Digital Marketing
Environment II

BFH HS24 - CAS AI – Gruppe 3
H. Gehrer, N. Hryciuk, S. Mavilio, M. Näpflin, H. Wermelinger

Notizen

18.03.2025

Vorgehensweise

Ausgangslage:
Agent x Environment
Challenge

Rethink
Business
Flow

Develop and
Observe

Ziel:
Realistisches Environment
Leichtgewichtiges Environment
Leichtgewichtiger Agent
Performanter Agent

Kein Ziel:
Optimierung nur durch
(brute force) Hyperparameter

Risikobereitschaft und Microfunding

Beware the Greedy Agent

Vorgaben durch Kunden

Kurse

Geld

Zeit

Business Case Vorgaben

Latenz
(Adds werden erst 1..n Tage nach Bestellung gesehen, im schlimmsten Fall erst nach Deadline Kampagen)

Expected Agent Behavior (Spannweite)

Adäquate und nachvollziehbare Empfehlungen

- Geld in kleinen Mengen über eine angemessene Zeit investieren

...

Creative Cheating

- Agent macht es auf eine völlig andere Art und Weise, wie wir es erwartet haben, aber trotzdem erfolgreich

...

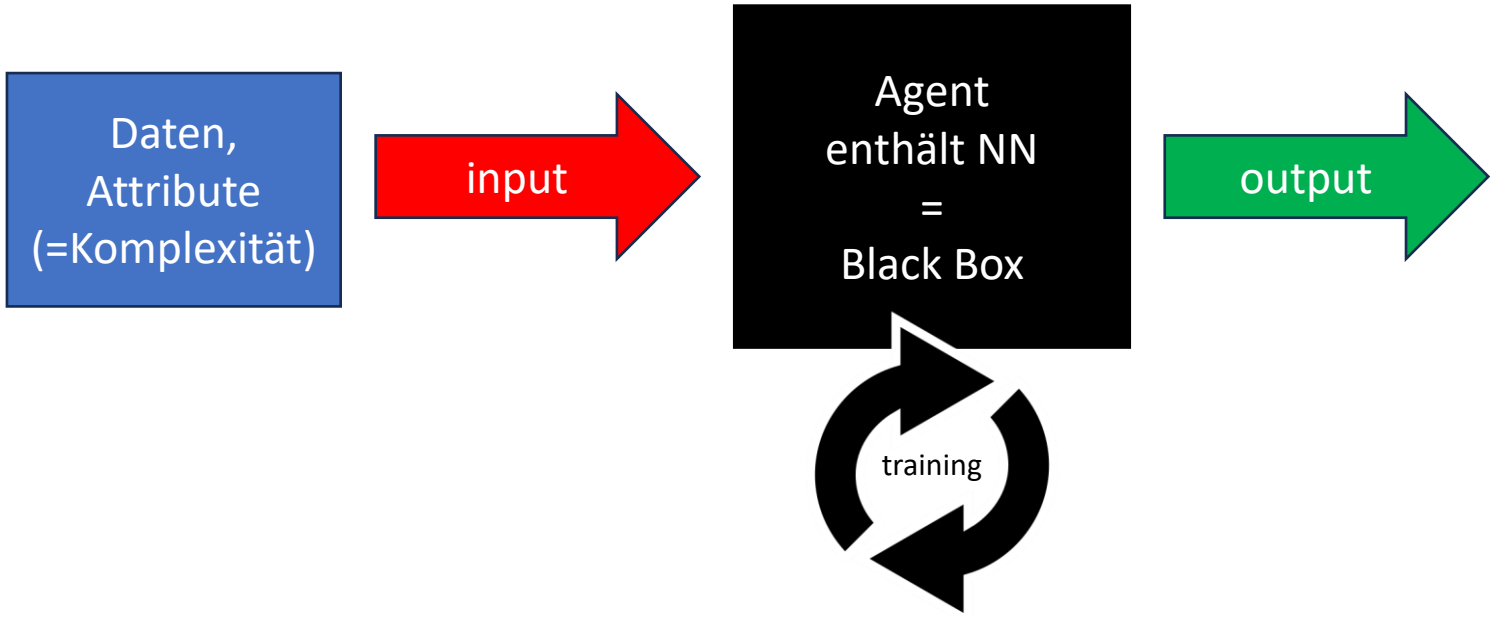
Risikoreiches Verhalten

- Alles auf einmal ausgeben
- Alles auf eine Karte setzen
- Alles auf einmal auf eine Karte setzen

Massnahme 1:
Variable ‚Risikobereitschaft'
RB [0.1 … 0.9] die am Anfang eines Trainings gesetzt wird.
D.h. Agent darf pro Schritt max. RB*max. Invest einsetzen

Massnahme 2:
Microfunding
Kampagne wird nicht von Anfang an frei gegeben, sondern in einzelnen kleinen Schritten finanziert
(NB: könnte USP von Agentur sein)

Komplexität und Performance



Performance (Rechenzeit) und
Output-Qualität abhängig von
Komplexität der Daten
(Menge, Qualität, Auswahl usw.)

