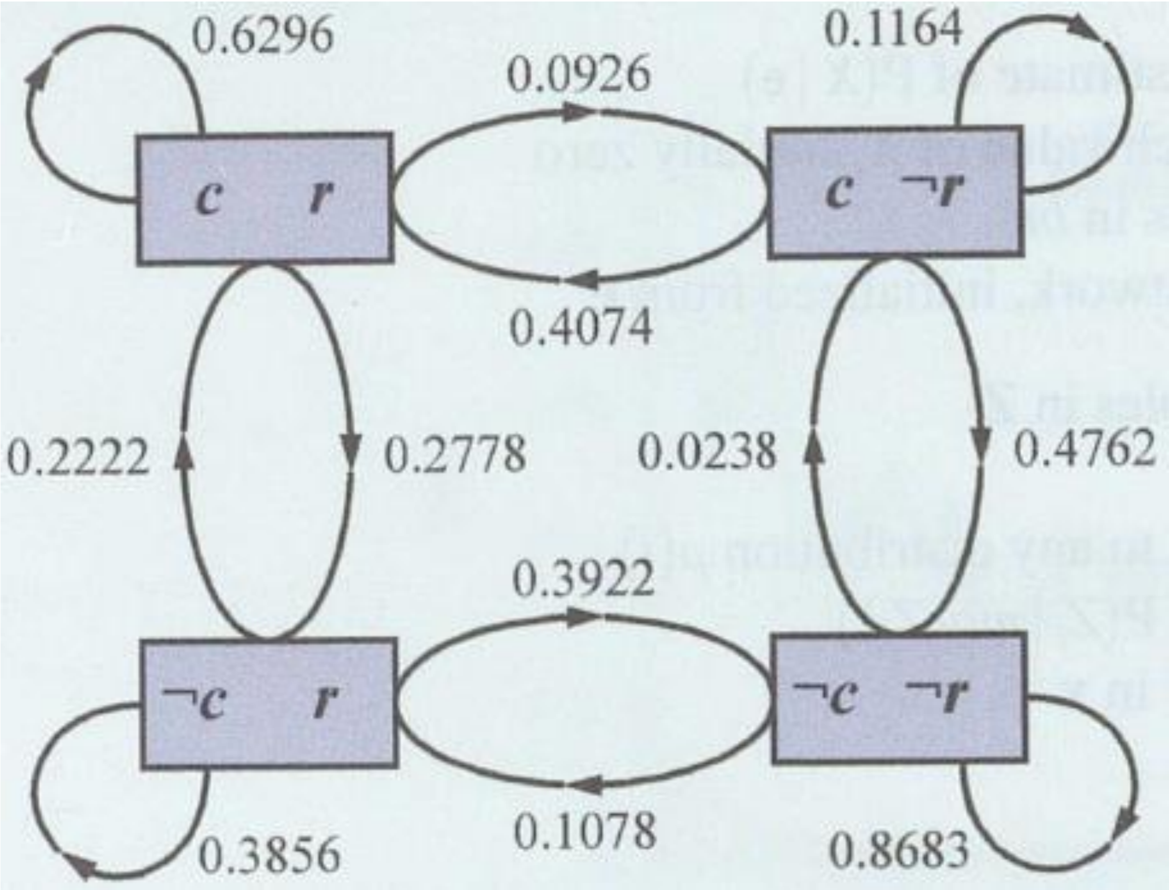


Illustration von Gibbs Sampling

- Alle Zustände und Transitionen für die Anfrage P(Rain|Sprinkler=True, WetGrass=True)
- Der Algorithmus wählt zufällig Variablen und wandert dann entsprechend den Wahrscheinlichkeiten der Kanten durch den Graph (rechts)





Effizienzüberlegungen zu Simulationsmethoden

- Unter welchen Bedingungen konvergieren Simulationsmethoden langsam?
- Wie kann man sie beschleunigen (um Faktor 100 bis 1000)?

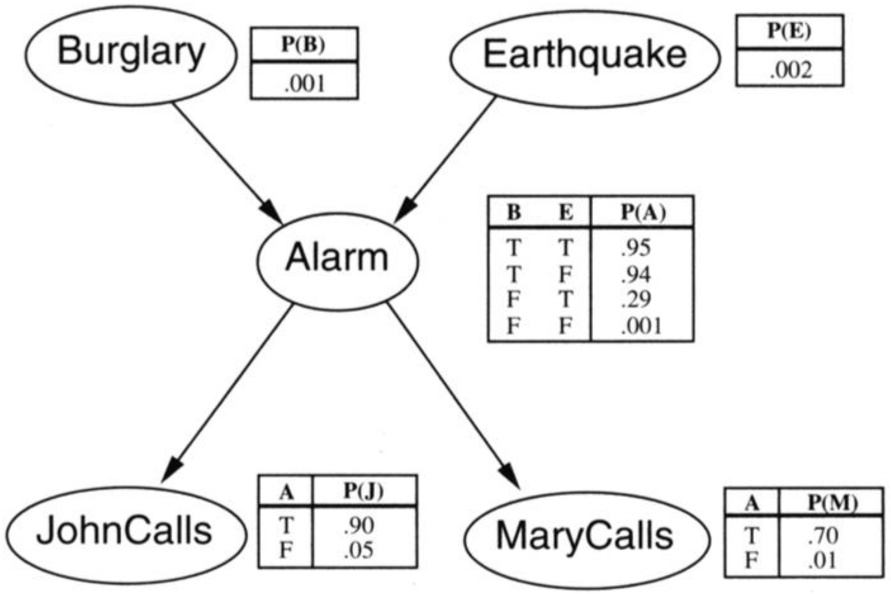
Effizienzüberlegungen zu Simulationsmethoden

- Unter welchen Bedingungen konvergieren Simulationsmethoden langsam?
  - Bei sehr geringen bedingten Wahrscheinlichkeiten braucht man sehr viele Durchläufe, damit man ausreichend Stichproben für diese erzeugen kann.

Effizienzüberlegungen zu Simulationsmethoden

Wie kann man sie beschleunigen (um Faktor 100 bis 1000)?

- Durch Kompilierung des Bayesschen Netzes in einen Zustandsautomat, z.B. für MCMC:
    - Schritt 1: Vorberechnung der Markov Blanket (MB) für jede freie Variable mit allen ihren Variablen/Wert-Konstellationen
    - Schritt 2: Umwandlung in Programmcode mit if-then-else-Abfragen der Variablen-Belegungen der MB für zu setzende Variable (s. rechts unten: die Variable „Earthquake" im „Earthquake-Netz" (s. rechts oben), deren MB die Variablen Alarm und Burglary umfasst)



$r \leftarrow$ a uniform random sample from $[0,1]$
if $Alarm = true$
  then if $Burglary = true$
    then return $[r < 0.0020212]$
    else return $[r < 0.36755]$
  else if $Burglary = true$
    then return $[r < 0.0016672]$
    else return $[r < 0.0014222]$

Chapter 14: Probabilistic Reasoning over Time

Anwendungen von zeitlichem, unsicherem Schließen

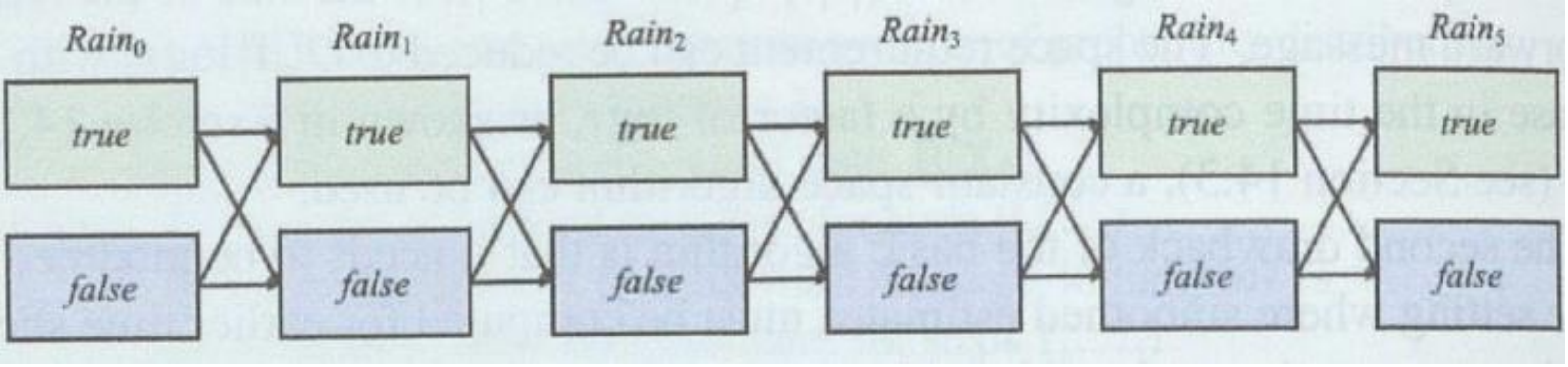
- Nennen Sie einige Anwendungen von zeitlichem, unsicherem Schließen:

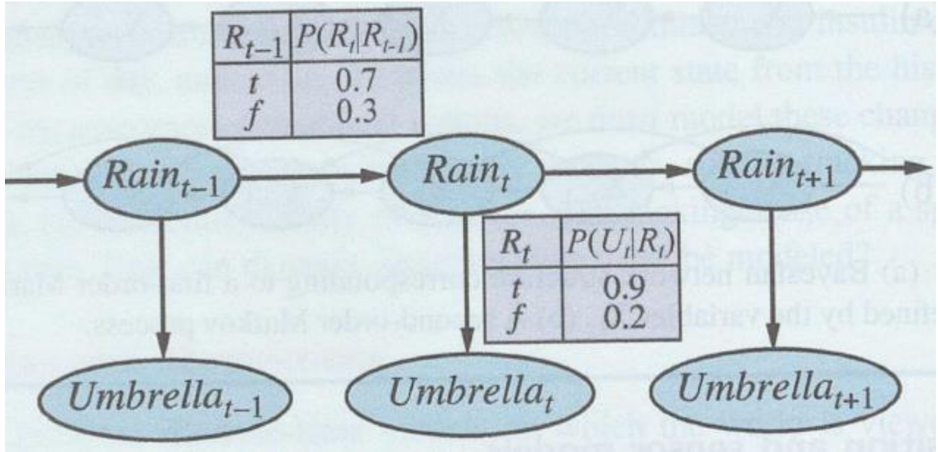
Anwendungen von zeitlichem, unsicherem Schließen

- Nennen Sie einige Anwendungen von zeitlichem, unsicherem Schließen:
  - NLP-Aufgaben: Übersetzung, Text-to-Speech, Speech-to-Text
  - Simulation geologischer Ereignisse, z.B. Continental-Drift
  - Tracking von Fahrzeugen, z.B. beim autonomen Fahren
  - Vorhersagen der Zukunft (??):
    - Wettervorhersage
    - Aktienkurse
    - wirtschaftlicher Entwicklungen, usw.
  - Simulation von Heizungsanlagen

Finding the Most Likely Sequence

- Assume we have the following observations: 5 days with umbrella except day 3
  - What is the most likely sequence for rain for the 5 days?





- **Recursive solution (Viterbi algorithm):** Compute most probable path to a state for a time step and expand it to the next time step (linear costs; similar to state estimation):

$m_{1:t+1} = \max_{x_{1:t}} P(x_{1:t}, X_{t+1}, e_{1:t+1}) =$

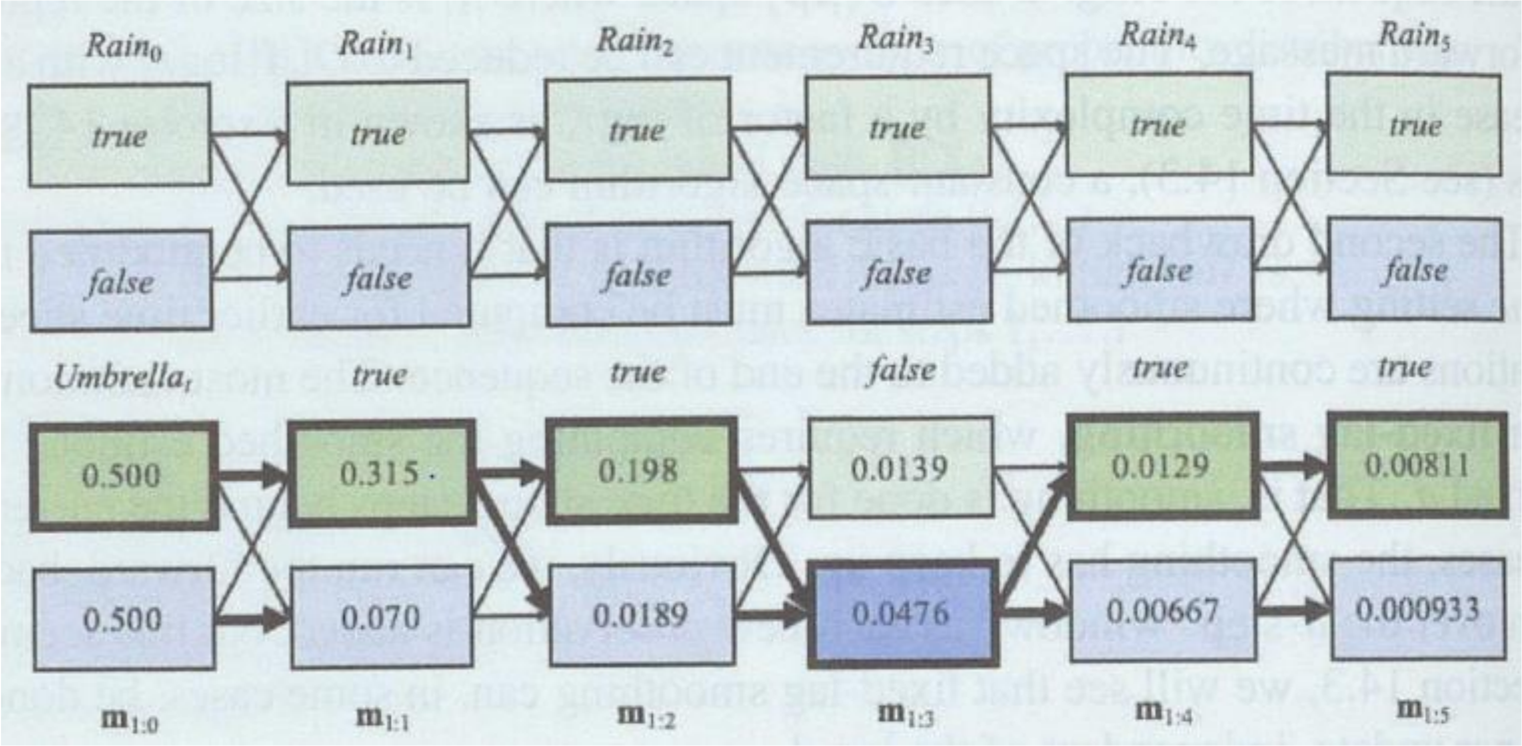
$\mathbf{P(e_{t+1} | X_{t+1})}$ sensor model

$\mathbf{\max_{x_t} P(X_{t+1} | x_t)}$ transition model

$\mathbf{\max_{x_{1:t-1}} P(x_{1:t-1}, x_t, e_{1:t})}$ Recursion t+1→t

Example for Viterbi Algorithm

- 5 days with umbrella except day 3;
- Most likely sequence for rain for the first 3 days (s. right)
- Continuation for day 4 and day 5?



Transition $m_{1:0}$ to $m_{1:1}$

case distinction (R=Rain; N=NoRain):

$P(R_{t-1})$ * transition * sensor

R→R: 0.5 * 0.7 * 0.9 = **0.3**

R→N: 0.5 * 0.3 * 0.2 = 0.03

N→R: 0.5 * 0.3 * 0.9 = 0.1

N→N: 0.5 * 0.7 * 0.2 = **0.07**

Transition $m_{1:1}$ to $m_{1:2}$

R→R: 0.3 * 0.7 * 0.9 = **0.2**

R→N: 0.3 * 0.3 * 0.2 = **0.02**

N→R: 0.07 * 0.3 * 0.9 = 0.02

N→N: 0.07 * 0.7 * 0.2 = 0.01

Transition $m_{1:2}$ to $m_{1:3}$

R→R: 0.2 * 0.7 * 0.1 = **0.01**

R→N: 0.2 * 0.3 * 0.8 = **0.05**

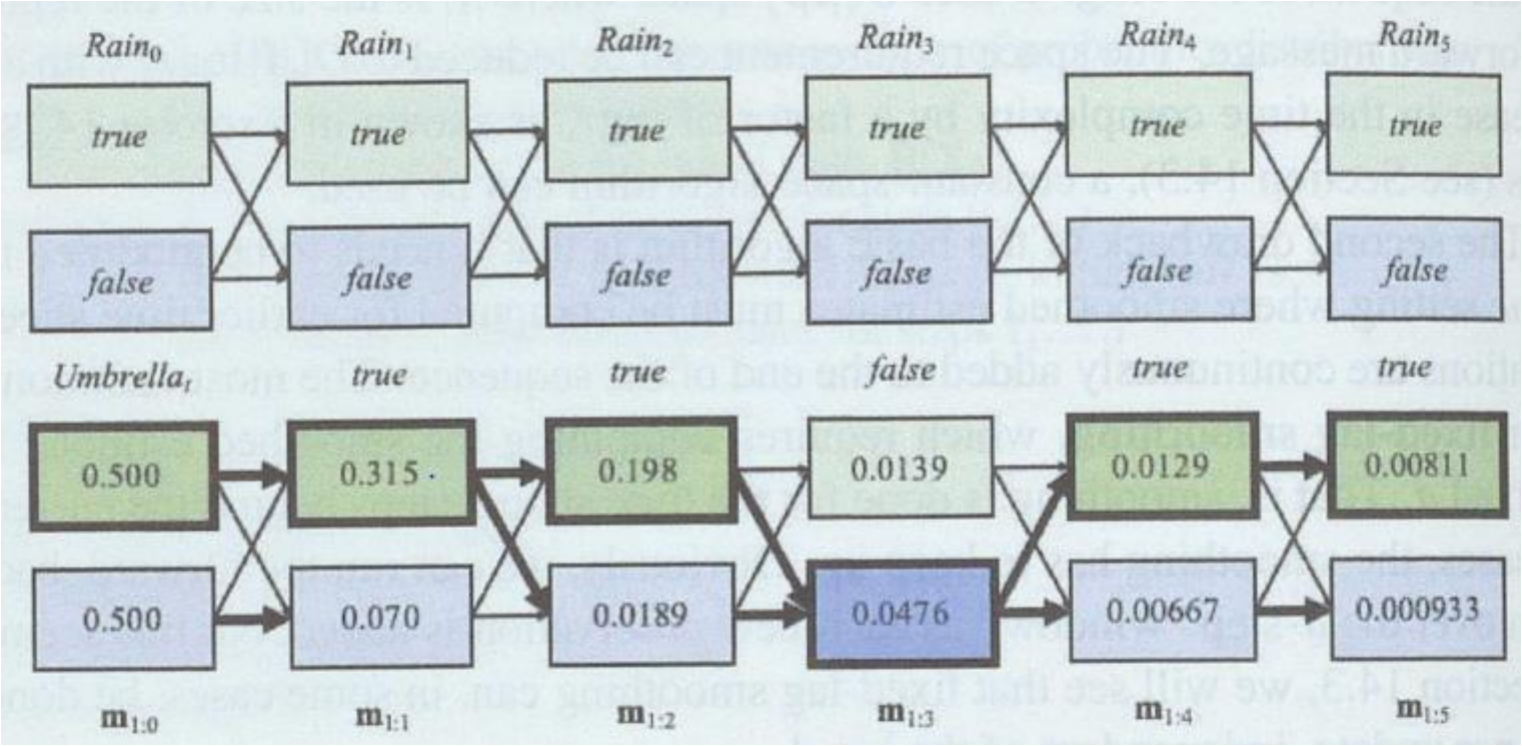
N→R: 0.02 * 0.3 * 0.1 = 0.0006

N→N: 0.02 * 0.7 * 0.8 = 0.01

Example for Viterbi Algorithm

Example continued: 5 days with umbrella except day 3

- What is the most likely sequence for rain for the 5 days?



Transition $m_{1:4}$ to $m_{1:5}$

R→R: 0.01 * 0.7 * 0.9 = **0.006**

R→N: 0.01 * 0.3 * 0.2 = 0.0006

N→R: 0.007 * 0.3 * 0.9 = 0.002

N→N: 0.007 * 0.7 * 0.2 = **0.001**

Transition $m_{1:0}$ to $m_{1:1}$

case distinction (R=Rain; N=NoRain):

$P(R_{t-1})$ * transition * sensor

R→R: 0.5 * 0.7 * 0.9 = **0.3**

R→N: 0.5 * 0.3 * 0.2 = 0.03

N→R: 0.5 * 0.3 * 0.9 = 0.1

N→N: 0.5 * 0.7 * 0.2 = **0.07**

Transition $m_{1:1}$ to $m_{1:2}$

R→R: 0.3 * 0.7 * 0.9 = **0.2**

R→N: 0.3 * 0.3 * 0.2 = **0.02**

N→R: 0.07 * 0.3 * 0.9 = 0.02

N→N: 0.07 * 0.7 * 0.2 = 0.01

Transition $m_{1:2}$ to $m_{1:3}$

R→R: 0.2 * 0.7 * 0.1 = **0.01**

R→N: 0.2 * 0.3 * 0.8 = **0.05**

N→R: 0.02 * 0.3 * 0.1 = 0.0006

N→N: 0.02 * 0.7 * 0.8 = 0.01

Transition $m_{1:3}$ to $m_{1:4}$

R→R: 0.01 * 0.7 * 0.9 = 0.006

R→N: 0.01 * 0.3 * 0.2 = 0.0006

N→R: 0.05 * 0.3 * 0.9 = **0.01**

N→N: 0.05 * 0.7 * 0.2 = **0.007**

Dynamic Belief Networks (DBN)

- Nennen Sie einen Vorteil von DBNs gegenüber Hidden Markov Models (HMMs) mit diskreten Variablen.
- Nennen Sie einen Vorteil von DBNs gegenüber Kalman Filters mit kontinuierlichen Variablen.

Dynamic Belief Networks (DBN)

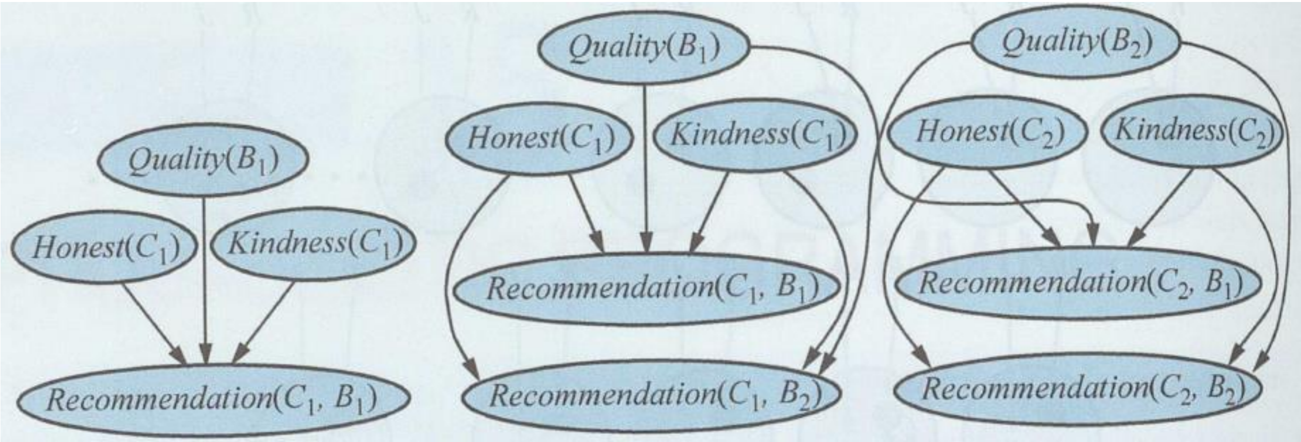
- Nennen Sie einen Vorteil von DBNs gegenüber Hidden Markov Models (HMMs) mit diskreten Variablen:
  - Kompaktere Representation: HMMs benötigen eine Wahrscheinlichkeitstabelle als Übergangsmatrix von einem Zustand zum nächsten, die exponentiell in der Anzahl der Variablen ist. Dafür sind die Inferenzalgorithmen für HMMs einfach und sehr effizienzt (z.B. Viterbi-Algorithmus)
- Nennen Sie einen Vorteil von DBNs gegenüber Kalman Filters (KFs) mit kontinuierlichen Variablen:
  - KFs benutzen multivariate Normalverteilungen (Gaussian distribution mit mehreren Variablen). Damit lassen sich nichtlineare Dichteverteilungen schlecht abbilden (z.B. das Verhalten eines Autos oder Fahrrad, wenn die Ampel vor ihm auf Gelb springt: Entweder Beschleunigen oder Abbremsen, aber keine Normalverteilung mit Weiterfahren als Mittelwert über viele Beobachtungen). In DBNs sind dagegen beliebige Dichteverteilungen möglich.

Chapter 15: Probabiliste Programming

Relational Probability Model (RPM) for Book Recommendations

Example: Recommendation (c,b) $\sim$ RecCPT (Honest(c), Kindnes (c), Quality (b))

- Recommendation conditional probability table (RecCPT) needs to specify all combinations of the values of its variables
- Values given by a type signature for the functions and predicates:
  - Recommendation: Customer x Book $\rightarrow$ {1, 2, 3, 4, 5}
  - Honest: Customer $\rightarrow$ {true, false}
  - Kindness: Customer $\rightarrow$ {1, 2, 3, 4, 5}
  - Quality: Book $\rightarrow$ {1, 2, 3, 4, 5}



- Prior probabilities for the random variables:
  - Honest (c) $\sim \langle 0.99, 0.01 \rangle$
  - Kindness (c) $\sim \langle 0.1, 0.1, 0.2, 0.3, 0.3 \rangle$
  - Quality (b) $\sim \langle 0.05, 0.2, 0.4, 0.2, 0.15 \rangle$
  - RecCPT for Recommendation (c,b) has 2 x 5 x 5 = 50 rows, each with 5 entries (250 in total)

Example: Rating Player Skill Levels

- Many competitive games have a rating for players‘ skill level, e.g. Elo rating for chess (beginner: around 800, world champion around 2800).
- We can develop a Bayesian rating scheme with the following functions and predicates:
  - Each player i has a skill level: Skill (i)
  - In each game g, the performance of i is: Performance (i, g)
  - Corresponding RPM:
    - Skill (i) $\sim$ Normalverteilung (μ, σ^2) [Gaussian Distribution (μ, σ^2)]
    - Performance (i,g) $\sim$ Normalverteilung (Skill (i), β^2) // β^2 variance of players actual performance
    - Win (i, j, g) = **if** Game (g, i, j) **then** (Performance (i, g) > Performance (j, g))

Elo Rating in Chess (and other Games) (aus Wikipedia)

Erwartungswert für Partie zwischen 2 Spielern A und B: Zahl zwischen 0 und 1

- kann man so interpretieren, dass bei 100 Partien die Anzahl der erzielten Punkte der beiden Spieler ihrem Erwartungswert * 100 entspricht.
- kann aus ELO-Differenz der beiden Spieler berechnet werden
  - als Tabelle (s. rechts)
  - als Formel für erwarteten Punktestand E_A für Spieler A, wobei R_A die Elo-Zahl von Spieler A und R_B die Elo-Zahl von Spieler B ist; $E_B = 1 - E_A$
- Erwartungswerte sind multiplikativ: Wenn E_A zu E_B = 3:1 (d.h. 0,75) und E_B zu E_C 2:1, dann ist E_A zu E_C 6:1.
- Berechnung der neuen Elo-Zahl R'_A von Spieler A nach einer Partie mit Ausgang S_A (1 = Gewinn; 0,5 = remis; 0 = Verlust) und Erwartungswert E_A gegen Spieler B:
  - **$R'_A = R_A + k\,(S_A - E_A)$**
  - k = 10 für Topspieler (Elo > 2400), k = 20 für gewöhnliche Spieler; k = 40 für Neulinge und Jugendliche (< 18).

