Reinforcement Learning

Grundlagen Grundbegriffe Grundübungen

Alfatraining Kurs 2026-01 Dozent: Karsten Flügge

Inverse Reinforcement Learning (IRL)

Inverse Reinforcement Learning (IRL)

Inverse Reinforcement Learning

≈ Inverse Optimal Control
≈ Inverse Optimal Planning

Bei Inverse Reinforcement Learning IRL wird das klassische Reinforcement Learning Problem **auf den Kopf gestellt**:

Gegeben das beobachtete **Verhalten** einer versteckten Policy π, was war die **Motivation, was war** der **Reward** für die gewählten Aktionen?

Inverse Reinforcement Learning (IRL)

"Warum tut er das?"

Man kann oft die Spielregeln eines Spiels erkennen anhand des Spielverlaufs und der Spielweise!

Ein Kind beobachtet Fußball, kennt aber die Regeln nicht. Es sieht, dass Spieler versuchen, den Ball in ein bestimmtes **Tor** zu bringen, vermeiden **Fouls**, und jubeln nach Treffern. Es erschließt die *implizite Belohnungsstruktur* (Ziel: Tore, Regeln: nicht mit der Hand spielen etc.) aus den Handlungen.

IRL Beispiel: Abseits Regel. (💡 Regeln und Belohnungen sind verwandt!)

Inverse Reinforcement Learning (IRL)

Gegeben das beobachtete **Verhalten** einer versteckten Policy **π**, was war die Motivation, was war der **Reward** für die gewählten Aktionen?

Beispiele

- **Tier Beobachtung**
- **Psychologie**
- **Anthropologie**
- **Experimentelle Wissenschaft** (Physik, Chemie, …)
- **Verkehrsflüsse** (versteckter reward: Döner)
- **Spiele / Gegner** (z.B. Fussball)
- Bildung: Motivation der Lehrer (Geld, Ideale, …)

Beispiele Inverse RL

Gegeben das beobachtete **Verhalten** einer Policy π, was war die Motivation, was war der **Reward** für die gewählten Aktionen?

Beispiele

- **Ökonomie** (Homo Economicus vs behavioral economics)
- Soziologie society (organic behavior), Gruppendynamik
- Trumpologie
- Ameisen Bienenstock Tanz
- Biologie: Zellverhalten (bei chemischen Agenten)
- Verkehr: Slalom/Hügel/Blitzer
- Astronomy Galaxy verhalten (reward ≈ physikalische Kräfte / Regeln (black matter!)
- Medizin (was war **Agent**, der die Krankheit ausgelöst hat: Viren, Bakterien)
- Autonomes Fahren (was ist Motivation der **anderen Fahrzeuge / Fußgänger (Agenten)**)
- Erdbeben? Agent = Planet?
- **Spiel: Gegenspieler verstehen** (⚡ reward von Umgebung bekannt? innerer reward?)
- => auf **Strategie** des Gegners reagieren!

Beispiele Inverse RL

Lerne Policy anderer Agenten

- Spiele lernen durch beobachten (andere beobachten)
- Aktienmarkt: Andere Akteure (welche **Strategie** fahren sie) DeepSeek : Liang Wen Feng
- Tiere: Hierarchie (alpha Wolf)
- Fußball Mannschaft 'ausspionieren' (alte Videos gucken) auch bei Boxen / alle Wettbewerbe?
- Spielautomaten (Umgebung vs Agenten?)
- Automatisches Reverse Engineering
- Sprache Lernen durch beobachten? (Sprache als policy? => LLM) 13te Krieger
- Autofahren Versicherungsbox **HUK24** policy der Fahrer => Rabatt!
- Verhalten Kopieren (Bionic) <=> Behavioural Cloning / Imitation Learning

Inverse Reinforcement Learning (IRL)

Formell:

Gegeben:

Ähnlich MDP ist IRL=**(S, A, <u>T</u>) <u>ohne reward</u>** MDP\R

Versteckte Policy **π**, welche wir versuchen zu finden/verstehen.