Regelbasiert und/oder NN-Crunching

Ansatz nasty

Keywordliste

Key words	Adds bestellt
A	0
B	0
C	0
D	3
F	0
G	7
H	0
I	42
J	0

input

Black Box

output

«Positiven Reward gibt es nur dann, wenn Agent nicht in einem bereits bestellten aber noch nicht verbuchten Keyword investiert!»

Ansatz friendly

Keywordliste

Key words	Adds bestellt
A	0
B	0
C	0
D	3
F	0
G	7
H	0
I	42
J	0

cleaning

Wir geben nur die Keywords weiter, wo der Agent überhaupt investieren darf.

Keywordliste

Key words	Adds bestellt
A	0
B	0
C	0
F	0
H	0
J	0

input

Black Box

output

Zeilen- vs. Kontext-Wissen

Zeilenorientierten Ansatz

- Loop Epoch (eine Kampagne)
  - Loop Schritt (jede Stunde)
    - Keywordliste einlesen
      - Loop Keywordliste
        - Zeile in Blackbox tun
        - Empfehlung kalkulieren
        - Reward berechnen

Agent empfiehlt pro Zeile,
aber
nur gemäss Analyse derselben Zeile

Kontextorientierten Ansatz

- Loop Epoch (eine Kampagne)
  - Loop Schritt (jede Stunde)
    - Keywordliste einlesen
    - Kontext berechnen (Durchschnitt, Median, Varianz usw.)
      - Loop Keywordliste
        - Zeile in Blackbox inkl. Kontext tun
        - Empfehlung kalkulieren
        - Reward berechnen

Agent empfiehlt pro Zeile,
aber
gemäss Analyse derselben Zeile
und des Kontextes (alle Zeilen)

Zeilen- vs. Kontext-Wissen

```python
def generate_synthetic_data(num_samples=1000):
    data = {
        "keyword": [f"Keyword_{i}" for i in range(num_samples)],
        "competitiveness": np.random.uniform(0, 1, num_samples),
        "difficulty_score": np.random.uniform(0, 1, num_samples),
        "organic_rank": np.random.uniform(1, 10, num_samples),
        "organic_clicks": np.random.randint(50, 5000, num_samples),
        "organic_ctr": np.random.uniform(0.01, 0.3, num_samples),
        "paid_clicks": np.random.randint(10, 3000, num_samples),
        "paid_ctr": np.random.uniform(0.01, 0.25, num_samples),
        "ad_spend": np.random.uniform(10, 10000, num_samples),
        "ad_conversions": np.random.uniform(0, 500, num_samples),
        "ad_roas": np.random.uniform(0.5, 5, num_samples),
        "conversion_rate": np.random.uniform(0.01, 0.3, num_samples),
        "cost_per_click": np.random.uniform(0.1, 10, num_samples),
        "cost_per_acquisition": np.random.uniform(5, 500, num_samples),
        "previous_recommendation": np.random.choice([0, 1], size=num_samples), unsere mepfehlung 0/1
        "impression_share": np.random.uniform(0.1, 1.0, num_samples), wieviel die seite erscheint, wieviel die adss erscheinen
        "conversion_value": np.random.uniform(0, 10000, num_samples)
    }
    return pd.DataFrame(data)
```

Reward Funktion
«Wenn Invest in eine Zeile empfohlen wird, bei der die Differenz Zeilenwert zu Durschnitt rel. klein, dann ist Reward negativ (Penalty)

Reward Debugging

Business Rules

Reward Rules

Feed in Environment

Feed in Agent

„Kann man Business Rules
1:1 in Reward Rules umsetzen?“

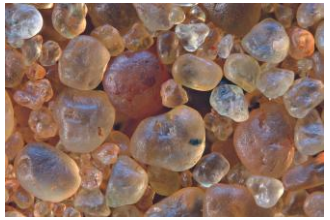
Nein! Beim Aussprechen der Business-Rules werden nur gewisse Daten berücksichtigt. Der Agent berücksichtigt aber immer alle Parameter. Dies kann zu kontradiktorischen Ergebnissen führen.