$R_A - R_B$	E_A	E_B
0	0,50	0,50
100	0,64	0,36
200	0,76	0,24
300	0,85	0,15
400	0,91	0,09

$$E_A = \frac{1}{1 + 10^{\frac{R_B - R_A}{400}}}$$

Beispiel für Elo-Berechnung

Der Schachspieler Alfred (Elo: 2806 = R_A) spielt gegen die Schachspielerin Berta (Elo: 2577 = R_B). Das entspricht einem Wertungsunterschied von R_A - R_B = 229. Daher ist der Erwartungswert E_A, dass Alfred gegen Berta im Mittel 0,789 Punkte pro Spiel bekommt:

$$E_A = \frac{1}{1 + 10^{(2577-2806)/400}} = 0{,}789$$

Nach einem Spiel gibt es drei Möglichkeiten (angenommen sei dabei $k = 10$):

a) Berta gewinnt – also $S_A = 0$. Die neuen Elo-Punktestände R'_A für Alfred und R'_B für Berta sind

$$R'_A = 2806 + 10 \cdot (0 - 0{,}789) \approx 2798, \qquad R'_B = 2577 + 10 \cdot (1 - 0{,}211) \approx 2585.$$

Alfred büßt acht Elo-Punkte ein, während Berta acht Elo-Punkte gewinnt.

b) Alfred gewinnt – also $S_A = 1$.

$$R'_A = 2806 + 10 \cdot (1 - 0{,}789) \approx 2808, \qquad R'_B = 2577 + 10 \cdot (0 - 0{,}211) \approx 2575.$$

Alfred erhält zwei Elo-Punkte, Berta verliert zwei.

c) Unentschieden – also $S_A = 0,5$.

$$R'_A = 2806 + 10 \cdot (0{,}5 - 0{,}789) \approx 2803, \qquad R'_B = 2577 + 10 \cdot (0{,}5 - 0{,}211) \approx 2580.$$

Alfred verliert drei Elo-Punkte, Berta gewinnt drei.

Chapter 16: Making Simple Decisions

Nützlichkeit (Utility)

- Nützlichkeitstheorie benötigt Quantisierung der Nützlichkeit aller Entscheidungsalternativen
- Berechnungsschema ist einfach:
  - Erwartungswert: $\sum$ Wahrscheinlichkeit * Nützlichkeit aller Alternativen
  - Seltene Alternativen mit sehr hoher oder niedriger Nützlichkeit können dominieren
- Kandidaten für quantisierte Nützlichkeitsfunktionen:
  - Geld
  - Zeit
  - Lebenserwartung (Überlebensrate in der Medizin)
    - adjustierte Lebenserwartung (Quali)
  - Micromort (Risikoabschätzung für Todesfälle z.B. bei Sicherheitsstrategien)
- Fallstricke der Nützlichkeitsfunktionen:
  - Oft nicht-linearer Wert (z.B. bei Geld)
  - Unmittelbarer Nutzen oft wichtiger als langfristiger Nutzen

Lösung für Wette

- Zwei Optimale Sequenzen nach [1,1], [1,2] und [2,1], die sich aus den bisherigen Berechnungen ergeben:
  - [3,1], ([4,1]), [3,2] : Risiko für Loch: 31%
  - [1,3], [3,1], ([4,1]), [3,2]: 0.31 + (0.69 * 0.2) = 0.448 = 44,8%
- Mittelwert beider Sequenzen, da diese initial gleich wahrscheinlich sind: 37,9% Risiko, in ein Loch zu fallen
- Wir können daher eine Wette von 4 € zu 6 € (mindestens 3,79 € zu 6,21 €) anbieten, dass unser Agent das Gold findet (d.h. wir bekommen 4 € bzw. zahlen 6 €)

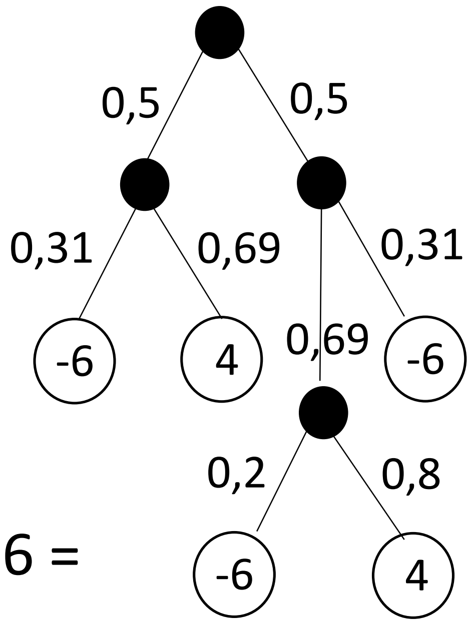
4	(B)			
3	(G)	Gold		
2	(B)			
1	Start	(B)	(B)	
	1	2	3	4

Wetten

- Das Design und die Evaluation der Chancen von Wetten basiert unmittelbar auf der Nützlichkeitstheorie.
  - Dabei werden zu Wahrscheinlichkeiten korrepondierende Nützlich-keiten berechnet, so dass zwei Alternativen gleich gut abschneiden.

	1	2	3	4
4	(B)			
3	(G)	Gold		
2	(B)			
1	Start	(B)	(B)	

- Beispiel (Wiederholung): Wette, dass Agent das Gold findet:
- Zwei optimale Sequenzen nach [1,1], [1,2] und [2,1]:
  - [3,1], ([4,1]), [3,2] : Risiko für Loch: 31%
  - [1,3], [3,1], ([4,1]), [3,2]: 0.31 + (0.69 * 0.2) = 0.448 = 44,8%
- Mittelwert beider gleich wahrscheinl. Sequenzen: 37,9% Risiko für Loch
- Wir können daher eine Wette von 4 € (Gewinn) zu 6 € (Verlust) (mind. 3,79 € zu 6,21 €) anbieten, dass Agent das Gold findet



- Berechnung durch Summe alle Erwartungswerte im Entscheidungsbaum:
- 0,5*0,31*-6 + 0,5*0,69*4 + 0,5*0,69*0,2*-6 + 0,5*0,69*0,8*4 + 0,5*0,31*-6 =

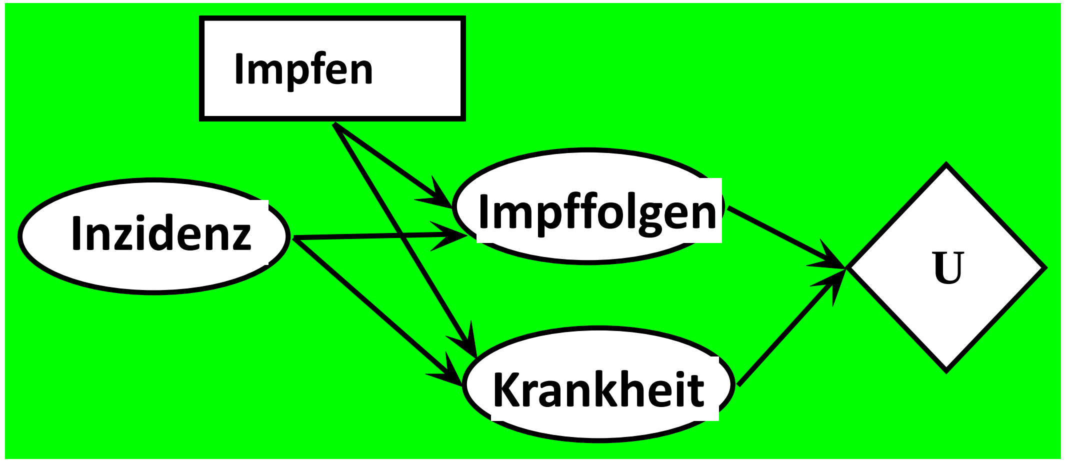
  -0,93 + 1,38 + -0,414 + 1,104 + -0,93 = **0.21**

Einfaches Beispiel

- Gegen eine Krankheit gibt es eine Impfung. Berechnen Sie die erwartete Nützlichkeit der Impfung unter folgenden Annahmen:
  - mit Impfung:
    - 90% nur leichte Beschwerden, die einen Tag dauern (Gewichtung -1)
    - 10% stärkere Beschwerden, die einem leichten Verlauf der Krankheit ähneln (-10)
    - zusätzlich: 40% bekommen die Krankheit trotz Impfung,
      - davon haben 90% einen leichten Verlauf (-10) und
      - 10% einen schweren Verlauf (-100)
  - ohne Impfung:
    - 20%: keine Krankheit
    - 70% Krankheit mit leichtem Verlauf (-10)
    - 9% Krankheit mit schwerem Verlauf (-100)
    - 1% Krankheit mit sehr schwerem Verlauf (-1000)
- Von welchen Annahmen hängt die erwartete Nützlichkeit der Impfung hauptsächlich ab?

Einfaches Beispiel als Entscheidungsnetzwerk

Entscheidungsnetzwerk (Decision network): CPTs (IF = Impffolgen; K = Krankheit)



Utilities (U)		-1	-10	-10	-100	-10000
Inzidenz	Impfen	P(IF1)	P(IF2)	P(K1)	P(K2)	P(K3)
H	Ja	0,9	0,1	0,4*0,9	0,4*0,1	0
H	Nein	0	0	0,7	0,09	0,01

Kosten(Impfung) + U(IF=x)*P(IF=x) + U(K=y)*P(K=y) = U

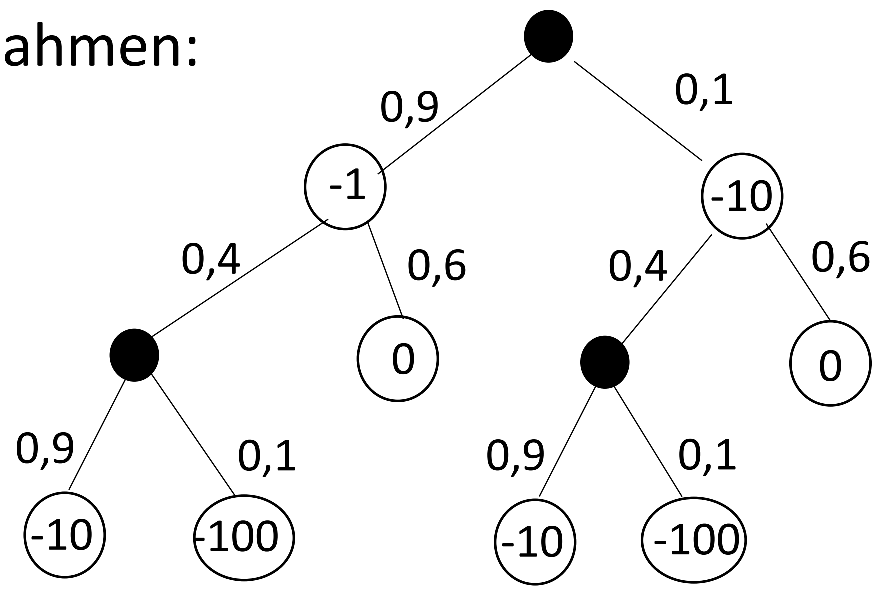
Inzidenz=Hoch, Impfen=Ja: 0 + -1*0,9 + -10*0,1 + -10*0.36 + -100*0,04 + 0 = **-9,5**

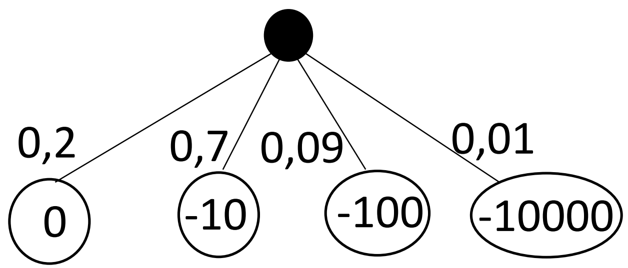
Inzidenz=Hoch, Impfen=Nein: 0 + 0 + 0 + -10*0.7 + -100*0,09 + -10000*0,01 = **-116**

$\Rightarrow$ *Impfen hat deutlich bessere Nützlichkeit*

Einfaches Beispiel als Entscheidungsbaum

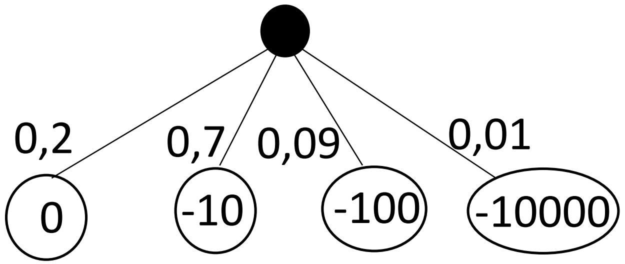
- Gegen eine Krankheit gibt es eine Impfung. Berechnen Sie die erwartete Nützlichkeit der Impfung unter folgenden Annahmen:
- mit Impfung:
  - 90% nur leichte Beschwerden, die einen Tag dauern (Gewichtung -1)
  - 10% stärkere Beschwerden, die einem leichten Verlauf der Krankheit ähneln (-10)
  - zusätzlich: 40% bekommen die Krankheit trotz Impfung, davon haben 90% einen leichten Verlauf (-10) und 10% einen schweren Verlauf (-100)
- ohne Impfung:
  - 20%: keine Krankheit
  - 70% Krankheit mit leichtem Verlauf (-10)
  - 9% Krankheit mit schwerem Verlauf (-100)
  - 1% Krankheit mit sehr schwerem Verlauf (-10000)



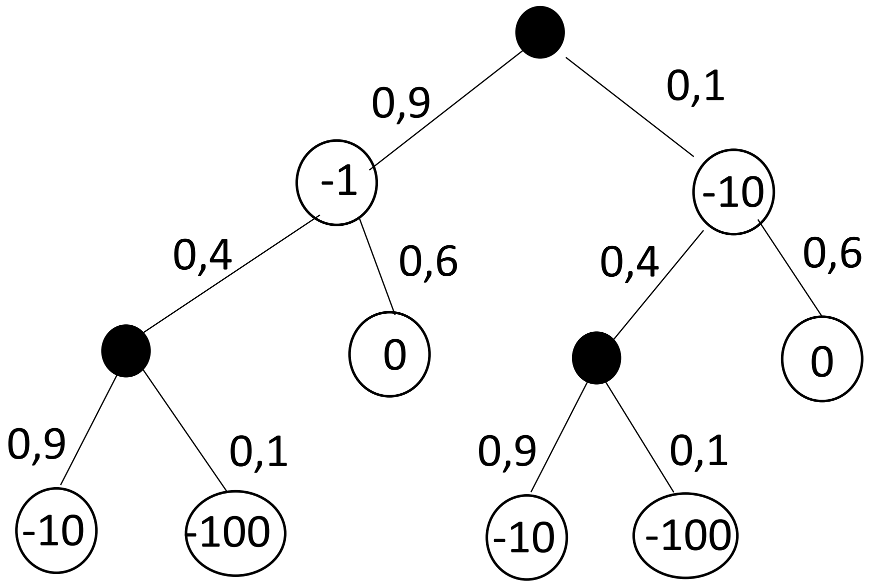


Lösung für einfaches Beispiel als Entscheidungsbaum

- **ohne Impfung:**
  - keine Krankheit: 0,2 * 0 = 0
  - Krankheit mit leichtem Verlauf: 0,7 * -10 = -7
  - Krankheit mit schwerem Verlauf: 0,09 * -100 = -9
  - Krankheit mit sehr schwerem Verlauf: 0,01 * -10000 = -100
  - **Expected Utility = -7 + -9 + -100 = -116**



- **mit Impfung:**
  - leichte Beschwerden: 0,9 * -1 = -0,9 +
    - ohne Krankheit: 0
    - mit leichter Krankheit: 0,9 * 0,4 * 0,9 * -10 = -3,24
    - mit schwerer Krankheit: 0,9 * 0,4 * 0,1 * -100 = -3,6
  - stärkere Beschwerden: 0,1 * -10 = -1 +
    - ohne Krankheit: 0
    - mit leichter Krankheit: 0,4 * 0,1 * 0,9 * -10 = -0,36
    - mit schwerer Krankheit: 0,4 * 0,1 * 0,1 * -100 = -0,04
  - **Expected Utility** = -0,9 + -3,24 + -3,6 + -1 + -0,36 + -0,4 = **-9,5**



Sensititätsanalyse

- Die erwartete Nützlichkeitkeit hängt hauptsächlich von der Wahrscheinlichkeit (0,01) und der Nützlichkeit (Schädlichkeit; -10000) der Krankheit ohne Impfung mit sehr schwerem Verlauf ab, auch von deren Abwesenheit bei der Impfung.

Knowledge Engineering Process

- It is useful to separate probabilities and utilities in a model
- **Knoweldge engineering process:**
  1. Sufficient understanding of the domain
  2. Development of causal model
     - Terminology (variables)
     - Causal relations (links)
     - Optional simplification by removing, aggregating oder seperating variables
  3. Assignment of probabilities
  4. Assignment of utilities
  5. Model evaluation and refinement with cases (gold standard)
  6. Sensitivity analysis of model (optionally add hyperparameters based on parameter ranges)
  7. Application to problem
  8. Sensitivity analysis of result (optionally with „worst case analysis“ based on ranges)

Komplexes Beispiel: Lockdown

- Lässt sich der Nutzen eines Lockdown berechnen?
- Welche Variablen müssen dafür erfasst werden?

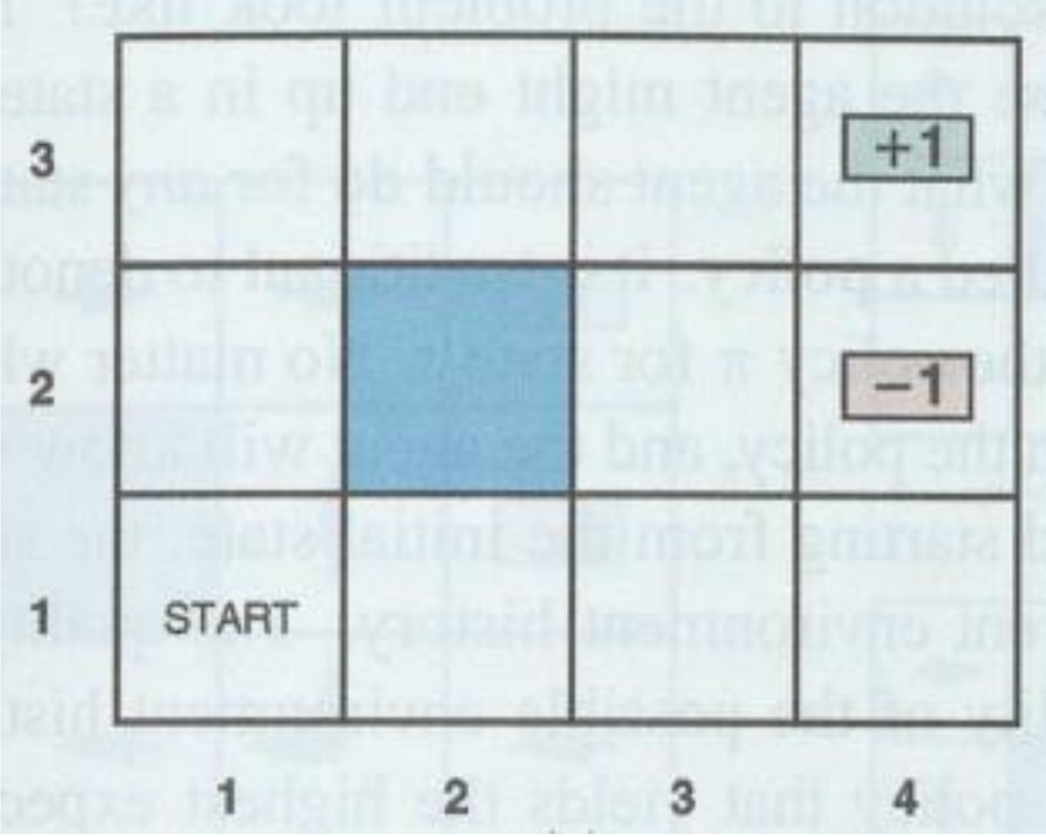
Beispiel Lockdown: Fragen zur Berechnung des Nutzens

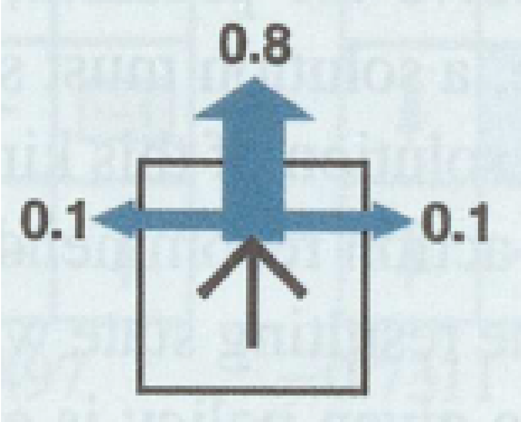
- Voraussetzung: Genaue Festlegegung der Art und Dauer eines Lockdowns
- Kosten (wie messen?):
  - Schaden für die Wirtschaft
  - Schaden für das Wohlbefinden von Individuen
  - Kosten für die Durchsetzung eines Lockdowns
- Nutzen (wie messen?):
  - Für die Wirtschaft: Vermeidung von Arbeitsausfällen
  - Für das Wohlbefinden von Individuen
    - Reduktion von Infizierten
    - Reduktion von Krankenhaus bzw. Intensiv-Station-Belegung
    - Reduktion von Langzeitschäden
    - Reduktion von Todesfällen
- Gesellschaftliche Auswirkungen: Schaden durch Konflikte; Nutzen durch Erwartungen
- Kurz- und langfristige Auswirkungen unterscheiden
- Wahrscheinlichkeitsniveaus für verschiedene Szenarien und deren Auswirkungen notwendig

Chapter 17. Making Complex Decisions

Example for Sequential Decision Problem

- The agent wants to maximize its reward
  - For reaching a terminal state (4,2) and (4,3) the reward is -1 resp. +1.
  - For each movement (action) the reward is -0.04
- In each state, the agent can choose among four actions (up, down, left, right) and has a probability of 80% to reach the intended state, and 20% to move at right angles to the intended direction
  - Collision with a wall results in no movement
- If the agent reaches the state (4,3) e.g. in 10 steps, its total utility is 9 * -0.04 + 1= +0.64





Value Iteration

- Compute utilities of states iteratively until the biggest change (δ) is very small ($\leq \varepsilon(1-\gamma)/\gamma$)
- Bellman update: $U_{i+1}(s) \leftarrow \max_{a \in A(s)} \sum_{s'} P(s'|s,a)\,[R(s,a,s') + \gamma U_i\,(s')]$ or
  $U_{i+1}(s) \leftarrow \max_{a \in A(s)}$ Q-value (mdp, s, a, U)

function VALUE-ITERATION(mdp, ϵ) **returns** a utility function
  inputs: mdp, an MDP with states S, actions $A(s)$, transition model $P(s'\,|\,s,a)$,
    rewards $R(s,a,s')$, discount γ
    ϵ, the maximum error allowed in the utility of any state
  local variables: U, U', vectors of utilities for states in S, initially zero
    δ, the maximum relative change in the utility of any state

  repeat
    $U \leftarrow U'$; $\delta \leftarrow 0$
    for each state s in S **do**
      $U'[s] \leftarrow \max_{a \in A(s)}$ Q-VALUE(mdp, s, a, U)
      if $|U'[s] - U[s]| > \delta$ **then** $\delta \leftarrow |U'[s] - U[s]|$
  until $\delta \leq \epsilon(1-\gamma)/\gamma$
  return U

Example for Value Iteration

U(3,3) = max { [0.8 *1 + 0.1 (-0.04 + U(3,2)) + 0.1 (-0.04 + U(3,3))], //Right

[0.8(-0.04 + U(3,3)) + 0.1 *1 + 0.1 (-0.04 + U(2,3))] //Up

[0.8(-0.04 + U(2,3)) + 0.1 (-0.04 + U(3,3)) + 0.1 (-0.04 + U(3,2))] //Left

[0.8(-0.04 + U(3,2)) + 0.1 * 1 + 0.1 (-0.04 + U(2,3))] } //Down

3	0	0	0	1
2	0		0	-1
1	0	0	0	0
	1	2	3	4

Computations 1. Iteration:

U(3,3): Right: 0.8*1 + 0.1*-0.04 + 0.1*-0.04 = 0.792

U(3,2): Left: 0.8*-0.04 + 0.1*-0.04 + 0.1*-0.04 = -0.04

… (other states = -0.04)

3			→	1
2			←	-1
1				↓
	1	2	3	4

3	–0.04	–0.04	0.792	1
2	–0.04		–0.04	-1
1	–0.04	–0.04	–0.04	–0.04
	1	2	3	4

Computations 2. Iteration:

U(3,3): Right: 0.8*1 + 0.1*(-0.04 -0.04) + 0.1*(0.792-0.04) = 0.8672

U(3,3): Left: **0.8*(-0.04-0.04) + 0.1*(0.792-0.04) + 0.1*(-0.04-0.04) = 0.0032**

U(3,2): Up: 0.8*(0.792-0.04) + 0.1*-1 + 0.1*(-0.04-0.04) = 0.4936

U(2,3): Right: 0.8*(0.792-0.04) + 0.2*(-0.04-0.04) = 0.5856

…

3		→	→	1
2			↑	-1
1				↓
	1	2	3	4

3	–0.08	0.5856	0.8672	1
2	–0.08		0.4936	-1
1	–0.08	–0.08	–0.08	–0.08
	1	2	3	4

Computations 3. Iteration:

…

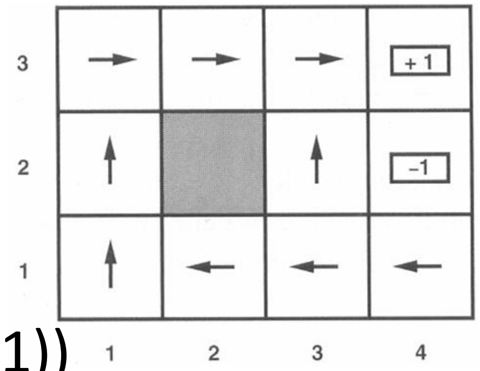
Policy Iteration

- Policy iteration algorithm alternates between two steps, beginning with some initial policy π
  - Policy evaluation: Given a policy π_i, calculate its uility U_l
  - Policy improvement: Calculate a new policy π_{i+1} based on U_i
- Termination, when the policy improvement step yields no change in the utilities

```
function POLICY-ITERATION(mdp) returns a policy
   inputs: mdp, an MDP with states S, actions A(s), transition model P(s' | s, a)
   local variables: U, a vector of utilities for states in S, initially zero
                    π, a policy vector indexed by state, initially random

   repeat
      U ← POLICY-EVALUATION(π, U, mdp)
      unchanged? ← true
      for each state s in S do
         a* ← argmax_{a∈A(s)} Q-VALUE(mdp, s, a, U)
         if Q-VALUE(mdp, s, a*, U) > Q-VALUE(mdp, s, π[s], U) then
            π[s] ← a*; unchanged? ← false
   until unchanged?
   return π
```

Methods for calculating the Utility in Policy Iteration



- **Analytic version:** Solving n equations with n unknowns with linear algebra methods in $O(n^3)$, since the policy is known
  - Example with π_i (1,1) = Up, π_i (1,2) = Up, etc.
    - $U_i(1,1) = 0.8(-0.04 + U_i(1,2)) + 0.1\,(-0.04 + U_i(2,1)) + 0.1\,(-0.04 + U_i(1,1))$
    - $U_i(1,2) = 0.8(-0.04 + U_i(1,3)) + 0.2\,(-0.04 + U_i(1,2))$
    - …
- **Iterative version:** Simplified version of Belman update in value iteration with given policy
  - $U_{i+1}(s) \leftarrow \sum_{s'} P(s'|s, \pi_i(s))\,[R(s, \pi_i(s), s') + \gamma U_i\,(s')]$ *// i.e. Q-value (mdp, s, $\pi_i(s)$, U)*
    - γ = Discount factor lowering future utilities; however in example: $\gamma = 1$
  - Must be repeated several times to produce next utility estimate

- **Asynchronous policy iteration:**
  - It is not necessary to update all states
  - Choosing any subset and perform either kind of updating of utilities is possible

Example Policy Iteration

Nur ein Durchlauf für Nützlichkeitsberechnung!