Dᵗ "**D**emonstrationen" also eine Menge von **D**ata:

Trajektorien $\mathbf{D}^{t}=(\tau_0, \ldots, \tau_n)$ $\tau=(\mathbf{s_0},\mathbf{a_0}, s_1, a_1, \ldots s_t)$ "Experten"

state-**feature** function ϕ= {fi : S => R} ≈ latent abstract compression

Gesucht:

Schätzung der **<u>Reward</u>funktion r(a | s)** der Umgebung.

Policy als optimal bezüglich R (Bellman Gleichungen / **π =** argmax_a E[R])

Schätzung der **Ursprünglichen Policy π** (*via reward!*) **"Apprentice Learning AL"**

MDP without reward: $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, \gamma)$

Apprentice(ship) Learning AL

Wenn man statt R gleich **π** sucht, nennt sich das **Apprentice**ship **Learning (AL)**

Idee: **Experten** machen verhalten vor, und wir als **Apprentice** imitieren.

Schätzung der **Rewardfunktion r(a | s)** der Umgebung.
Policy als optimal bezüglich R (Bellman Gleichungen / **π =** argmax_a E[R])
Schätzung der **Ursprünglichen Policy π** (*via reward!*) **"Apprentice Learning AL"**

MDP without reward: $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, \gamma)$

Unterschied Model Based

Model based: MDP (S, A, γ, p, R)
reward wird bei jedem step **gegeben** und immer **genutzt**, manchmal **gespeichert** im model
suche Policy via model estimator p(…) "**dynamics**" für simulations / rollouts
p(s,a)->s' oder p(s,a)->(s',r) p erlernt

orthogonal!

IRL Gegeben: MDP (S, A, Dᵗ) <u>ohne reward</u>
Dᵗ "Demonstrationen" also eine Menge von
Trajektorien $D^t=(\tau_0, \ldots, \tau_n)$ $\tau=(s_0,a_0, s_1, a_1, \ldots s_t)$ suche Policy

geht auch model based! p(S,A)->S'

Gesucht:

Schätzung der **Ursprünglichen Policy π** (via)
Schätzung der **Rewardfunktion r(a I s)** der Umgebung.

MDP without reward: $\mathcal{M} = (\mathcal{S}, \mathcal{A}, T, \gamma)$

Inverse Reinforcement Learning (IRL)

Können in RL nicht (sehr) verschiedene rewards zu ähnlichen (gleichen) Policies führen?
JA klar (Motivation: Geld, Ideologie, Liebe, ...)

Gibt es nicht also ganz verschiedene Interpretationen der Handlungen?
JA

Inverse Reinforcement Learning (IRL)

Können in RL nicht (sehr) verschiedene rewards zu ähnlichen (gleichen) Policies führen? JA!

Beispiel: Lunar Lander: reward für Landung oder
Beispiel: Reward shaping (*2)
Beispiel: Belohnung / Bestrafung (-100)

Gibt es nicht also ganz verschiedene Interpretationen der Handlungen? JA klar

Inverse Reinforcement Learning (IRL)

Können in RL verschiedene rewards zu ähnlichen Policies führen?

JA, Aus beobachtetem Verhalten folgt keine eindeutige Intention:

- **Zwei Agenten könnten identisch handeln, aber aus völlig unterschiedlichen Gründen (z. B. Sicherheit vs. Effizienz).**

Gibt es nicht also ganz verschiedene Interpretationen der Handlungen?

Ja, Lösungen von IRL sind <u>äquivalenzklassen-artig</u>

💡 Daher IRL "Problem" ;)

Inverse Reinforcement Learning (IRL)

Konsequenz: Ill-posedness (nicht wohldefiniert)

