Service Operations Management – ZF

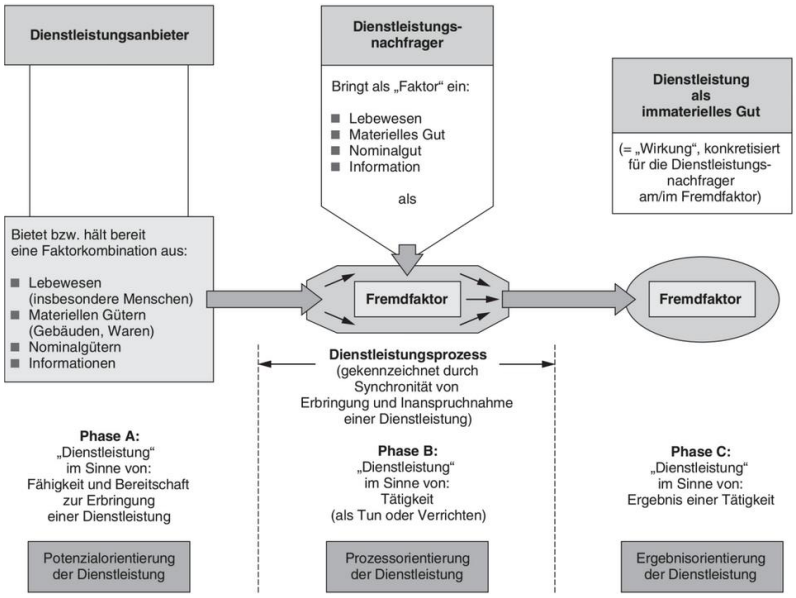
Definition von Dienstleistungen

Drei wesentliche Eigenschaften

- Potenzial-, Prozess-, und Ergebnisorienierung

Begriffsverständnis

- Dienstleistungen sind selbstständige, marktfähige Leistungen, die mit Bereitstellung (z. B. Versicherungsleistungen) und/oder dem Einsatz von Leistungsfähigkeiten (z. B. Friseurleistungen) verbunden sind (Potenzialorientierung).
- Interne (z. B. Geschäftsräume, Personal, Ausstattung) und externe Faktoren (liegen nicht im Einflussbereich des Dienstleisters) werden im Rahmen des Erstellungsprozesses kombiniert (Prozessorientierung).



- Faktorenkombination des Dienstleistungsanbieters wird mit Ziel eingesetzt, an den externen Faktoren, an Menschen (z. B. Kunden) und deren Objekten (z. B. Auto des Kunden) nutzenstiftende Wirkungen (z. B. Inspektion beim Auto) zu erzielen (Ergebnisorientierung).

Service Operations Management – SOM

Definition SOM

- SOM befasst sich mit Aktivitäten, Entscheidungen und Verantwortlichkeiten von Operations Managern in Serviceorganisationen.
- Es beinhaltet die Bereitstellung von Dienstleistungen und Mehrwert für Kunden oder Benutzer, um sicherzustellen, dass sie die richtigen Erfahrungen und die gewünschten Ergebnisse erhalten.
- Geht darum Kundenbedürfnisse zu verstehen, Serviceprozesse zu verwalten, sicherzustellen, dass die Ziele der Organisation erreicht werden und auf kontinuierliche Verbesserung Dienstleistungen zu achten.

Generisches Input–Transformation–Output Modell

- **Betriebliche Perspektive:** Materialien, Wissen, Personal, Technologie, Einrichtungen und Kunden
- **Betriebliche Perspektive als Input für: Prozess:** Erlebnis, wiederum als Output für… Kundenperspektive
- **Kundenperspektive**: Produkte, Vorteile, Emotionen, Einschätzungen und Absichten

Vereinbarkeit von Operations- und Kundenperspektive

- **Betriebliche Perspektive:** Die betriebliche Perspektive betont die effiziente Nutzung von Ressourcen und Prozessen im Input-Transformations-Output Prozess
- **Kundenperspektive**: Die Kundenperspektive legt den Schwerpunkt auf die «Produkte», Erlebnis und Vorteile, wie sie vom Kunden wahrgenommen werden.

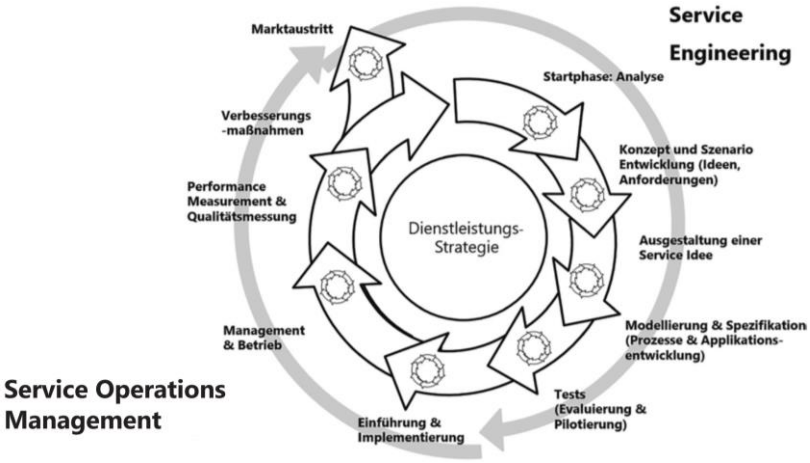
→ Zielkonflikt zwischen effizientem Ressourceneinsatz und Kundenwert

Beispiele für Operations- und Kundenperspektive

Service	Betriebliche Perspektive			Kundenperspektive		
Beispiel	**Inputs**	**Prozess**	**Output**	**Erlebnis**	**«Produkte»**	**Vorteile**
Operation	Hausarzt, Krankenschwestern, Chirurg, Bett, Operationssaal	Diagnose, Operation, Nachsorge	Hüftgelenkersatz	Einfache, einfühlsame und schmerzfreie Behandlung	Eine funktionierende Hüfte	Größere Mobilität
Bildung	Vorlesungen, Bibliothek, Computer, Seminarräume	Zeitplanung, Vorlesungen, Prüfungen, Benotung	Informationen, Dia-Pakete, Abschlüsse	Unvergessliche und nützliche Dozenten/Seminare	Wissen, Selbstvertrauen und Fähigkeiten	Bessere Berufsaussichten/Fähigkeiten
Beratung	Berater, Informationen, Fähigkeiten, Wissen	Datenerfassung und -analyse	Präsentationen, Berichte	Hilfreiche und zeitnahe Diskussionen und Beratung	Lösungen	Geringere Kosten und größerer wirtschaftlicher Erfolg

Beispielhafter Gesamtprozess für die Entwicklung und den Betrieb von Dienstleistungen

- Aufgabe von SOM: Von Einführung & Implementierung bis Marktaustritt, der Rest ist Service Engineering.
- Welche Phasen lassen sich im SOM unterscheiden? Einführung & Implementierung, Management & Betrieb, Performancemessung & Verbesserung



Performance Management als Teilgebiet des SOM: Data Envelopment Analysis (DEA)

Einführung und Definition

- Ziel: Vergleich der Effizienz verschiedener Serviceeinheiten, die identische Dienstleistungen bereitstellen
- Explizite Berücksichtigung des Verbrauchs verschiedener Ressourcen (Inputs) und verschiedener Leistungskriterien (Outputs)
- Lineare Programmierung
  - Maximierung der Effizienz einer Serviceeinheit (Verhältnis von Outputs zu Inputs)
  - Vergleich Effizienz Serviceeinheit mit anderen Serviceeinheiten, die gleiche Dienstleistung bereitstellen
- Ziele:
  - Identifikation von Serviceeinheiten mit der höchsten Effizienz (= 100 %): relativ effiziente Einheiten
  - Identifikation von Serviceeinheiten mit niedrigerer Effizienz (< 100 %): ineffiziente Einheiten
  - Messung der relativen Effizienz durch Vergleich von ineffizienter zu relativ effizienten Einheiten

Beispiel Filialen einer Restaurantkette

- Verglichen werden 6 Filialen.
- Inputs: Arbeitsstunden und Materialeinsatz in €
- Für jeweils 100 Mahlzeiten wird in jeder Filiale eine unterschiedliche Kombination von Material und Arbeitsstunden benötigt. Die Qualität ist bei allen Filialen gleich.

Filiale	Verkaufte Mahlzeiten	Arbeitsstunden	Materialeinsatz in €
1	100	2	200
2	100	4	150
3	100	4	100
4	100	6	100
5	100	8	80
6	100	10	50

Technologie

- Technologie = Menge der technisch möglichen Input-Output-Kombinationen
- Annahme Konvexität: beliebige Serviceeinheiten derselben Technologie können konvex kombiniert werden.
- $(x', y')\ und\ (x'', y'') \in T\ und\ \lambda \in [0,1] \rightarrow \lambda(x', y') + (1-\lambda)(x'', y'') \in T$

Effizienter Rand

- Eine Serviceeinheit ist genau dann effizient, wenn es nicht möglich ist, einen Input bei gegebenem Output zu reduzieren oder einen Output bei gegebenem Input zu erhöhen.
- Der effiziente Rand eines Technologieraums stellt die Menge aller (auf Basis der Annahmen möglichen) effizienten Serviceeinheiten dar.
- Effizienter Rand = Umschlag, der die nicht effizienten Einheiten umhüllt („data envelopment“)

Operationalisierung der Effizienz einer Serviceeinheit

- Serviceeinheit e
- M Outputs (O_{1e} bis O_{Me})
- N Inputs (I_{1e} bis I_{Ne})
- Kernidee: Effizienz (E_e) bedeutet Verhältnis von Outputs zu Inputs
- Nutzung von Bedeutungsgewichten für die Outputs (u_1 bis u_M) und die Inputs (v_1 bis v_N)
- $E_e = \frac{u_1 O_{1e} + u_2 O_{2e} + \cdots + u_M O_{Me}}{v_1 I_{1e} + v_2 I_{2e} + \cdots + v_N I_{Ne}} \rightarrow \max E_e = \frac{u_1 O_{1e} + u_2 O_{2e} + \cdots + u_M O_{Me}}{v_1 I_{1e} + v_2 I_{2e} + \cdots + v_N I_{Ne}}$
- Zielfunktion der linearen Programmierung: Maximierung der Effizienz der Serviceeinheit e
- Optimierung konfiguriert die Bedeutungsgewichte für die Outputs (u_1 bis u_M) und die Inputs (v_1 bis v_N).
- Für jede Serviceeinheit e wird getrennt eine Optimierung durchgeführt.
- **Nebenbedingungen**: $\frac{u_1 O_{1k} + u_2 O_{2k} + \cdots + u_M O_{Mk}}{v_1 I_{1k} + v_2 I_{2k} + \cdots + v_N I_{Nk}} \leq 1, k = 1,2,\ldots,K$, ZF ist mit Index e und NB mit Index k
- Für alle Serviceeinheiten ($k = 1,2,\ldots,K$) darf die Effizienz 1 nicht überschreiten (nicht über 100 % liegen).
- **Nebenbedingungen:** $u_j \geq 0, j = 1,2,\ldots,M$ und $v_i \geq 0, i = 1,2,\ldots,N$
- Bedeutungsgewichte für Outputs (u_1 bis u_M) und die Inputs (v_1 bis v_N) müssen nicht-negativ (≥ 0) sein.

- Umformen in LP: $\max E_e = u_1 O_{1e} + u_2 O_{2e} + \cdots + u_M O_{Me}$
- LP NB: $v_1 I_{1e} + v_2 I_{2e} + \cdots + v_N I_{Ne} = 1, \quad u_1 O_{1k} + u_2 O_{2k} + \cdots + u_M O_{Mk} - (v_1 I_{1k} + v_2 I_{2k} + \cdots + v_N I_{Nk}) \leq 0$ $k = 1,2,\ldots,K, \quad u_j \geq 0, j = 1,2,\ldots,M$ und $v_i \geq 0, i = 1,2,\ldots,N$
- **Beispiel:** Effizienzberechnung der Filiale 1
- Bei Filiale 2 ändert sich nur eine NB, ZF bleibt hier gleich (da Output bei allen gleich) $v_1 2 + v_2 200 = 1$ wird zu $v_1 4 + v_2 150 = 1$
- **Interpretation Ergebnisse Filiale 4:** S_4 muss sich um $1 - 0.889 = 0.111$ (11.1 %) verbessern, um eine relative Effizienz zu erreichen.
- Für jede reduzierte Arbeitsstunde erhöht sich die Effizienz um 0.0555.

Service-einheit	Effizienz (E)	Bedeutungs-gewicht Arbeitsstunde(v_1)	Bedeutungs-gewicht Materialeinsatz in € (v_2)
S_1	1.000	.1667	.0033
S_2	0.857	.1428	.0028
S_3	1.000	.0625	.0075
S_4	0.889	.0555	.0067
S_5	0.901	.0568	.0068
S_6	1.000	.0000	.0200

$$\max E(S_1) = u_1 100$$
$$u_1 100 - v_1 2 - v_2 200 \leq 0$$
$$u_1 100 - v_1 4 - v_2 150 \leq 0$$
$$u_1 100 - v_1 4 - v_2 100 \leq 0$$
$$u_1 100 - v_1 6 - v_2 100 \leq 0$$
$$u_1 100 - v_1 8 - v_2 80 \leq 0$$
$$u_1 100 - v_1 10 - v_2 50 \leq 0$$
$$v_1 2 + v_2 200 = 1$$
$$u_1, v_1, v_2 \geq 0$$

- Um effizient zu sein, müssten $\frac{0.111}{0.0555} = 2$ Arbeitsstunden eingespart werden, falls alles andere konstant bleibt
- Für jeden reduzierten Euro Materialeinsatz erhöht sich die Effizienz um 0.0067. Um effizient zu werden, müssten $\frac{0.111}{0.0067} = 16.57$ € eingespart werden (falls alles andere konstant bleibt).

Revenue Management

- Revenue Management umfasst Techniken, um **begrenzten Ressourcen** verschiedenen Arten von Kunden zu **unterschiedlichen Preisen zuzuweisen**, um den **Unternehmensumsatz** zu **maximieren**.
  ➔ Science of squeezing every possible dollar from customers
- Alternative Bezeichnungen: Ertragsmanagement, Preisoptimierung oder Nachfragemanagement
- Revenue Management ist am **effektivsten** wenn
  - das Produkt nur begrenzt haltbar ist und im Vorhinein verkauft werden kann
  - die Kapazität limitiert ist und nur mit viel Aufwand erhöht werden kann
  - der Markt bzw. die Kunden in Segmente eingeteilt werden können
  - die variablen Kosten gering sind
  - Preise angepasst werden können
  - die Nachfrage über die Zeit schwankt und zum Zeitpunkt der Entscheidung unbekannt ist

Kapazitätsmanagement als Balanceakt

Unzureichende Ressourcennutzung

- Teure Ressourcen, die keine Einnahmen erzielen, führen zu schlechten finanziellen Ergebnissen.
- Kunden sind misstrauisch gegenüber Diensten, die scheinbar nicht ausgelastet sind.
- Servicemitarbeiter können demotiviert werden, wenn die Unterauslastung anhält.

Überlastete Ressourcen

- Ein plötzlicher Ansturm von Kunden in ein Geschäft führt dazu, dass sich die Wartezeiten erhöhen.
- Mitarbeiter, die ständig überlastet sind, machen mehr Fehler und gehen.
- Mitarbeiter müssen Aufgaben ausführen, mit denen sie nicht vertraut sind.

Drei Strategien für das Kapazitätsmanagement

Level capacity/Niveaukapazität

- In diesem Fall werden knappe oder teure Ressourcen auf einem konstanten Niveau gehalten, und die Organisation muss die Folgeprobleme für die Kundenzufriedenheit und die operative Servicequalität bewältigen.
- Z.B Überbuchungsmanagement

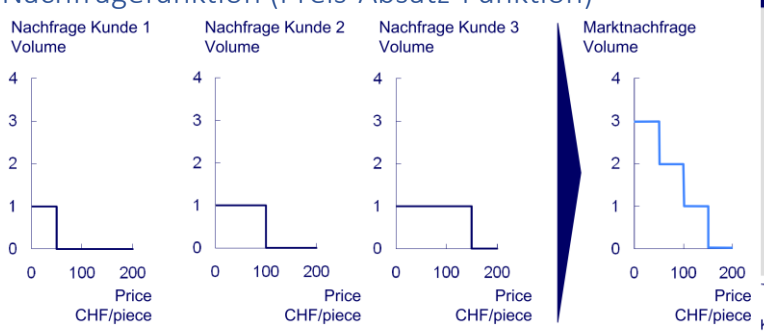
Chase capacity/Verfolgungskapa.