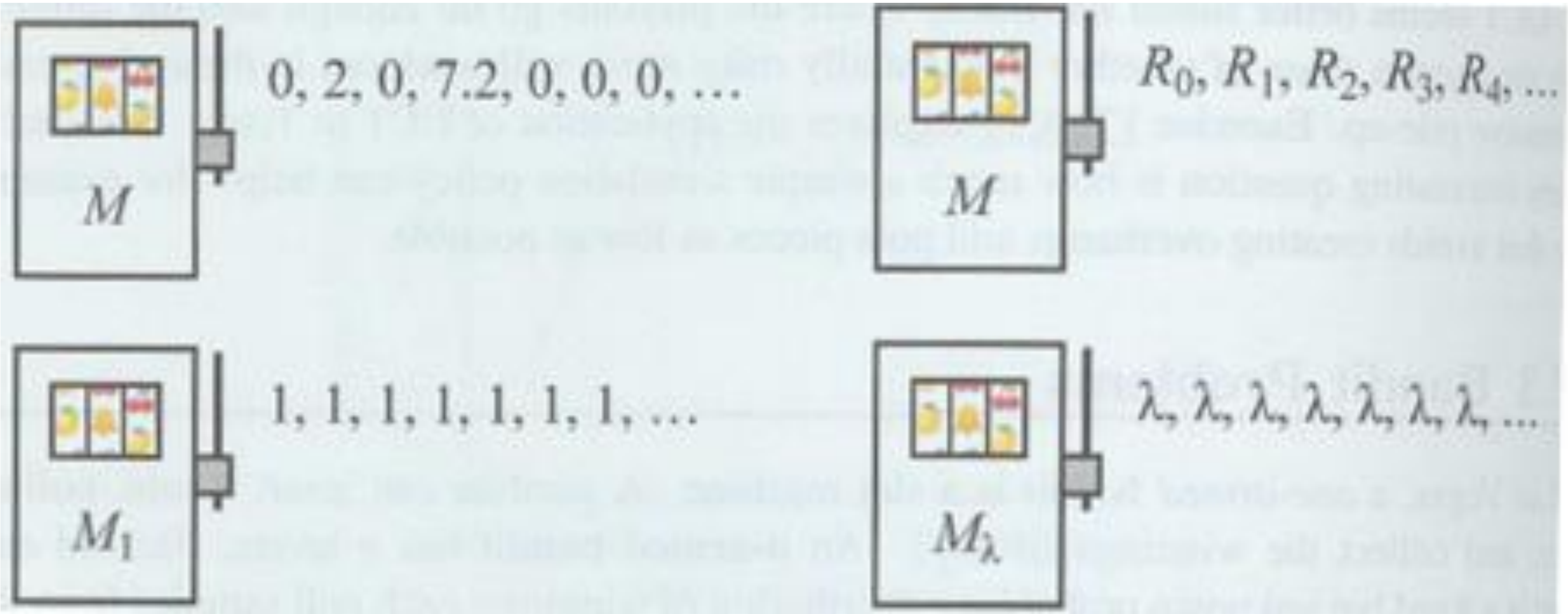
- Initiale Nützlichkeiten von 0 für Zustände und initiale Politik (↑ = „Up“)
- Neu berechnete Nützlichkeiten auf Basis der initialen Politik und resultierende geänderte Politik (**5 Zustände**; z.B. ändert sich Politik in Feld[3,3] von ↑ auf →)
- Neu berechnete Nützlichkeiten (**Formel für Feld [3,3] als Beispiel**) auf Basis der geänderten Politik und resultierende erneut geänderte Politik (**2 Zustände**)
- Neu berechnete Nützlichkeiten, die keine Änderungen in der Politik ergeben → **Terminierung**

J37 =-0,04 + 0,8*(K32+0,04)+0,1*J32+0,1*J33

	G	H	I	J	K	L	M	N	O	P	Q
27	3	0	0	0	1		3	↑	↑	↑	1
28	2	0		0	-1		2	↑		↑	-1
29	1	0	0	0	0		1	↑	↑	↑	↑
30		1	2	3	4			1	2	3	4
31											
32	3	–0,040	–0,040	0,064	1		3	↑	→	→	1
33	2	–0,040		–0,136	-1		2	↑		←	-1
34	1	–0,040	–0,040	–0,040	–0,808		1	↑	↑	←	←
35		1	2	3	4			1	2	3	4
36											
37	3	–0,080	0,003	0,785	1		3	→	→	→	1
38	2	–0,080		–0,146	-1		2	↑		↑	-1
39	1	–0,080	–0,080	–0,090	–0,249		1	↑	←	←	←
40		1	2	3	4			1	2	3	4
41											
42	3	–0,112	0,588	0,856	1		3	→	→	→	1
43	2	–0,120		–0,088	-1		2	↑		↑	-1
44	1	–0,120	–0,121	–0,128	–0,233		1	↑	←	←	←
45		1	2	3	4			1	2	3	4

Banditen-Probleme

- Formales Modell für Investitionsentscheidungen, z.B.
  - Kauf von Aktien
  - Förderung von Projekten
  - Optimierung von Arbeitszeit für unterschiedliche Tätigkeiten
  - …
- Jeder Bandit M (jede Option) liefert Belohnungen entsprechend einer unbekannten Verteilung
- Der Wert eines Banditen M wird mit dem Wert eines Standard-Banditen M_λ verglichen, der eine konstante Belohnung liefert (λ; z.B. λ =1)
- Optimal ist eine Strategie, die zu einem Zeitpunkt T den Banditen M wählt, solange seine durchschnittliche Belohnung höher ist als die von M_λ, und danach zu M_λ wechselt (hier nach T = 4)



Gittins Index

What is the value of λ of the one-armed bandit, so that an optimal strategy (with the best stopping time) is eqivalent to stop immediately?

$$\max_{T>0} E\left[\left(\sum_{t=0}^{T-1} \gamma^t R_t\right) + \sum_{t=T}^{\infty} \gamma^t \lambda\right] = \sum_{t=0}^{\infty} \gamma^t \lambda$$

which simplifies to

Gittins index:

$$\lambda = \max_{T>0} \frac{E\left(\sum_{t=0}^{T-1} \gamma^t R_t\right)}{E\left(\sum_{t=0}^{T-1} \gamma^t\right)}$$

Computation of Gittins index for example:

T	1	2	3	4	5	6
R_t	0	2	0	7.2	0	0
$\sum \gamma^t R_t$	0.0	1.0	1.0	1.9	1.9	1.9
$\sum \gamma^t$	1.0	1.5	1.75	1.875	1.9375	1.9687
ratio	0.0	0.6667	0.5714	1.0133	0.9806	0.9651

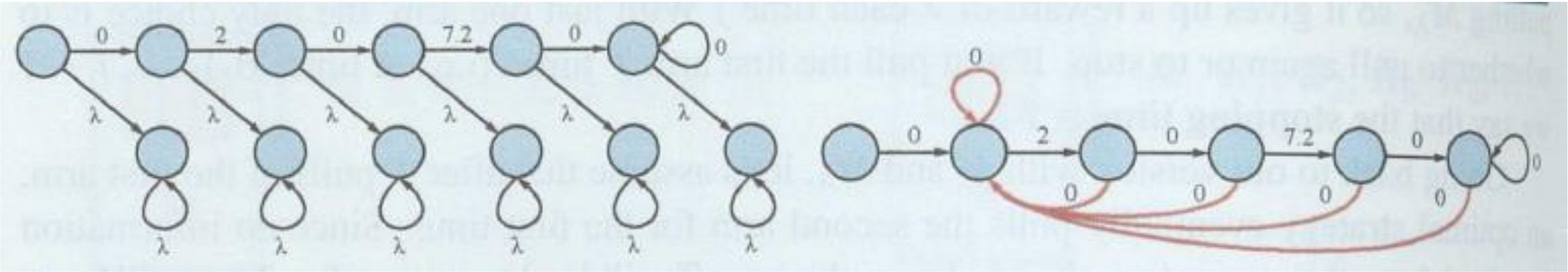
Gittins Index

- Berechne durchschnittlichen Gewinn pro Zeiteinheit mit Berücksichtigung von Discount-Faktor γ
- Der maximale Wert ist der Gittins-Index (hier bei T=4):
  - bei γ = 0,5 $\rightarrow$ 1,01
  - bei γ = 0,9 $\rightarrow$ 2,05
- Alternative Berechnung als MDP

T	1	2	3	4	5	6	γ
R_t	0	2	0	7,2	0	0	
$\sum \gamma^t R_t$	0	1	1	1,9	1,9	1,9	0,5
$\sum \gamma^t$	1	1,5	1,75	1,875	1,9375	1,9688	
Ratio	0	0,667	0,571	1,013	0,9806	0,9651	

T	1	2	3	4	5	6	
R_t	0	2	0	7,2	0	0	
$\sum \gamma^t R_t$	0	1,8	1,8	7,049	7,0488	7,0488	0,9
$\sum \gamma^t$	1	1,9	2,71	3,439	4,0951	4,6856	
Ratio	0	0,947	0,664	2,05	1,7213	1,5044	

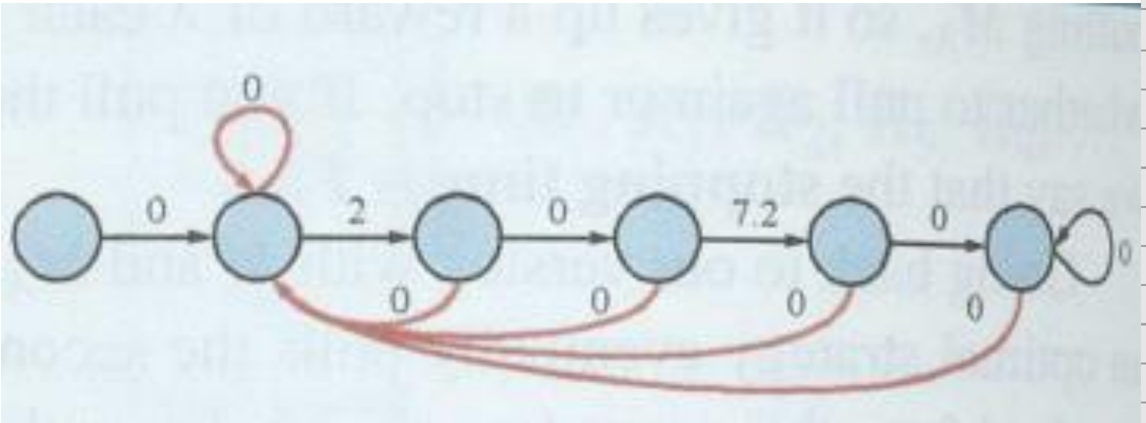
Gittins Index als MDP berechnen



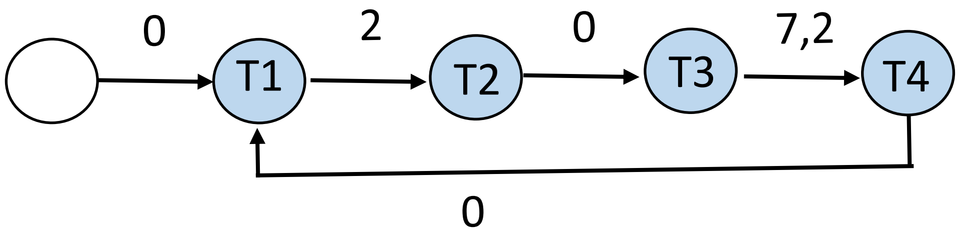
- Zu jedem Zeitpunkt kann man die Belohnung des Banditen bekommen oder auf Dauer zur Default-Belohnung λ wechseln (links)
- Das kann man als Restart-MDP darstellen (rechts), bei dem von jedem Zustand zum Startzustand zurückkehren kann. Die optimale Strategie des Restart-MDP hat als Wert des Start-Zustandes, multipliziert mit (1-γ) den Gittins-Index (d.h. befindet sich im Gleichgewicht).

Value-Iteration für Restart-MDP mit Bsp.-Daten

T	1	2	3	4
R_t	0	2	0	7,2
Utility	0	2	0	7,2
Utility	1	2	3,6	7,2
Utility	1	3,8	3,6	7,7
Utility	1,9	3,8	3,85	7,7
Utility	1,9	3,925	3,85	8,15
Utility	1,9625	3,925	4,075	8,15
Utility	1,9625	4,038	4,075	8,181
Utility	2,0188	4,038	4,091	8,181
Utility	2,0188	4,045	4,091	8,209
Utility	2,0227	4,045	4,105	8,209
Utility	2,0227	4,052	4,105	8,211
Utility	2,0262	4,052	4,106	8,211
Utility	2,0262	4,053	4,106	8,213
Utility	2,0264	4,053	4,107	8,213



Vereinfacht zu folgendem Automaten mit Knoten T1 bis T4 und Value Iteration in Excel



Links: Ergebnis mit Discount-Faktor γ = 0,5

U (T1) = 2,026; λ = U(T1)*(1- γ) = 1,013

Rechts: Ergebnis mit Discount-Faktor γ = 0,9

U (T1) = 19,86; λ = U(T1)*(1- γ) = 1,99 (2,05)

T	1	2	3	4
R_t	0	2	0	7,2
Utility	0	2	0	7,2
Utility	1,8	2	6,48	7,2
Utility	1,8	7,832	6,48	8,82
Utility	7,0488	7,832	7,938	8,82
Utility	7,0488	9,144	7,938	13,54
Utility	8,2298	9,144	12,19	13,54
Utility	8,2298	12,97	12,19	14,61
Utility	11,674	12,97	13,15	14,61
Utility	11,674	13,83	13,15	17,71
Utility	12,448	13,83	15,94	17,71
Utility	12,448	16,34	15,94	18,4
Utility	14,708	16,34	16,56	18,4
Utility	14,708	16,91	16,56	20,44
Utility	15,216	16,91	18,39	20,44
Utility	15,216	18,55	18,39	20,89
Utility	16,699	18,55	18,81	20,89
Utility	16,699	18,92	18,81	22,23
Utility	17,032	18,92	20,01	22,23
Utility	17,032	20,01	20,01	22,53
Utility	18,005	20,01	20,28	22,53
Utility	18,005	20,25	20,28	23,4
Utility	18,224	20,25	21,06	23,4
Utility	18,224	20,96	21,06	23,6
Utility	18,862	20,96	21,24	23,6
Utility	18,862	21,12	21,24	24,18
Utility	19,005	21,12	21,76	24,18
Utility	19,005	21,58	21,76	24,3
Utility	19,424	21,58	21,87	24,3
Utility	19,424	21,69	21,87	24,68
Utility	19,518	21,69	22,21	24,68
Utility	19,518	21,99	22,21	24,77
Utility	19,793	21,99	22,29	24,77
Utility	19,793	22,06	22,29	25,01
Utility	19,855	22,06	22,51	25,01

Chapter 18. Multiagent Decision Making

Example of Maximin for Two-finger Morra

Maximin technique:
- Generate n linear equations for n actions (linear programming problem)

	O: *one*	O: *two*
E: *one*	$E = +2, O = -2$	$E = -3, O = +3$
E: *two*	$E = -3, O = +3$	$E = +4, O = -4$

- Define the mixed strategy for E „one“ with probability p and „two“ with probability 1-p
- The outcome for E is:
  - If O chooses always „one“: 2p -3(1-p)
  - If O chooses always „two“: -3p + 4(1-p)
  - In the intersection holds: 5p – 3 = 4 – 7p, i.e. p=7/12 („one“) for E
- Similar for O with the same result (7/12: „one“, 5/12: „two“)
- Result: for O: +1/12; for E: -1/12
- Justification: Mixed strategies for O are not better, if E reveals its strategy
  - Mixed strategy for O (p U_{one} + (1-p) U_{two}) cannot be better than the better of U_{one} or U_{two}

Maximin Technique für Schere, Stein, Papier

	Schere	Stein	Papier
Schere	0	-1	1
Stein	1	0	-1
Papier	-1	1	0

- Schere, Stein, Papier mit Spielern A und B
- Gemischte Strategie für A:
  - P(Schere) = p; P(Stein) = q; P(Papier) = 1 - p – q
- Ergebnis für A:
  - Wenn B immer Schere wählt: p*0 + q*1 + (1-p-q)*(-1) = q-1+p+q= -1+p+2q
  - Wenn B immer Stein wählt: p*(-1) + q*0 + (1-p-q)*1 = -p+1-p-q= 1-2p-q
  - Wenn B immer Papier wählt: p*1 + q*(-1) + (1-p-q)*0 = p-q
  - Im Schnittbereich: p=1/3; q=1/3
  - $\Rightarrow$ Ergebnis: 0
- Ergebnis für B äquivalent

Maximin Technique für Schere, Stein, Papier, Brunnen

	Schere	Stein	Papier	Brunnen
Schere	0	-1	1	-1
Stein	1	0	-1	-1
Papier	-1	1	0	1
Brunnen	1	1	-1	0

- Schere, Stein, Papier, Brunnen mit Spielern A und B
- Gemischte Strategie für A:
  - P(Schere) = p; P(Stein) = q;
  - P(Papier) = r; P(Brunnen) = 1 - p - q - r
- Ergebnis für A:
  - Wenn B immer Schere wählt: p*0 + q*1 + r*(-1) + (1-p-q-r)*1 = q-r+1-p-q-r= 1-p-2r
  - Wenn B immer Stein wählt: p*(-1) + q*0 + r*1 + (1-p-q-r)*1 = -p+r+1-p-q-r= 1-2p-q
  - Wenn B immer Papier wählt: p*1 + q*(-1) + r*0 + (1-p-q-r)*-1 = p-q-1+p+q+r= -1+2p+r
  - Wenn B immer Brunnen wählt: p*(-1) + q*(-1) + r*1 + (1-p-q-r)*0 = -p-q+r
  - Im Schnittbereich: ?????

Auflösung der Gleichungen

- $1\text{-}p\text{-}2r = 1\text{-}2p\text{-}q = \text{-}1\text{+}2p\text{+}r = \text{-}p\text{-}q\text{+}r$
- $1\text{-}p\text{-}2r = \text{-}1\text{+}2p\text{+}r \Rightarrow 2 = 3p + 3r \Rightarrow p = 2/3 - r$
- $1\text{-}p\text{-}2r = \text{-}p\text{-}q\text{+}r \Rightarrow 1 = \text{-}q\text{+}3r \Rightarrow q = 3r - 1$
- $1\text{-}p\text{-}2r = 1\text{-}2p\text{-}q \Rightarrow 2r = p\text{+}q \Rightarrow 2r = 2/3 – r + 3r - 1 \Rightarrow 1 = 2/3 \Rightarrow$ **Widerspruch!**

- **Warum??**

Begründung für Widerspruch

	Schere	Stein	Papier	Brunnen
Schere	0	-1	1	-1
Stein	1	0	-1	-1
Papier	-1	1	0	1
Brunnen	1	1	-1	0

- Wir haben die Annahme von rationalen Spielern gemacht.
- Ein rationaler Spieler würde nie Stein wählen, sondern stattdessen Brunnen.
- Daher entfällt die Annahme „wenn B immer Stein wählt“ im MaxiMin-Algorithmus
- Wenn man diese Annahme weglässt und die Wahrscheinlichkeit P(Stein) = q = 0 setzt, liefert der Algorithmus ein Ergebnis.
- Allerdings ist er dann äquivalent zu Schere, Stein, Papier, wobei Stein durch Brunnen ersetzt wurde.
- Psychologische Spielstrategien sind davon unberührt.

Maximin Technique für Schere, Stein, Papier, Brunnen

	Schere	Stein	Papier	Brunnen
Schere	0	-1	1	0
Stein	1	0	-1	-1
Papier	-1	1	0	1
Brunnen	0	1	-1	0

- Schere, Stein, Papier, Brunnen mit Spielern A und B
- Gemischte Strategie für A:
  - P(Schere) = p; P(Stein) = q;
  - P(Papier) = r; P(Brunnen) = s
- Ergebnis für A:
  - Wenn B immer Schere wählt: p*0 + q*1 + r*(-1) + s*0 = q-r
  - Wenn B immer Stein wählt: p*(-1) + q*0 + r*1 + s*1 = -p+r+s
  - Wenn B immer Papier wählt: p*1 + q*(-1) + r*0 + s*(-1) = p-q-s
  - Wenn B immer Brunnen wählt: p*0 + q*(-1) + r*1 + s*0 = -q+r
  - Im Schnittbereich: ?????

Auflösung der Gleichungen

- q-r = -p+r+s = p-q-s= -q+r
- q-r = -q+r $\Rightarrow$ q = r

- 0 = -p+q+s = p-q-s
- s=p-q
- 1 = p+q+q+s=p+q+q+p-q = 2p+q

- Zusätzliches Constraint: {p,q,r,s} zwischen 0 und 1
- **Viele Lösungen mit p zwischen 1/2 und 1/3 : z.B.**
  - **p=s=0,5 und q=r=0 oder**
  - **p=0,4 und s=q=r=0,2 oder**
  - **p=0,35 und q=r=0,3 und s=0,05 oder**
  - **p=1/3 und q=r=1/3 und s=0**

	Schere	Stein	Papier	Brunnen
Schere	0	-1	1	0
Stein	1	0	-1	-1
Papier	-1	1	0	1
Brunnen	0	1	-1	0

Korrektheitstest in Excel

- Multiplikation der Payoff-Matrix mit der Wahrscheinlichkeitsmatrix der Strategie
- Für jede (reine) Gegenstrategie (d.h. jede Zeile) ist das Ergebnis 0

K3 =B3*G3+C3*H3+D3*I3+E3*J3

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1		p	q	r	s		p=s=0,5 und q=r=0				Ergebnis		p=0,4 und s=q=r=0,2				Ergebnis
2		Schere	Stein	Papier	Brunnen		p	q	r	s			p	q	r	s	
3	Schere	0	-1	1	0		0,5	0	0	0,5	0		0,4	0,2	0,2	0,2	0
4	Stein	1	0	-1	-1		0,5	0	0	0,5	0		0,4	0,2	0,2	0,2	0
5	Papier	-1	1	0	1		0,5	0	0	0,5	0		0,4	0,2	0,2	0,2	0
6	Brunnen	0	1	-1	0		0,5	0	0	0,5	0		0,4	0,2	0,2	0,2	0
7																	
8		Schere	Stein	Papier	Brunnen		p=0,35; q=r=0,3; s=0,05				Ergebnis		p=q=r=1/3; s=0				Ergebnis
9	Schere	0	-1	1	0		0,35	0,3	0,3	0,05	0		0,33	0,33	0,33	0,00	0
10	Stein	1	0	-1	-1		0,35	0,3	0,3	0,05	0		0,33	0,33	0,33	0,00	0
11	Papier	-1	1	0	1		0,35	0,3	0,3	0,05	0		0,33	0,33	0,33	0,00	0
12	Brunnen	0	1	-1	0		0,35	0,3	0,3	0,05	0		0,33	0,33	0,33	0,00	0

Lernen von Entscheidungsbäumen

- Sie haben für 14 Fälle aufgezeichnet, ob Sie abhängig der diskreten Werte für Temperatur und Regenprognose einen Ausflug machen (s. rechts).
- Lernen Sie einen Entscheidungsbaum und klassifizieren Sie die drei neuen Fälle (unten rechts).

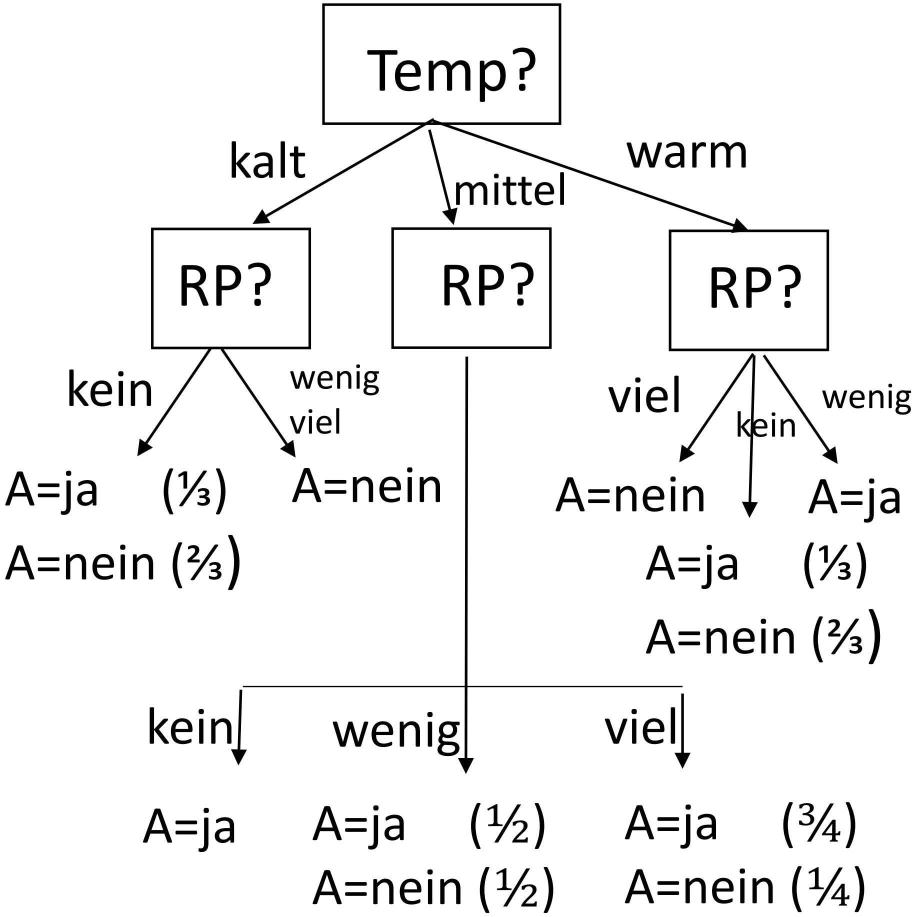
	Tem	Temperatur			Regenprognose			Ausflug	
	Cels	kalt	mittel	warm	kein	wenig	viel	ja	nein
Fall1	10		x			x		x	
Fall2	20			x			x		x
Fall3	15		x		x			x	
Fall4	5	x			x			x	
Fall5	3	x				x			x
Fall6	8	x				x			x
Fall7	25			x		x		x	
Fall8	12		x		x			x	
Fall9	22			x			x		x
Fall10	11		x			x			x
Fall11	0	x			x				x
Fall12	1	x					x		x
Fall13	6	x				x			x
Fall14	8	x			x				x
Neu1	12		x			x		??	??
Neu2	8	x			x			??	??
Neu3	13		x				x	??	??

- Entropie (Dataset) = H(Dataset) = **B (p/(p+n))** = B (5/(5+9)) = B(5/14)
- Entropie (Boolsche Variable, die mit P(q) wahr ist) = **B(q) = -(q $\log_2$ q + (1-q) $\log_2$ (1-q))**
- B(5/14) = -(5/14 $\log_2$ 5/14 + 9/14 $\log_2$ 9/14) = -(0,36*(-1,5) + 0,64*(-0,64))=0,94
- **Gain(A) = B(p/(p+n)) - Remainder (A) = B(p/(p+n)) -** $\sum_{k=1}^{d} \frac{p_k+n_k}{p+n} B\left(\frac{p_k}{p_k+n_k}\right)$
- Gain (RP) = 0,94 - [5/14 * B(3/5) + 6/14 * B(2/6) + 3/14 * B(0/3)] =
  = 0,94 - [0,36 * 0,97 + 0,43 * 0,92 + 0,21 * 0] =
  = 0,94 - [0,35 + 0,4 + 0] = **0,94-0,75=0,19**
- Gain (Temp) = 0,94 - [7/14 * B(1/7) + 4/14 * B(3/4) + 3/14 * B(1/3)] =
  = 0,94 - [0,5 * 0,59 + 0,29 * 0,81 + 0,21 * 0,92] =
  = 0,94 - [0,30 + 0,23 + 0,19] = **0,94-0,72=0,22**

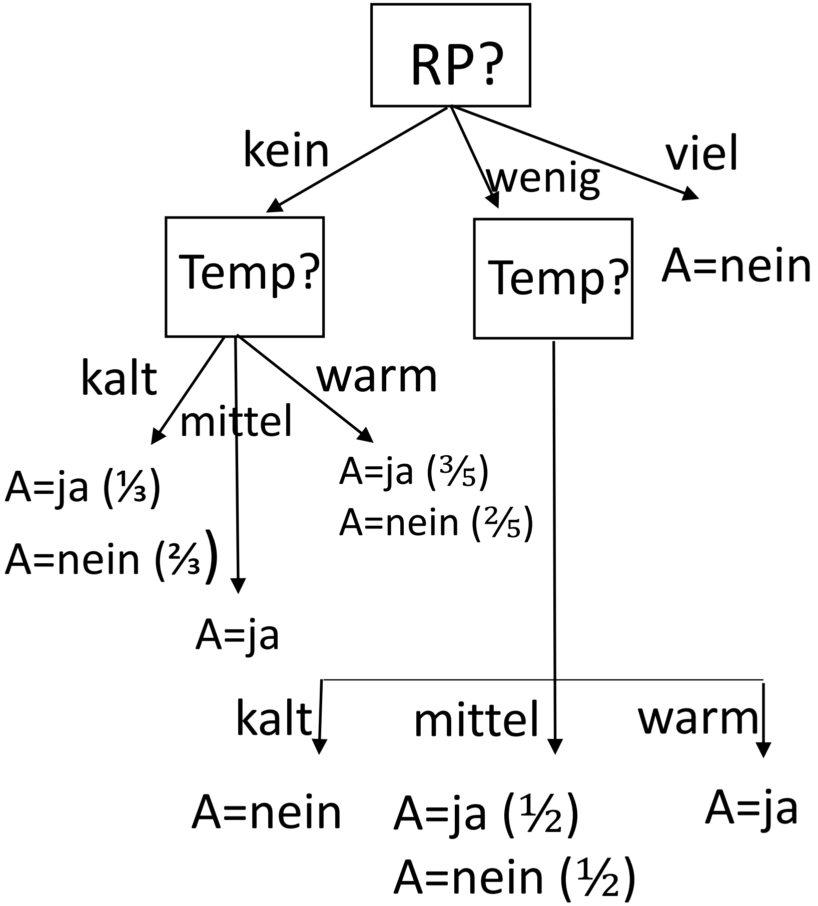
$\Rightarrow$ Gain (Temperatur) leicht besser als Gain (Regenprognose)

	Tem	Temperatur			Regenprognose			Ausflug	
	Cels	kalt	mittel	warm	kein	wenig	viel	ja	nein
Fall1	10		x			x		x	
Fall2	20			x			x		x
Fall3	15		x		x			x	
Fall4	5	x			x			x	
Fall5	3	x				x			x
Fall6	8	x				x			x
Fall7	25			x		x		x	
Fall8	12		x		x			x	
Fall9	22			x			x		x
Fall10	11		x			x			x
Fall11	0	x			x				x
Fall12	1	x					x		x
Fall13	6	x				x			x
Fall14	8	x			x				x
Neu1	12		x			x		??	??
Neu2	8	x			x			??	??
Neu3	13		x				x	??	??