Reward Debugging

Project02 > Grp3 > _digital_marketing_env.py > SiteAnalyticsSimulation > run

```python
class DigitalMarketingEnv(EnvBase):

    def _compute_reward(self, action, sample:torch.tensor):
        is_buy = action != 0
        bought_ads = action

        overall_success = self.course_bookings # Ganzzahl 0-25
        available_budget = 100 / self.budget * self.spent_amount
        overall_success += available_budget # Bonus für verfügbares Budget in %
        if self.course_bookings >= self.maximum_class_size:
            overall_success += 100 # Bonus für erfolgreichen Abschuss
        if self.spent_amount > self.budget:
            overall_success -= 100  # Penalty für Budgetüberschreitung
        if self.data_builder.current_generation >= self.generations_per_epic:
            overall_success -= 100 # Penalty für Kampagnenabbruch

        difficulty = sample[DIFFICULTY_SCORE_FIELD]
        organic_rank = sample[ORGANIC_RANK_FIELD]
        is_search_prefered = organic_rank <= 5

        difficulty_reward = (10 - difficulty) / 10
        if not is_buy and not is_search_prefered:
            difficulty_reward *= -1

        if is_buy and is_search_prefered:
            # Kauf, obwohl in den Suchergebnissen gut platziert
            organic_rank_penalty = (5 - organic_rank) * bought_ads * -1
        else:
            organic_rank_penalty = 0

        ad_continget = sample[AD_CONTINGENT_FIELD]
        if is_buy and ad_continget > 0:
            # Kauf, obwohl noch nicht alle Inserate verkauft sind.
            ad_hoarding_penalty = ad_continget * bought_ads * -1
        else:
            ad_hoarding_penalty = 0

        iteration_count_penalty = self.data_builder.current_generation * -0.1

        reward = overall_success + difficulty_reward + organic_rank_penalty + ad_hoarding_p
        self.data_builder.log_reward(self.current_keyword, reward) # prüfen: müsste hier ni

        if self.data_builder.current_generation % 10 == 0:
            log.app_logger().info(f" Reward: {reward} for action {action}: (overall_succes

        return reward
```

Während des Trainierens …



Aaargh! Ein Sandkorn im System …
Agent lernt nicht hinreichend gut!
…
‚Zu viele Köche verderben den Brei‘
…
Wie finden wir raus, ob es kontradiktorische Regeln gibt, welche ein Lernen durch den Agenten verhindern?

→ **Reward-Debugging**

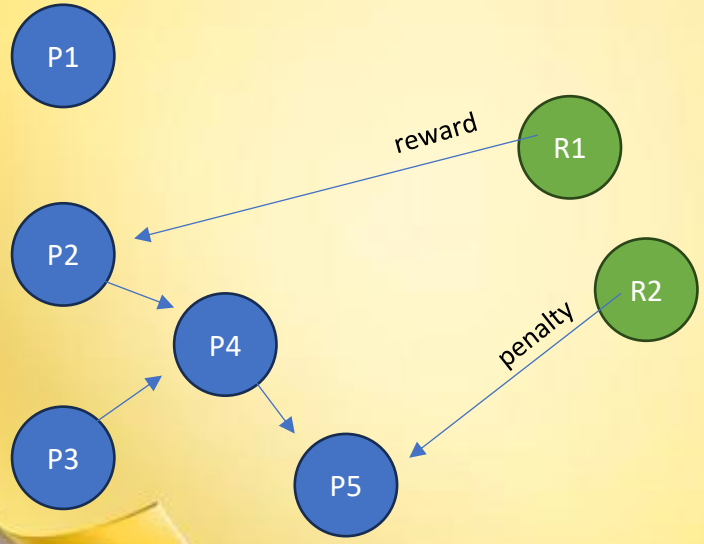
Reward Debugging

Divide-and-conquer
d.h. Agent mit einzelnen Rewards trainieren, einzelne Rewards nur schrittweise neu hinzufügen.

Reward Monitoring.
Reward-Mechanismus während der Laufzeit transparenter / nachvollziehbarer machen.

Reward and Penalty Graph

D.h. Abhängigkeiten zwischen Daten (P) und Rewards (R) zeigen, um logische Wiedersprüche zu umgehen



...