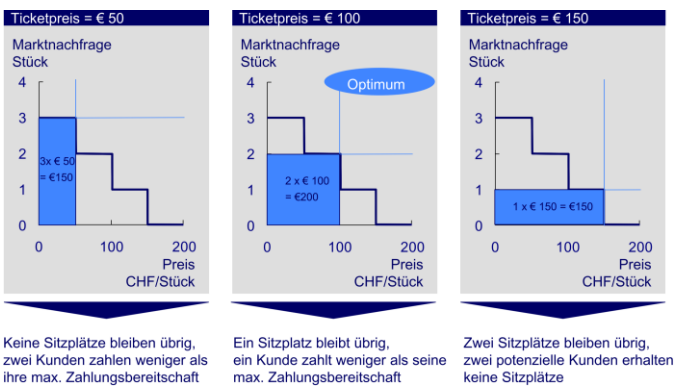
- Serviceorganisation versucht, Angebot und Nachfrage aufeinander abzustimmen, indem sie Flexibilität in Betrieb einbaut. Oberstes Ziel: eine hohe Serviceverfügbarkeit oder schnelle Reaktion auf die effizienteste Weise zu gewährleisten.
- Z.B. Workforce Scheduling

Demand management/Nachfrage

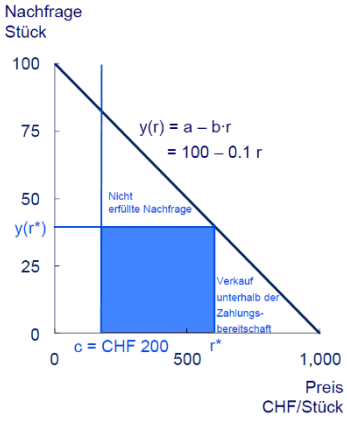
- Anstatt die Kapazität des Servicebetriebs zu ändern, beeinflusst Organisation das Nachfrageprofil, um die Belastung der Ressourcen zu "glätten".
- Z.B. Preissegmentierung

Nachfragefunktion (Preis-Absatz-Funktion)





Kontinuierliche Nachfragefunktion mit einem Segment



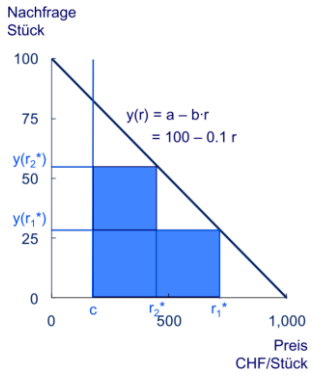
Notation

- r = Preis
- a = Nachfrage bei Preis = 0 (Y-Achsenabschnitt)
- b = negative Steigung der Nachfragefunktion
- c = variable Kosten
- Steigung: $b = -\frac{\Delta y}{\Delta x} = -\frac{y_1 - y_2}{x_1 - x_2}$, (x_1, y_1) und (x_2, y_2) auf Nachfragekurve
- Prohibitivpreis $r_0 = x_1 + \frac{y_1}{b}$, $(x_1,\ y_1)$ auf Nachfragekurve
- Y-Achsenabschnitt: $a = b \cdot r_0$
- Lineare Nachfragefunktion: $y(r) = a - b \cdot r$
- Bestimme den optimalen Preis r^*, so dass der Gewinn maximiert wird $r^* = \frac{a+bc}{2b}$
- Maximaler Gewinn: $\Pi(r^*) = \frac{(a-bc)^2}{4b}$

Beispiel Preisoptimierung mit einem Segment

- Sie schätzen, dass bei einem Preis von CHF 2 etwas 9200 Kunden und bei einem Preis von CHF 10 etwa 6000 Kunden gewinnen können. Berechnen Sie die Nachfragekurve.
- Steigung: $b = -\frac{\Delta y}{\Delta x} = -\frac{y_1 - y_2}{x_1 - x_2} = -\frac{9200-6000}{2-10} = -\frac{3200}{-8} = 400$
- Prohibitivpreis $r_0 = x_1 + \frac{y_1}{b} = 2 + \frac{9200}{400} = 2 + 23 = 25$
- Y-Achsenabschnitt: $a = b \cdot r_0 = 400 \cdot 25 = 10000$ → Funktion $y(r) = a - b \cdot r = 10000 - 400r$
- Optimaler Preis: $r^* = \frac{a+bc}{2b} = \frac{10000+400 \cdot 10}{2 \cdot 400} = 17.5$, Max. Gewinn: $\Pi(r^*) = \frac{(a-bc)^2}{4b} = \frac{(10000-400 \cdot 10)^2}{4 \cdot 400} = 22'500$

Preisoptimierung: Kontinuierliche Nachfragefunktion mit 2 Segmenten

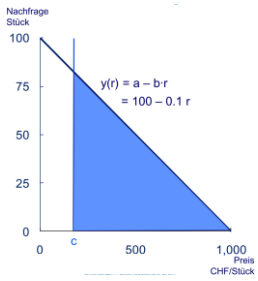


- Gibt nun zwei optimale Preise, je einen pro Segment: $r_1^*,\ r_2^*$
- $\Pi(r_1, r_2) = (r_1 - c)y(r_1) + (r_2 - c)\big(y(r_2) - y(r_1)\big) =$
  $= ar_1 - br_1^2 + bcr_2 - br_2^2 + br_1r_2 - ac$
- $\frac{d\Pi(r_1,r_2)}{dr_1} = 0 \rightarrow r_1^* = \frac{2a+bc}{3b}$, $\frac{d\Pi(r_1,r_2)}{dr_2} = 0 \rightarrow r_2^* = \frac{a+2bc}{3b}$
- $\Pi(r_1^*, r_2^*) = \frac{(a-bc)^2}{3b}$

Beispiel Preisoptimierung mit zwei Segmenten

- Sie wissen, dass beim Preis von CHF 0 etwas 10000 Kunden den Dienst nutzen würden und dass die Nachfragekurve eine negative Steigung von 400 Kunden/CHF hat (variable Kosten: CHF 10). Bestimmen Sie die optimalen Preise und den Gewinn bei der Einführung von zwei Segmenten. → $(a = 10000,\ b = 400,\ c = 10)$
- $r_1^* = \frac{2a+bc}{3b} = \frac{2 \cdot 10000+400 \cdot 10}{3 \cdot 400} = 20$, $r_2^* = \frac{a+2bc}{3b} = \frac{10000+2 \cdot 400 \cdot 10}{3 \cdot 400} = 15$
- $\Pi(r_1^*, r_2^*) = \frac{(a-bc)^2}{3b} = \frac{(10000-400 \cdot 10)^2}{3 \cdot 400} = 30'000$

Preisoptimierung: Kontinuierliche Nachfragefunktion mit individuellen Segmenten



- Ziel: Bestimme den Gewinn unter individueller Preissetzung Lösung
- Preis für den die Nachfrage auf 0 sinkt: $y(r_0) = a - br_0 = 0,\ r_0 = \frac{a}{b}$
- Maximaler Gewinn: $\Pi^* = \frac{1}{2}(r_0 - c)y(c) = \frac{(a-bc)^2}{2b}$

Beispiel Preisoptimierung mit individuellen Segmenten

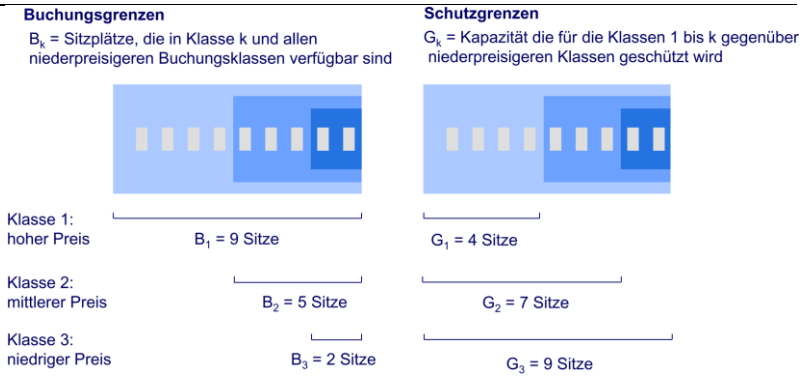
- Sie wissen, dass beim Preis von CHF 0 etwas 10000 Kunden den Dienst nutzen würden und dass die Nachfragekurve eine negative Steigung von 400 Kunden/CHF hat. Bestimmen Sie den theoretisch erzielbaren Gewinn bei vollständiger Preisdifferenzierung mit individuellen Segmenten.
- Maximaler Gewinn: $\Pi^* = \frac{1}{2}(r_0 - c)y(c) = \frac{(a-bc)^2}{2b} = \frac{(10000-400 \cdot 10)^2}{2 \cdot 400} = 45'000$

Segmentierung

- Segmentierung = Mechanismus, um sicherzustellen, dass Kunden mit hoher Zahlungsbereitschaft nicht ins Niedrigpreissegment einbezogen werden.
- Segmentierungen aufgrund:
  - Zeit: Zeit zwischen dem Kauf und dem Verbrauch, Zeitpunkt des Verbrauchs oder Lieferzeit
  - Ort: Land/Stadt des Konsums oder Ort des Konsums
  - Flexibilität: Änderungen/ Stornierungen oder Ort
  - Gruppen: Alter, Status oder Mitgliedschaften
  - Varianten: Aufwertung einer Basisvariante oder Versionen

Buchungssteuerung – Geschachtelte Politik

- Rechnerischer Zusammenhang von Buchungsgrenzen und Schutzgrenzen:
  $G_k = \begin{cases} B_1 - B_{k+1} & k = 1,2,\dots,K-1 \\ B_1 (:= C) & k = K \end{cases}$
- Implementierung:
  - Klasse k Nachfrage y_k tritt auf
  - Erfüllte Nachfrage: $\min(y_k, B_k)$
  - Verkleinere alle Buchungsgrenzen um $\min(y_k, B_k)$ bis sie 0 erreichen.
- Alternativer Ansatz Schutzgrenzen:
  - $G_k = \begin{cases} B_1 - B_{k+1} & k = 1,2,\dots,K-1 \\ B_1 (:= C) & k = K \end{cases}$
  - G_k = Kapazität die für die Klasse 1 bis k gegenüber niederpreisigeren Klassen geschützt wird.
  - B_k = Sitzplätze, die in Klasse k und allen niederpreisigeren Buchungsklassen verfügbar sind.



Buchungsgrenzen – Zwei Beispiele ungeordnete Nachfrage

Nachfrage	B.grenzen B_1	B_2	B_3	Akzeptiert 1	2	3	Abgelehnt 1	2	3
	9	5	2	0	0	0	0	0	0
1x Klasse 2	8	4	1	0	1	0	0	0	0
1x Klasse 1	7	3	0	1	1	0	0	0	0
3x Klasse 1	4	0	0	4	1	0	0	0	0
1x Klasse 3	4	0	0	4	1	0	0	0	1
2x Klasse 1	2	0	0	6	1	0	0	0	1

Nachfrage	B.grenzen B_1	B_2	B_3	Akzeptiert 1	2	3	Abgelehnt 1	2	3
	11	7	3	0	0	0	0	0	0
2x class 1	9	5	1	2	0	0	0	0	0
5x class 3	8	4	0	2	0	1	0	0	4
2x class 1	6	2	0	4	0	1	0	0	4
3x class 2	4	0	0	4	2	1	0	1	4
4x class 3	4	0	0	4	2	1	0	1	8

Bestimmung der optimalen Buchungsgrenzen

Optimierungsproblem

- **Gegeben:**
  - C: Kapazität
  - r_1 und r_2: Preise für 2 Klassen ($r_1 > r_2$)
  - Y_1: stochastische Nachfrage für Premiumklasse mit Verteilungsfunktion F_1
  - B_1: Buchungsgrenze für Klasse 1 ($= C$)
- **Gesucht:**
  - B_2: Buchungsgrenze für niedrigpreisige Klasse ($= C - G_1$)
- **Annahmen:**
  - Nachfrage der niedrigpreisigen Kunden tritt vor der Nachfrage der hochpreisigen Kunden auf
  - Nachfrage der niedrigpreisigen Kunden übersteigt die Kapazität C
- **Vorgehen:**
  - B_2 festlegen
  - Tickets für Klasse 2 verkaufen
  - Tickets für Klasse 1 verkaufen

Standardnormalverteilung

- $F^{-1}_{N(\mu,\sigma)}(x) = \mu + \sigma F^{-1}_{N(0,1)}(x)$
- Wert $F(z)$ suchen, der am nächsten bei der gesuchten Wahrscheinlichkeit liegt. Dann den Wert in der z Spalte übernehmen.
- Eine Zufallsvariable ist normalverteilt mit Mittelwert 3 und Standardabweichung 5 $F_{N(3,25)}$. Bei welchem Wert ist die Wahrscheinlichkeit, dass diese Zufallsvariable kleiner oder gleich diesem Wert ist, gleich 30 %?
- $F^{-1}_{N(3,25)}(x) = 3 + 5 \cdot F^{-1}_{N(0,1)}(0.3) = 3 + 5 \cdot (-0.5) = 0.5$

z	$f_{N(0,1)}(z)$	$F_{N(0,1)}(z)$	z	$f_{N(0,1)}(z)$	F(z)	z	$f_{N(0,1)}(z)$	F(z)	z	$f_{N(0,1)}(z)$	F(z)
-3,2	0,0024	0,0007	-1,6	0,1109	0,0548	0,0	0,3989	0,5000	1,6	0,1109	0,9452
-3,1	0,0033	0,0010	-1,5	0,1295	0,0668	0,1	0,3970	0,5398	1,7	0,0940	0,9554
-3,0	0,0044	0,0013	-1,4	0,1497	0,0808	0,2	0,3910	0,5793	1,8	0,0790	0,9641
-2,9	0,0060	0,0019	-1,3	0,1714	0,0968	0,3	0,3814	0,6179	1,9	0,0656	0,9713
-2,8	0,0079	0,0026	-1,2	0,1942	0,1151	0,4	0,3683	0,6554	2,0	0,0540	0,9772
-2,7	0,0104	0,0035	-1,1	0,2179	0,1357	0,5	0,3521	0,6915	2,1	0,0440	0,9821
-2,6	0,0136	0,0047	-1,0	0,2420	0,1587	0,6	0,3332	0,7257	2,2	0,0355	0,9861
-2,5	0,0175	0,0062	-0,9	0,2661	0,1841	0,7	0,3123	0,7580	2,3	0,0283	0,9893
-2,4	0,0224	0,0082	-0,8	0,2897	0,2119	0,8	0,2897	0,7881	2,4	0,0224	0,9918
-2,3	0,0283	0,0107	-0,7	0,3123	0,2420	0,9	0,2661	0,8159	2,5	0,0175	0,9938
-2,2	0,0355	0,0139	-0,6	0,3332	0,2743	1,0	0,2420	0,8413	2,6	0,0136	0,9953
-2,1	0,0440	0,0179	-0,5	0,3521	0,3085	1,1	0,2179	0,8643	2,7	0,0104	0,9965
-2,0	0,0540	0,0228	-0,4	0,3683	0,3446	1,2	0,1942	0,8849	2,8	0,0079	0,9974
-1,9	0,0656	0,0287	-0,3	0,3814	0,3821	1,3	0,1714	0,9032	2,9	0,0060	0,9981
-1,8	0,0790	0,0359	-0,2	0,3910	0,4207	1,4	0,1497	0,9192	3,0	0,0044	0,9987
-1,7	0,0656	0,0446	-0,1	0,0656	0,4602	1,5	0,0656	0,9332	3,1	0,0656	0,9990

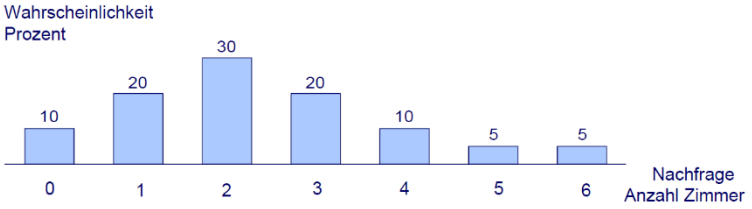
Zusammenfassung der optimalen Buchungsgrenzen

- Wenn wir B_2 auf $B_2 + 1$ erhöhen, ändert sich der erwartet Umsatz um:
  $E[\Delta U] = r_2 F_1(C - B_2) + (r_2 - r_1)\big(1 - F_1(C - B_2)\big) \Leftrightarrow E[\Delta U] = r_2 - r_1\big(1 - F_1(C - B_2)\big)$
- Wir erhöhen B_2 so lange bis die Umsatzänderung nicht mehr steigt: $E[\Delta U] = r_2 - r_1\big(1 - F_1(C - B_2)\big) = 0$
  $\Leftrightarrow F_1(C - B_2) = 1 - \frac{r_2}{r_1} \Leftrightarrow C - B_2 = F_1^{-1}\left(1 - \frac{r_2}{r_1}\right) \Leftrightarrow B_2^* = C - F_1^{-1}\left(1 - \frac{r_2}{r_1}\right)$
- $F(G_1) = \sum_{y=0}^{G_1} f(y)$
- Optimale Schutzgrenze: $G_1^* = C - B_2^* = F_1^{-1}\left(1 - \frac{r_2}{r_1}\right)$
- **Littlewoods Regel:** Die Wahrscheinlichkeit, dass die Nachfrage der Buchungsklasse 1 höher als die Schutzgrenze der Buchungsklasse 1 ist, sollte mit dem Verhältnis der Preise $\frac{r_2}{r_1}$ übereinstimmen.