Entscheidungsbaum



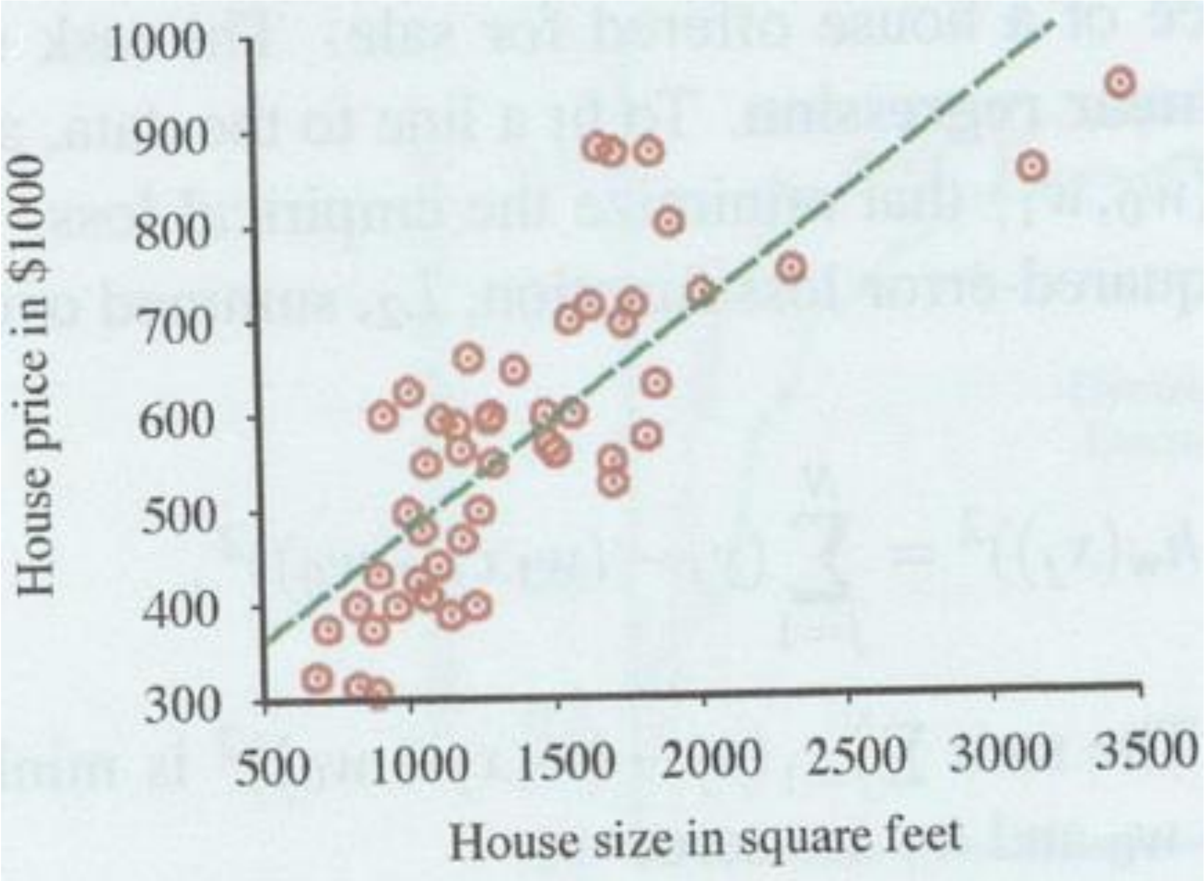
	Tem	Temperatur			Regenprognose			Ausflug	
	Cels	kalt	mittel	warm	kein	wenig	viel	ja	nein
Fall1	10		x			x		x	
Fall2	20			x			x		x
Fall3	15		x		x			x	
Fall4	5	x			x			x	
Fall5	3	x				x			x
Fall6	8	x				x			x
Fall7	25			x		x		x	
Fall8	12		x		x			x	
Fall9	22			x			x		x
Fall10	11		x			x			x
Fall11	0	x			x				x
Fall12	1	x					x		x
Fall13	6	x				x			x
Fall14	8	x			x				x
Neu1	12		x			x		??	??
Neu2	8	x			x			??	??
Neu3	13		x				x	??	??



- **Neu1: Ja: 50%; Nein: 50%**
- **Neu2: Ja: 33%, Nein: 67%**
- **Neu3:**
  - **Links: Ja: 75%, Nein: 25%**
  - **Rechts: Nein: 100%**

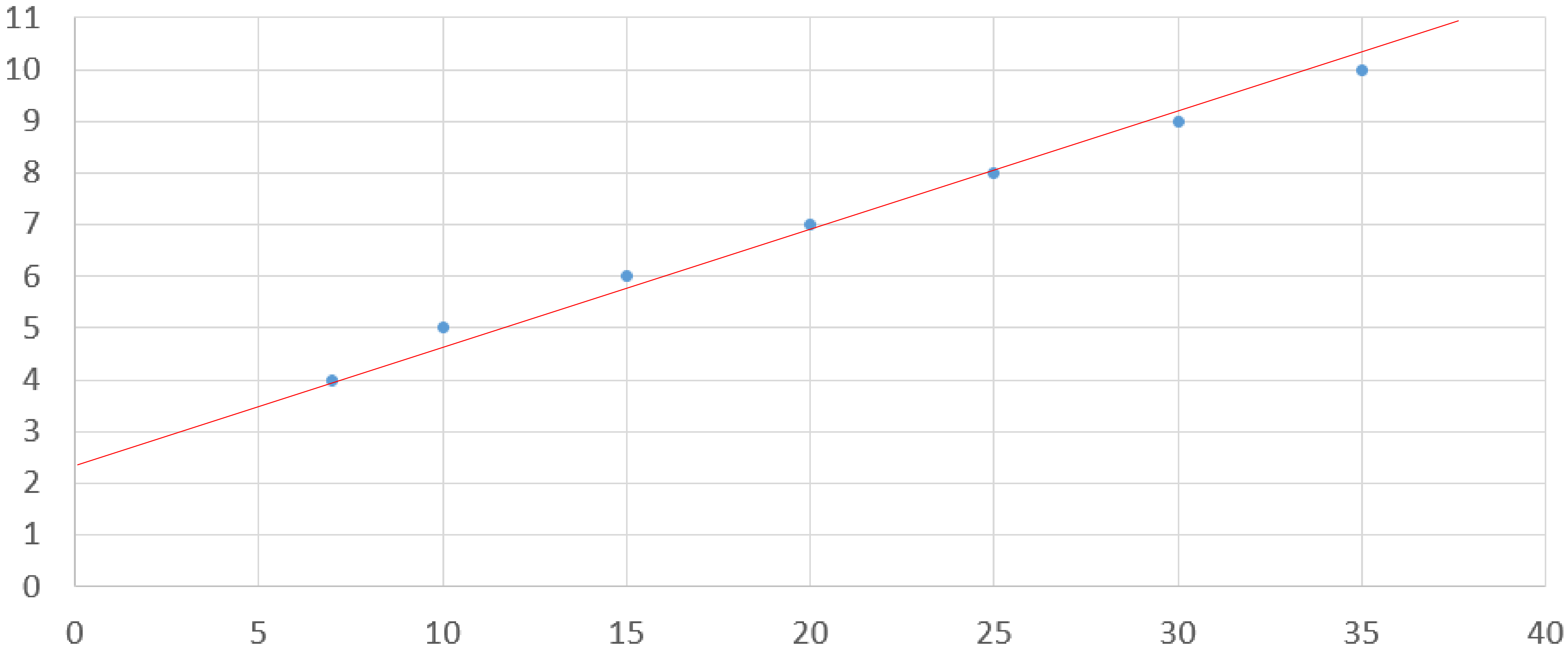
Linear Regression

- Task
  - Given: Data sets with numerical variables (s. figure)
  - Sought: Formula for predicition of one variable given the others
- Univariate linear regression (straight line)
  - Computed by equations with exact solution
    - Lösung: w_1= 0,232; w_0= 246
  - **Computed by gradient descent**



Linear Regression

- **Beispieldaten (7 Punkte):**
- (4,7),(5,10),(6,15),(7,20),(8,25), (9,30), (10,35)



Gradient Descent Algorithm

$w \leftarrow$ any point in the parameter space

While not converged

for each w_i in w do $\quad w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} Loss(\mathbf{w})$ (α: Learning rate (step size))

Computing the partial derivatives for w_0 and w_1 :

Chain rule $\partial\, g(f(x)/\partial x = g'(f(x)\partial f(x)/\partial x$

$$\frac{\partial}{\partial w_i} Loss(\mathbf{w}) = \frac{\partial}{\partial w_i}(y - h_{\mathbf{w}}(x))^2 = 2(y - h_{\mathbf{w}}(x)) \times \frac{\partial}{\partial w_i}(y - h_{\mathbf{w}}(x))$$
$$= 2(y - h_{\mathbf{w}}(x)) \times \frac{\partial}{\partial w_i}(y - (w_1 x + w_0)) .$$

Applying this to w_0 and w_1 yields: $\frac{\partial}{\partial w_0} Loss(\mathbf{w}) = -2(y - h_{\mathbf{w}}(x))$; $\quad \frac{\partial}{\partial w_1} Loss(\mathbf{w}) = -2(y - h_{\mathbf{w}}(x)) \times x$

with update and learning rate α : $w_0 \leftarrow w_0 + \alpha\,(y - h_w(x))$; $\quad w_1 \leftarrow w_1 + \alpha\,(y - h_w(x))\, x$

Batch Gradient Descent

- An update is also possible for N training examples instead of just one training example:
  - Minimize the sum of individual losses:

$$w_0 \leftarrow w_0 + \alpha \sum_j \left(y_j - h_{\mathbf{w}}\left(x_j\right)\right); \qquad w_1 \leftarrow w_1 + \alpha \sum_j \left(y_j - h_{\mathbf{w}}\left(x_j\right)\right) \times x_j.$$

  - „Batch gradient descent learning rule“ for deterministic gradient descent
    - Rather slow
- Faster variant: **Stochastic gradient descent (SGD)**
  - Select randomly a small number of training examples „minibatch“
  - Update weights w_0 and w_1 for the sum analogous to formula for one training example
  - Repeat procedure till convergence
    - Problem: Convergence of minibatch SGD not guranteed, may oscillate
      - Improvement: Schedule of decreasing learning rate α

Programmierung von LR-GA in Python

- Nächste Folie: Daten und Datengenerierung
- Übernächste Folie: Parameter und Hilfsfunktionen
  - Parameter:
    - w0 und w1: Zufällige Startwerte für w0 und w1,
    - a: Schrittweite alpha (α),
    - iter: Anzahl der Iterationen
    - batch: Größe des Batches (muss kleiner sein als die Anzahl der Punkte)
  - Für Batch-Gradient werden n Zufallspunkte aus der Menge aller Punkte ausgewählt
  - Die Fehlersummer für den Gradienten wird für die Gewichte w0 und w1 berechnet
- Dritte Folie: Drei Varianten des Gradientenabstiegs:
  - Gradient wird aus Summe aller Punkte berechnet
  - Gradient wird aus jedem Punkt einzeln berechnet
  - Gradient wird aus einer zufälligen Teilmenge der Punkte („batch“) berechnet

Lineare Regression mit Gradientenabstieg (LR-GA): Daten

```
import random
import time

def generierePunkte (w0, w1, anzahl, rauschen):
    punkte = []
    for i in  range (0, anzahl):
        epsilon = random.uniform(-1*rauschen,rauschen)        # Zufallszahl im Intervall
        y = random.random()                                   # Zufallszahl zwischen 0 und 1
        punkte.append ([y, w0+y*w1+epsilon])
    return punkte

#punkte = [(7, 4),(10, 5), (15, 6),(20, 7),(25, 8), (30, 9), (35, 10)]
#punkte = [(700, 400),(1000,500), (1500,600), (2000,700),(2500,800), (3000, 900), (3500,1000)]
#punkte = [(0.7, 0.4),(1, 0.5), (1.5, 0.6),(2, 0.7),(2.5, 0.8), (3, 0.9), (3.5, 1)]
punkte = generierePunkte (0.25, 0.23, 100, 0.01)
```

LR-GA: Parameter und Hilfsfunktionen


```
w0=0
w1=1
a=0.01
iter = 1000
batch = 10

def zufallspunkte (anzahl, punkte):
    neupunkte = []
    for i in range (0, anzahl):
        neupunkte.append
(random.choice(punkte))
    return neupunkte

    # a = Lernrate alpha
    # batch: Anzahl Punkte für Batch-Gradient
```

```
def summefehlerW0 (w0, w1, punkte):
    summe = 0
    for p in punkte:
        summe += p[1] - (w0 + w1*p[0])
    return summe/len(punkte)

def summefehlerW1 (w0, w1, punkte):
    summe = 0
    for p in punkte:
        summe += (p[1] - (w0 + w1*p[0]))*p[0]
    return summe/len(punkte)

# Punkt: [x, y]
# mit p[0] = x  und p(1] = y
# Normierung der Summenfehler(!)
```

LG-GA: 3 Varianten für Gradientenabstieg


```
def gradientenabstieg (w0,w1,punkte):
# Gradient über Summe der Punkte
    for i in range(0, iter):
        summeW0 = summefehlerW0 (w0,w1, punkte)
        summeW1 = summefehlerW1 (w0,w1, punkte)
        for p in punkte:
            w0 += a*summeW0
            w1 += a*summeW1
    return [w0, w1]

def gradientenabstieg1 (w0,w1,punkte):
# Gradient über einzelne Punkte
    for i in range(0, iter):
        for p in punkte:
            w0 += a*(p[1] - (w0 + w1*p[0]))
            w1 += a*(p[1] - (w0 + w1*p[0]))*p[0]
    return [w0, w1]
```

```
def gradientenabstieg2 (w0,w1,punkte):
# Gradient über batch von Punkten (zufällige Auswahl)
    for i in range(0, iter):
        neupunkte = zufallspunkte (batch, punkte)
        summeW0 = summefehlerW0 (w0,w1, neupunkte)
        summeW1 = summefehlerW1 (w0,w1, neupunkte)
        for p in neupunkte:
            w0 += a*summeW0
            w1 += a*summeW1
    return [w0, w1]
```

Ergebnisse für die 3 LG-RA Varianten

Ergebnis: w_0=0, w_1=1, α = 0.01, iter=1 000, batch=10, **100** generierte Punkte mit w_0=0,25, w_1=0,23, ε = 0,01
G0: W_0= 0,251; w_1= 0,230; Dauer: 0,053 Sekunden
G1: W_0= 0,251; w_1= 0,230; Dauer: 0,041 Sekunden
G2: W_0= 0,250; w_1= 0,231; Dauer: 0,016 Sekunden

Ergebnis: iter=100, ansonsten wie oben
G0: W_0= 0,248; w_1= 0,233; Dauer: 0,012 Sekunden
G1: W_0= 0,248; w_1= 0,234; Dauer: 0,010 Sekunden
G2: W_0= 0,026; w_1= 0,612; Dauer: 0,007 Sekunden $\rightarrow$ viel zu wenig Iterationen

Ergebnis: w_0=0, w_1=1, α = 0.01, iter=1 000, batch=3, **10** generierte Punkte mit w_0=0,25, w_1=0,23, ε = 0,01
G0: W_0= 0,252; w_1= 0,223; Dauer: 0,017 Sekunden
G1: W_0= 0,252; w_1= 0,223; Dauer: 0,012 Sekunden
G2: W_0= 0,223; w_1= 0,284; Dauer: 0,010 Sekunden $\rightarrow$ zu wenig Iterationen

Ergebnis: iter=10000, ansonsten wie oben
G0: W_0= 0,253; w_1= 0,222; Dauer: 0,116 Sekunden
G1: W_0= 0,253; w_1= 0,222; Dauer: 0,045 Sekunden
G1: W_0= 0,253; w_1= 0,222; Dauer: 0,045 Sekunden

Ergebnisse für die 3 LG-RA Varianten

Ergebnis: w_0=0, w_1=1, α = 0.01, iter=1 000, batch=3, **7** gegebene Punkte zwischen 0 und 1; ε = 0,01
G0: W_0= 0,278; w_1= 0,208; Dauer: 0,015 Sekunden
G1: W_0= 0,278; w_1= 0,208; Dauer: 0,011 Sekunden
G2: W_0= 0,274; w_1= 0,211; Dauer: 0,009 Sekunden

Ergebnis: iter=10 000, ansonsten wie oben
G0: W_0= 0,278; w_1= 0,208; Dauer: 0,045 Sekunden
G1: W_0= 0,278; w_1= 0,207; Dauer: 0,029 Sekunden
G2: W_0= 0,278; w_1= 0,208; Dauer: 0,046 Sekunden

Ergebnis: iter=10 000, α = 0.001, ansonsten wie oben
G0: W_0= 0,2775; w_1= 0,2083; Dauer: 0,053 Sekunden
G1: W_0= 0,2776; w_1= 0,2082; Dauer: 0,030 Sekunden
G2: W_0= 0,2735; w_1= 0,2100; Dauer: 0,044 Sekunden

Ergebnis: iter=100 000, ansonsten wie oben
G0: W_0= 0,27753; w_1= 0,20826; Dauer: 0,332 Sekunden
G1: W_0= 0,27759; w_1= 0,20821; Dauer: 0,240 Sekunden
G2: W_0= 0,27751; w_1= 0,20829; Dauer: 0,410 Sekunden

Erweiterung für multivariate Regression (1)

```
def mgenerierePunkte (w0, w1, w2, anzahl, rauschen):
   mpunkte = []
   for i in  range (0, anzahl):
      epsilon = random.uniform(-1*rauschen,rauschen)
      x = random.random()
      y = random.random()
      mpunkte.append ([x,y, w0+x*w1+y*w2+epsilon])
   return mpunkte

mpunkte = mgenerierePunkte (0.2, 0.2, 0.2, 100, 0.01)
```

```
def mgradientenabstieg1 (w0,w1,w2, punkte):
# Gradient über einzelne Punkte
   for i in range(0, iter):
      for p in punkte:
         soll = p[2]
         ist = w0 + w1*p[0] + w2*p[1]
         w0 += a*(soll - ist)
         w1 += a*(soll - ist)*p[0]
         w2 += a*(soll - ist)*p[1]
   return [w0, w1, w2]
```

```
#Parameter:
w0=0
w1=1
w2=1
a=0.01
iter = 1000
```

Aufruf:

```
print ('MGradientenabstieg über Einzelfehler:')
time01 = time.time()
print ('W0, W1 und W2: ',mgradientenabstieg1 (w0, w1, w2, mpunkte))
time02 = time.time()
print ('Dauer: ',time02-time01, ' Sekunden.')
```

Ergebnis:

```
MGradientenabstieg über Einzelfehler:
W0, W1 und W2:  [0.19692471499765626,
0.20318451225762207, 0.20180243438219717]
Dauer:  0.1080164909362793  Sekunden.
```

Erweiterung von multivariate Regression (2)


```
def msummefehlerW0 (w0, w1, w2, punkte):
   summe = 0
   for p in punkte:
      summe += p[2] - (w0 + w1*p[0] + w2*p[1])
   return summe/len(punkte)

def msummefehlerW1 (w0, w1, w2, punkte):
   summe = 0
   for p in punkte:
      summe += (p[2] - (w0 + w1*p[0] + w2*p[1]))*p[0]
   return summe/len(punkte)

def msummefehlerW2 (w0, w1, w2, punkte):
   summe = 0
   for p in punkte:
      summe += (p[2] - (w0 + w1*p[0] + w2*p[1]))*p[1]
   return summe/len(punkte)
```

```
def mgradientenabstieg (w0,w1,w2, punkte):
# Gradient über Summe der Punkte
   for i in range(0, iter):
      summeW0 = msummefehlerW0 (w0,w1,w2, punkte)
      summeW1 = msummefehlerW1 (w0,w1,w2, punkte)
      summeW2 = msummefehlerW2 (w0,w1,w2, punkte)
      for p in punkte:
         w0 += a*summeW0
         w1 += a*summeW1
         w2 += a*summeW2
   return [w0, w1, w2]
```

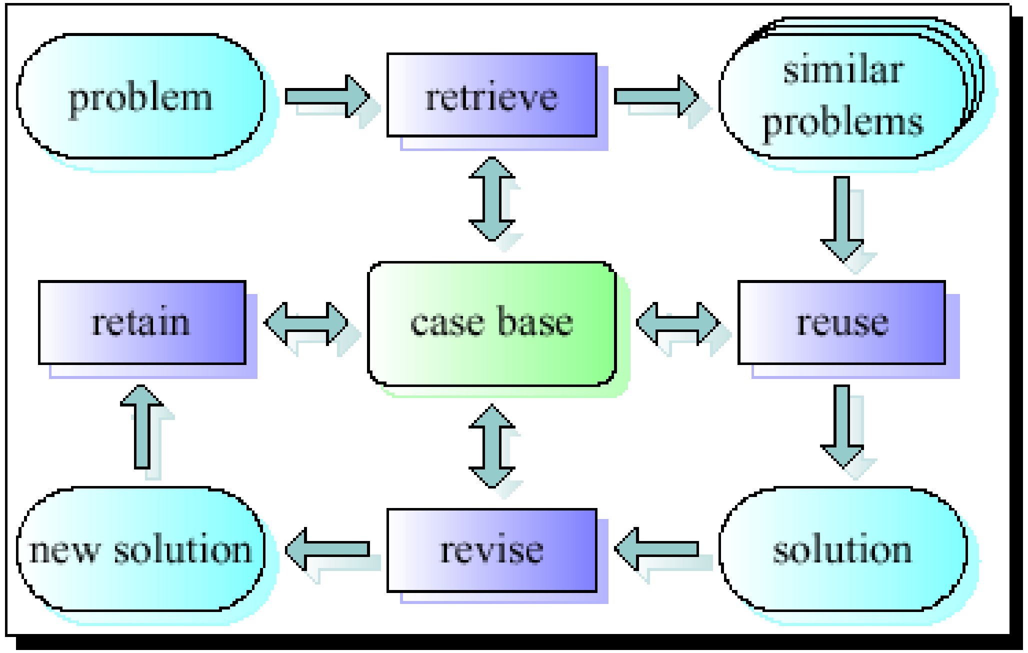
```
def mgradientenabstieg2 (w0,w1,w2, punkte):
# Gradient über batch von Punkten (zufällige Auswahl)
#   for i in range(0, int(iter*len(punkte)/batch)):
   for i in range(0, iter):
      neupunkte = zufallspunkte (batch, punkte)
      summeW0 = msummefehlerW0 (w0,w1,w2, punkte)
      summeW1 = msummefehlerW1 (w0,w1,w2, punkte)
      summeW2 = msummefehlerW2 (w0,w1,w2, punkte)
      for p in neupunkte:
         w0 += a*summeW0
         w1 += a*summeW1
         w2 += a*summeW2
   return [w0, w1, w2]
```

Ergebnisse für die 3 LG-RA Varianten

Ergebnis: w_0=0, w_1=w_2=1, α = 0.01, iter=1 000, batch=10, **100** generierte Punkte mit w_0=w_1=w_2=0,2, ε = 0,01
G0: W_0= 0,1995; w_1= 0,2007; w_2= 0,1999; Dauer: 0,122 Sekunden
G1: W_0= 0,1996; w_1= 0,2007; w_2= 0,1999; Dauer: 0,116 Sekunden
G2: W_0= 0,1970; w_1= 0,2027; w_2= 0,2026; Dauer: 0,116 Sekunden

Ergebnis: iter=100, ansonsten wie oben
G0: W_0= 0,1963; w_1= 0,2050; w_2= 0,2019; Dauer: 0,016 Sekunden
G1: W_0= 0,1961; w_1= 0,2054; w_2= 0,2019; Dauer: 0,016 Sekunden
G2: W_0= -0,183; w_1= 0,553; w_2= 0,566; Dauer: 0,016 Sekunden $\rightarrow$ viel zu wenig Iterationen

Fallbasiertes Schließen

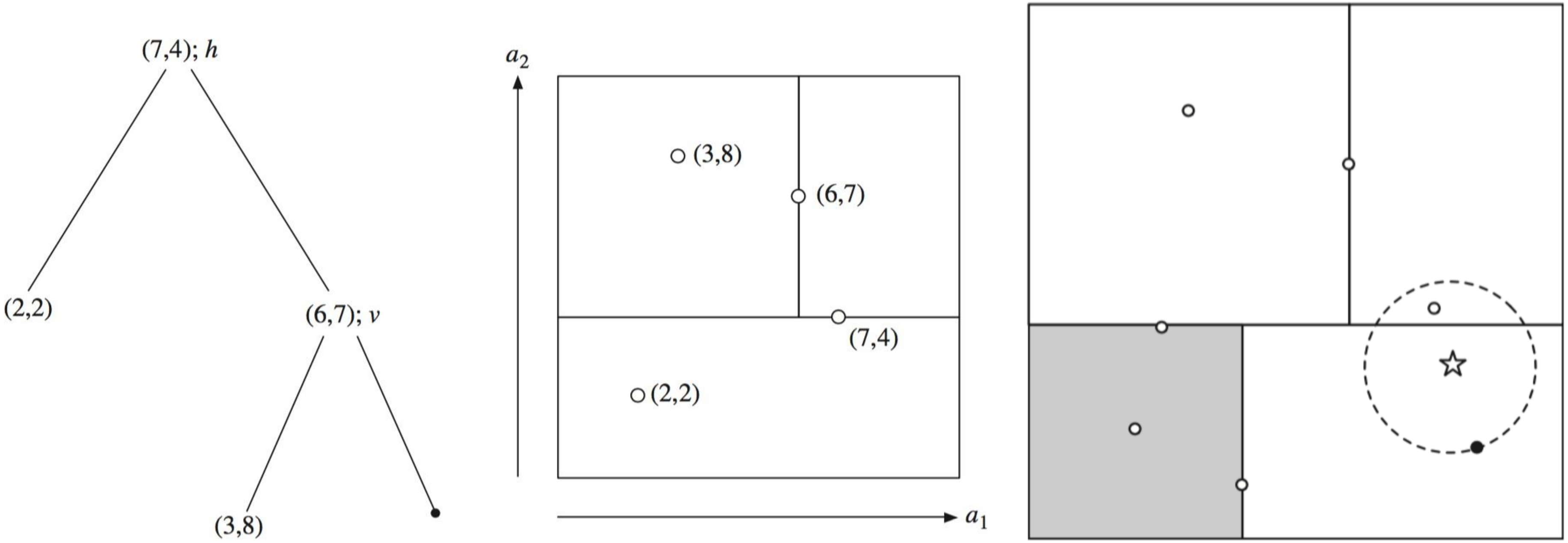


	Neuer Fall	Alter Fall 1	Alter Fall 2	Alter Fall 3
Autotyp	Marke A	Marke B	Marke A	Marke C
Km-Stand	100.000	110.000	95.000	105000
Benzinverbrauch	7	8	8	13
Motor ruckelt	ja	ja	nein	ja
Springt nicht an	meistens	manchmal	immer	meistens
Geräusche	Klopfen	Klingeln und Klopfen	keine	unbekannt
Lösung	?	Zündkerzen verbraucht	Batterie leer	C-Turbo defekt

k-d-Trees zur Effizienzsteigerung des Fall-Retrievals

- Laufzeiteffizienz ohne Optimierung: O(k*n)   [k = Anzahl Attribute n = Anzahl Fälle]

- Verbesserungsidee:
  - ○ **k-d trees** (k-dimensional tree): Vorberechnung eines Baumes, der die Fallmenge jeweils anhand einer von k Dimensionen (k= Anzahl Attribute) aufteilt.
    - Bei der Suche eines ähnlichen Falles mit max-Differenz von ε können dann größere Bereiche des Baumes (und deren Fälle) abgeschnitten werden.
    - Beispiel (s.u.): 4 Fälle mit zwei numerischen Attributen

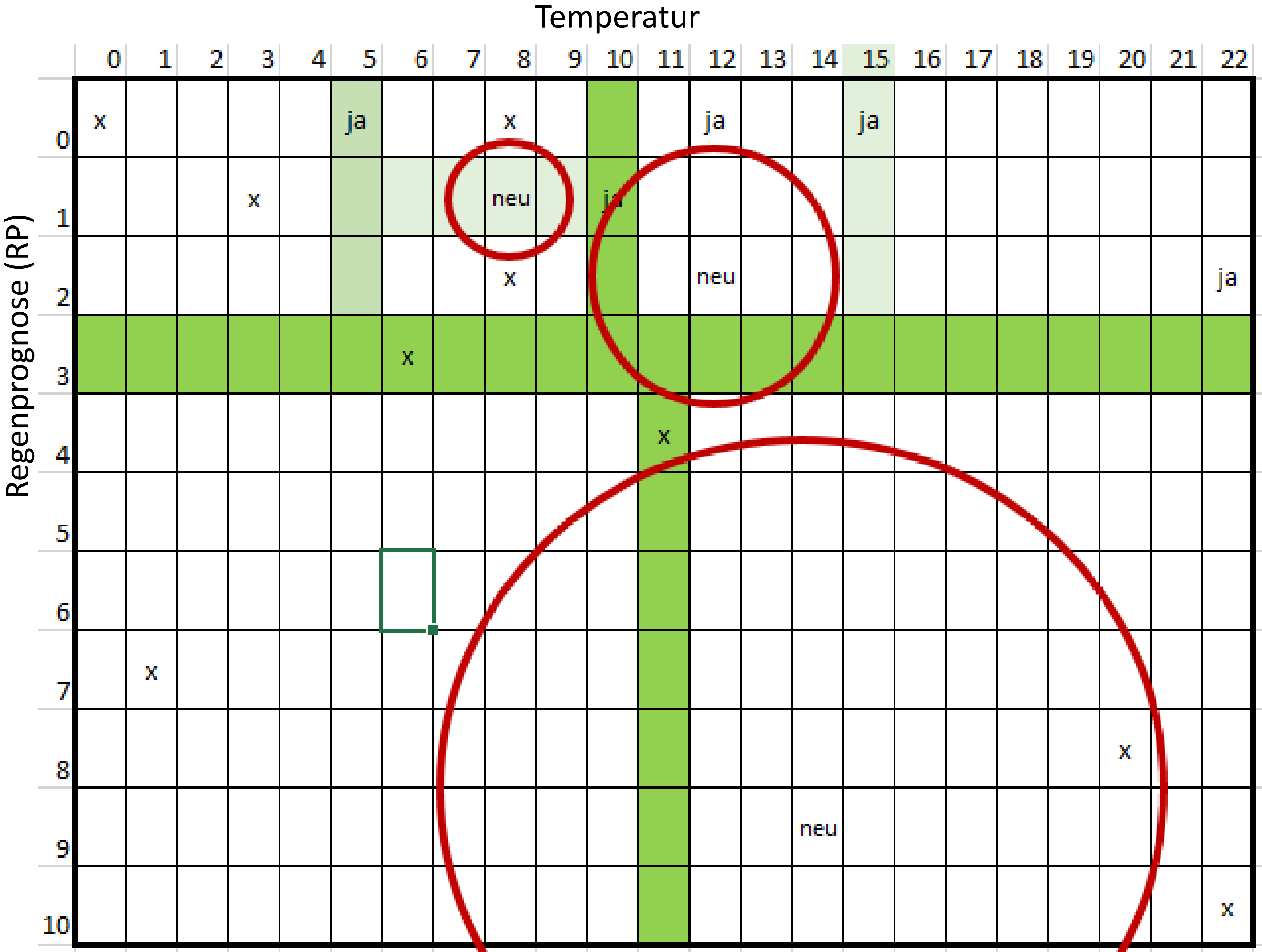


Beschreibung k-d-Trees

- **Struktur:** Suchbaum, der rekursiv anhand einer Dimension aufgespalten ist, bis sich in einem Blatt nur ein oder kein Fall befindet.
- **Generierung:** Berechne die Dimension mit der größten Streuung (Standardabweichung). Berechne den Fall, der bezüglich dieser Dimension zum Mittelwert am nächsten liegt. Füge bei diesem Wert eine Trenn-Hyperebene (Gerade im 2-dimensionalen Fall) ein. Der Fall selbst liegt auf der Trenn-Hyperebene bzw. Geraden. Wiederhole solange, bis in jedem Blatt sich nur noch ein oder kein Fall befindet.
- **Inkrementelles Update:** Füge einen neuen Fall in den k-d-tree an der entsprechenden Stelle ein (falls Bereich leer) oder füge eine neue Trenn-Hyperebene hinzu (falls Bereich besetzt war). Re-Balanciere k-d-tree gelegentlich.
- **Suche:** Für neuen Fall wird im k-d-tree durch Traversieren des Baumes der passende Bereich gefunden und dort der dazugehörige Fall, der als Kandidat für den ähnlichsten Fall gilt. Dies muss jedoch überprüft werden, indem bei jeder Entscheidung im Baum (rekursiv bis zur Wurzel des Baumes) geschaut wird, ob in dem Nachbarbereich theoretisch ähnlichere Fälle liegen können (Schneidet der Kreis um den neuen Fall mit Radius zu dem bisher ähnlichsten Fall eine Bereichsgrenze, d.h. Hyperebene bzw. Gerade?). Falls ja, muss in diesem Bereich ebenfalls gesucht werden, falls nein, kann der Bereich und alle Unterbereiche ausgeschlossen werden.

Anwendung k-d-Trees für Fallvergleich (KNN; k=2)

	Tem	RP	Ausflug	
	Cels	mm	ja	nein
Fall1	10	1	x	
Fall2	20	8		x
Fall3	15	0	x	
Fall4	5	0	x	
Fall5	3	1		x
Fall6	8	2		x
Fall7	22	2	x	
Fall8	12	0	x	
Fall9	22	10		x
Fall10	11	4		x
Fall11	0	0		x
Fall12	1	7		x
Fall13	6	3		x
Fall14	8	0		x
Erwartungswert (MW)	10	2,714		
Standardabweichung	7,3	3,338		
Variationskoeffizient	0,7	1,23		
Neu1	12	2	??	??
Neu2	8	1	??	??
Neu3	14	9	??	??



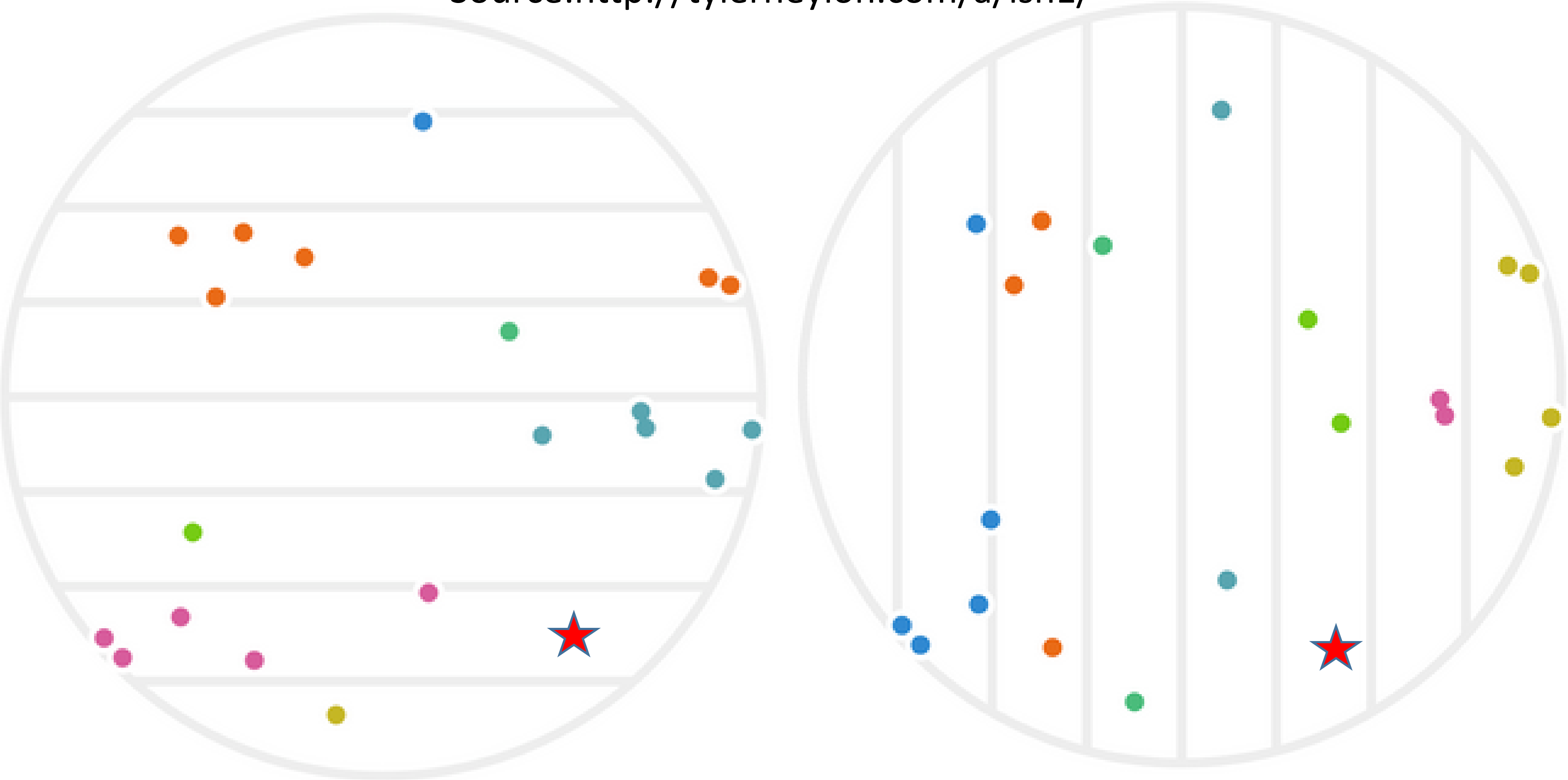
Locality-sensitive Hashing for KNN

- **Hash codes** maps data or arbritary size into fixed size values (bins) allowing a fast look-up
  - Different data should be mapped to different bins
- **Locality-sensitve hashing (LSH)** tries to map similar data into the same bin

Locality-sensitive Hashing: Simple Example

Source:http://tylerneylon.com/a/lsh1/

- Two simple projections on two-dimensional data
- Equal colors: Examples are mapped in the same bin of the projection (same hash code)
- New Case ★ requires examination of violet (left) and green (right) cases

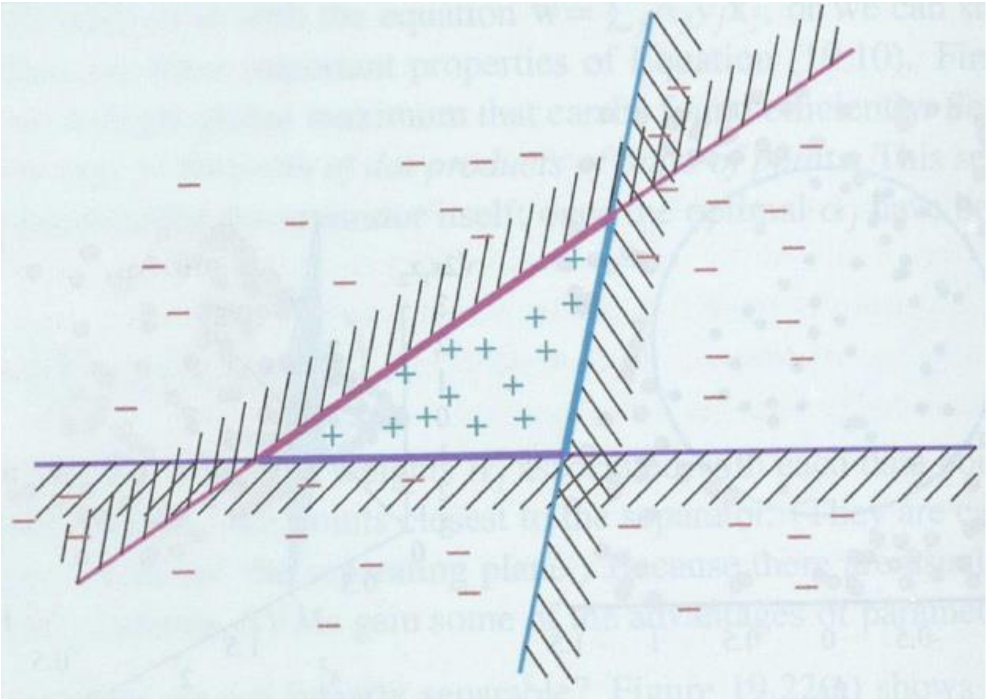


- For a large real world problems finding nearest neighbors in a dataset for 13 million images with 512 dimensions, locality-sensitive hashing had to examine only a few thousand images

Ensemble Learning

- Idea: Select a collection or ensemble of hypotheses $h_1, h_2, \ldots, h_n$ and combine their predictions by averaging, voting or even by another level of machine learning
  - Individual hypotheses: **Base models**
  - Their combination: **Ensemble model**

- Advantages
  - **Reduce bias:** An ensemble might be more expressive than the base models, e.g. the ensemble of three linear classifiers can represent a nonlinear, triangular region by voting
  - **Reduce variance:** If the errors of different base classifiers are independant, the ensemble has a much smaller error for statistical reasons
  - **Broadly applicable!**



Techniques for Creating Ensembles

- **Bagging:** Generate different base models by training a learning algorithm with different data
  - by different partitions in training, validation and test data
  - by using different data sets
  - by different combinations of pretraining and training
- **Stacking:** Use multiple base models from different model classes trained on the same data
  - e.g. decision trees, Bayes and KNN for same data
  - e.g. different types of Neural Nets
- **Random forests:** Generate different decision trees by varying attribute & split point value choices
- **Boosting:** Generate different base models with the same data by increasing the weight of misclassified examples (equivalent to add identical misclassified examples) and decreasing the weight of correctly classified examples
- ***Mixing:*** Combine different decision makers (independantly developed)

Chapter 20: Learning Probabilistic Models

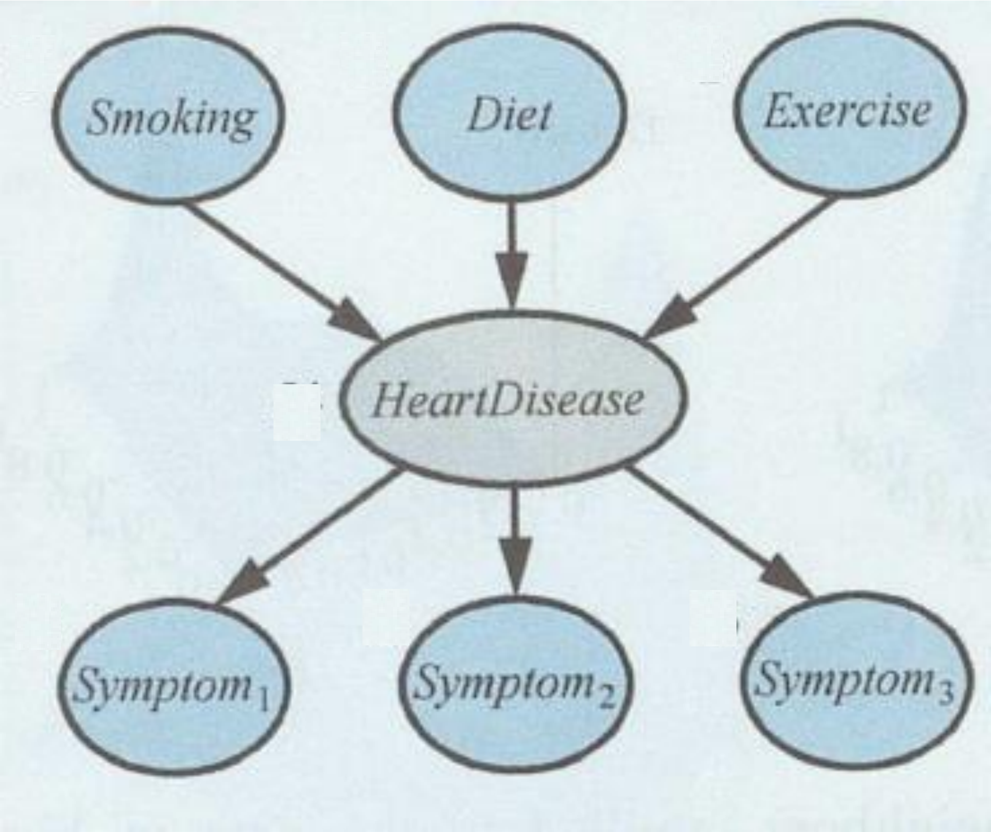
Expectation-Maximization (EM) Algorithm

Goal: Lern parameters for hidden (unobserved) variables

Idea: Guess parameters of model, generate data based on given variables and model and adapt model paramters with generated data; repeat until model is stable

Example: HeartDisease (HD) is not observable, only the 6 blue variables are observable

- Given: n cases with the 6 observable variables
- Guess probability tables for HD, S1, S2, S3
- Add to all cases values the HD variable inferred from probability tables and given variables
- Compute probability tables for the Bayes Net
- Repeat until the values of the probability tables do not change very much



EM-Algorithm

EM-Algorithms iterates between two steps till convergence:

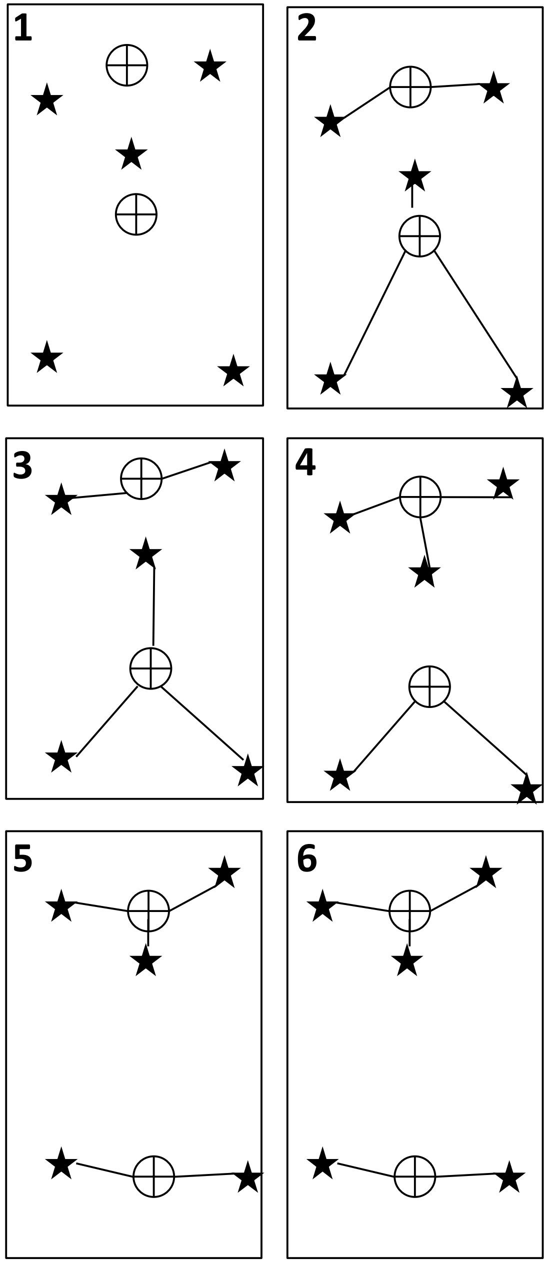
- **Initialisation:** Start with a model, i.e. a specification of all parameters (random or guessed)
- **E-Step (Expectation):** From model and given data, hypothetical data for the hidden variables are computed
- **M-Step (Maximation):** From hypothetical values of hidden variables and given data, new parameters of the model are computed
- **Termination:** Iterate E- and M-step until no relevant changes of the model occur

EM converges to a local maximum, which is often but not always good

- Depends a.o. from the inital parameter guess in the initialization.

Standard Example for EM: Clustering

- **Clustering:**
  - Given: A set of points in an n-dimensional space
  - Sought: A set of c clusters (represented by the cluster centroids)
- **EM-Clustering-Algorithm:**
  1. **Initialization:** Choose c cluster centroids randomly (1)
  2. **Expectation**: Assign each point to the next cluster centroid (2, 4)
  3. **Maximation**: Recompute centroids based on the assigned points (3, 5)
  4. **Terminination:** Repeat E- and M-Steps, until no point changes its centroid (6)
- Variants for computing the cluster centroids:
  - Computing the mean (K-Means-algorithm)
  - Computing the median (K-Median-algorithm)
  - Choosing data points as centroids (K-Mediods-algorithm)
  - Density distribution (DB-SCAN)

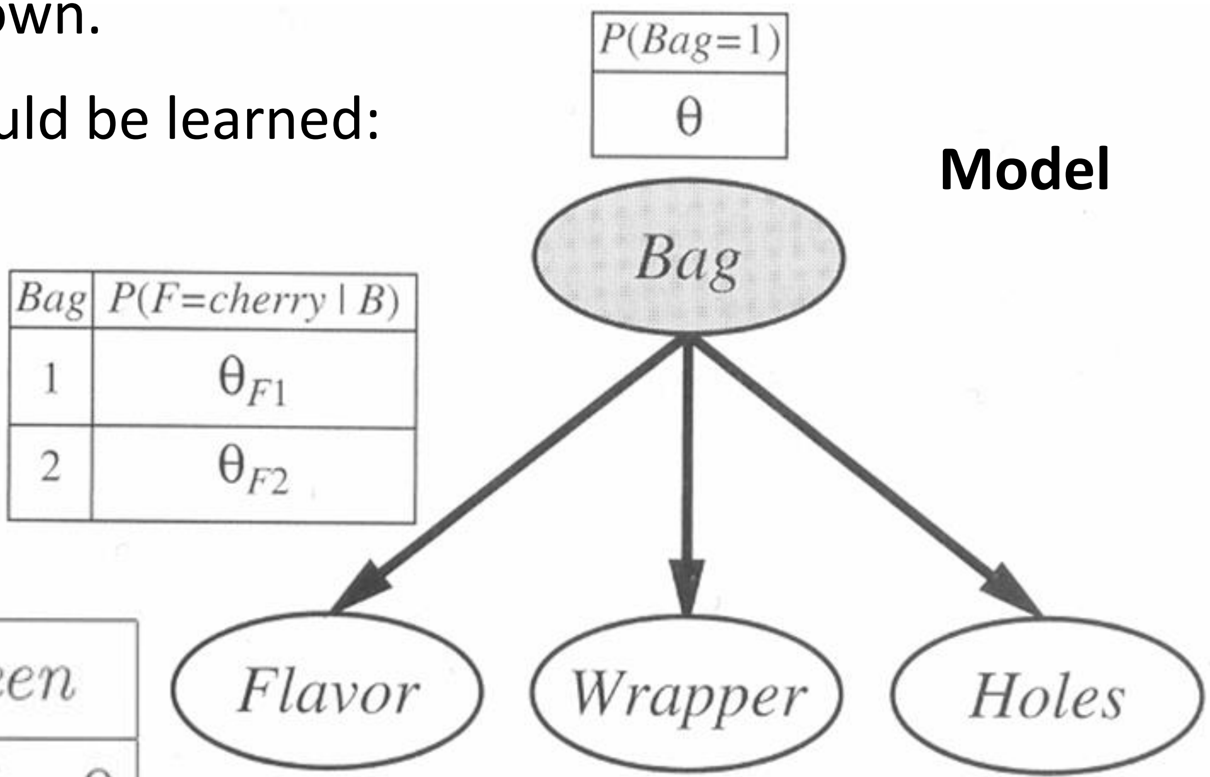


Example for EM for Bayesian Networks

- Task: From two bags are 1000 candies taken (s. table). The candies have 3 attributes: Flavor (cherry, lime), Wrapper (red, green), Hole (yes, no). The bags have a different distribution of candy attributes, but it is unknown.