Beispiel Optimierung Reservierung (diskrete Nachfrage)

Frage: Wieviele Zimmer sollten für die hochpreisigen Kunden reserviert werden? Annahmen:

- Kapazität $C = 100$ Zimmer für eine Nacht
- Keine Stornierungen, Überbuchungen etc. sind erlaubt
- Zwei Kundenklassen: Preise sind vorher festgelegt
  - Klasse 1: Business Kunden, Preis $r_1 = 159$ CHF
  - Klasse 2: Niedrigpreisige Kunden (Touristen) zahlen $r_2 = 49$ CHF
- Nachfrage der niedrigpreisigen Kunden tritt vor der Nachfrage der hochpreisigen Kunden auf
- Nachfrage der niedrigpreisigen Kunden übersteigt die Kapazität C
- Nachfrageverteilung der hochpreisigen Kunden ist bekannt



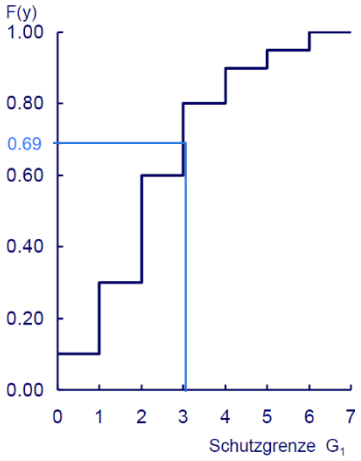
Lösung

- $r_1 = 159, r_2 = 49$
- $1 - \frac{r_2}{r_1} = 1 - \frac{49}{159} = 0.69$
- $G_1^* = 3$

Was passiert wenn

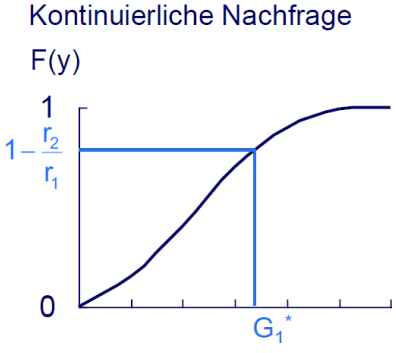
- r_1 steigt → die Schutzgrenze G_1 steigt
- r_2 steigt → die Schutzgrenze G_1 sinkt

y	f(y)	F(y)
0	0.10	0.10
1	0.20	0.30
2	0.30	0.60
3	0.20	0.80
4	0.10	0.90
5	0.05	0.95
6	0.05	1.00



Kontinuierliche Nachfrage

- Der Falle der kontinuierlichen Nachfrage kann ähnlich bearbeitet werde wie der Fall der diskreten Nachfrage
- $F(G_1) = \int_{y=0}^{G_1} f(y)\, dy$



Beispiel Kontinuierliche Nachfrage

- Kapazität $C = 160$ Zimmer für eine Nacht
- zwei Kundenklassen: Preise sind vorher festgelegt
  - Klasse 1: Business Kunden, Preis $r_1 = 300$
  - Klasse 2: niedrigpreisige Kunden (Touristen) zahlen $r_2 = 100$
- Nachfrage Klasse 1 ist normalverteilt mit einem Mittelwert von 35 und einer Standardabweichung von 10
- Alle vorherigen Annahmen gelten weiterhin

Lösung

- $1 - \frac{r_2}{r_1} = 1 - \frac{100}{300} = 0.67$
- $F(z) = F(0.67)$ Wert in Tabelle suchen, der am nächsten bei 0.67 liegt. Dann z-Wert nehmen, $z = 0.4$
- $G_1^* = \mu_1 + z \cdot \sigma_1 = 35 + 0.4 \cdot 10 = 39$
- Wir reservieren 39 Zimmer für die Kundenklasse 1 und verkaufen $160 - 39 = 121$ Zimmer an die Touristen.

EMSR und Überbuchungsmanagement

Kapazitätsoptimierung mit K Buchungsklassen Expected Marginal Seat Revenue (EMSR)

Annahmen

- K Buchungsklassen
- Einzige Kapazitätsbedingung: C
- Klassen sind bestimmt sodass gilt $r_1 > r_2 > \cdots > r_K$
- Nachfrage wird erfüllt nach der sequenziellen Reihenfolge der Buchungsklassen
- Verschachtelte Schutzgrenzen
- Auf jeder Ebene wird Zwei Klassen Problem gelöst

Lösung durch Verwendung von Heuristiken

- **EMSR-a**
  - Expected marginal seat revenue - Version a
  - Aggregierte **Schutzgrenzen**
- **EMSR-b**
  - Expected marginal seat revenue - Version b
  - Aggregierte **Nachfrage**

Exptected marginal seat revenue – Version a → Aggregierte Schutzgrenzen

1. Berechne Schutzgrenzen $G_{k+1,l}$, welche die Klasse l vor der Klasse $k+1$ schützt mit Hilfe der Littlewood's Regel ($k+1 > l$): $G_{k+1,l} = F_l^{-1}\left(1 - \frac{r_{k+1}}{r_l}\right)$
2. Aggregiere die Schutzgrenzen $G_k = \sum_{l=1}^{k} G_{k+1,l}$ $G_{k+1,l}$ → Paarweise Schutzgrenzen
3. Berechne die Buchungsgrenzen $B_k = C - G_{k-1}$

Beispiel: Drei Buchungsklassen, $C = 100$

Normalverteilung in Standardnormalverteilung umwandeln: $F_{N(3,25)}^{-1}(x) = 3 + 5 \cdot F_{N(0,1)}^{-1}(0.3) = 3 + 5 \cdot (-0.5) = 0.5$

Klasse k	Preis r_k	Mittelwert μ_k	Stdabw. σ_k	F
1	CHF 150	30	10	$F_1 = F_N(30{,}10^2)$
2	CHF 100	50	20	$F_2 = F_N(50{,}20^2)$
3	CHF 50	100	20	$F_3 = F_N(100{,}20^2)$

Buchungsgrenze Klasse 3

$G_{3,1} = F_1^{-1}\left(1 - \frac{r_3}{r_1}\right) = F_1^{-1}\left(1 - \frac{50}{150}\right) = 30 + 10 \cdot 0.4 = 34$

$G_{3,2} = F_2^{-1}\left(1 - \frac{r_3}{r_2}\right) = F_2^{-1}\left(1 - \frac{50}{100}\right) = 50 + 20 \cdot 0 = 50$

$G_2 = G_{3,1} + G_{3,2} = 34 + 50 = 84$

$B_3 = C - G_2 = 100 - 84 = 16$

Buchungsgrenze Klasse 2

$G_{2,1} = F_1^{-1}\left(1 - \frac{r_2}{r_1}\right) = F_1^{-1}\left(1 - \frac{100}{150}\right) =$

$= 30 + 10 \cdot -0.4 = 26$

$G_1 = G_{2,1} = 26$

$B_2 = C - G_1 = 100 - 26 = 74$

Buchungsgrenze Klasse 1

$B_1 = C = 100$

Exptected marginal seat revenue – Version b → Aggregierte Nachfrage

1. Berechne die aggregierte durchschnittliche Nachfrage $\bar{\mu}_i$ und die Standardabweichung $\bar{\sigma}_i$ für Klassen 1, … , i:

   $\bar{\mu}_i = \sum_{k=1}^{i} \mu_k, \qquad \bar{\sigma}_i = \sqrt{\sum_{k=1}^{i} \sigma_k^2}$
2. Berechne den aggregierten Preis als gewichteten Durchschnitt $\bar{r}_i = \frac{\sum_{k=1}^{i} r_k \cdot \mu_k}{\sum_{k=1}^{i} \mu_k}$
3. Berechne Schutzgrenzen für aggregierte Nachfrage mit Littlewood's Regel: $G_i = F_{\bar{\mu}_i,\bar{\sigma}_i}^{-1}\left(1 - \frac{r_{i+1}}{\bar{r}_i}\right)$

Beispiel: Drei Buchungsklassen, $C = 100$, gleiche Zahlen wie oben

Buchungsgrenze Klasse 3

$\bar{\mu}_2 = \mu_1 + \mu_2 = 30 + 50 = 80$

$\bar{\sigma}_2 = \sqrt{10^2 + 20^2} = 22.36$

$\bar{r}_2 = \frac{r_1 \cdot \mu_1 + r_2 \cdot \mu_2}{\mu_1 + \mu_2} = \frac{150 \cdot 30 + 100 \cdot 50}{30 + 50} = 118.75$

$G_2 = F_{\bar{\mu}_2,\bar{\sigma}_2}^{-1}\left(1 - \frac{r_3}{\bar{r}_2}\right) = F_{80,22.36}^{-1}\left(1 - \frac{50}{118.75}\right) = 80 + 22.36 \cdot 0.195$

$= 84.36$

$B_3 = C - G_2 = 100 - 84 = 16$

Buchungsgrenze Klasse 2

$\bar{\mu}_1 = \mu_1 = 30$

$\bar{\sigma}_1 = \sigma_1 = 10$

$\bar{r}_1 = r_1 = 150$

$G_1 = F_{\bar{\mu}_1,\bar{\sigma}_1}^{-1}\left(1 - \frac{r_2}{\bar{r}_1}\right) = F_{30,10}^{-1}\left(1 - \frac{100}{50}\right) =$

$= 30 + 10 \cdot -0.43 = 25.7$

$B_2 = C - G_1 = 100 - 26 = 74$

Buchungsgrenze Klasse 1

$B_1 = C = 100$

Zusammenfassung

- Eine optimale Lösung erfordert eine dynamische Programmierung
- Beide Ansätze sind Heuristiken
- EMSR-a aggregiert die Schutzgrenzen während EMSR b die Nachfrage aggregiert
- Die Berechnung des gewichteten Durchschnittspreises ist eine **kritische Annahme**
- In der Praxis ist EMSR-b weiterverbreitet
- Experimentelle Studien zeigen, dass keine der beiden Methoden die andere dominiert

Überbuchungsmanagement

Überbuchung ist besonders wichtig, wenn die Anzahl an No Shows hoch ist und die Stornierung keine Kosten nach sich zieht. Überbuchung wird angewendet bei

- Flugzeugsitzplätzen, die vor der Reise reserviert werden
- Autovermietung, bei der die Autos vor dem Tag der Vermietung reserviert werden
- Hotelzimmern oder Konzerttickets die im Vorhinein verkauft werden

Vorgehen wenn Kunden abgewiesen werden müssen:

- Hotels: alternative Unterkunft, Entschädigung
- Fluglinien: alternative Flüge
- Autovermietung: Hochstufung
- Werbung: alternative Sendezeiten, Preisnachlass

Minimiere die No Show Anzahl

- verlange frühzeitige Zahlung
- erhöhe Stornierungsgebühren
- erhöhe Wechselgebühren

Gründe für die Stornierung

- Doppelbuchung
- Verpasste Verbindungen
- Andere Gründe

Modell für das Überbuchungsmanagement

- C: Kapazitätbegrenzung
- B: Buchungsgrenze
- r: Preis für eine verkaufte Einheit
- p: Strafkosten

Weitere Annahmen:

- Jeder erschienene Kunde zahlt r, auch wenn keine Kapazität übrigbleibt
- Anzahl von No Shows ist X
- Dichtefunktion der No Shows ist $h(x)$ und Verteilungsfunktion der No Shows ist $H(x)$
- Dichtefunktion der Nachfrage Y ist $f(y)$ und die Verteilungsfunktion $F(y)$

Mögliche Fälle:

	Fall	Umsatzänderung	Beschreibung
•	$Y \leq B$	$-$	kein Mehrumsatz, da die Nachfrage unter der Buchungsgrenze liegt
•	$X \leq B - C$	$r - p$	weniger No-Shows als erwartet, Strafkosten p fallen an.
•	$X > B - C$	r	mehr No-Shows als überbuchte Plätze

Wir erhöhen B bis die Strafkosten mit dem Umsatz übereinstimmen: $E[\Delta U] = r - pH(B - C) = 0$

Optimale Buchungsgrenze: $B^* = C + H^{-1}\left(\frac{r}{p}\right)$

Optimale Buchungsgrenze bei normalverteilen Nachfrage: $B^* = C + \mu + \sigma \cdot F^{-1}_{(0,1)}, \left(\frac{r}{p}\right)$

Beispiel

- Kapazität 100
- Nachfrage übersteigt eindeutig die Kapazität
- Umsatz für ein vergebenes Zimmer beträgt 90 €
- Kosten, die auftreten, wenn der Kunde in ein anderes Hotel umgebucht werden muss und für zukünftige freie Unterbringung betragen 390 €
- No-Shows folgen einer Normalverteilung mit $\mu = 11.73$ und $\sigma = 4.74$

Lösung:

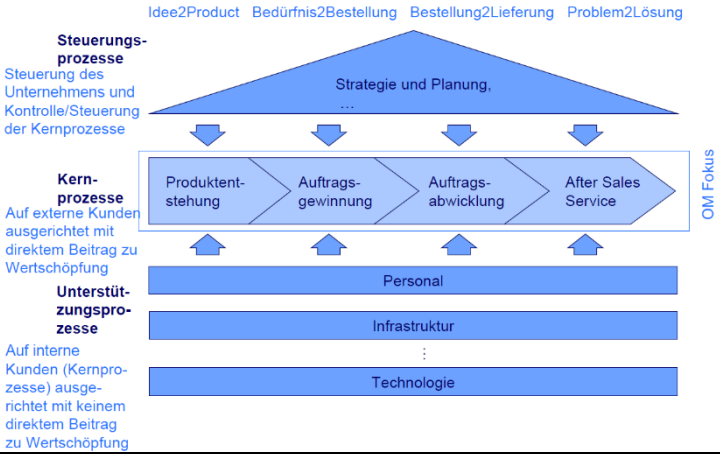
- $\frac{r}{p} = \frac{90}{390} = 0.23$ $z = -0.7$ $B^* = C + \mu_X + \sigma_X \cdot F^{-1}_{(0,1)}\left(\frac{r}{p}\right) = 100 + 11.73 + 4.47 \cdot -0.7 = 108$
- Überbuchung beträgt: $B - C = 108 - 100 = 8$

Definition Prozess

Eine Folge von Aktivitäten zur Erstellung einer Leistung, mit einem Anfang, einem Ende und einem Ziel.

Beispiele Prozesse	Anfang	Ende	Ziel
Telefonische Auftragsannahme PC-Hersteller	Anruf eines Kunden	Gewonnener oder verlorener Auftrag	Umsetzung verbal geäußerter Kundenwünsche in Aufträge
Herstellung PC	Freigabe eines Fertigungsauftrags	Verpackung PC	Umsetzung eines Fertigungsauftrags in ein versandfähiges Produkt

Kategorisierung von Geschäftsprozessen



Messung der Prozessleistung

- **Kapazität:** Auftragsannahme, Produktion
- **Zeit:** Durchlaufzeit, Lieferzeit
- **Kosten:** Personal, Material
- **Qualität:** Ausschuss, Zuverlässigkeit
- **Flexibilität:** Mengenflexibilität, Produktflexibilität

Zielkonflikte im Prozessmanagement

Beispiel Callcenter (siehe FAP): Durchschnittliche Wartezeit in Minuten vs. Anzahl Bedienplätze (Kapazität)

➔ Warum ist Geschäftsprozessmanagement wichtig? Zielkonflikt zwischen Kapazität und Qualität

Process Mining

Process Mining: Schließt die Lücke zwischen Prozessmodellierung und Data Science. Inhalt Process Mining:

- **Prozessmodellierung** (simulation, verification, optimization, etc.)
- **Complianceanalyse** (z.B. Detektion von Abweichungen von der Prozessvorgabe)
- **Data Science** (data mining, machine learning, business intelligence)
- **Performanceanalyse** (z.B. Identifikation von Engpässen)

Datenbasis: Event Log (= Ereignisprotokoll)

- **Case:** Logische Gruppierung von Aktivitäten (z. B. eine Bestellung)
- **Aktivität:** Aktiver Handlungsschritt eines Prozesses (z.B. Bestellungsabbruch)
- **Zeitstempel:** Zeitpunkt des Starts (und ggf. zusätzlich der Beendigung) einer Aktivität
- **Event:** Datensatz bestehend aus Case, Aktivität und Zeitstempel (sowie ggf. Zusatzinformationen)
- **Event Log:** Sammlung von Events eines Prozesses

Zusammenfassung Event log

- Ein Event Log (Ereignisprotokoll) besteht aus Events (Ereignissen), die sich jeweils auf einen Case (Fall), eine Aktivität und einen Zeitpunkt beziehen.
- Ein Case (Fall) beschreibt eine individuelle Instanz der Abfolge von Aktivitäten.

Trace (Datenspur): Aktivitäten eines Cases werden in einem Kontrollfluss zusammengefasst

Beispiel
Alle Aktivitäten mit der selben Ordernummer ergeben je eine Trace

order number	activity	timestamp	user	product	quantity
9901	register order	22-1-2014@09.15	Sara Jones	iPhone5S	1
9902	register order	22-1-2014@09.18	Sara Jones	iPhone5S	2
9903	register order	22-1-2014@09.27	Sara Jones	iPhone4S	1
9901	check stock	22-1-2014@09.49	Pete Scott	iPhone5S	1
9901	ship order	22-1-2014@10.11	Sue Fox	iPhone5S	1
9903	check stock	22-1-2014@10.34	Pete Scott	iPhone4S	1
9901	handle payment	22-1-2014@10.41	Carol Hope	iPhone5S	1
9902	check stock	22-1-2014@10.57	Pete Scott	iPhone5S	2
9902	cancel order	22-1-2014@11.08	Carol Hope	iPhone5S	2

9901 → ⟨register_order, check_stock, ship_order, handle_payment ⟩
9902 → ⟨register_order, check_stock, cancel_order⟩, 9903 → ⟨register_order, check_stock⟩

Reduzierung des Event Logs auf Sammlung (multiset) von Traces
[⟨register_order, check_stock, ship_order, handle_payment⟩,
⟨register_order, check_stock, cancel_order⟩, ⟨register_order, check_stock⟩]

Vereinfachter Event Log

- Ein vereinfachtes EventLog ist ein Multiset von Traces (d.h. dieselbe Trace kann mehrmals auftreten).
- Eine Trace ist die Aktivitätensequenz eines Cases (d.h. wir betrachten lediglich die Reihenfolge der Aktivitäten der einzelnen Cases).
- **Beispiel:** $L_1 = [\langle a,b,c,d\rangle^3,\ \langle a,c,b,d\rangle^2,\ \langle a,e,d\rangle]$
  - L_i: Log, Index i steht für Log ID
  - ⟨ ⟩: ist eine Trace, die Potenz steht für die Häufigkeit
  - Ein Element in der Klammer ist eine Aktivität

➔ **Was ist die Datenbasis für das Process Mining? Event Log als Multiset von Traces**

Beziehungen

- **Direkte Nachfolge** (x gefolgt von y) **x > y** iff in some trace x is directly followed by y.
- **Kausalität** (x führt zu y) **x→y** iff x > y and not y > x.
- **Parallelität** (x parallel zu y) **x||y** iff x > y and y > x
- **Unabhängigkeit** (x und y sind unabhängig) **x#y** iff not x>y and not y>x.
- **Beispiel:** $L_1 = [\langle a,b,c,d\rangle^3,\ \langle a,c,b,d\rangle^2,\ \langle a,e,d\rangle]$

a>b a>c a>e b>c b>d c>b c>d e>d | a→b a→c a→e b→d c→d e→d | b||c c||b | b#e e#b c#e a#d …

Lösung:

	a	b	c	d	e
a	$\#_{L_1}$	$\rightarrow_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$
b	$\leftarrow_{L_1}$	$\#_{L_1}$	$\|\|_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
c	$\leftarrow_{L_1}$	$\|\|_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
d	$\#_{L_1}$	$\leftarrow_{L_1}$	$\leftarrow_{L_1}$	$\#_{L_1}$	$\leftarrow_{L_1}$
e	$\leftarrow_{L_1}$	$\#_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$

Footprint des Event Logs L_1

- Matrixschreibweise von Beziehungen
- Obere Dreiecksmatrix ausfüllen, untere dann übernehmen und Kausalitätspfeil kehren
- Matrix ist bis auf Pfeilrichtung symmetrisch

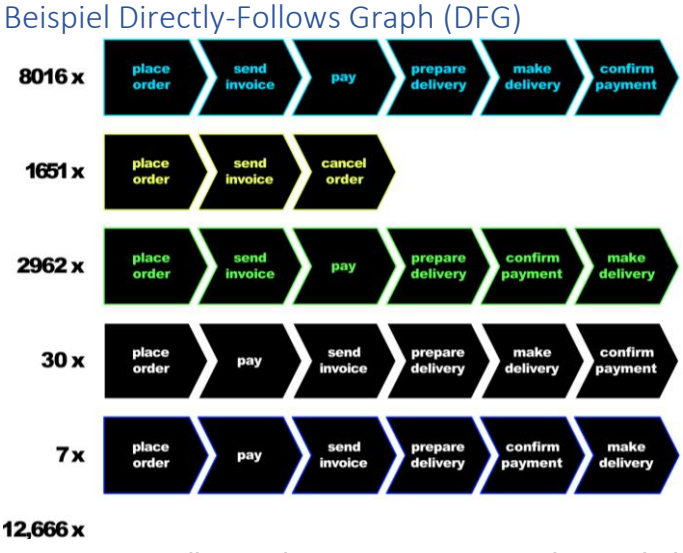
Directly Follows Graph (DFG) und Petri-Netze

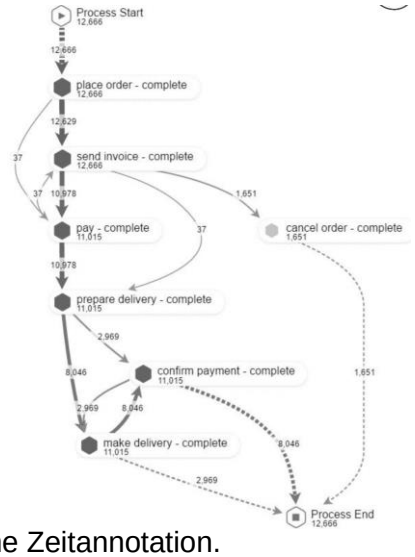
Directly Follows Graph (DFG): Formale Definition

Der Directly-Follows Graph (DFG) eines Event Logs L ist definiert als $G(L) = (A_L, >_L, A_{L\,start}, A_{L\,end})$ mit:

- $A = \{a \in \sigma | \sigma \in L\}$ die Menge aller Aktivitäten in L,
- $>_L = \{(a,b) \in A \times A | a >_L b\}$ die Menge aller Direkte-Nachfolge-Beziehungen,
- $A_{L\,start}$ $\{a \in A | \exists_{\sigma \in L} a = first(\sigma)\}$ die Menge aller Startaktivitäten
- $A_{L\,end}$ $\{a \in A | \exists_{\sigma \in L} a = last(\sigma)\}$ die Menge aller Endaktivitäten.
- $G(L)$: DFG des Event Logs L
- L: Event Log (d.h. ein Multiset von Traces)
- σ: individuelle Trace in L
- a, b: individuelle Aktivitäten in L
- $first(\sigma)$: erste Aktivität des Traces σ
- $last(\sigma)$: letzte Aktivität des Traces σ

Beispiel Directly-Follows Graph (DFG)





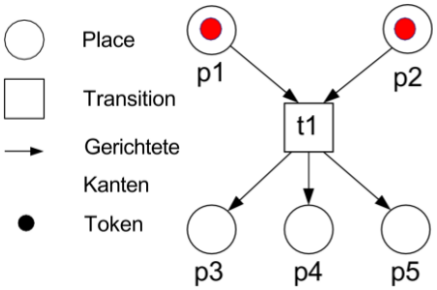
- Im DFG gibt es eine Frequenzannotation und eine Zeitannotation.
- Im Bild oben ist die Frequenzannotation verwendet worden.
- Dabei werden die Häufigkeiten der direkten Nachfolge notiert.
- Bei der Zeitannotation wird der Durchschnitt der Zeit für die direkte Nachfolge notiert.
- Problem: Directly-Follows Graphs können keine Gleichzeitigkeit (concurrency) erkennen.
- Wie erstellt man einen DFG? Visualisierung der direkten Nachfolge (Aktivitäten = Kästen, Nachfolge = Pfeil)

Petri-Netz

- Petri-Netze sind älteste und am besten untersuchte Prozessmodellierungssprache, die Modellierung von Parallelität ermöglicht.
- Obwohl die grafische Notation intuitiv und einfach ist, sind Petri-Netze ausführbar und können mit vielen Analysetechniken analysiert werden
- Ein Petri-Netz ist ein Graph, der aus **Stellen** (States) und **Transitionen** (Transitions) besteht.
- Die Netzwerk**struktur** ist **statisch**, aber gemäß der Schaltregeln können Token durch das Netzwerk fließen.
- Der **Zustand** eines Petri-Netzes wird durch die **Verteilung** von **Token** über Stellen bestimmt und wird als dessen **Markierung** bezeichnet.
- **Stellen** repräsentieren **Zustände** eines Prozesses und **Transitionen Zustandsübergänge** (Aktivitäten).
- Stellen und Transitionen sind durch gerichtete Kanten verbunden, wobei eine Kante genau eine Stelle mit einer Transition oder umgekehrt verbindet.
- Stellen können Token enthalten. Token (Spielstein oder Marke) geben den Zustand eines Prozesses an.
- **Transitionen erzeugen** und/oder **konsumieren Token**.

Enabling (Schaltfähigkeit) und Firing (Schalten)

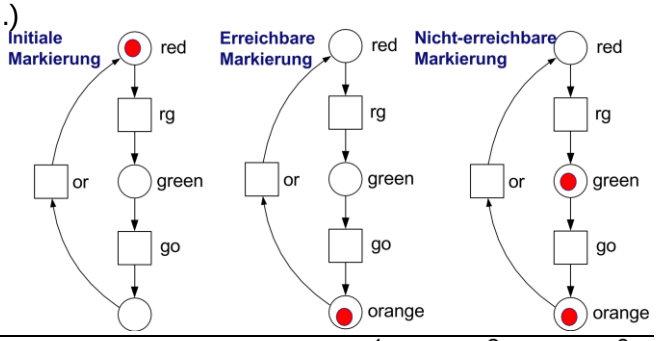
- Transition ist schaltfähig (enabled), wenn jeder Input-Place (unmittelbar vorgelagerte Stelle) einen Token enthält.
- Eine schaltfähige Transition kann schalten (fire) indem jeweils ein Token der Input-Places konsumiert und jeweils ein Token in jedem Output-Place (unmittelbar nachgelagerte Stelle) produziert werden.
- Schalten ist atomisch! (Tokenkonsumption und -produktion gleichzeitig)
- Rechts Beispiel einer schaltfähigen Transition
- Bei Schaltung im Beispiel erhalten p3, p4 und p5 je einen Token.



Initiale Markierung, erreichbare Markierungen und finale Markierung

- Eine **Markierung** ist die Verteilung von Token auf Stellen des Petri-Netzes.
- **Initiale Markierung**: Markierung des Petri-Netzes vor der Ausführung von Schaltungen.
- **Erreichbare Markierungen**: Markierungen des Petri-Netzes die durch Ausführung von Schaltungen von der initialen Markierung aus erreichbar sind.
- **Finale Markierung**: Markierung des Petri-Netzes, nachdem alle möglichen Schaltungen durchgeführt wurden. (Beachte: es können auch keine oder mehrere sein.)

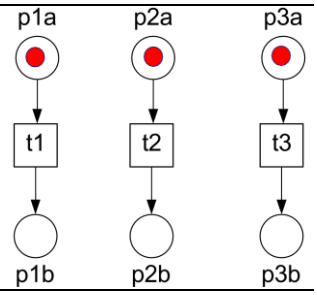
Beispiel Initiale Markierung, erreichbare Markierungen und finale Markierung



Was passiert bei mehreren Transitionen?

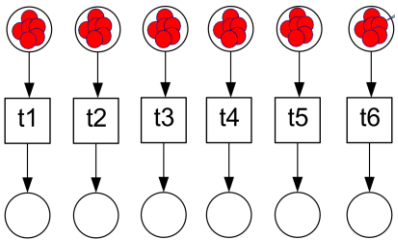
Alle drei Transitionen sind schaltfähig und können in beliebiger Reihenfolge schalten.
Wir nehmen eine «interleaving semantic» (dt. Verschränkung) an, d.h.
Transitionen schalten NICHT gleichzeitig.

➔ $2^3 = 2 \cdot 2 \cdot 2 = 8$ Erreichbare Markierungen (= Verteilung Token im Petri-Netz)

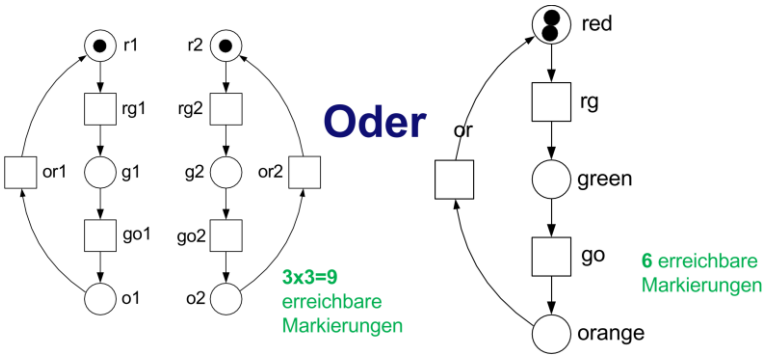


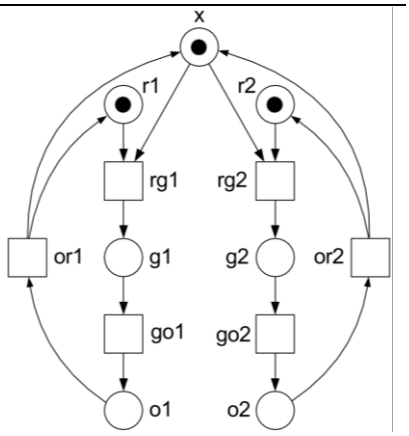
Zustandsexplosion bei mehreren schaltfähigen Transitionen/vielen Tokens

- Im Beispiel je 6 Tokens
- Anzahl Tokens in einer Stelle ist zwischen 0 und 6 → 7
- 6 Transitionen t1 bis t6
- $7^6 = 117'649$ erreichbare Markierungen



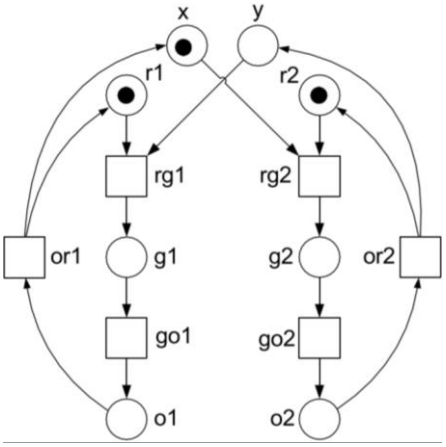
Wie modellieren wir zwei Verkehrsampeln?





Rechts sichere Gestaltung, durch Aufbau können nicht beide Ampel gleichzeitig grün sein.
Nicht deterministisch! (d.h. dieselbe Ampel kann mehrmals hintereinander auf grün schalten)

Wie modelliert man eine alternierende Ampelschaltung?



Process Discovery mit dem Alpha Algorithmus

Drei Anwendungsmuster für Process Mining

- Idee: Prozessmodelle und Event Logs in Bezug zueinander setzen
- Play-in
- Play-out
- Replay

Play-out

- Simulation
- Workflowautomation
- Qualitätssicherung

Play-In

- Process Discovery: Erlernen von De-facto-Prozessmodellen aus beobachtetem Verhalten.

Replay

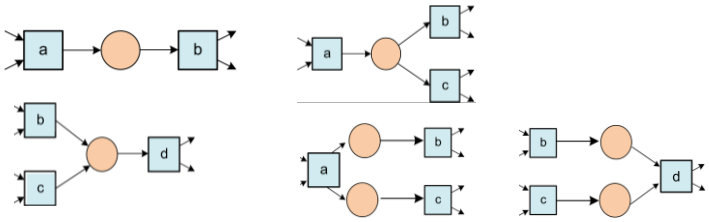
- conformance
- checking bottleneck
- analysis prediction
- Erweitertes Modell mit Zeiten, Frequenzen, etc.
- Diagnostik
- Vorhersagen
- Empfehlungen
- Modelliertes und beobachtetes Verhalten abstimmen:
  - Die wichtigste Form des Process Mining!
  - Konfrontation zwischen Modell und Realität.

Process Discovery: Annahme: Event Log enthält alle möglichen Traces des Event Logs.

Grundidee des Alpha-Algorithmus

Identifikation von Stellen zwischen den Transitionen

- sequence pattern: a→b
- XOR-split pattern: a→b, a→c, and b#c
- XOR-join pattern: b→d, c→d, and b#c
- AND-split pattern: a→b, a→c, and b||c
- AND-join pattern: b→d, c→d, and b||c



Alpha-Algorithmus Einleitung

- Fähigkeit zur Erkennung und Beschreibung komplexer Kontrollflüsse
- Schleifen („Loops“)
- Parallele Transitionen
- Acht Schritte
- Unabhängigkeit von Transitionen
- Verhältnismässig einfach

1. $T_L = \{t \in T \mid \exists_{\sigma \in L}\, t \in \sigma\}$,
2. $T_I = \{t \in T \mid \exists_{\sigma \in L}\, t = first(\sigma)\}$,
3. $T_O = \{t \in T \mid \exists_{\sigma \in L}\, t = last(\sigma)\}$,
4. $X_L = \{(A, B) \mid A \subseteq T_L \wedge A \neq \emptyset \wedge B \subseteq T_L \wedge B \neq \emptyset \wedge \forall_{a \in A} \forall_{b \in B} a \rightarrow_L b \wedge \forall_{a_1, a_2 \in A} a_1 \#_L a_2 \wedge \forall_{b_1, b_2 \in B} b_1 \#_L b_2\}$,
5. $Y_L = \{(A, B) \in X_L \mid \forall_{(A', B') \in X_L} A \subseteq A' \wedge B \subseteq B' \Rightarrow (A, B) = (A', B')\}$,
6. $P_L = \{p_{(A,B)} \mid (A, B) \in Y_L\} \cup \{i_L, o_L\}$,
7. $F_L = \{(a, p_{(A,B)}) \mid (A, B) \in Y_L \wedge a \in A\} \cup \{(p_{(A,B)}, b) \mid (A, B) \in Y_L \wedge b \in B\} \cup \{(i_L, t) \mid t \in T_I\} \cup \{t \in T_O\}$, and
8. $\alpha(L) = (P_L, T_L, F_L)$.