- The following 7 parameters of the model should be learned:
  θ: P(Bag=1)
  θ_{F1}: P(F=cherry | Bag1)
  θ_{F2}: P(F=cherry | Bag2)
  θ_{W1}: P(W=red | Bag1) θ_{H1}: P(H=1 | Bag1)
  θ_{W2}: P(W=red | Bag2) θ_{H2}: P(H=1 | Bag2)

Model



Data	$W = red$		$W = green$	
	$H=1$	$H=0$	$H=1$	$H=0$
$F = cherry$	273	93	104	90
$F = lime$	79	100	94	167

Solution with EM (1)

1. **Initialization:** Guess the parametes, e.g. $\theta = 0.6$; $\theta_{F1}=\theta_{W1}=\theta_{H1}=0.6$; $\theta_{F2}=\theta_{W2}=\theta_{H2}=0.4$
2. **Expectation**: Compute, how many candies with different attributes came from bag1
   - Distribute the 273 cherry red candies with hole between bag1 and bag2:
   - bag1: 0.6*0.6*0.6*0.6 = 0.1296
   - bag2: 0.4*0.4*0.4*0.4 = 0.0256
   - proportion bag1 of cherry red candies with hole = 0.1296 / (0.1296+0.0256) ≈ 83,5%
   - Guessed number of cherry red candies with hole from bag1 = 0.835 * 273 ≈ 227,97
   - Repeat for all 8 candy combinations; results for bag1:

bag1	W=red		W=green	
	H=1	H=0	H=1	H=0
F=cherry	227,97	64,38	72,00	45,00
F=lime	54,69	50,00	47,00	51,38

Data	$W = red$		$W = green$	
	$H=1$	$H=0$	$H=1$	$H=0$
$F = cherry$	273	93	104	90
$F = lime$	79	100	94	167

Results for bag2 (difference of bags1 and data):

bag2	W=red		W=green	
	H=1	H=0	H=1	H=0
F=cherry	45,03	28,62	32,00	45,00
F=lime	24,31	50,00	47,00	115,62

Solution with EM (2)

3. **Maximization:** From these numbers, recompute the parameters of the model:
   - θ = sum bag1/1000 ≈ 0.61243
   - θ_{F1} = sum (F=cherry from bag1) / sum (bag1) = 409.35/612.43 = 0.6684
   - θ_{W1}= sum (W=red from bag1) / sum (bag1) = 397,05/612,43 = 0.6483
   - θ_{H1}= sum (H=1 from bag1) / sum (bag1) = 401,66/612,43 = 0.6558
   - 1-θ = sum bag2/1000 ≈ 0.3876
   - θ_{F2}= sum (F=cherry from bag2) / sum (bag2) = 150.65/387.57 = 0.3887
   - θ_{W2}= sum (W=red from bag2) / sum (bag2) = 147.95/387.57 = 0.3817
   - θ_{H2}= sum (H=1 from bag2) / sum (bag2) = 148.34/387.57 = 0.3827

bag1	W=red		W=green	
	H=1	H=0	H=1	H=0
F=cherry	227,97	64,38	72,00	45,00
F=lime	54,69	50,00	47,00	51,38

	$W = red$		$W = green$	
	$H = 1$	$H = 0$	$H = 1$	$H = 0$
$F = cherry$	273	93	104	90
$F = lime$	79	100	94	167

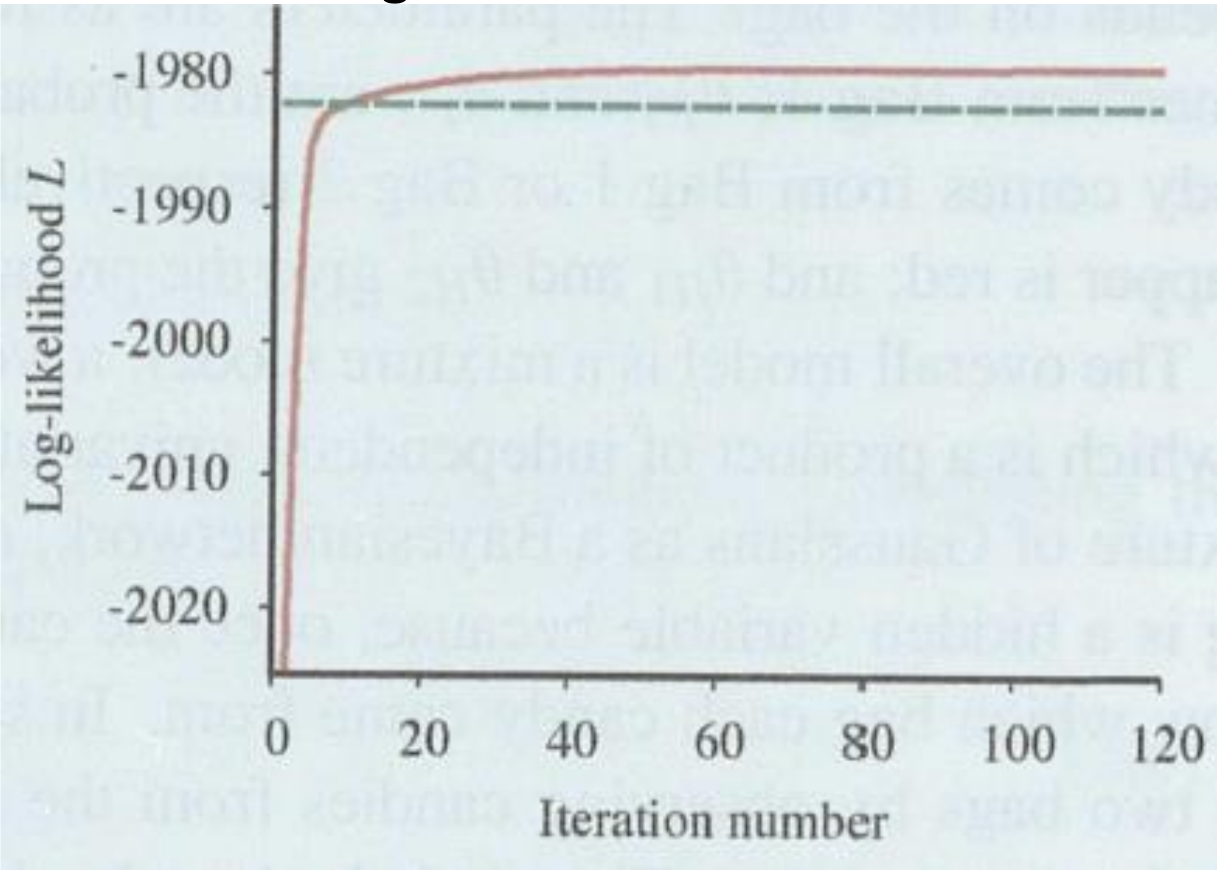
bag2	W=red		W=green	
	H=1	H=0	H=1	H=0
F=cherry	45,03	28,62	32,00	45,00
F=lime	24,31	50,00	47,00	115,62

Solution with EM (3)

- Iterate the expectation and maximization steps until no relevant changes of the model occur

- We can measure the fit of the model to the data by computing the likelihood, that the exact numbers of the data are generated by the model.
- Since this is a very small probability, we take the logarithm of the likelihood (Log-likelihood)
    - Log-likelihood of initial model = -2044
    - Log-likelihood after 1. iteration = -2021
    - Log-likelihood after 10. iteration > -1982
        - Better fit than original model (-1982): $\theta = 0.5$; $\theta_{F1}=\theta_{W1}=\theta_{H1}=0.8$; $\theta_{F2}=\theta_{W2}=\theta_{H2}=0.3$

Red line: learned model after n iterations
Green line: original model



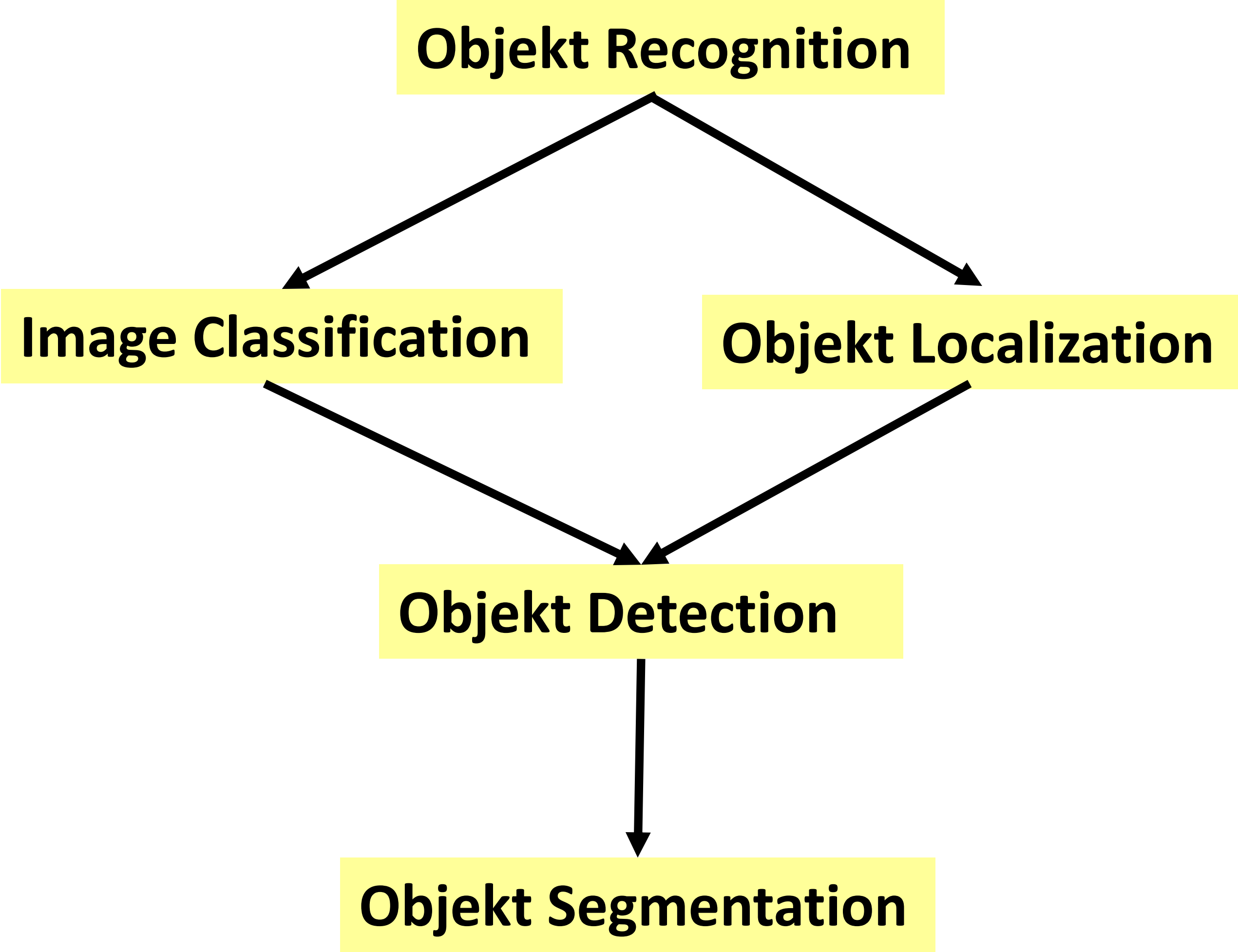
Limits of EM Algorithm

- In the example, the EM-algorithm recovered 7 parameters θ, θ_{F1},θ_{W1},θ_{H1}, θ_{F2},θ_{W2},θ_{H2} from 7 (2^3 -1) observed counts in the data (the 8th count can be computed from the others).
- If each candy is described by two attributes (e.g. omitting the holes), 5 parameters (θ, θ_{F1},θ_{W1}, θ_{F2},θ_{W2}) must be derived from 3 (2^2 -1) counts, which is impossible
    - The two-attribute model is not **identifiable**
- Even for the three-attribute model, there are two symmetric solution, if bag1 and bag2 are exchanged. This kind of non-identifiability is unavoidable with variables that are never observed.

Chapter 21+: Deep Learning Architectures

- Vision
- Natural language Processing

Deep Learning Tasks for Vision

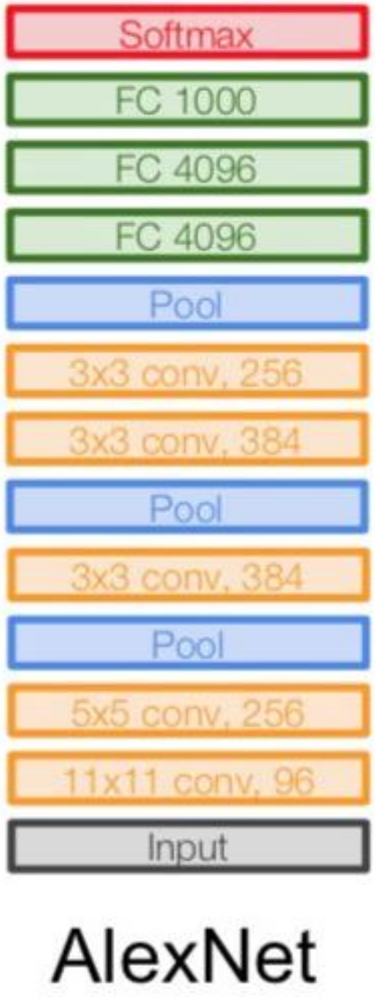


Deep Learning Architectures for Image Classification

- Basis: Convolutional-, Pooing-, Fully Connected and Softmax-Layers für Merkmalserkennung, Dimensionsreduktion und finale Klassifikation
- Verschiedene Architekturen
    - **AlexNet**
    - **VGGNet**
    - **ResNet**
    - **DenseNet**
    - **Inception**
    - **EfficientNet**

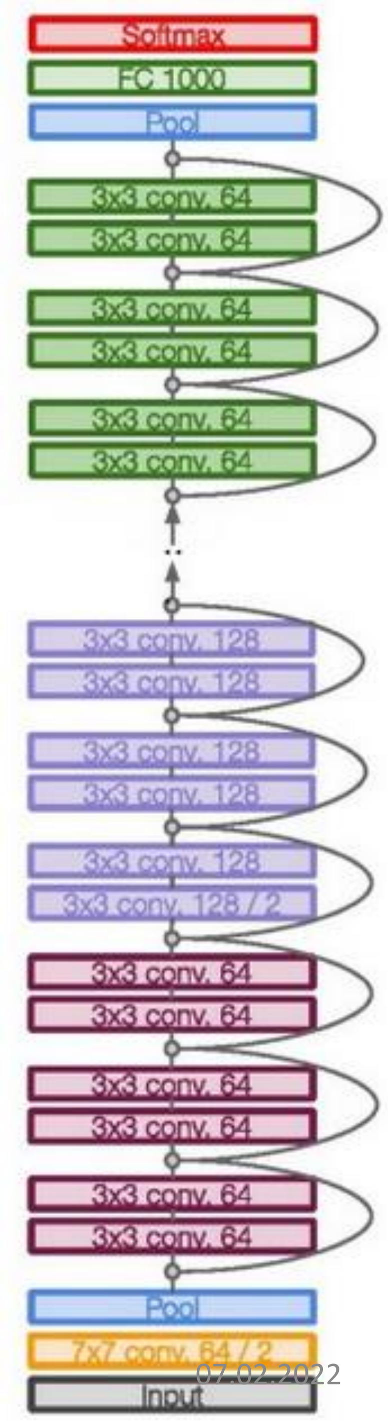
aus: https://towardsdatascience.com/architecture-comparison-of-alexnet-vggnet-resnet-inception-densenet-beb8b116866d

AlexNet und VGGNet

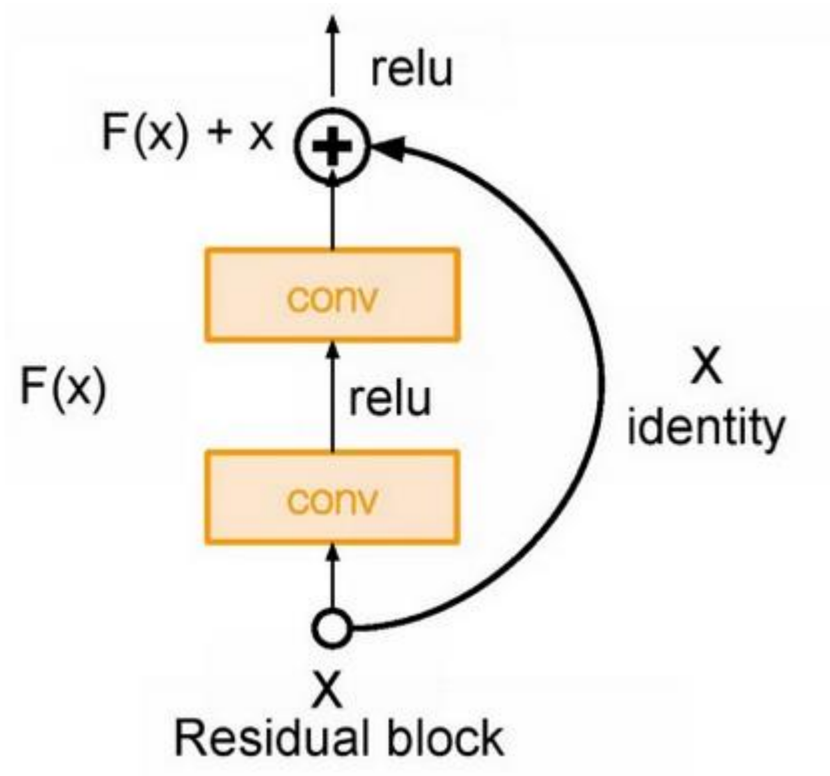


- Erstes Modell: **AlexNet:** Sieger Image Net Competition ILSVRC 2012 (links)
- Verbesserung: **VGGNet:** Mehr Ebenen mit kleineren Convolutions (meist 3x3): Sieger Image Net Competitior ILSVRC 2014 (rechts)
- Weitere Verbesserungen zur Vermeidung des Vanishing Gradient Problems:
  - ResNet
  - DenseNet
  - Inception
- Optimierung der Netzgröße: EfficientNet

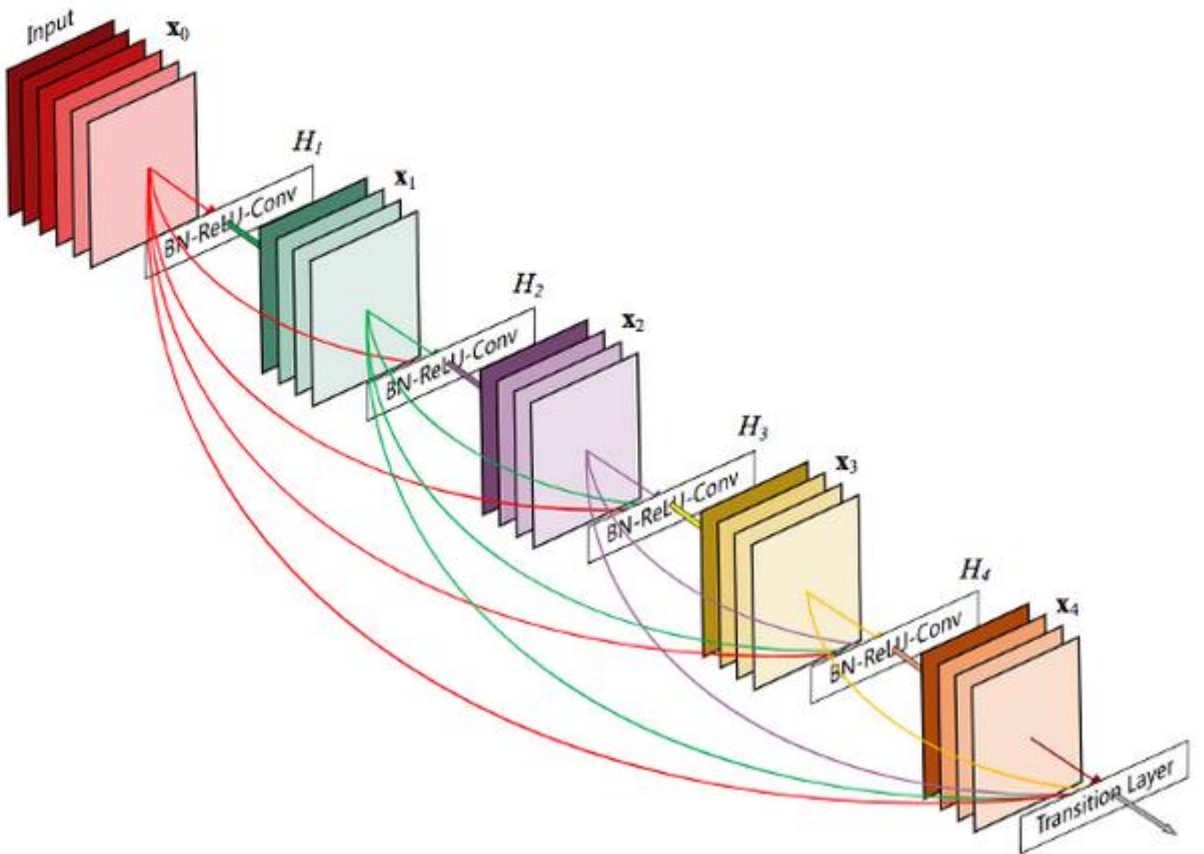
ResNet und DenseNet



ResNet (Sieger ILSVRC 2015):
Hauptmerkmal: Residual Block
Ein Layer wird nicht durch den nächsten ersetzt, sondern wird zum (übernächsten) Layer addiert



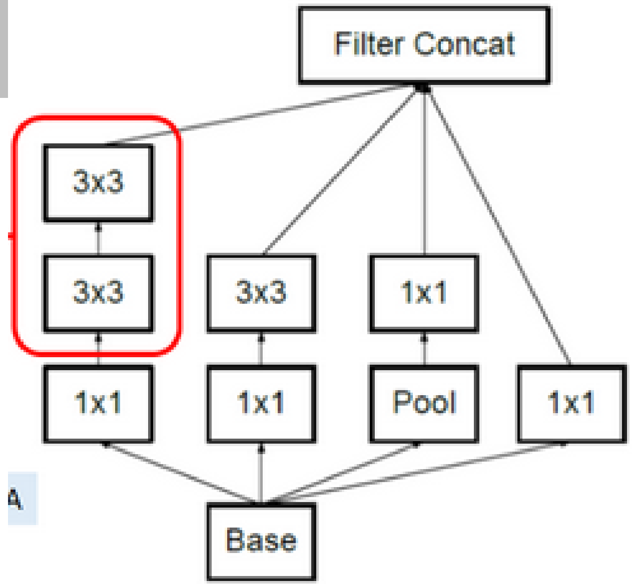
DenseNet:
Noch stärkere Wiederverwendung früherer Ergebnisse durch dichtere Verknüpfung.
Benötigt weniger Parameter als ResNet.

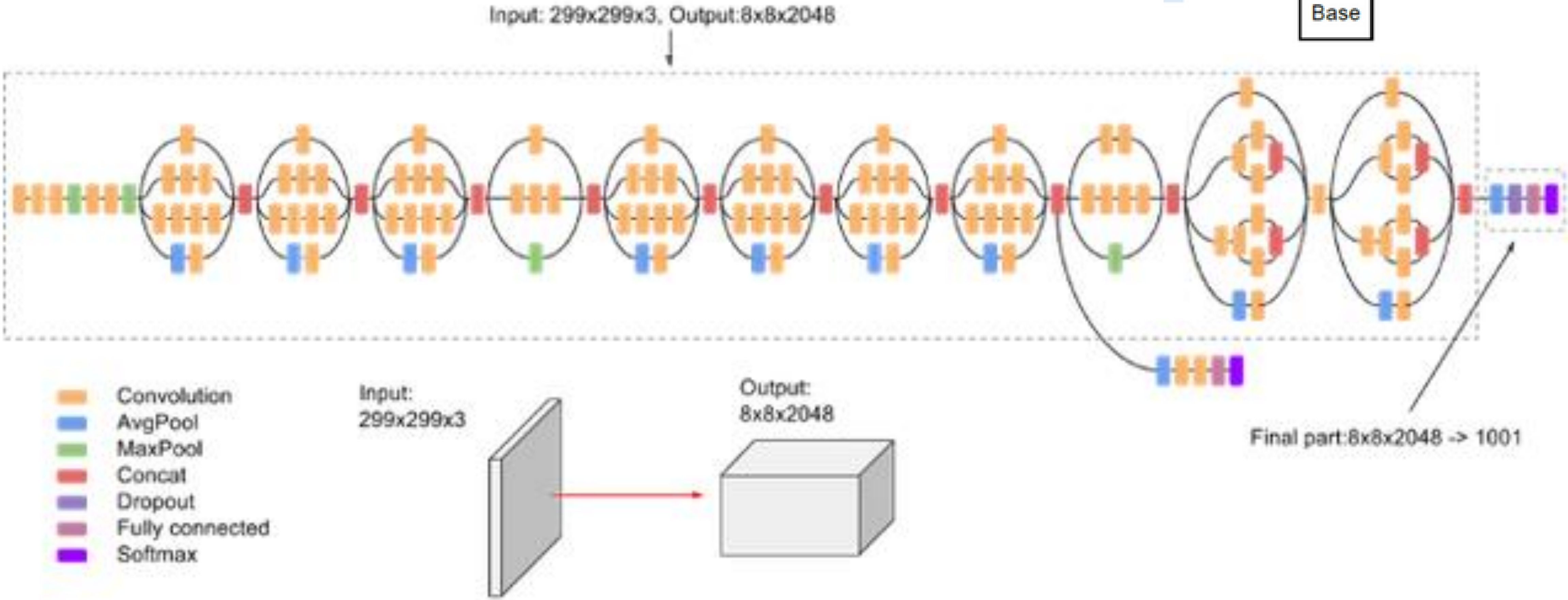


Inception

Inception: Mischung verschiedener Ideen mit symmetrischen und asymmetrischen Blöcken

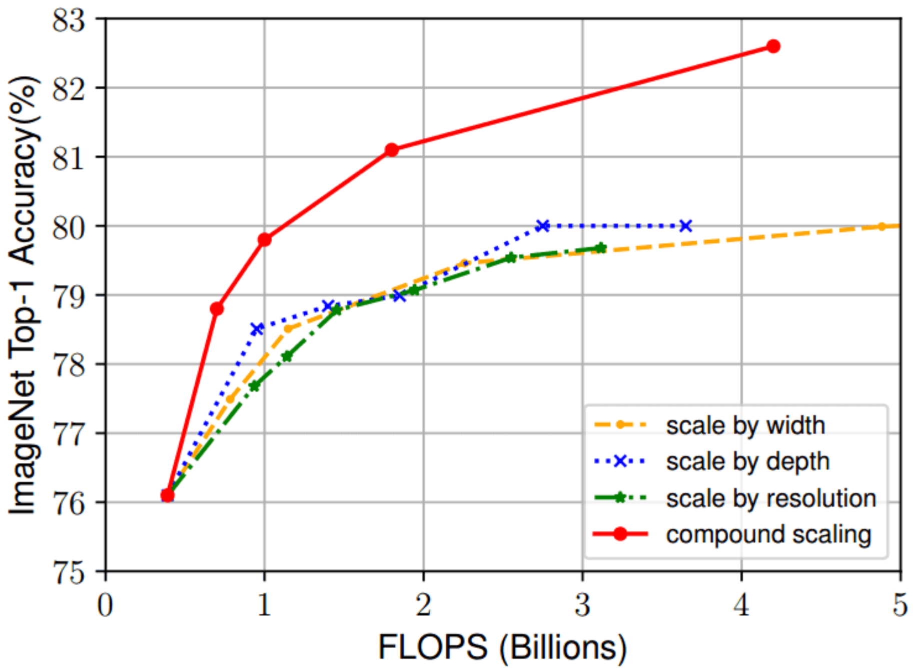
- Inception Modul verknüpft verschiedene Filter (Convolutions)





EfficientNet

aus: https://theaisummer.com/cnn-architectures/



Individual upscaling:

- With **more layers** (depth) one can capture richer and more complex features, but such models are hard to train (due to the vanishing gradients)
- **Wider networks** are much easier to train. They tend to be able to capture more fine-grained features but saturate quickly.
- By training with **higher resolution images**, convnets are in theory able to capture more fine-grained details. Again, the accuracy gain diminishes for quite high resolutions.