Schritt 1

- $T_L = \{t \in T | \exists_{\sigma \in L} t \in \sigma\}$
- Alle Aktivitäten (= Transitionen) eines Event Logs werden in einer Menge zusammengefasst.
- t = activity / transition
- L = event log
- T = set of all activities in the real world
- T_L = Set of activities detected in the event log
- σ = trace in the event log L

Beispiel:

$$L_1 = [\langle a,b,c,d\rangle^3, \langle a,c,b,d\rangle^2, \langle a,e,d\rangle]$$

- $T_L = \{a, b, c, d, e\}$
- Alle Aktivitäten vom Event Log werden in einer Menge zusammengefasst, die als Menge der Transitionen des Petri-Netz angesehen wird.

	a	b	c	d	e
a	$\#_{L_1}$	$\rightarrow_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$
b	$\leftarrow_{L_1}$	$\#_{L_1}$	$\|\|_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
c	$\leftarrow_{L_1}$	$\|\|_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$
d	$\#_{L_1}$	$\leftarrow_{L_1}$	$\leftarrow_{L_1}$	$\#_{L_1}$	$\leftarrow_{L_1}$
e	$\leftarrow_{L_1}$	$\#_{L_1}$	$\#_{L_1}$	$\rightarrow_{L_1}$	$\#_{L_1}$

Schritt 2

- $T_I = \{t \in T | \exists_{\sigma \in L} t = first(\sigma)\}$
- Identifikation Menge der Starttransitionen – d.h. die ersten Aktivitäten aller Traces: $\langle t_1, \dots, t_n\rangle, \dots, \langle t'_1, \dots, t'_m\rangle$:
- t activity/transition
- T_I : set of start activities (I stands for „in“)
- L: event log
- σ = trace in the event log L

Beispiel $T_I = \{a\}$ Menge aller Starttransitionen

Schritt 3

- $T_O = \{t \in T | \exists_{\sigma \in L} t = last(\sigma)\}$,
- Identifikation Menge der Endtransitionen – d.h. die letzten Aktivitäten aller Traces : $\langle t_1, \dots, t_n\rangle, \dots, \langle t'_1, \dots, t'_m\rangle$
- t: activity/transition
- T_O: set of end activities (O stands for „out“)
- L: event log
- σ = trace in the event log L

Beispiel $T_O = \{d\}$ Menge aller Endtransitionen

Schritt 4-6

Ziel: Identifizierung von Stelle p(A,B) in dem folgendes beschrieben wird:

- Menge von vorgelagerten Transitionen („input transitions“) einer Stelle: $A = \{a_1, a_2, \dots, a_m\}$
- Menge von nachtgelagerten Transitionen („output transitions) einer Stelle: $B = \{b_1, b_2, \dots, b_n\}$

Schritt 4

- $X_L = \{(A,B) \mid A \subseteq T_L \wedge A \neq \emptyset \wedge B \subseteq T_L \wedge B \neq \emptyset \wedge \forall_{a \in A} \forall_{b \in B} a \rightarrow_L b \wedge \forall_{a_1,a_2 \in A} a_1 \#_L a_2 \wedge \forall_{b_1,b_2 \in B} b_1 \#_L b_2\}$
- Finde Paare (A,B) von Mengen von Aktivitäten, so dass
  - zwischen jedem Element $a \in A$ und jedem Element $b \in B$ eine Kausalitätsbeziehung besteht ($a \rightarrow_L b$),
  - alle Elemente in A unabhängig sind ($a_1 \#_L a_2$), und alle Elemente in B unabhängig sind ($b_1 \#_L b_2$).

Beispiel

1. Alle Kausalitätsbeziehungen: $(\{a\},\{b\}), (\{a\},\{c\}), (\{a\},\{e\}), (\{b\},\{d\}), (\{c\},\{d\}), (\{e\},\{d\})$
2. Zusammenfassung zu Mengen mit mehr als einer Aktivität: $(\{a\},\{b,e\}), (\{a\},\{c,e\}), (\{b,e\},\{d\}), (\{c,e\},\{d\})$
   Achtung, zusammengefasst Elemente (also alles in einer geschweiften Klammer) müssen unabhängig voneinander sein, sonst dürfen sie nicht zusammengefasst werden!
3. Beide Mengen zusammenfassen: $X_{L_1} = \{(\{a\},\{b\}), (\{a\},\{c\}), (\{a\},\{e\}), (\{b\},\{d\}), (\{c\},\{d\}), (\{e\},\{d\}), (\{a\},\{b,e\}), (\{a\},\{c,e\}), (\{b,e\},\{d\}), (\{c,e\},\{d\})\}$

Schritt 5

- $Y_L = \{(A,B) \in X_L \mid \forall_{(A',B') \in X_L} A \subseteq A' \wedge B \subseteq B' \Rightarrow (A,B) = (A',B')\}$
- Lösche aus der Menge X_L alle Paare (A,B) die nicht maximal sind (d.h. Teilmenge von anderer Menge)!

Beispiel $Y_L = \{(\{a\},\{b,e\}), (\{a\},\{c,e\}), (\{b,e\},\{d\}), (\{c,e\},\{d\})\}$

Schritt 6

- $P_L = \{p_{(A,B)} \mid (A,B) \in Y_L\} \cup \{i_L, O_L\}$
- Bestimmung der Menge der Stellen: Jedes Paar (A,B) aus YL repräsentiert eine Stelle $p(A,B)$. Um einen Ende-zu-Ende Prozess abzubilden, wird eine Startstelle i_L und eine Endstelle o_L ergänzt.

Beispiel: $P_{L1} = \{p_{(\{a\},\{b,e\})}, p_{(\{a\},\{c,e\})}, p_{(\{b,e\},\{d\})}, p_{(\{c,e\},\{d\})}, i_{L1}, O_{L1}\}$

Schritt 7

- $F_L = \{(a, p_{(A,B)}) \mid (A,B) \in Y_L \wedge a \in A\} \cup \{(p_{(A,B)}, b) \mid (A,B) \in Y_L \wedge b \in B\} \cup \{(i_L, t) \mid t \in T_I\} \cup \{(t, o_L) \mid t \in T_O\}$
- Identifikation der gerichteten Kanten: Verbinde jede Stelle $p(A,B)$ mit jeder Transition a der Menge A vorgelagerter Transitionen und mit jeder Transition b der Menge B nachgelagerter Transitionen. Ergänze zudem jeweils eine Kante zwischen der Startstelle i_L und den Anfangstransitionen $t \in T_I$ und eine Kante zwischen jeder Endtransition $t \in T_O$ und der Endstelle o_L.

Beispiel: $F_{L_1} = \{(a, p(\{a\},\{b,e\})), (p(\{a\},\{b,e\}, b), (p(\{a\},\{b,e\}, e), (a, p(\{a\},\{c,e\})), \dots, (i_L, a), (d, O_L)\}$

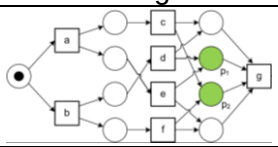
Schritt 8

- $\alpha(L) = (P_L, T_L, F_L)$ → $\alpha(L)$: Petri net
- P_L: Menge Stellen (Schritt 6)
- T_L: Menge Transitionen (Schritt 1)
- F_L: Menge Kanten (Schritt 7)

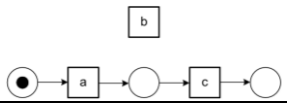
Einschränkung α Algorithmus:

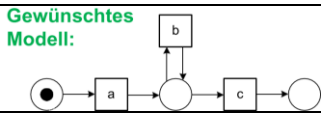
- Der Alpha-Algorithmus kann einfache Prozessmuster aus Event Logs ermitteln.

- implizite (d.h. verzichtbare) Stellen!

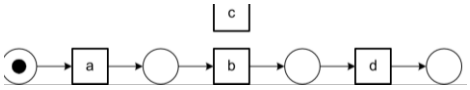


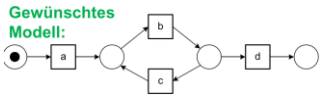
- Schleifen der Länge 1



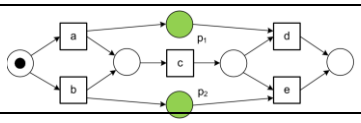


- Schleifen der Länge 2



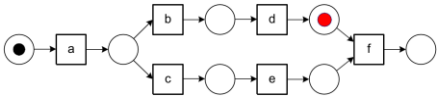


- Nicht-lokale Abhängigkeiten



- Representational Bias (stille Transition)

- Petri-Netze mit Deadlocks (Blockierung)



Conformance Checking

- Replay: conformance checking
- Konformität = Übereinstimmung des tatsächlichen Verhaltens mit der Norm

Drei Kriterien für die Bewertung der Qualität des entdeckten Modells

Fitness

- Perfekte Fitness, wenn alle Traces im Event Log vom Modell von Anfang bis Ende wiedergegeben werden.
- Log-Ebene: Anteil der Traces im Event Log, die vollständig wiedergegeben werden können
- Trace-Ebene: Anteil der Events im Trace, die im Modell tatsächlich möglich sind

Einfachheit

- Das einfachste Modell, das das im Event Log beobachtete Verhalten erklären kann, ist das beste Modell.
- Die Komplexität des Modells kann durch die Anzahl der Stellen, Transitionen und Kanten definiert werden.

Präzision

- Ein Modell ist präzise, wenn es nicht "zu viel" Verhalten ("Underfitting") zulässt.
- Anteil der Traces im Event Log an allen Traces, die gemäss Prozessmodell möglich sind.

Footprint Konformität

- Idee: Vergleichen Sie die Footprints von Log und Modell
- Footprint Konformität $(FC) = 1 - \frac{Anzahl\ Unterschiede}{Anzahl\ Zellen\ im\ Footprint}$
- (1 = perfekte Konformität), (0 = schlechtest mögliche Konformität)

Einschränkungen der Footprint Konformität

- Häufigkeiten der Traces werden nicht verwendet.
- Das Verhalten wird nur indirekt betrachtet (nur Direkte-Nachfolge-Verhältnisse).
- Ziel der Footprint Konformität ist es, Fitness und Präzision in einer einzigen Metrik zu erfassen.

Token-based Replay (token-basiertes Abspielen)

Quantifizierung der Fitness eines Petrinetzes N in Bezug auf einen Trace σ:

- p = produced tokens
- m = missing tokens
- σ = trace
- c = consumed tokens
- r = remaining tokens
- N = petri net
- Formel: $fitness(\sigma, N) = \frac{1}{2}\left(1 - \frac{m}{c}\right) + \frac{1}{2}\left(1 - \frac{r}{p}\right)$

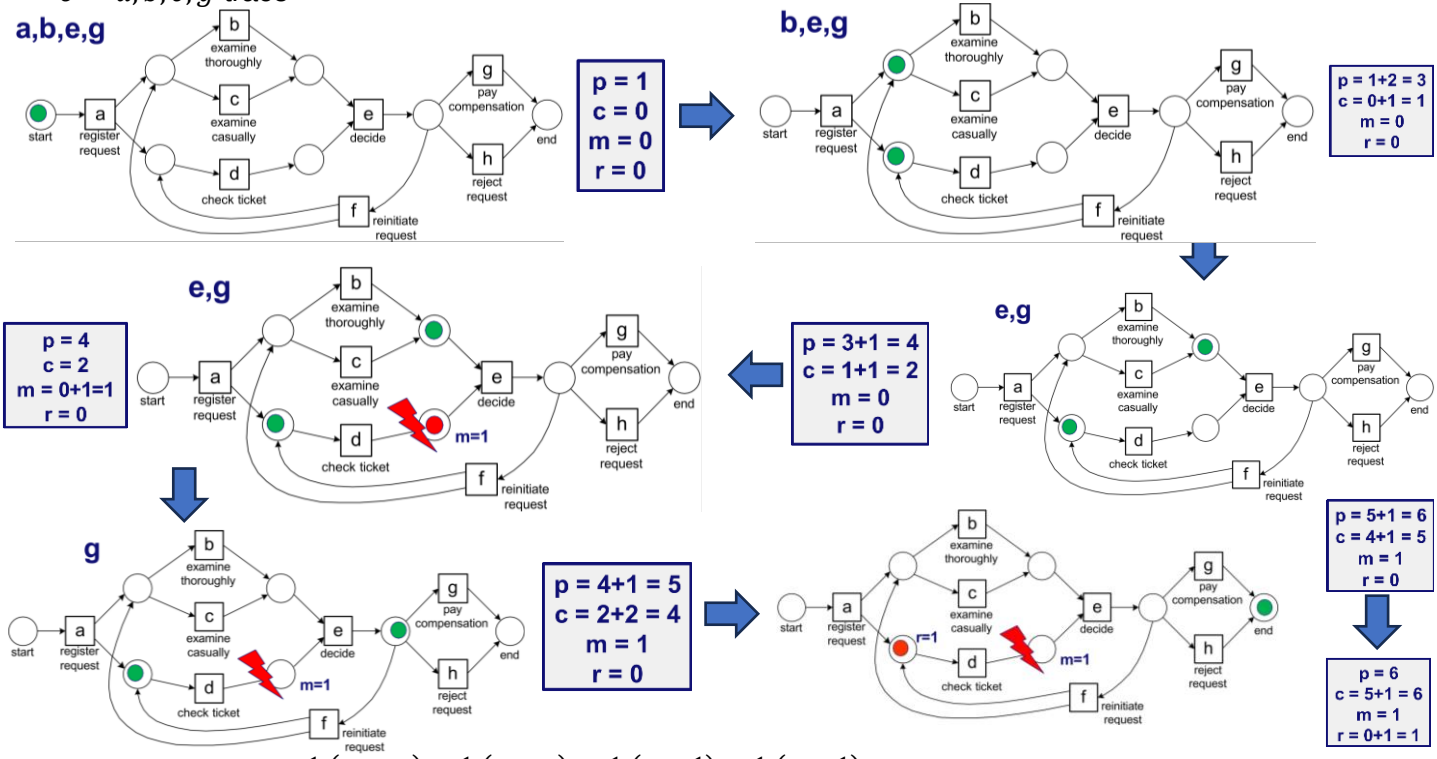
Zählen von produzierten und konsumierten Token

Initialisierung und Finalisierung:

- Zu Beginn wird ein Token **produziert** für die Startstelle: $p = 1$
- Am Ende wird ein Token **konsumiert** aus der Endstelle (auch wenn es ein missing token ist!): $c' = c + 1$.

Beispiel Kostenerstattungsprozess

- $\sigma = a, b, e, g$ trace



→ $fitness(\sigma, N) = \frac{1}{2}\left(1 - \frac{m}{c}\right) + \frac{1}{2}\left(1 - \frac{r}{p}\right) = \frac{1}{2}\left(1 - \frac{1}{6}\right) + \frac{1}{2}\left(1 - \frac{1}{6}\right) = 0.83333$

Quantifizierung der Token-based Replay Fitness auf Log-Ebene

- $fitness(L, N) = \frac{1}{2}\left(1 - \frac{missing\ tokens}{consumed\ tokens}\right) + \frac{1}{2}\left(1 - \frac{remaining\ tokens}{produced\ tokens}\right) = \frac{1}{2}\left(1 - \frac{\Sigma_{\sigma \in L} L(\sigma) \cdot m_{N,\sigma}}{\Sigma_{\sigma \in L} L(\sigma) \cdot c_{N,\sigma}}\right) + \frac{1}{2}\left(1 - \frac{\Sigma_{\sigma \in L} L(\sigma) \cdot r_{N,\sigma}}{\Sigma_{\sigma \in L} L(\sigma) \cdot p_{N,\sigma}}\right)$
- L: Event Log
- N: Petrinetz
- σ: Trace
- $L(\sigma)$: Häufigkeit von σ
- $m_{N,\sigma}$: Missing Tokens bei Wiedergabe σ auf N
- $r_{N,\sigma}$: Remaining Tokens bei Wiedergabe σ auf N
- $c_{N,\sigma}$: Consumed Tokens bei Wiedergabe σ auf N
- $p_{N,\sigma}$: Produced Tokens bei Wiedergabe σ auf N
- Intuition: Anteil der Events im Event Log, die vom Petrinetz korrekt abgespielt werden können.

Beispiel: $L = \{< abd >, < acbd >^3, < abcd >^3\}$ →

$< abd > : p = 5, c = 5, m = 1, r = 1;\ < acbd > : p = 6, c = 6, m = 0, r = 0;\ <$ abcd $> :$ p $= 6,$ c $= 6,$ m $= 0,$ r $= 0$

Aggregation of Log-Ebene: $\Sigma p: 1 * 5 + 3 * 6 + 3 * 6 = 41;\ \Sigma c: 1 * 5 + 3 * 6 + 3 * 6 = 41;\ \Sigma m: 1 * 1 + 3 * 0 + 3 * 0 = 1;\ \Sigma r: 1 * 1 + 3 * 0 + 3 * 0 = 1$ → $fitness(L, N) = (1/2\ (1 - 1/41)) + (1/2\ (11/41)) = 40/41 = 0.975$

Limitationen der Token-based Replay Fitness

Token flooding (Überflutung mit Token)

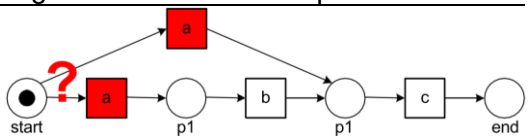
Konformitätswerte aufgrund von «Token-Flooding» manchmal zu optimistisch:

- Wenn der Trace stark vom Modell abweicht, werden während Wiedergabe viele fehlende Token eingefügt.
- Als Ergebnis all der hinzugefügten Token werden viele Transitionen schaltfähig.
- Daher ist es wahrscheinlich, dass auch abweichende Events zu einer schaltfähigen Transition passen.
- Dies führt zu einer irreführenden Diagnose, da unerwünschte Teile des Modells schaltfähig werden können.
- Daher kann Fitnesswert bei geringer Konformität des Modells in Bezug auf einen Trace zu optimistisch sein.

Keine eindeutig bezeichnete Transitionen:

- Fitness geht von eindeutig bezeichneten Transitionen aus.
- **Beispiel:**

$$L_{11} = [\langle a, b, c \rangle^{20}, \langle a, c \rangle^{30}]$$



Organizational Mining

- Organizational Mining-Ziel:
  - Erweitern von Prozessmodellen mit Informationen zur Organisationsebene.
  - Aktivitäten zu Rollen, Gruppen oder anderen organisatorischen Entitäten (Einheiten) zu ordnen
- Wie kann man Ressourcendaten in Ereignisprotokollen nutzen?

Herausforderung 1: Rollenidentifikation aus Daten

- Kompakte Darstellung des Ereignisprotokolls mit Hervorhebung des Ressourcenattributs jedes Ereignisses

Beispiel für Rollenidentifikation

a = Antrag registrieren, b = gründlich prüfen,
c = beiläufig prüfen, d = Ticket prüfen,
e = entscheiden, f = Antrag erneut stellen,
g = Entschädigung zahlen und h = ablehnen Antrag

case id	trace
1	$\langle a^{Pete}, b^{Sue}, d^{Mike}, e^{Sara}, h^{Pete} \rangle$
2	$\langle a^{Mike}, d^{Mike}, c^{Pete}, e^{Sara}, g^{Ellen} \rangle$
3	$\langle a^{Pete}, c^{Mike}, d^{Ellen}, e^{Sara}, f^{Sara}, b^{Sean}, d^{Pete}, e^{Sara}, g^{Ellen} \rangle$
4	$\langle a^{Pete}, d^{Mike}, b^{Sean}, e^{Sara}, h^{Ellen} \rangle$
5	$\langle a^{Ellen}, c^{Mike}, d^{Pete}, e^{Sara}, f^{Sara}, d^{Ellen}, c^{Mike}, e^{Sara}, f^{Sara}, b^{Sue}, d^{Pete}, e^{Sara}, h^{Mike} \rangle$
6	$\langle a^{Mike}, c^{Ellen}, d^{Mike}, e^{Sara}, g^{Mike} \rangle$

Ressourcen-Aktivitäts-Matrix

- Aktivitäten anhand der Anzahl verwendeter Ressourcenattribute aus den Cases darstellen

	a	b	c	d	e	f	g	h
Ellen	1	0	1	2	0	0	2	1
Mike	2	0	3	4	0	0	1	1
Pete	3	0	1	3	0	0	0	1
Sara	0	0	0	0	9	3	0	0
Sean	0	2	0	0	0	0	0	0
Sue	0	2	0	0	0	0	0	0

Normalisierte Ressourcen-Aktivitäts-Matrix

- Dividieren die absoluten Häufigkeiten durch die Anzahl der Fälle (Cases)
- Interpretation der Ressourcen-Aktivitäts-Matrix:
- Häufigkeit, mit der eine Ressource eine Aktivität pro Case ausführt.

	a	b	c	d	e	f	g	h
Ellen	0.166667	0	0.166667	0.333333	0	0	0.333333	0.166667
Mike	0.333333	0	0.5	0.666667	0	0	0.166667	0.166667
Pete	0.5	0	0.166667	0.5	0	0	0	0.166667
Sara	0	0	0	0	1.5	0.5	0	0
Sean	0	0.333333	0	0	0	0	0	0
Sue	0	0.333333	0	0	0	0	0	0

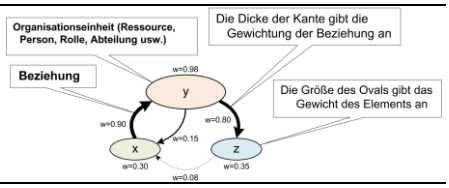
- **Beispiel:** Durchschnittliche Aktivität e wird von Sara 1,5-mal pro Case ausgeführt.
- Die Ressourcen-Aktivitäts-Matrix bietet grundlegende Erkenntnisse darüber, «wer was tut».

Soziales Netzwerk

- Soziometrie: Präsentiert Daten zu zwischenmenschlichen Beziehungen in Diagrammform oder Matrizen.
- Jacob Levy Moreno nutzte solche Techniken 1930, um Studenten besser in Wohnhäuser einzuteilen.
- Kanten: Gewicht oder (invertierter) Abstand.
- Metriken zur Angabe der Wichtigkeit:
  - Zentralität, o Nähe, o Betweenness «Dazwischenheit»
- Identifizierung von Cliquen.

Soziale Netzwerkanalyse

Wie interpretiert man ein Soziales Netzwerk?



- Knoten repräsentieren soziale Entitäten,
- Kanten repräsentieren Beziehungen,
- Stärke der Kanten beschreibt Intensität der Beziehung

Frage: Welche Ressourcen ähneln sich?

- Angenommen, Sie müssen drei Gruppen mit ähnlichen Ressourcen erstellen. Was wären das für Gruppen?
- **Beispiel:** Intuitiv gäbe dies die Gruppen: {Pete, Mike, Ellen}, {Sue, Sean}, {Sara}

Ressourcen als Vektoren

- Zeilen aus Matrix als Vektor nehmen, sodass eine Zeile ein Vektor mit je einer Ressource bildet.

Methoden zur Berechnung von Vektordistanzen

Standard Metriken von «Entfernung» können verwendet werden, z. B.

- Euklidische Distanz: $d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2}$
- Manhatten Distanz: $d(p,q) = \sum_i |p_i - q_i|$
- Pearson Korrelation Koeffizient: $Korr(X,Y) = \frac{E\left((X-E(X))(Y-E(Y))\right)}{\sqrt{E\left((X-E(X))^2\right)E\left((Y-E(Y))^2\right)}}$

Beispiel: Berechnen die Manhattan Distanz zwischen Sean und Sara.

$P_{Sara} = (0, 0, 0, 0, 1.5, 0.5, 0, 0)$ $P_{Sean} = (0, 0.333333, 0, 0, 0, 0, 0, 0)$

➔ $d(P_{Sara}, P_{Sean}) = \sum_i |p_i - q_i| = 0 + 0.333333 + 0 + 0 + 1.5 + 0.5 + 0 + 0 = 2.333333$

Invertieren den Wert (y = 1/x), um ein Ähnlichkeitsmaß zwischen P_{Sara} und P_{Sean} zu erzeugen.

➔ $similarity(P_{Sean}, P_{Sara}) = \frac{1}{2.3333} = 0.428571$

Beispiel: Manhattan-Distanzen für Ressourcenaktivitätsmatrix → Invertierte Werte für Ähnlichkeitsmass

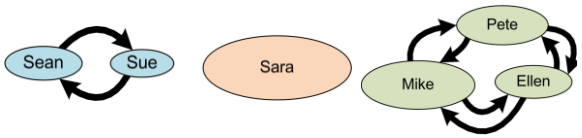
Resource	Ellen	Mike	Pete	Sara	Sean	Sue
Ellen	0	1	0.833333	3.166667	1.5	1.5
Mike	1	0	0.833333	3.833333	2.166667	2.166667
Pete	0.833333	0.833333	0	3.333333	1.666667	1.666667
Sara	3.166667	3.833333	3.333333	0	2.333333	2.333333
Sean	1.5	2.166667	1.666667	2.333333	0	0
Sue	1.5	2.166667	1.666667	2.333333	0	0

Resource	Ellen	Mike	Pete	Sara	Sean	Sue
Ellen	inf	1	1.2	0.315789	0.666667	0.666667
Mike	1	inf	1.2	0.26087	0.461538	0.461538
Pete	1.2	1.2	inf	0.3	0.6	0.6
Sara	0.315789	0.26087	0.3	inf	0.428571	0.428571
Sean	0.666667	0.461538	0.6	0.428571	inf	inf
Sue	0.666667	0.461538	0.6	0.428571	inf	inf

- inf bedeutet perfekte Ähnlichkeit.
- Ähnlichkeiten >= 1 in rot.
- Selbstähnlichkeiten auf der Diagonalen können ignoriert werden.

Soziales Netzwerk auf der Grundlage der Ähnlichkeit von Profilen

- Ähnlichkeit ist der Kehrwert von Entfernung.
- Ressourcen, die ähnliche Aktivitäten ausführen, sind miteinander verknüpft.
- Selbstähnlichkeit wird ignoriert.



Clustering von Ressourcen zum Identifizieren von Rollen

Beispielhafte Methoden:

- k-means clustering
- agglomerative
- hierarchical clustering

Herausforderung 2: Identifizierung von Teams anhand von Daten

Ressource Case Matrix

- Aufzählung, wie oft in einem Case eine einzelne Person/Ressource involviert ist.
- Oft ist es nur interessant, ob eine Ressource an einem Case beteiligt ist oder nicht

Case_ID	1	2	3	4	5	6
Ellen	0	1	2	1	2	1
Mike	1	2	1	1	3	3
Pete	2	1	2	1	2	0
Sara	1	1	3	1	5	1
Sean	0	0	1	1	0	0
Sue	1	0	0	0	1	0

Berechnen eines Ähnlichkeitsmaßes aus der Resource Case Matrix

1. Binarization (alle Werte die grösser als 1 sind, werden auf 1 gesetzt, 1 bleibt und 0 bleibt auch 0)
2. Euklidische Distanzen berechnen (Matrix)
3. Werte der Distanzmatrix invertieren

Case_ID	1	2	3	4	5	6
Ellen	0	1	2	1	2	1
Mike	1	2	1	1	3	3
Pete	2	1	2	1	2	0
Sara	1	1	3	1	5	1
Sean	0	0	1	1	0	0
Sue	1	0	0	0	1	0

Case_ID	1	2	3	4	5	6
Ellen	0	1	1	1	1	1
Mike	1	1	1	1	1	1
Pete	1	1	1	1	1	0
Sara	1	1	1	1	1	1
Sean	0	0	1	1	0	0
Sue	1	0	0	0	1	0

Resource	Ellen	Mike	Pete	Sara	Sean	Sue
Ellen	0	1	1.414214	1	1.732051	2.236068
Mike	1	0	1	0	2	2
Pete	1.414214	1	0	1	1.732051	1.732051
Sara	1	0	1	0	2	2
Sean	1.732051	2	1.732051	2	0	2
Sue	2.236068	2	1.732051	2	2	0

Resource	Ellen	Mike	Pete	Sara	Sean	Sue
Ellen	inf	1	0.707107	1	0.57735	0.447214
Mike	1	inf	1	inf	0.5	0.5
Pete	0.707107	1	inf	1	0.57735	0.57735
Sara	1	inf	1	inf	0.5	0.5
Sean	0.57735	0.5	0.57735	0.5	inf	0.5
Sue	0.447214	0.5	0.57735	0.5	0.5	inf

Identifikation von Teams mit der Resource Case Matrix

- Verwenden Sie die Ähnlichkeitsmaße, um ein Ressourcennetzwerk zu zeichnen. Berücksichtigen Sie nur Ähnlichkeiten von über 0,6 (rot markiert).
- $\{Ellen, Mike, Pete, Sara\}, \{Sean\}, \{Sue\}$

Resource	Ellen	Mike	Pete	Sara	Sean	Sue
Ellen	inf	1	0.707107	1	0.57735	0.447214
Mike	1	inf	1	inf	0.5	0.5
Pete	0.707107	1	inf	1	0.57735	0.57735
Sara	1	inf	1	inf	0.5	0.5
Sean	0.57735	0.5	0.57735	0.5	inf	0.5
Sue	0.447214	0.5	0.57735	0.5	0.5	inf

Fortgeschrittene Anwendungen des Process Mining

- Organizational Mining-Ziel: Erweitern von Prozessmodellen mit Daten auf Organisationsebene.
- Herausforderung 3: Identifizierung potenzieller **Ressourcenengpässe**
- Identifikation von «Handovers» («Die Arbeit wird von einer Ressource an eine andere übergeben»)
- Die kausalen Abhängigkeiten im Footprint werden verwendet, um Handovers im Event Log zu zählen.

Simplifiziertes Handover Mass Formel

- $p_1, p_2 = Ressourcen$ $\triangleright_L$ = Handover im Log L "p_1 hands over to p_2"
- $|p_1 \triangleright_c^1 p_2|$ = Anzahl direkte Nachfolge einer Aktivität von p_1 durch eine Aktivität von p_2 in c
- $p_1 \triangleright_L p_2 = \frac{(\sum_{c \in L} |p_1 \triangleright_c^1 p_2|)}{(\sum_{c \in L} (|c|-1))}$
- Theoretische maximale Anzahl an Handovers: $c = Case$, $|c| = Anzahl\ Aktivitäten$ im Case, $L = Log$
- Handover zeigt wieviel Prozent von wo nach wo läuft. Umso grösser Wert, umso wichtiger → Bottleneck

Beispiel:

- $L = \{< Bestellung^{Sue}, Servieren^{Alan}, Kassieren^{Sara} >, < Bestellung^{Cara}, Servieren^{Alan}, Kassieren^{Sara} >\}$
- Berechne $Alan \triangleright_L Sara$
- $Alan \triangleright_L Sara = \frac{(\sum_{c \in L} |Alan \triangleright_c^1 Sara|)}{(\sum_{c \in L}(|c|-1))} = \frac{2}{4} = 0.5$
- $(\sum_{c \in L}(|c|-1)) = (|3|-1) + (|3|-1) = 4$ → theoretische Maximale Anzahl Übergaben
- $(\sum_{c \in L} |Alan \triangleright_c^1 Sara|) = (1) + (1) = 2$ → Wie oft übergibt Alan an Sara im ersten und zweiten Case?
- Berechne $Cara \triangleright_L Alan = \frac{(\sum_{c \in L} |Cara \triangleright_c^1 Alan|)}{(\sum_{c \in L}(|c|-1))} = \frac{0+1}{4} = 0.25$
- **Bottleneck ist hier Alan zu Sara**

	Alan	Cara	Sara	Sue
Alan	0.00	0.0	0.5	0.0
Cara	0.25	0.0	0.0	0.0
Sara	0.00	0.0	0.0	0.0
Sue	0.25	0.0	0.0	0.0

Berechnung Degree Centrality

- Summe der Gewichte der Verbindungen, die ein bestimmter Knoten hat (eingehend und ausgehend)
- $C_D(i) = \sum_{j=1}^{n}(w_{i,j} + w_{j,i})$ Degree Centrality des Knotens i
- i, j: Knoten
- n: Anzahl Knoten im Netzwerk
- $w_{i,j}$: Gewicht der gerichteten Kante von i nach j
- Pro Knoten (Ressource) Spalten- und Zeilensumme bilden und diese addieren

Beispiel (von oben weitergeführt):

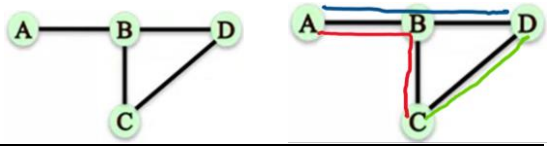
- $C_D(Alan) = 1$
- $C_D(Cara) = 0.25$
- $C_D(Sara) = 0.5$
- $C_D(Sue) = 0.25$
- Alan und (zu einem geringerem Masse) Sara erhalten und übergeben die meisten Arbeitsschritte. Bei ihrem Ausfall bleiben viele Arbeitsschritte unerledigt.

Berechnung Betweenness Centrality

- Anteil der kürzesten Wege zwischen zwei beliebigen Knoten, die an einem bestimmten Knoten vorbeiführen
- Annahme: Wichtige Knoten verbinden andere Knoten miteinander
- $c_{btw}(v) = \sum_{s,t \in N} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}}$
- N: Netzwerk
- $\sigma_{s,t}$: Anzahl an kürzesten Pfaden zwischen Knoten s und t
- $\sigma_{s,t}(v)$: Anzahl an kürzesten Pfaden zwischen Knoten s und t durch v

Beispiel

- $c_{btw}(B) = \sum_{s,t \in N} \frac{\sigma_{A,D}(B)}{\sigma_{A,D}} = \frac{\sigma_{A,C}(B)}{\sigma_{A,C}} + \frac{\sigma_{C,D}(B)}{\sigma_{C,D}} = \frac{1}{1} + \frac{1}{1} + \frac{0}{1} = 2$



Berechnung Betweenness Centrality für Handovers: Umwandlung Handover- in Distanznetzwerk

- Betweenness Centralitiy wird invertiert, wodurch Umwandlung bereits stattfindet
- $Distanz = \frac{1}{Nähe}$ → Matrix invertieren

Customer Lifetime Value – CLV

Definition CLV

- Gesamter finanzieller Beitrag von der aktuellen Periode in die Zukunft – d. h. Einnahmen abzüglich Kosten – eines Kunden über seine zukünftige Lebensdauer im Unternehmen
- spiegelt zukünftige Rentabilität des Kunden wider
- Barwert der kumulierten Cashflows eines Kunden
- Vergleichbar mit dem Kapitalwert von Maschinen

Ziel CLV

- hilft einem Unternehmen zu wissen, wie viel es in die Kundenbindung investieren kann, um eine positive Kapitalrendite zu erzielen
- Fokussieren von Investitionen auf Kunden, die den maximalen Gewinn bringen
- Entscheidung über kundenspezifische Kommunikationsstrategien

Basis-CLV-Modell – Barwert der wiederkehrenden Margen

- $CLV = \sum_{t=1}^{T} m \cdot \frac{1}{(1+i)^t}$ CLV: Customer Lifetime Value
- t: Zeitintervall (z.B. jährlich)
- T: Zeithorizont (= Gesamtzahl der betrachteten Zeitintervalle)
- m: Marge (= Umsatz – Kosten) des Kunden im Zeitintervall t
- i: Zinssatz für das Zeitintervall t

CLV-Modell mit Retention Rate und endlicher Anzahl von Perioden («On-Off-Modell»)

- $CLV = \sum_{t=1}^{T} m \cdot \frac{1}{(1+i)^t} \cdot r^t$
- Zusätzlich: r: Retentionsrate (= Wahrscheinlichkeit des Verbleibs, zwischen 0 und 1)

Modellannahmen:

- Der Kunde entscheidet nach jeder Periode, ob er den Dienst fortsetzen oder einstellen möchte
- $r = 0.9$ bedeutet, dass Kunde mit Wahrscheinlichkeit 90 % bleibt und mit Wahrscheinlichkeit 10 % geht
- Ein Kunde, der gegangen ist, ist für immer verloren
- Endliche Anzahl von Perioden

CLV-Modell mit Retention Rate und unendlichem Zeithorizont

- $CLV = \frac{m}{(1+i)} \cdot r + \frac{m}{(1+i)^2} \cdot r^2 + \frac{m}{(1+i)^3} \cdot r^3 + \cdots = m \sum_{t=1}^{\infty} \left(\frac{r}{1+i}\right)^t = m \frac{r}{1+i-r}$

CLV-Modell mit fixen Akquisitionskosten und unendlichem Zeithorizont

- $CLV = m \frac{r}{1+i-r} - AC$
- AC: Akquisitionskosten

Beispiel – CLV mit unendlichem Zeithorizont

Was ist die Schwelle der Akqusitionskosten, ab der eine Kundenbeziehung unrentabel wird?