New Idea in **EfficientNet**:

- let's instead scale up network depth (more layers), width (more channels per layer), resolution (input image) simultaneously. This is known as **compound scaling.**

Object Detection

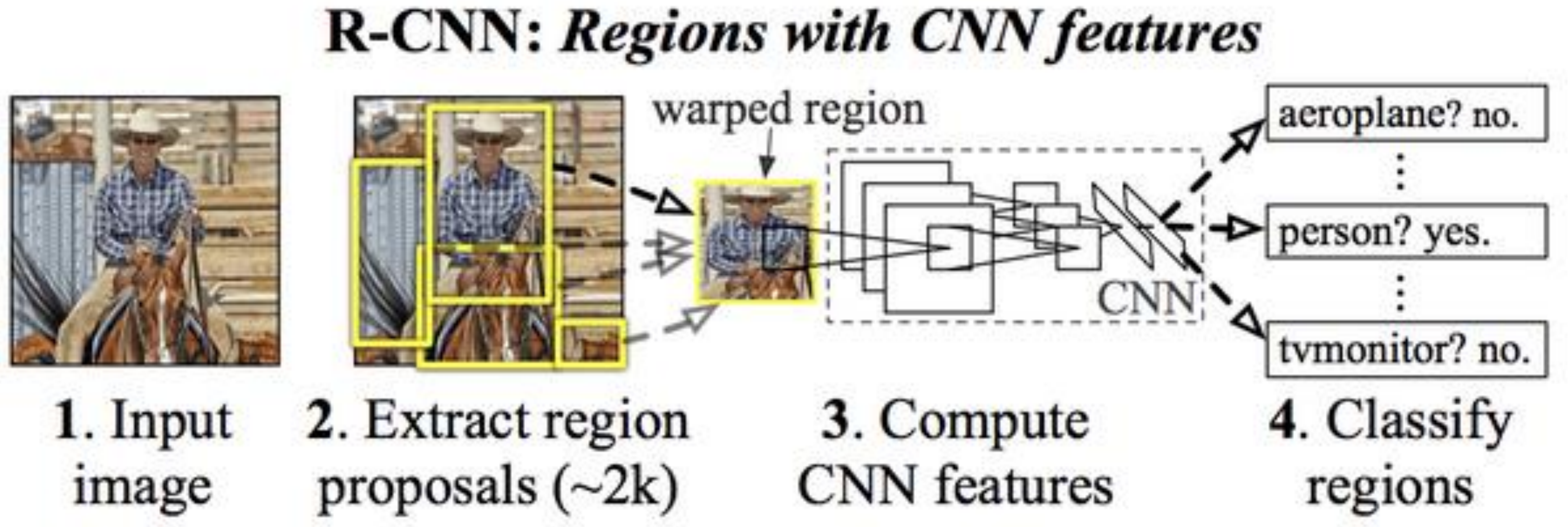
- Two-Stages Models:
  - R-CNN
  - Fast R-CNN
  - Faster R-CNN
- One-Stage Models
  - YOLO familiy (You Only Look Once)

R-CNN

R-CNN (2014): Rich feature hierarchies for accurate object detection and semantic segmentation

R-CNN consists of Three Modules:

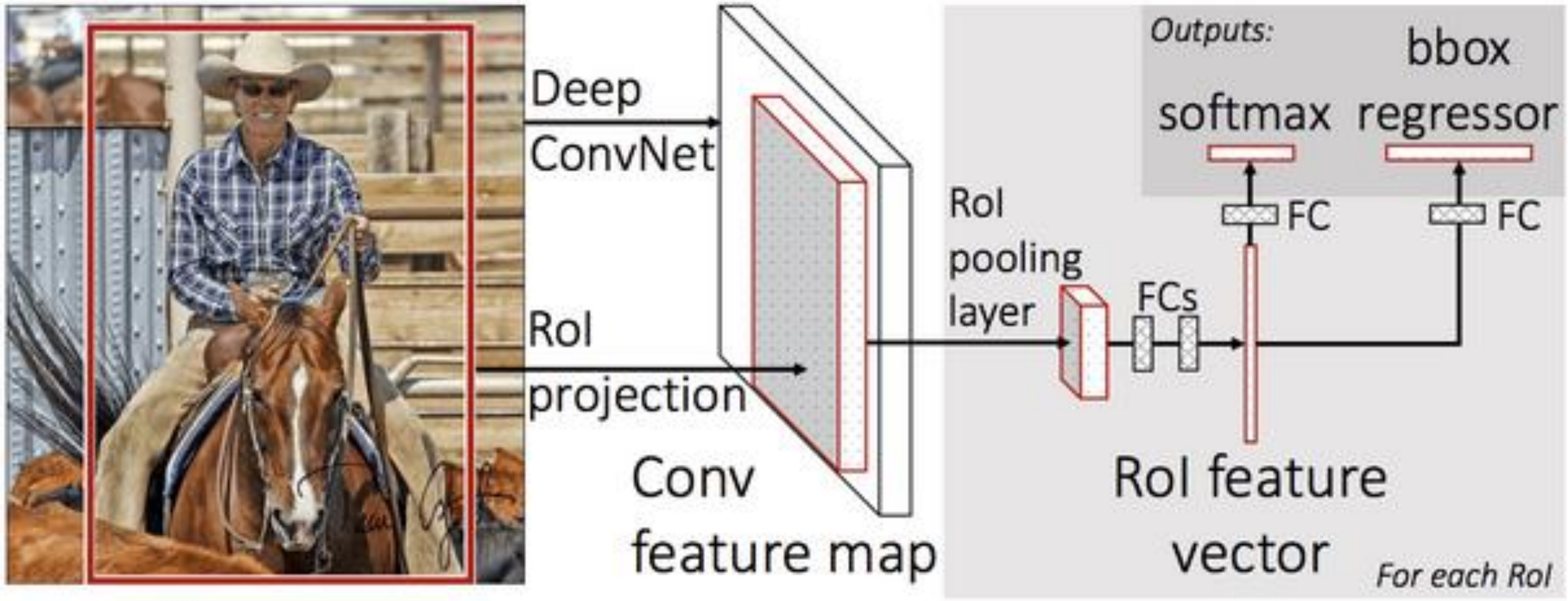
1. Region Proposal
2. Feature Extractor (originally AlexNet was used)
3. Classifier



https://machinelearningmastery.com/object-recognition-with-deep-learning/

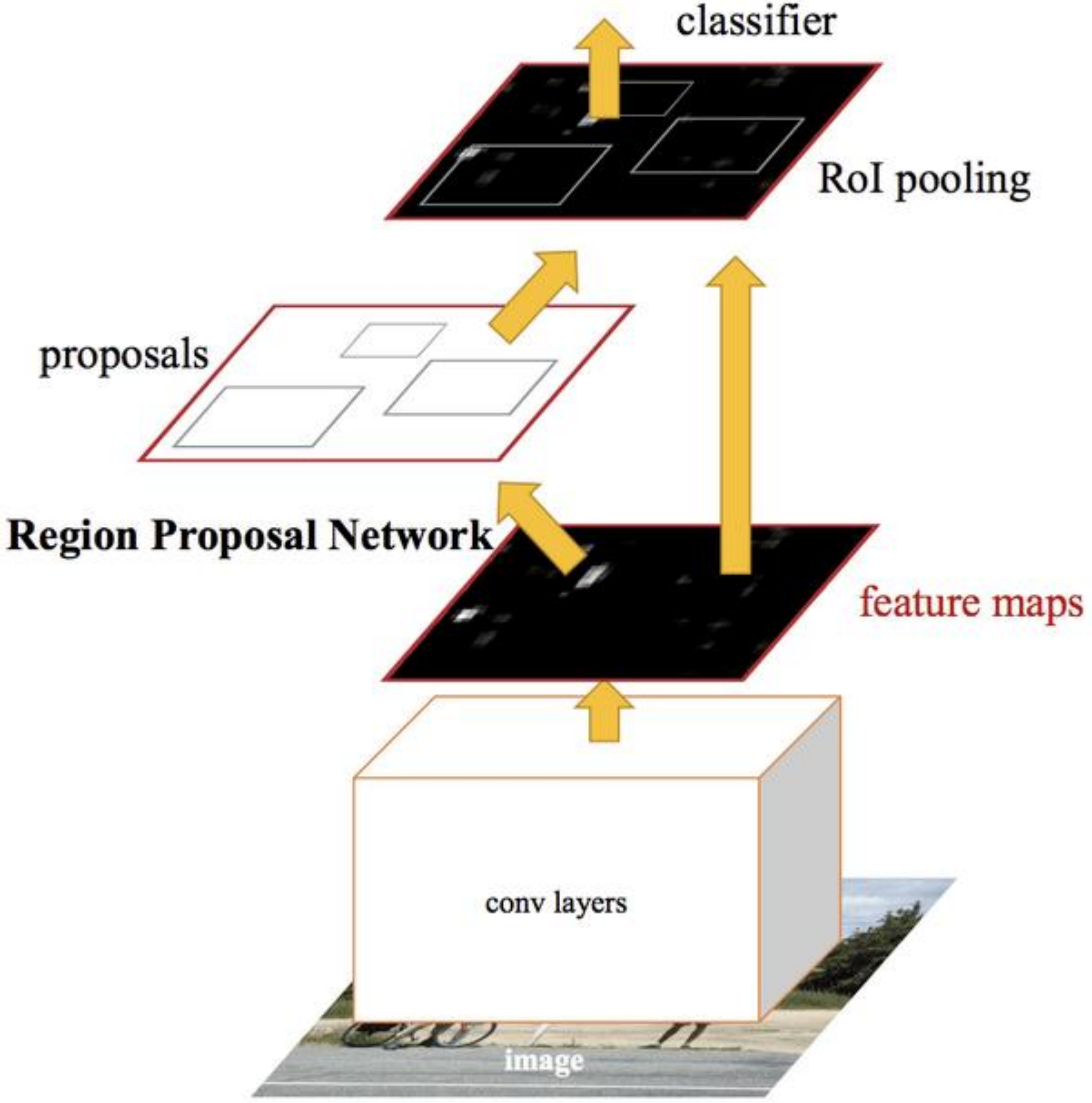
Fast R-CNN

- Problems of R-CNN:
  - **Training is a multi-stage pipeline** with three separate models.
  - **Training is expensive**: Training Deep CNN for many region proposals per image very slow.
  - **Object detection is also slow**: Predictions with deep CNN on many region proposals
- Improvement: **Fast R-CNN**
  - Single Model with reuse
  - Set of region proposals
  - Feature extraction with pre-training CNN like VGG-16
  - Result: Region of Interest (RoI) pooling layers
  - Interpretation by FC (Fully connected layer)
  - For each RoI: Class prediction and bounding box computation



Faster R-CNN

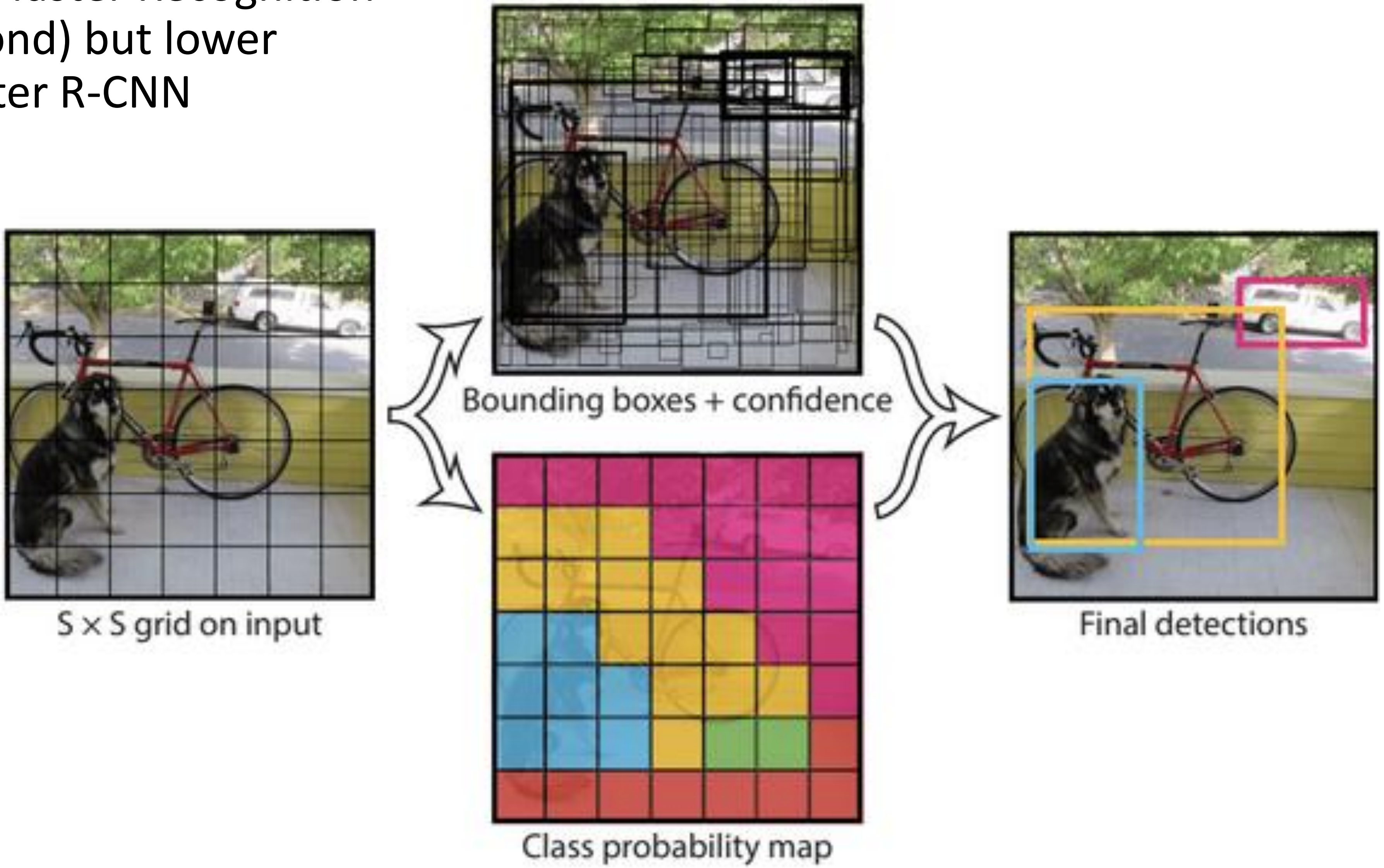
- Further improvement in speed of training and dection
- Winner Object Recognition and Detection Image Net Competition ILSVRC 2015
- Single model with two modules based on the output of a deep CNN
  - **Module 1: Region Proposal Network** (CNN) for region proposal
  - **Module 2: Fast R-CNN** for extracting features from the proposed regions and outputting the bounding box and class labels.



YOLO (You Only Look Once)

Single neural Network with faster Recognition Time (> 45 Frames per second) but lower accurracy compared to Faster R-CNN

- Image split in grid of cells. Each cell respon-sible for predicting a bounding box (x, y coordinate, width, height, confidence) if the center of a bounding box falls within the cell
- Combination to the finald set of bounding boxes and class labels



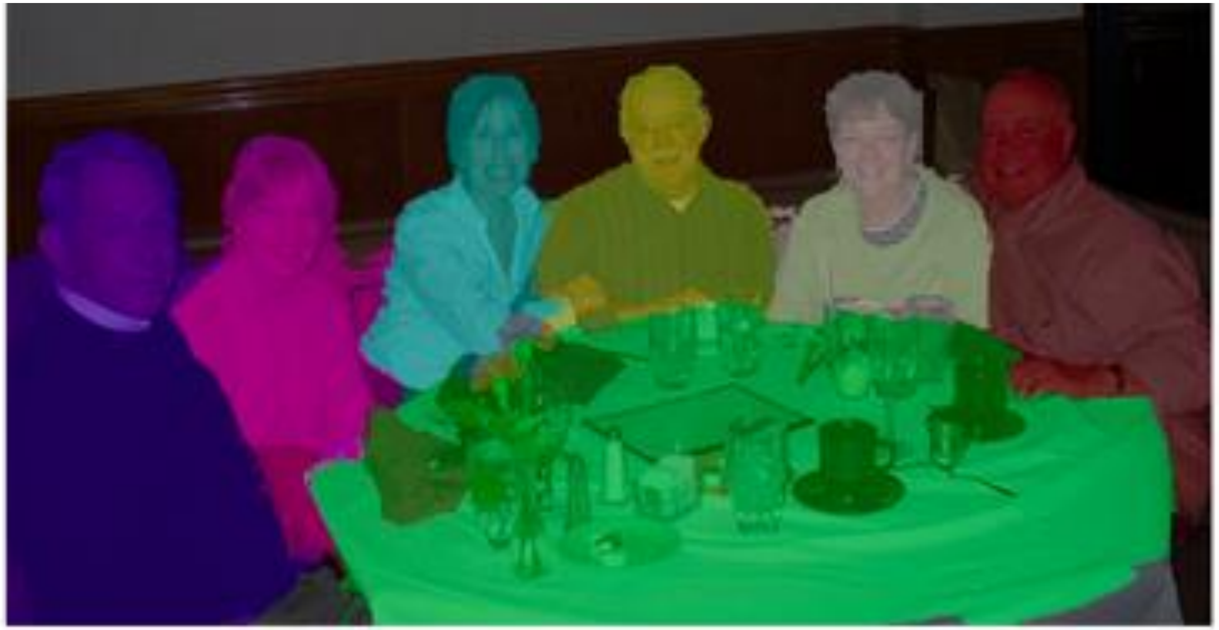
YOLO improvements

- YOLOv2 improvements to YOLO:
  - Like Faster R-CNN, anchor boxes are used (with pre-defined shapes), which are tailored durching training
  - Predicted representation of bounding box is changed to allow small variations
    - instead of coordinates, offsets relative to a grid cell are predicted
- YOLOv3: Minor improvements (e,g. deeper feature detector)
- YOLOv4 und YOLOv5: further small improvements

Image Segmentation

- Every pixel is labeled with an object type
  - Semantic segmentation: All objects of the same type get the same class label
  - Instance segmentation: Similar objects get their own seperate labels

Semantic Segmentation

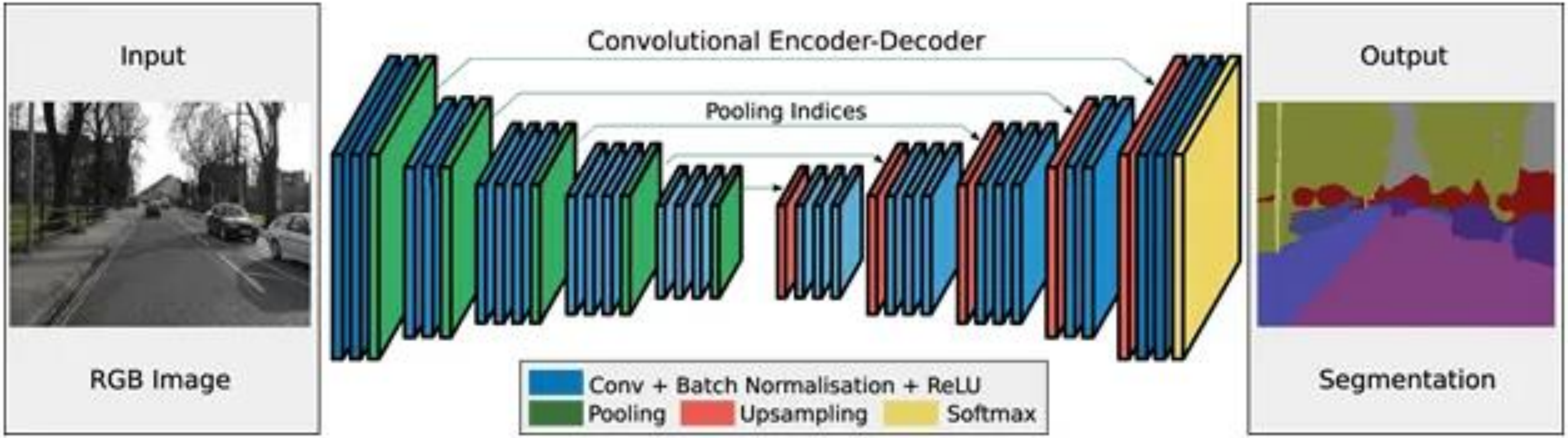


Instance Segmentation

https://neptune.ai/blog/image-segmentation

Image segmentation architectures

- The basic architecture consists of an encoder and a decoder (with skip projections)

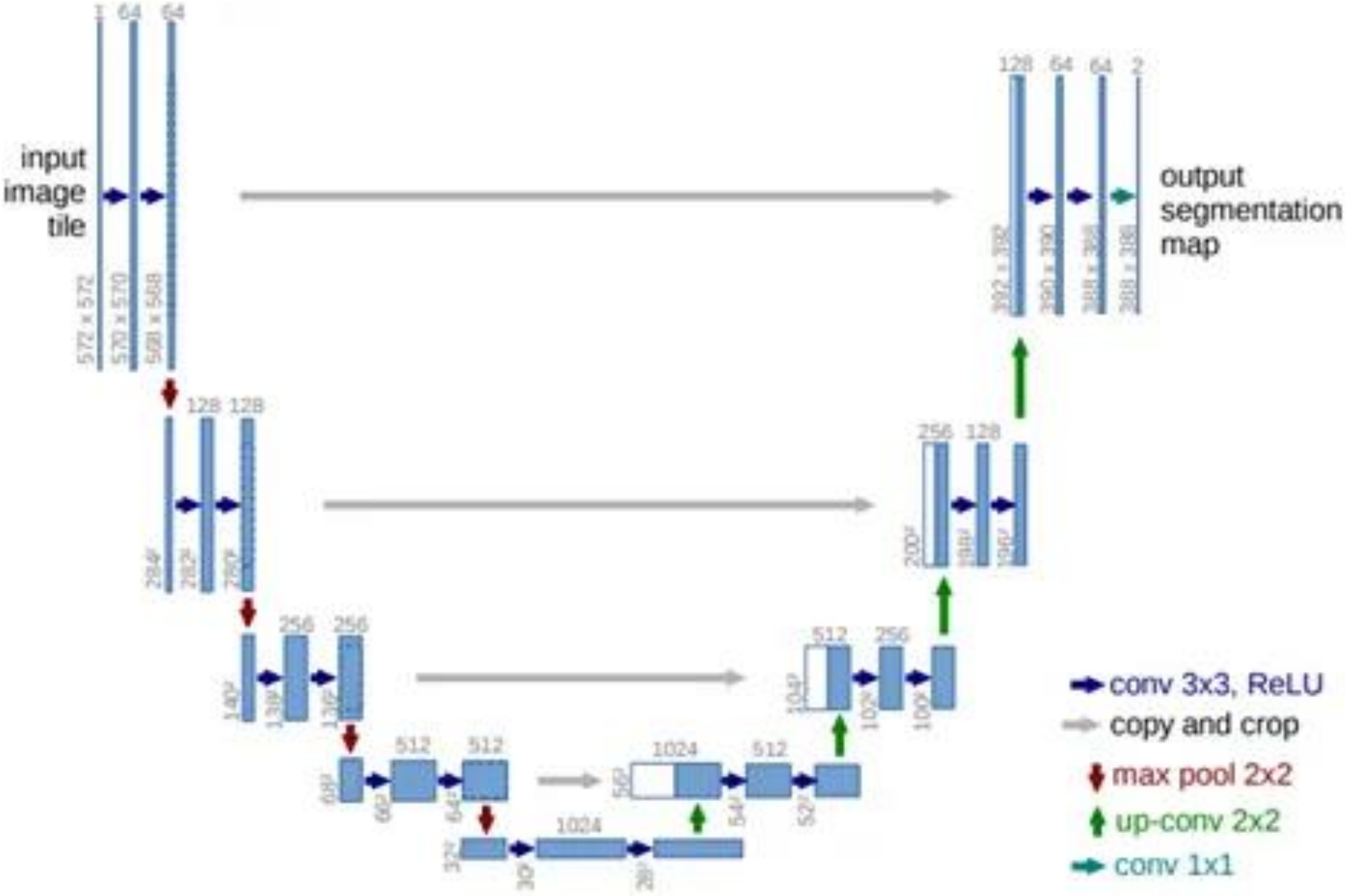


- Concrete architectures:
  - **U-Net**
  - **FastFCN** (Fast Fully Convolutional Network)
  - **Mask R-CNN**
  - Gated-SCNN
  - DeepLab

U-Net

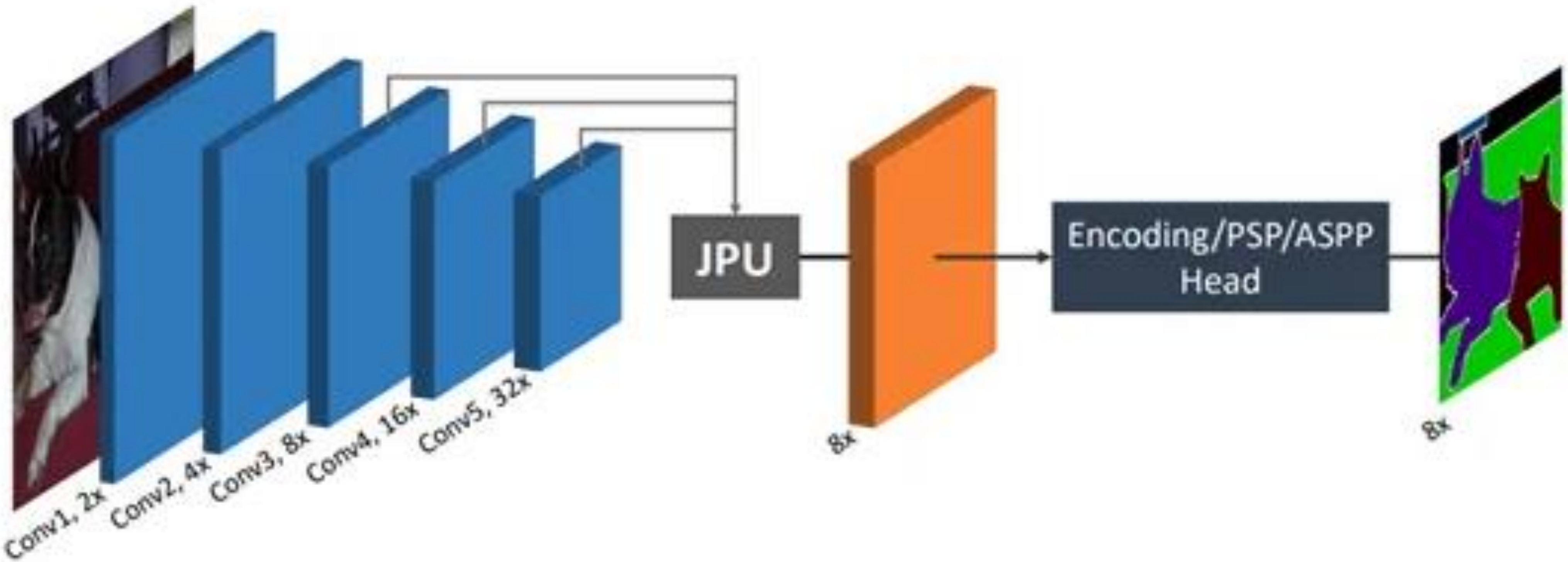
Two parts:

- left part: Encoder: Contraction for capturing context)
- right part: Decoder: Expansion for precise localication



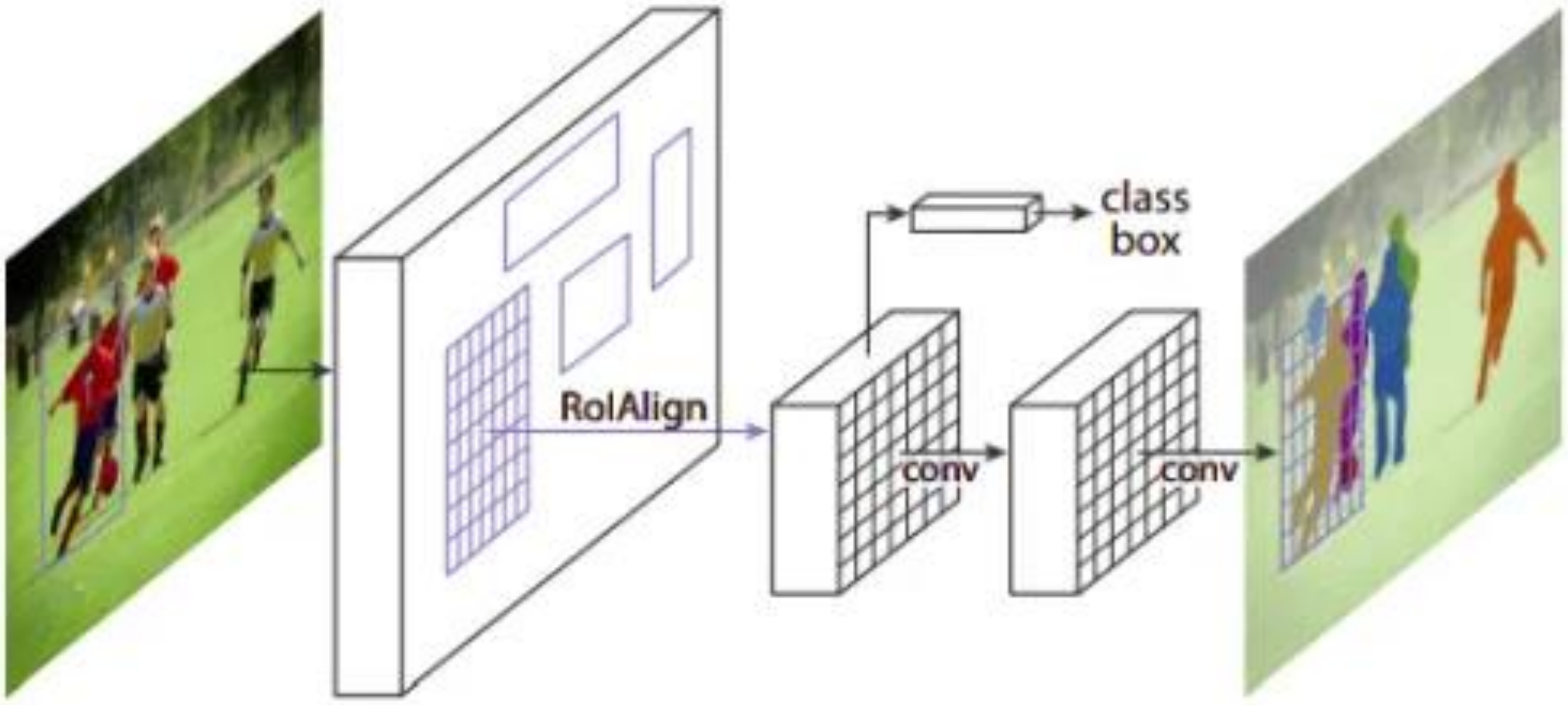
FastFCN (Fast Fully Convolutional Network)

- A Joint Pyramd Upsampling (JPU) is used to replace dilated convolutions requiring less memory and time.
- Downsampling: Fully convolutional Network
- Upsampling: JPU