- Umsatz pro Kunde und Periode: CHF 5
- Kosten pro Kunde und Periode: CHF 3
- Retention Rate pro Periode: 0.65
- Zinssatz pro Periode: 0.05

➔ $0 = m \frac{r}{1+i-r} - AC \rightarrow AC = m \frac{r}{1+i-r} = (5-3) \frac{0.65}{1+0.05-0.65} = 3.25$

Customer Equity Summe der Customer Lifetime Values aller Kunden

- $CE = \sum_{c=1}^{C} CLV_c = \sum_{c=1}^{C} ((\sum_{t=1}^{T} (m_c \cdot \frac{r_c^t}{(1+i)^t})) - AC_C)$
- CE: Customer Equity (= Summe einzelnen CLVs)
- c: Kundenindex
- C: Gesamtzahl der Kunden
- CLV_c : Customer Lifetime Value des Kunden c
- m_c : Feste Marge m pro Periode für Kunden c
- i: Zinsrate
- r_c : Bindungsrate des Kunden C
- AC_c : Anschaffungskosten für Kunde c

Customer Equity Summe der Customer Lifetime Values aller Kunden (unendlicher Zeithorizont und alle Kunden mit gleichen Daten)

- $CE = C \cdot \left(m \cdot \frac{r}{1+i-r} - AC\right)$

Beispiel – Customer Equity

Betrachten Sie das CLV-Modell der letzten Übung mit Akquisitionskosten von 1.
Was ist die Schwelle der Akqusitionskosten, ab der eine Kundenbeziehung unrentabel wird?

- Umsatz pro Kunde und Periode: CHF 5
- Kosten pro Kunde und Periode: CHF 3
- Retention Rate pro Periode: 0.65
- Zinssatz pro Periode: 0.05

Wie hoch ist das Customer Equity, wenn Sie 5000 Kunden haben?

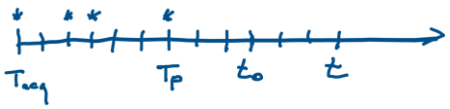
➔ $CLV = m \frac{r}{1+i-r} - AC = (5-3) \frac{0.65}{1+0.05-0.65} - 1 = 3.25 - 1 = 2.25, \quad CE = C \cdot CLV = 5000 \cdot 2.25 = 11250$

Kritik am CLV-Modell mit Retention Rate und unendlichem Zeithorizont

Eine feste Bindungsrate ist in den meisten Fällen in nicht vertraglichen Umgebungen (z. B. Friseur, Restaurant, Autoreparatur) unrealistisch. Z.B. wegen:

- **Aktualitätseffekt** (Intuition hier: je jünger die letzte Transaktion, desto höher die Transaktionswahrscheinlichkeit/Bindungsrate)
- **Häufigkeitseffekt** (Intuition hier: Budgetbeschränkungen und abgenutzter Neuheitseffekt verringern die Transaktionswahrscheinlichkeit/Bindungsrate)
- Kunden sind nicht "**für immer verloren**", sondern können zu einem späteren Zeitpunkt wiederkommen

Kaufwahrscheinlichkeit mit Aktualitäts- und Häufigkeitseffekten



- $P_{act}(t) = \left(\frac{T_p - T_{acq}}{t - T_{acq}}\right)^n$
- P_{act}: Wahrscheinlichkeit der Kundenaktivität
- T_p: Zeitperiode des letzten Kaufs
- T_{acq} : Zeitperiode der Kundenakquise
- t: Zeitperiode, für die die Wahrscheinlichkeit berechnet wird
- n: Anzahl der Käufe («*»)
- t_0 : Gegenwärtige Periode

CLV mit Aktualitäts- und Häufigkeitseffekt

- $CLV = \sum_{t=1}^{T} P_{act}(t) \cdot \frac{m}{(1+i)^t}$

Beispiel:

- Berechnen Kaufwahrscheinlichkeit eines Kunden, der in Periode -12 akquiriert wurde für die Periode 5.
- Der Kunde hat 4 Käufe getätigt, den letzten in Periode -3.

- $T_p = -3$
- $T_{acq} = -12$
- $n = 4$
- $t = 5$
- $P_{act}(t) = \left(\frac{T_p - T_{acq}}{t - T_{acq}}\right)^n = P_{act}(5) = \left(\frac{-3--12}{5--12}\right)^4 = \left(\frac{9}{17}\right)^4 = 0.079$

Zusammenhang zwischen Loyalität und Abwanderung

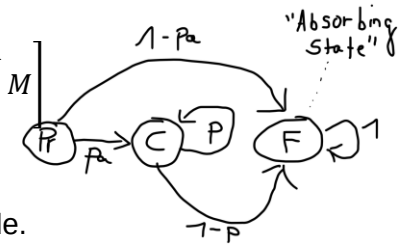
- churn rate = 1 – retention rate → Abwanderungsrate = 1 - Bindungsrate
- Abwanderungsrate nimmt mit der Zeit ab (langjährige Kunden haben mehr Wert als Neukunden)

Markov-Kette

- Eine Markov-Kette ist ein stochastischer Prozess. (z.B. Kauf, Kundenverbleib)
- Ziel bei der Anwendung von Markov-Ketten ist es, Wahrscheinlichkeiten für das Eintreten zukünftiger Ereignisse anzugeben. (z.B. Kaufwahrscheinlichkeit -> Erwarteter Gewinn)
- Eine Markov-Kette ist definiert, dass durch Kenntnis einer nur begrenzten Vorgeschichte (hier: nur aktueller Zustand des Systems) ebenso gute Prognosen über die zukünftige Entwicklung möglich sind wie bei Kenntnis der gesamten Vorgeschichte des Prozesses. (z.B. Kundenstatus als potenzieller Kunde, aktiver Kunde oder abgewanderter Kunde)
- Modellierung mit diskreter Zeit (z.B. Monate oder Jahre)
- Modellierung mit endlicher Zustandsmenge (z.B. Kundenstatus)

Markov-Ketten Ansatz zur Modellierung des CLV – Visualisierung

- $p_r: Prospect$, $C: Customer$, $F: Former\ Customer$ $\quad P = p_{i,j} = \begin{bmatrix} 0 & p_a & 1-p_a \\ 0 & p & 1-p \\ 0 & 0 & 1 \end{bmatrix}$, $Revenue\ Vec = R = \begin{bmatrix} -A \\ NC - M \\ 0 \end{bmatrix}$



- Übergangsmatrix P: $p_{i,j}$ ist die Wahrscheinlichkeit des Zustandsübergangs von i nach j in der folgenden Periode.
- Revenue Vector R: r_i ist der Gewinn im Zustand i innerhalb einer Zeitperiode.
- A: Akquisitionskosten pro Kunde und Zeitperiode («acquisition cost»)
- NC: Umsatz pro Kunde und Zeitperiode («net contribution»)
- M: Kundenbindungskosten pro Kunde und Zeitperiode («remarketing cost»)

Potenzierung P

- $P^0 = I = Einheitsmatrix$
- $P^1 = P$ Übergangswahrscheinlichkeit nach einer Periode
- $P^2 = P \cdot P$ Übergangswahrscheinlichkeit nach zwei Perioden
- $P^n =$ Übergangswahrscheinlichkeiten nach n Perioden

Beispiel

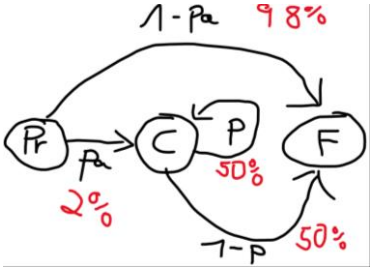
Ein Restaurant möchte den CLV von Interessenten im Vergleich zu Bestandskunden ermitteln. Annahmen:

- Kosten von CHF 0.30 pro Flyer für Werbungneukungen.
- 2% Reaktion auf Flyer.
- Durchschnittlicher Umsatz pro Kunde von CHF 35 und Kosten von CHF 30, was Nettobeitrag CHF 5 ergibt.
- 5% Treuerabatt für Bestandskunden mit 50%iger Rückkehrwahrscheinlichkeit.
- Interessenten reagieren entweder auf den Flyer oder nicht und werden dann verlorene Kunden.
- Verlorene Kunden können nicht zurückgewonnen werden.
- Keine Abzinsung zukünftiger Zahlungen.

Lösung

$p_r: Prospect$, $C: Customer$, $F: Former\ Customer$ $\quad P = p_{i,j} = \begin{bmatrix} 0 & 0.02 & 0.98 \\ 0 & 0.5 & 0.5 \\ 0 & 0 & 1 \end{bmatrix}$, $Revenue\ Vec = R = \begin{bmatrix} -0.30 \\ 3.25 \\ 0 \end{bmatrix}$

$M = 35 \cdot 0.05 = 1.75,\ NC - M = 5 - 1.75 = 3.25$



Bestandskunde zu Beginn («Monat 0»)

$$P^0 \cdot R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -0.30 \\ 3.25 \\ 0 \end{bmatrix} = \begin{bmatrix} -0.30 \\ 3.25 \\ 0 \end{bmatrix}$$

Bestandskunde nach einem Monat

$$P^0 \cdot R + P^1 \cdot R = \begin{bmatrix} -0.30 \\ 3.25 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0.02 & 0.98 \\ 0 & 0.5 & 0.5 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -0.30 \\ 3.25 \\ 0 \end{bmatrix} = \begin{bmatrix} -0.235 \\ 4.875 \\ 0 \end{bmatrix}$$

Übergangswahrscheinlichkeiten für zwei Monate

$$P^0 \cdot R + P^1 \cdot R + P^2 \cdot R = \begin{bmatrix} -0.235 \\ 4.875 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0.01 & 0.99 \\ 0 & 0.25 & 0.75 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -0.30 \\ 3.25 \\ 0 \end{bmatrix} = \begin{bmatrix} -0.2025 \\ 5.6875 \\ 0 \end{bmatrix}$$

CLVs (=Gewinnerwartungswerte) nach T Zeitperioden

- $V^T = \sum_{t=0}^{T} [(1+d)^{-1} P]^t R$
- V^T : Spaltenvektor der erwarteten CLVs nach T Zeitperioden
- T: Anzahl Zeitperioden
- d: Zinsrate pro Zeitperiode
- R: Gewinnvektor
- t: Zeitperiodenindex
- P: Übergangsmatrix

CLVs (=Gewinnerwartungswerte) nach unendlicher Anzahl von Zeitperioden (nicht prüfungsrelevant)

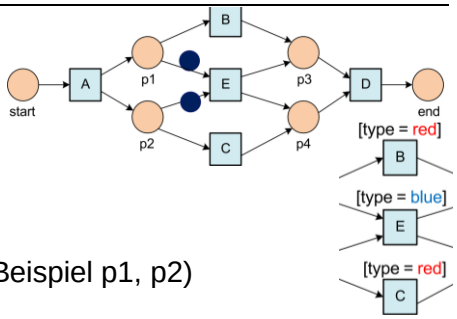
$$V \equiv \lim_{T \to \infty} V^T = \{I - (1+d)^{-1} P\}^{-1} R$$

Decision Point Minings

Voraussetzung Decision Point Mining

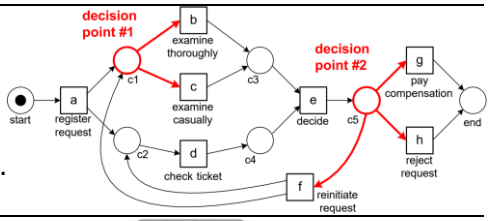
- Eingabe:
  - Ereignisprotokoll
  - Prozessmodell
- Vermutung: Protokoll und Modell wurden abgeglichen.
  - Mapping der Aktivitätsnamen in Protokoll und Modell.
  - Jede Spur kann mit einem Pfad durch das Modell zugeordnet werden.

Beispiel



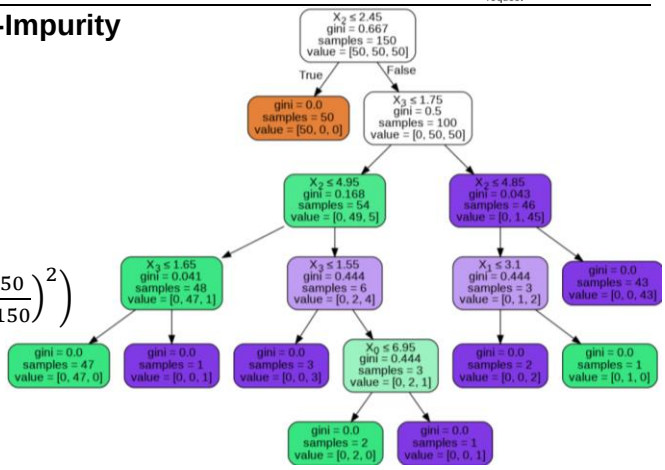
- Gibt blaue und rote Tokens im Beispiel
- Rot verläuft A→B→C→D, blau hingegen A→E→D
- Guards sorgen dafür, dass der richtige Weg eingeschlagen wird (unter der Annahme, dass Fälle einen Datenattributtyp mit dem Wert Blau oder Rot haben)
- Wenn rot dann B+C; Wenn blau dann E;
- Stellen mit mehreren Ausgangsbögen bilden Entscheidungspunkte (Im Beispiel p1, p2)

Decision Point Analysis - Klassifikationsproblem



- Antwortvariable: welche Aktivität soll ausgeführt werden (b oder c)
- Prädiktorvariablen: alles, was mit dem Kontext dieser Entscheidung zusammenhängt, z. B. das Alter des Kunden, die Ressource, die die vorherige Aktivität ausgeführt hat, das Wetter usw.

Klassifikation mit Entscheidungsbaum Berechnung Gini-Impurity



- Berechnung Gini-Impurity (bei Kategorialvariablen):
- $I_G(p) = 1 - \sum_{i=1}^{J} p_i^2$
- I_G: Gini Impurity
- p: Wahrscheinlichkeitsverteilung
- J: Anzahl Kategorien
- p_i: Wahrscheinlichkeit der Kategorie i
- **Beispiel:** $I_G(p) = 1 - \sum_{i=1}^{J} p_i^2 = 1 - \left(\left(\frac{50}{150}\right)^2 + \left(\frac{50}{150}\right)^2 + \left(\frac{50}{150}\right)^2\right)$
- $= 1 - 3 \cdot \frac{1}{9} = \frac{2}{3}$
- p_{left}, p_{right}: Wahrscheinlichkeit links/rechts
- $I_{G,A}(p)$ = Gini Impurity einer Teilung mit Attribut A
- n: Anzahl Proben in Knoten
- A: Aufteilungsattribut
- Wird Datensatz mit Wahrscheinlichkeitsverteilung p eines übergeordneten Knotens auf einem Attribut A in zwei Teilmengen mit den Wahrscheinlichkeitsverteilungen p_{left} und p_{right} mit den Größen n_{links} und n_{rechts} kann die Gini Impurity einer Aufteilung kann definiert werden als $I_{G,A}(p) = \frac{n_{left}}{n} I_G(p_{left}) + \frac{n_{right}}{n} I_G(p_{right})$
- **Beispiel erste Unterteilung links:** $I_G(p) = 1 - \left(\left(\frac{50}{50}\right)^2 + \left(\frac{0}{50}\right)^2 + \left(\frac{0}{50}\right)^2\right) = (1 - 1) = 0$
- **Beispiel erste Unterteilung rechts:** $I_G(p) = 1 - \left(\left(\frac{0}{100}\right)^2 + \left(\frac{50}{100}\right)^2 + \left(\frac{50}{100}\right)^2\right) = 1 - \left(0 + \frac{1}{4} + \frac{1}{4}\right) = \frac{1}{2}$
- **Beispiel:** $I_{G,A}(p) = \frac{n_{left}}{n} I_G(p_{left}) + \frac{n_{right}}{n} I_G(p_{right}) = \frac{50}{150} \cdot 0 + \frac{100}{150} \cdot \frac{1}{2} = \frac{1}{3}$

Gini-Gewinn

- Die Gini-Gewinn misst die Verringerung der Verunreinigungen bei einer Aufteilung und sollte maximiert werden: $GiniGain(A) = I_{G(p)} - I_{G,A}(p)$
- Beispiel: $\frac{2}{3} - \frac{1}{3} = \frac{1}{3}$

Entscheidungspunktanalyse - Klassifizierungsproblem

- Beispiel Ausgabenprozess: Gründlich oder beiläufig prüfen
- Mit welchem Split beginnen?
- Impurity im Ausgangsknoten: $I_G(p) = 1 - \left(\left(\frac{4}{9}\right)^2 + \left(\frac{5}{9}\right)^2\right) = 1 - \frac{41}{81} = \frac{40}{81}$
- Casually Examined: $\frac{3}{4} \cdot (1 - \left(\left(\frac{2}{3}\right)^2 + \left(\frac{1}{3}\right)^2\right) + \frac{6}{9} \cdot \left(1 - \left(\left(\frac{2}{6}\right)^2 + \left(\frac{4}{6}\right)^2\right)\right) = \frac{3}{9} \cdot \frac{4}{9} + \frac{6}{9} \cdot \frac{4}{9} = \frac{36}{81}$
- Year 2023: $\frac{4}{9} \cdot \left(1 - \left(\left(\frac{1}{4}\right)^2 + \left(\frac{3}{4}\right)^2\right)\right) + \frac{5}{9} \cdot \left(1 - \left(\left(\frac{3}{5}\right)^2 + \left(\frac{2}{5}\right)^2\right)\right) = \frac{4}{9} \cdot \frac{6}{16} + \frac{5}{9} \cdot \frac{12}{25} = \frac{13}{30}$

Casually_Examined	Amount_High	Year_2023	Examination
0	1.0	0.0	Thorough
0	0.0	1.0	Casual
0	1.0	1.0	Casual
1	1.0	1.0	Thorough
0	1.0	0.0	Thorough
0	1.0	0.0	Casual
1	1.0	0.0	Casual
1	1.0	0.0	Thorough
0	0.0	1.0	Casual

Kollaboratives Filtern

Herausforderung: Welchen Wert hat eine Dienstleistung für einen Kunden?

- Soll welches Angebot wählen? Spezifikation Dienstleistung (Service Value Proposition / Service Package)
- Wert
  - Allgemeine Hypothese: Der Kunde wählt das Angebot mit dem höchsten «Wert»
  - Hinweis: Im Prinzip ist der Wert eine mehrdimensionale Variable!
  - Was sollte also «höchster Wert» bedeuten?
- Möglicher Ansatz: explizite Modellierung der Präferenz
  - Das ist eine eindimensionale Größe!
  - Es integriert alle Wertdimensionen

Definition Kollaboratives Filtern

DEFINITION

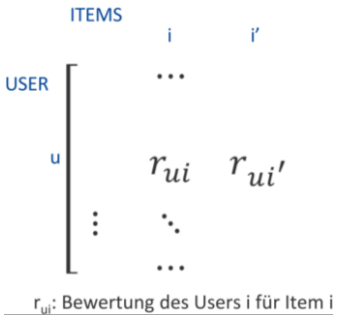
- Beim kollaborativen Filtern werden Verhaltensmuster von Benutzergruppen ausgewertet, um auf die Interessen Einzelner zu schließen.
- Ziel: automatische Schätzung (Filtern) von Benutzerinteressen bzw. -bewertungen.
- Annahme: Wenn zwei Personen dieselben Vorlieben zu ähnlichen Produkten haben, sind sie sich auch in anderen Produkten einig.
- Daher auch der Begriff Kollaboration: Möchte man wissen, welche Meinung ein Nutzer A zu einem Artikel hat, betrachtet man welche Meinung andere Nutzer zu diesem Artikel haben.
- Die anderen Nutzer „arbeiten zusammen“ um die Frage zu lösen welcher Meinung wohl Nutzer A ist.

BEISPIELE

- Amazon
- eBay
- Google News
- iTunes
- Netflix
- Spotify

Datenbasis User Rating Matrix – URM

- Matrix der von Nutzern gegebenen Bewertungen.
- **Einziger Input für den CF Algorithmus.**
- Ein Beispiel für von Nutzern explizit gegebenen Bewertungen ist die Bewertung 1 bis 5 (Sterne.)
- Beispiel für implizite Bewertungen ist 1 für «gekauft» und 0 für «nicht gekauft».



Beispiel User Rating Matrix movielens Basisdaten → → → →

	userId	movieId	rating
0	1	1	4.0
1	1	3	4.0
2	1	6	4.0
3	1	47	5.0
4	1	50	5.0
...	...	...	...
100831	610	166534	4.0
100832	610	168248	5.0
100833	610	168250	5.0
100834	610	168252	5.0
100835	610	170875	3.0

100836 rows × 4 columns

Präferenzmessung Explizite und explizite Ansätze

Explizit: Vom Nutzer formulierte Präferenz.

- Rating
- Review
- Vote

Implizit: Vom Nutzerverhalten abgeleitete Präferenz

- Click
- Purchase
- Follow

Nutzerbasiertes Kollaboratives Filtern

- **Kernidee:** «Schlage vor, was ähnliche Kunden gut fanden.»
- Operationalisierung der Nutzerähnlichkeit → «Vergleich der **Zeilen** der User Rating Matrix»
- $s_{uv} = \frac{\sum_i (r_{ui} \cdot r_{vi})}{\sqrt{\sum_i r_{ui}^2} \cdot \sqrt{\sum_i r_{vi}^2}}$ Kosinus-Ähnlichkeit, liegt zwischen 1 (identisch) und 0 (unabhängig).
- u, v: User
- i: Item Index
- r_{ui}: Bewertung des Users u für Item i
- s_{uv}: Ähnlichkeit zwischen den Usern u und v

Rechenbeispiel Nutzerähnlichkeit beim Speiseangebot

→ 3: gut, 2: neutral und 1: schlecht

→ $s_{Alessio,Gabi} = ?$

	Fleisch	Vegan	Vegetarisch
Alessio	1.0	??	3.0
Gabi	1.0	3.0	3.0
Klaus	3.0	1.0	1.0

$$s_{Alessio,Gabi} = \frac{r_{Alessio,Fleisch} \cdot r_{Gabi,Fleisch} + r_{Alessio,Vegi} \cdot r_{Gabi,Vegi}}{\sqrt{r_{Alessio,Fleisch}^2 + r_{Alessio,Vegi}^2} \cdot \sqrt{r_{Gabi,Fleisch}^2 + r_{Gabi,Vegi}^2}} = \frac{1 \cdot 1 + 3 \cdot 3}{\sqrt{1^2 + 3^2} \cdot \sqrt{1^2 + 3^2}} = \frac{10}{\sqrt{10} \cdot \sqrt{10}} = \frac{10}{10} = 1$$

$$s_{Alessio,Klaus} = \frac{1 \cdot 3 + 3 \cdot 1}{\sqrt{1^2 + 3^2} \cdot \sqrt{3^2 + 1^2}} = \frac{3}{\sqrt{10} \cdot \sqrt{10}} = \frac{6}{10} = \frac{3}{5}$$

→ NA’s werden hier ausgeschlossen, resp mit 0 gerechnet.

Nutzerbasiertes Kollaboratives Filtern – Schätzung der Bewertung

- $\hat{r}_{ui} = \frac{\sum_{v \in KNN(u)}(r_{vi} \cdot s_{vu})}{\sum_{v \in KNN(u)}(s_{vu})} = \frac{\sum_v(r_{vi} \cdot s_{vu})}{\sum_v(s_{vu})}$
- u: User für den geschätzt wird
- $\hat{r}_{ui}$: geschätzte Bewertung für User u und Item i
- v: User Index
- $KNN(u)$: K zu u ähnlichste Nutzer
- r_{vi}: Bewertung des Users v für Item i
- s_{vu}: Ähnlichkeit zwischen Usern v und u

Rechenbeispiel Nutzerbasiertes CF – Berücksichtigung aller Kunden («Summe über alle Kunden»)

→ $s_{Alessio,Gabi} = 1$ und $s_{Alessio,Klaus} = \frac{3}{5}$

→ $\hat{r}_{Alessio,Vegan} = ?$

$$\hat{r}_{Alessio,Vegan} = \frac{r_{Gabi,Vegan} \cdot s_{Gabi,Alessio} + r_{Klaus,Vegan} \cdot s_{Klaus,Alessio}}{s_{Gabi,Alessio} + s_{Klaus,Alessio}} = \frac{3 \cdot 1 + 1 \cdot \frac{3}{5}}{1 + \frac{3}{5}} = \frac{\frac{18}{5}}{\frac{8}{5}} = \frac{9}{4}$$

Beispiel Nutzerbasiertes CF – Berücksichtigung des ähnlichsten Kunden

→ $s_{Alessio,Gabi} = 1 = max$ ist somit der ähnlichste Kunde

→ $\hat{r}_{Alessio,Vegan} = ?$

$$\hat{r}_{Alessio,Vegan} = \frac{r_{Gabi,Vegan} \cdot s_{Gabi,Alessio}}{s_{Gabi,Alessio}} = \frac{3 \cdot 1}{1} = 3$$

Itembasiertes Kollaboratives Filtern

- Bekannt aus dem Onlineeinkauf: Nachdem Objekt in Warenkorb hinzugefügt wird erscheint die Meldung → «wird oft zusammen gekauft» mit neuen Produktvorschlägen
- «Schlage Produkte vor, die ähnlich sind zu für mich interessanten Produkten.»
- «Vergleich der **Spalten** der User Rating Matrix»
- $s_{ij} = \frac{\sum_u(r_{ui} \cdot r_{uj})}{\sqrt{\sum_u r_{ui}^2} \cdot \sqrt{\sum_u r_{uj}^2}}$ Kosinus-Ähnlichkeit, liegt zwischen 1 (identisch) und 0 (unabhängig).
- u: User Index
- i, j: Items
- r_{ui}: Bewertung des Users u für Item i
- s_{ij}: Ähnlichkeit zwischen den Items i und j

Rechenbeispiel Itemähnlichkeit

→ $s_{Fleisch,Vegan} = ?$

$$s_{Fleisch,Vegan} = \frac{r_{Gabi,Fleisch} \cdot r_{Gabi,Vegan} + r_{Klaus,Fleisch} \cdot r_{Klaus,Vegan}}{\sqrt{r_{Gabi,Fleisch}^2 + r_{Klaus,Fleisch}^2} \cdot \sqrt{r_{Gabi,Vegan}^2 + r_{Klaus,Vegan}^2}} = \frac{1 \cdot 3 + 3 \cdot 1}{\sqrt{1^2 + 3^2} \cdot \sqrt{3^2 + 1^2}} = \frac{6}{\sqrt{10} \cdot \sqrt{10}} = \frac{6}{10} = \frac{3}{5}$$

$$s_{Vegi,Vegan} = \frac{3 \cdot 3 + 1 \cdot 1}{\sqrt{3^2 + 1^2} \cdot \sqrt{3^2 + 1^2}} = \frac{10}{\sqrt{10} \cdot \sqrt{10}} = \frac{10}{10} = 1$$

Itembasiertes Kollaboratives Filtern – Schätzung der Bewertung

- $\hat{r}_{ui} = \frac{\sum_{j \in KNN(i)}(r_{uj} \cdot s_{ij})}{\sum_{j \in KNN(i)}(s_{ij})} = \frac{\sum_v(r_{vi} \cdot s_{vu})}{\sum_v(s_{vu})}$
- u: User für den geschätzt wird
- $\hat{r}_{ui}$: geschätzte Bewertung für User u und Item i
- j: Item Index
- $KNN(i)$: K ähnlichste Items zu Item i
- r_{uj}: Bewertung des Users u für Item j
- s_{ij}: Ähnlichkeit zwischen Item i und j

Rechenbeispiel Itembasiertes CF – Berücksichtigung aller Items («Summe über alle Items»)

→ $s_{Fleisch,Vegan} = \frac{3}{5}$ und $s_{Vegi,Vegan} = 1$

→ $\hat{r}_{Alessio,Vegan} = ?$

$$\hat{r}_{Alessio,Vegan} = \frac{r_{Alessio,Fleisch} \cdot s_{vegan,Fleisch} + r_{Alessio,Vegi} \cdot s_{vegan,vegi}}{s_{vegan,Fleisch} + s_{vegan,vegi}} = \frac{1 \cdot \frac{3}{5} + 3 \cdot 1}{\frac{3}{5} + 1} = \frac{9}{4}$$

Herausforderungen beim Kollaborativen Filtern

- Kleine Vergleichsmenge
- Festlegung der Anzahl an verwendeten Nachbarn
- Positivitäts-/Negativitätsverzerrung bei Bewertung
- Berechnungsaufwand bei der Schätzung

Problematik bei kleiner Vergleichsmenge – Nutzung eines Shrink Terms

- Problem bei der Nutzung der Kosinus-Ähnlichkeit: Vertrauen Sie Ähnlichkeiten nur, wenn viele Benutzer die gleiche Meinung teilen. => Nutzung eines Shrink Terms H (gute Werte zwischen 1 und 10)

Nutzerähnlichkeit:

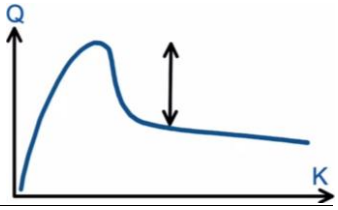
$$s_{uv} = \frac{\sum_i(r_{ui} * r_{vi})}{\sqrt{\sum_i r_{ui}^2} * \sqrt{\sum_i r_{vi}^2} + H}$$

Itemähnlichkeit:

$$s_{ij} = \frac{\sum_u(r_{ui} * r_{uj})}{\sqrt{\sum_u r_{ui}^2} * \sqrt{\sum_u r_{uj}^2} + H}$$

Festlegung der Anzahl an Vergleichsnutzern/-items bei der Ähnlichkeitsbestimmung («K in KNN»)

- Qualität ist abhängig von der Anzahl an verwendeten Nachbarn K
- Alternativ zur direkten Anzahl an berücksichtigen Vergleichsnutzern/items lässt sich auch eine minimale Ähnlichkeit als Schwellwert definieren. Alle Vergleichsnutzer/items mit einer Ähnlichkeit unter diesem Schwellwert werden ignoriert.



Positivitäts- oder Negativitätsverzerrung bei expliziten Bewertungen (nutzerbasiertes CF)

- Bei expliziten Bewertungen müssen wir die unterschiedliche Art und Weise berücksichtigen, wie Benutzer ihre Meinung äußern («Positivitäts- oder Negativitätsverzerrung»)

PEARSON CORRELATION COEFFICIENT

$$s_{uv} = \frac{\Sigma_i((r_{ui} - \bar{r}_u) * (r_{vi} - \bar{r}_v))}{\sqrt{\sum_i (r_{ui} - \bar{r}_u)^2} * \sqrt{\sum_i (r_{vi} - \bar{r}_v)^2}}$$

u,v: User
r_{ui}: Bewertung des Users i für Item i
$\bar{r}_u$: Mittlere Bewertung des Users u
s_{uv}: Ähnlichkeit zwischen Nutzern u und v

Schätzung der Bewertung

- Die Bewertung, die für einen Nutzer schätzen, beruht auf der Bewertung ähnlicher Nutzer plus der durchschnittlichen Bewertung dieses Nutzers.
- **Rating Delta:** $\hat{r}_{ui} - \bar{r}_u = \frac{\sum_{v \in KNN(u)} \left((r_{vi} - \bar{r}_v) \cdot s_{vu} \right)}{\sum_{v \in KNN(u)} (s_{vu})}$

Nutzerbasiertes und Itembasiertes Kollaboratives Filtern unterscheiden sich im Berechnungsaufwand

NUTZERBASIERTES CF

- Sollen für einen unbekannten Nutzer Bewertungen geschätzt werden, müssen die Nutzerähnlichkeiten s_{vu} **neu berechnet** werden.
- Hoher Berechnungsaufwand bei Schätzung
- Gedächtnisbasierter Schätzansatz: memory base

ITEMBASIERTES CF

- Sollen für einen unbekannten Nutzer Bewertungen geschätzt werden, müssen die Itemähnlichkeiten s_{ij} **NICHT neu berechnet** werden.
- Geringer Berechnungsaufwand bei Schätzung
- Modellbasierter Schätzansatz: model based

Stand der Arbeit

SW	Vorlesung	AB/Aufgabe	Skript	ZF
1	☒	☒	☐	☒
2	☒	☒	☐	☒
3	☒	☒	☐	☒
4	☒	☒	☐	☒
5	☒	☒	☐	☒
6	☒	☒	☐	☒
7	☒	☒	☐	☒
8	☒	☒	☐	☒
9	☒	☒	☐	☒
10	☒	☒	☐	☒
11	☒	☒	☐	☒
12	☒	☒	☐	☒
13	☐	☐	☐	☐
14	☐	☐	☐	☐

Modulauflagen

- SEP 100 %
- Schriftlich papierbasiert (90 Minuten, rund 90 Punkte)
- „Closed Book“ (Fokus der Prüfung: analytisches Verständnis der vorgestellten Modelle und Methoden)
- Eine Formelsammlung mit allen relevanten Formeln wird vorher publiziert und bereitgestellt.
- Notenbonus aus den Übungen wird bei der Prüfungsbewertung berücksichtigt.
- Die (gerundete) Prüfungsnote stellt die Modulnote dar.
- Hilfsmittel: Es sind nicht-programmierbare Taschenrechner erlaubt. Prüfung ist so konzipiert, dass für die meisten Aufgaben kein Taschenrechner benötigt wird.
- Bonus: Falls Sie 70% der Aufgaben erfolgreich eingereicht haben, erhalten Sie einen Bonus von 0.5 auf die Endnote der Prüfung.