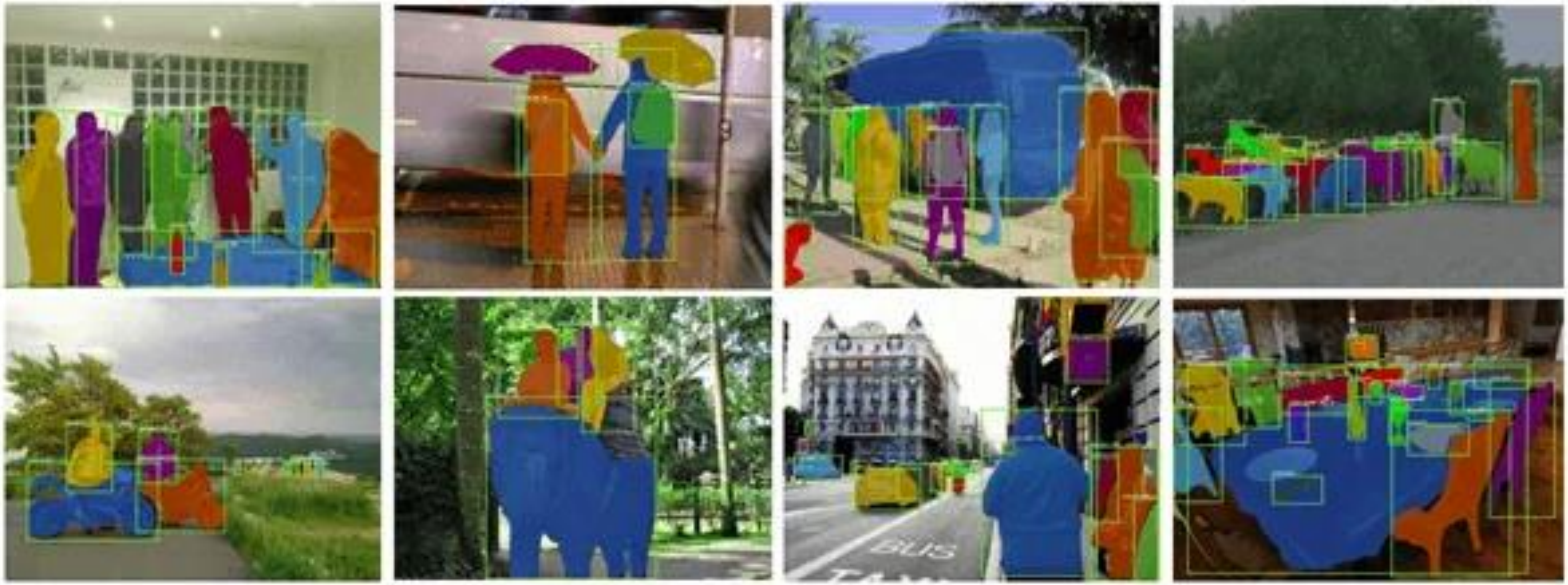


Mask R-CNN

- Architecture is an expansion for Faster R-CNN
  - Bounding Boxes of objects are refined to pixel-wise labeling



Examples of Mask R-CNN on COCO test set



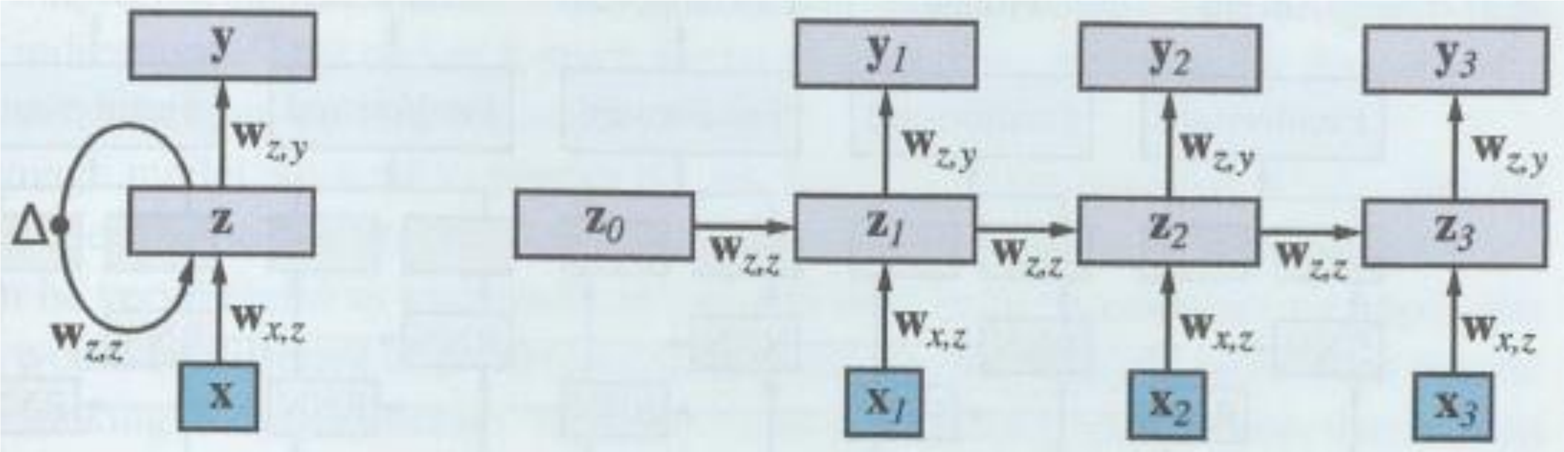
- For image classification, ResNet-101 is used

Natural Language Processing

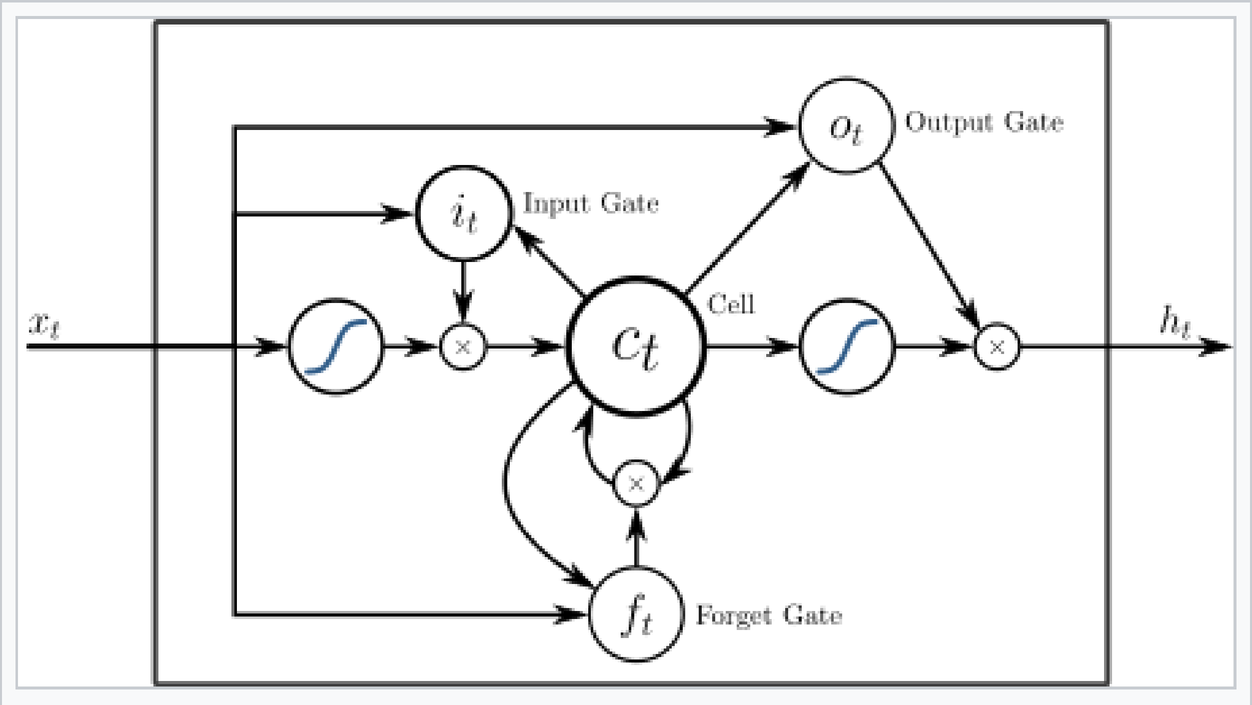
- Tasks:
  - **Text transformations**
    - Speech-to-Text
    - Text-to-Speech
    - Language Translation
    - Text Paraphrasation
  - **Information extraction:** Extract structured data for a given ontology
  - **Meaning reconstruction**: Represent the meaning of a text in an appropriate ontology

Methods for Text Transformations

- Hidden Markov Models (HMM)
- Recurrent Neural Nets (RNN)



- Long Short-Term Memory (LSTM)



- Transformer Architecture (see next slides)

Attentional Sequence-to-Sequence Model

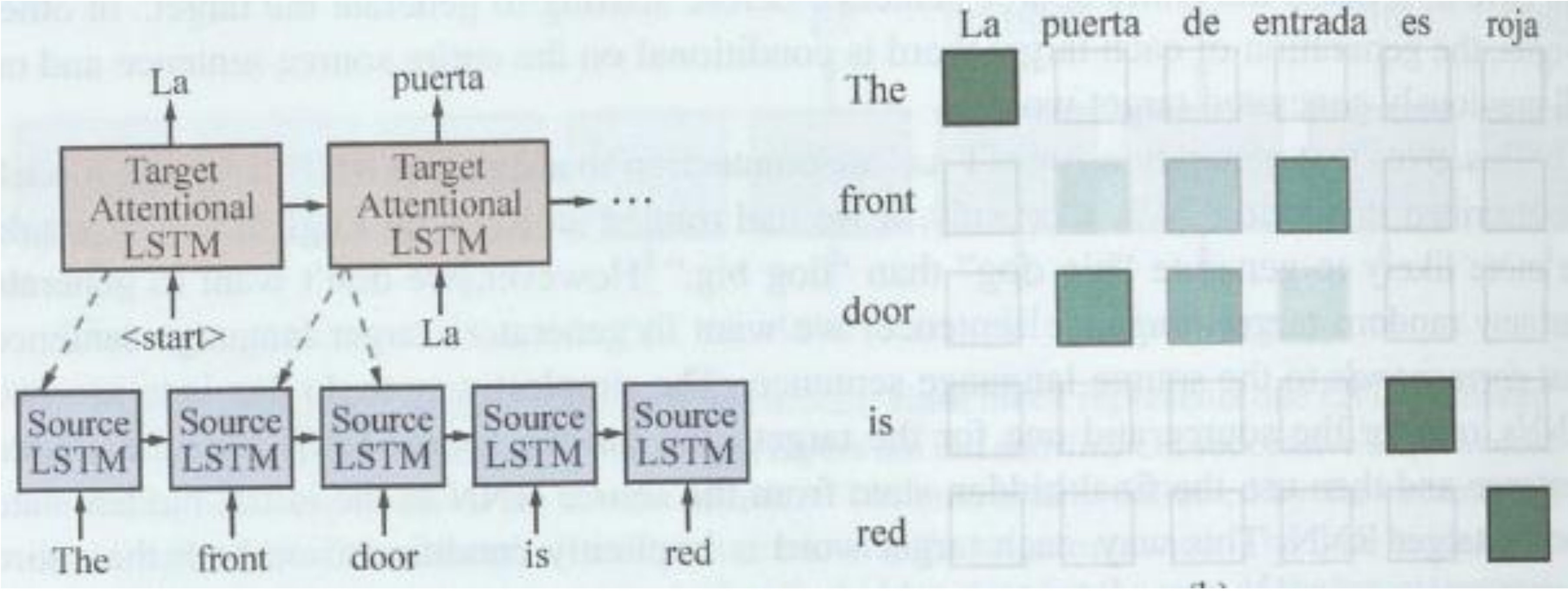
Standard RNN: $\mathbf{h}_i = \text{RNN}(\mathbf{h}_{i-1}, \mathbf{x}_i)$; **Attentional RNN**: $\mathbf{h}_i = \text{RNN}(\mathbf{h}_{i-1}, [\mathbf{x}_i;\mathbf{c}_i])$

where $[\mathbf{x}_i;\mathbf{c}_i]$ is the concatenation of the input ($\mathbf{x}_i$) and context vectors ($\mathbf{c}_i$):

$c_i = \sum_j a_{ij} \cdot s_j$ // context vector as weighted average over source vector word s_j

$a_{ij} = e^{r_{ij}} / (\sum_k e^{r_{ik}})$ // normalized attention scores (probabilities) using softmax over all words

$r_{ij} = h_{i-1} \cdot s_j$ // raw attention score for the target state h_{i-1} and a source vector word s_j



Extensions in Transformer Architecture (1)

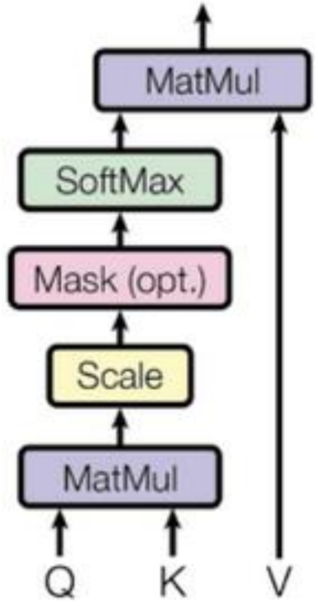
- Vaswani et al.: Attention is all you need, 2018.
- Key concept of tranformer architecture: **Self-attention**:
  - Allows to model long-distance context with a sequential dependency
  - Extends the attention mechanism (from target RNN to source RNN) so that each sequence of hidden states also attends to itself (source to source, target to target)

Extensions in Transformer Architecture (2)

Applying self-attention with a matrix just formed by a dot product of the input vector not optimal, since a dot product between a vector and itself is always (too) high

- Therefore, the input is first projected into 3 different representations using three different weight matrices. They are learnt from final translationen losses in the training data and represent arbitrary vectors (the names query and key are not helpful in understanding).
  - **Query vector $\mathbf{q}_i = \mathbf{W}_q\mathbf{x}_i$**
  - **Key vector $\mathbf{k}_i = \mathbf{W}_k\mathbf{x}_i$**
  - **Value vector $\mathbf{v}_i = \mathbf{W}_v\mathbf{x}_i$**
- Structure of self attention-matrix a_{ij} resp. $A = Q^T \cdot K$:

	The	front	door	is	red
The	$k_1 \cdot q_1$	$k_1 \cdot q_2$	$k_1 \cdot q_3$	$k_1 \cdot q_4$	$k_1 \cdot q_5$
front	$k_2 \cdot q_1$	…			
door	$k_3 \cdot q_1$				
is	$k_4 \cdot q_1$				
red	$k_5 \cdot q_1$				



Propagation of vectors:

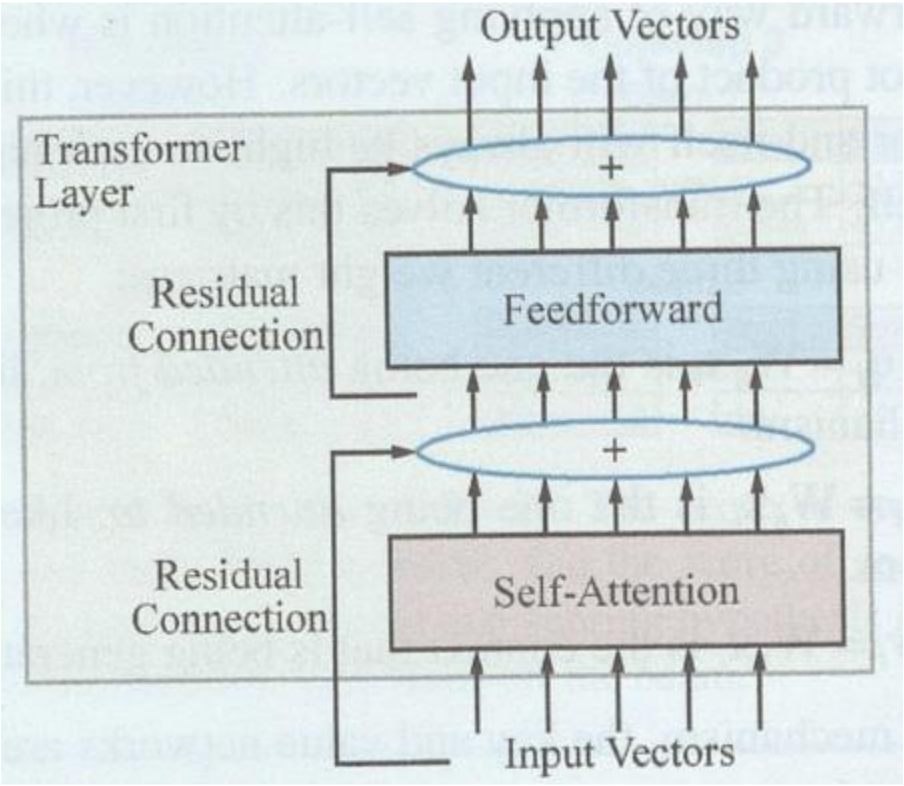
$r_{ij} = q_i \cdot k_j / \sqrt{d}$
$a_{ij} = e^{r_{ij}} / (\sum_k e^{r_{ik}})$
$c_i = \sum_j a_{ij} \cdot v_j$

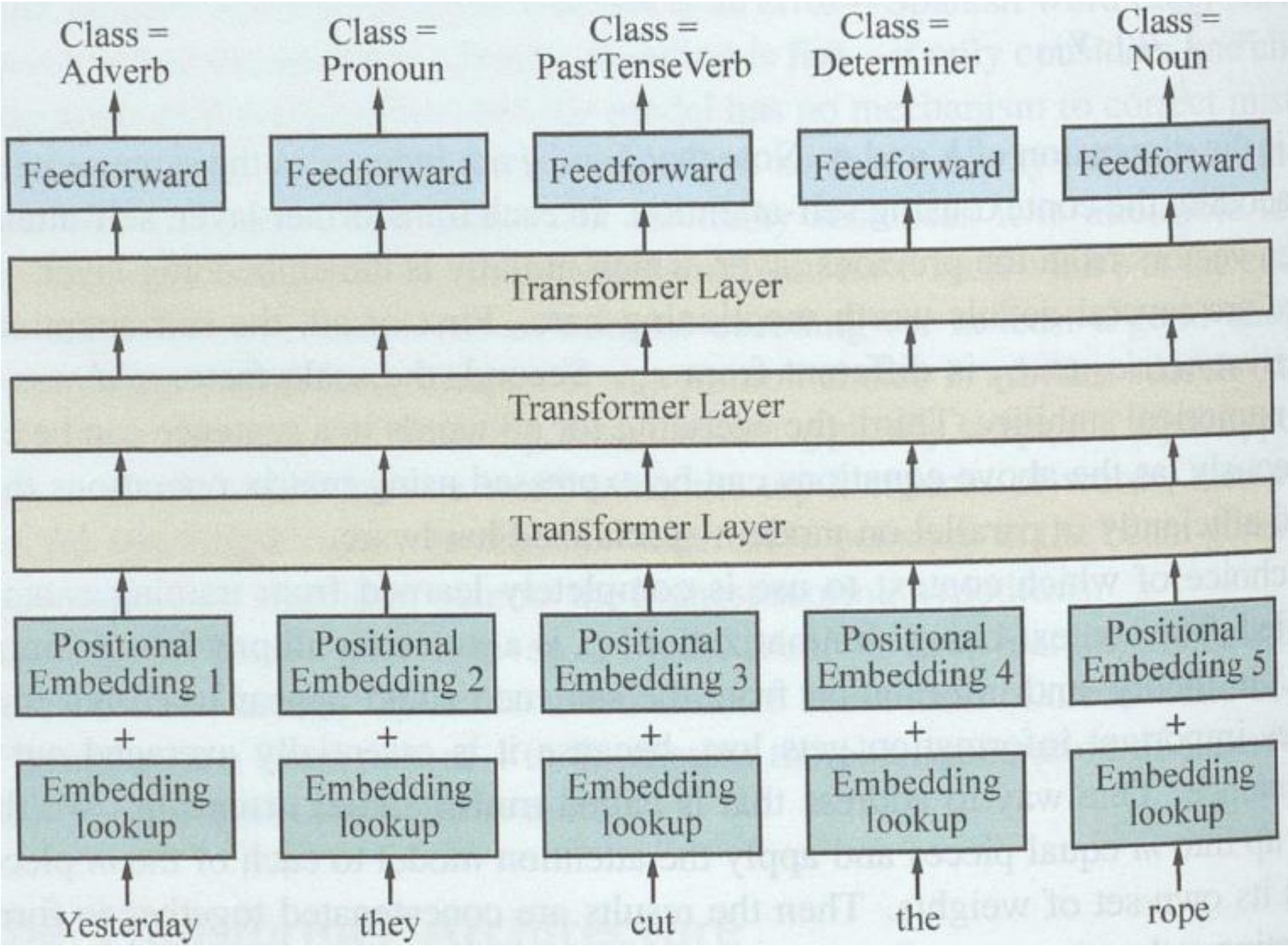
Extensions in Transformer Architecture (3)

- The context is the completely learned from training examples.
- The context-based summarization $\mathbf{c_i}$, is a sum over all previous positions in the sentence.
- In long sequences, information is averaged over the whole sentence and might get lost
- A possible solution is **multiheaded attention**.
  - Instead of one attention matrix, 8 attention matrices are learnt (not necessarily from different parts of the sentence)
  - Each matrix has ist own set of weights.
  - The results are concatenated together to form $\mathbf{c_i}$.

Extensions in Transformer Architecture (4)

- Right: Transformer for POS-Tagging
- Below: One transformer layer consists of self attention, feedforward network and residual connections





Extensions in Transformer Architecture (5)

- Transformer Models might have > 10^{11} parameters, due to the quadratic nature of attention matrices and many layers.

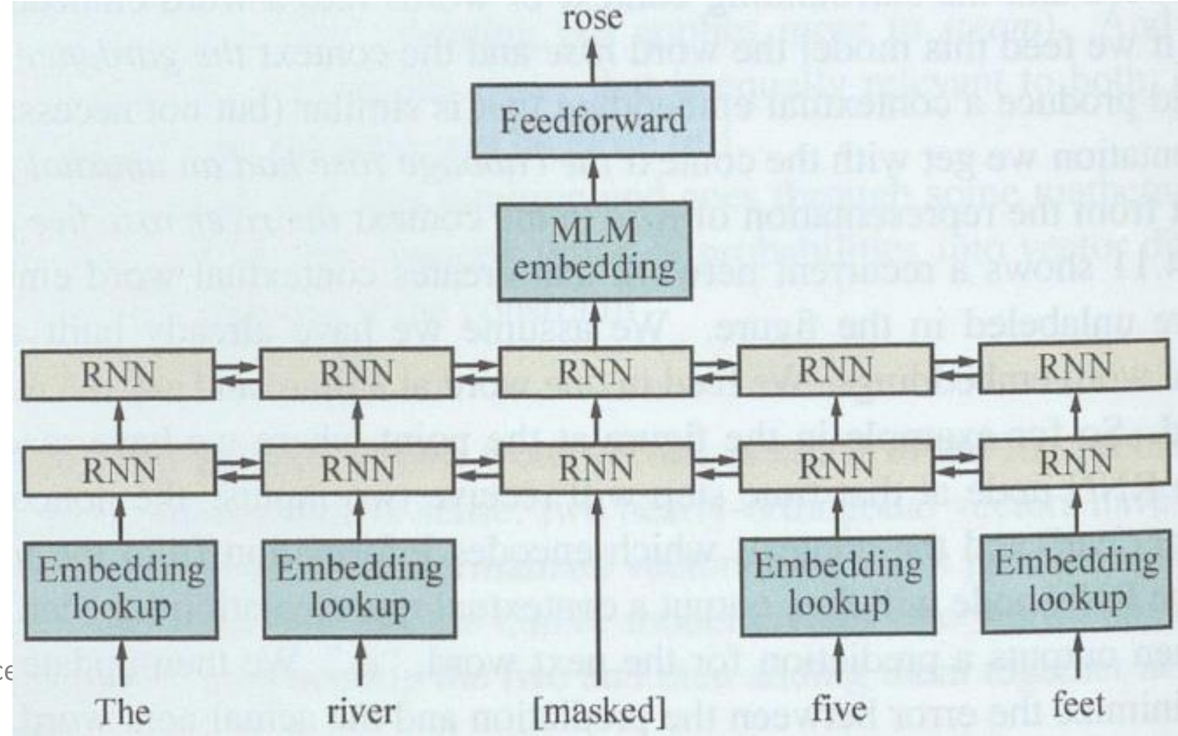
Word Representations

- One hot encoding: Word as is
  - Improvement: Reduce word to ground form (lemmatization)
- Vector encoding: Represent a word as a vector, so that similar words have similar vectors
  - Can be learned with generated data by masking and reconctructing words in sentences

- Vector models
  - Word2Vec (2013)
  - GloVe (Global Vectors for Word Representation; 2014)
  - BERT (Pre-training of Deep Bidirectional Transformers for Language Understanding)
  - RoBERTa (Robustly Optimized BERT Pretraining Approach)
  - ALBERT: A Lite BERT for Self-supervised Learning of Language Representations
  - GPT2, GPT3
  - …

Learning Word Embeddings as Vectors

- Based on co-occurrence of words
- Based on co-occurence with third words: Two words are similar, if they both appear in the context of the same other words (**GloVe** approach: *Glo*bal *Ve*ctors)
  - e.g. *ice* and *steam*: Word *solid* co-occurs more with *ice*, *gas* more with *steam* and *water* with both
  - Compute $P_{w,ice}/P_{w,steam}$ for many third words w
  - Convert the ratio to vectors fulfilling the embedding constraint $E_i \cdot E_j = \log(P_{ij})$
- Based on **masked language models (MLM)**: pretrain a bidirectional model by masking one word input and predict the masked word from the context (right)
- Learn different embeddings for ambiguous words by including the context of the word (e.g. *rose* as flower or as verb in past tense)



Chapter 22: Reinforcement Learning

Example for TD Learner

Update Formula:

$U^{\pi}(s) \leftarrow U^{\pi}(s) +$
$\alpha\, [R(s,\pi(s),s') + \gamma\, U^{\pi}(s') - U^{\pi}(s)]$;

set $\gamma = 1$; $\alpha = 0.9$ (constant for simplicity; but $\alpha = 1$ for first visit of state)

$$(1,1) \xrightarrow[Up]{-.04} (1,2) \xrightarrow[Up]{-.04} (1,3) \xrightarrow[Right]{-.04} (1,2) \xrightarrow[Up]{-.04} (1,3) \xrightarrow[Right]{-.04} (2,3) \xrightarrow[Right]{-.04} (3,3) \xrightarrow[Right]{+1} (4,3)$$

$$(1,1) \xrightarrow[Up]{-.04} (1,2) \xrightarrow[Up]{-.04} (1,3) \xrightarrow[Right]{-.04} (2,3) \xrightarrow[Right]{-.04} (3,3) \xrightarrow[Right]{-.04} (3,2) \xrightarrow[Up]{-.04} (3,3) \xrightarrow[Right]{+1} (4,3)$$

$$(1,1) \xrightarrow[Up]{-.04} (1,2) \xrightarrow[Up]{-.04} (1,3) \xrightarrow[Right]{-.04} (2,3) \xrightarrow[Right]{-.04} (3,3) \xrightarrow[Right]{-.04} (3,2) \xrightarrow[Up]{-1} (4,2)$$

First sequence (backwards):

(3,3): 1

(2,3): 0.96

(1,3): 0.92

(1.2): 0.88

(1,3): 0.92+0.9(-0.04+0.88-0.92) = 0.92-0.9*0.08 = 0.848

(1,2): 0.88+0.9(-0.04+0.848-0.88) = 0.88-0.9*0.072 = 0.8152

(1,1): 0.7752

Second sequence:

(3,3): 1 + 0.9(1 - 1) = 1

(3,2): 0.96

(3,3): 1 + 0.9(-0.04 + 0.96 - 1) = 1-0.9*0.08 = 0.928

(2.3): 0.888

(1,3): 0.848 + 0.9(-0.04 + 0.848 - 0.888) =
0.848 - 0.9*0.08 = 0.776

(1,2): 0.8152 + 0.9(-0.04 + 0.776 - 0.8152) =
0.8152 - 0.9*0.0792 = 0.74392

(1,1): 0.7752 + 0.9(-0.04 + 0.74392 - 0.7752) =
0.7752 - 0.9*0.07128 = 0.711048