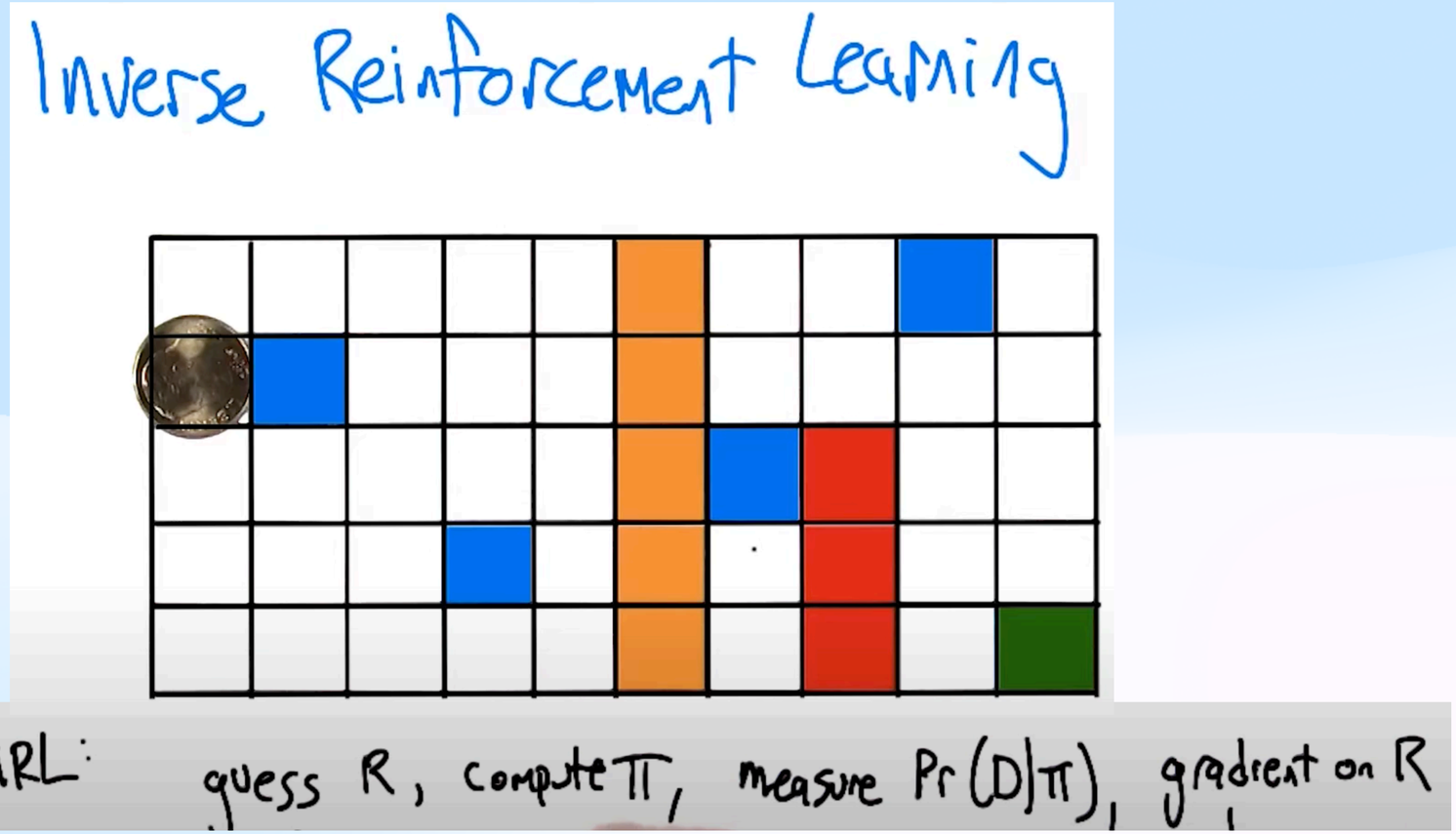
Das IRL-Problem ist **fundamental unbestimmt** ohne (optionale) Zusatzannahmen:

- ***Feature*-Selektion** reward beeinflusst Interpretation
- **Regularisierung**, z. B. Entropiemaximierung (Ziebart), bevorzugt “uninformative” Rewards
- **Priors** in Bayesian IRL steuern Interpretation
- **Policies** handeln *wahrscheinlich* **optimal**

Inverse Reinforcement Learning (IRL)

https://www.youtube.com/watch?v=h7uGyBcIeII

Maximum Likelihood IRL



https://github.com/Silviatulli/contrastive_examples_MLIRL TODO: Simplify Example!

Inverse Reinforcement Learning (IRL)

Ein Algorithmus zum Lösen des Problems:

Maximum Likelihood IRL (*ML*IRL)

bootstrapping R = 0 => Q = 0

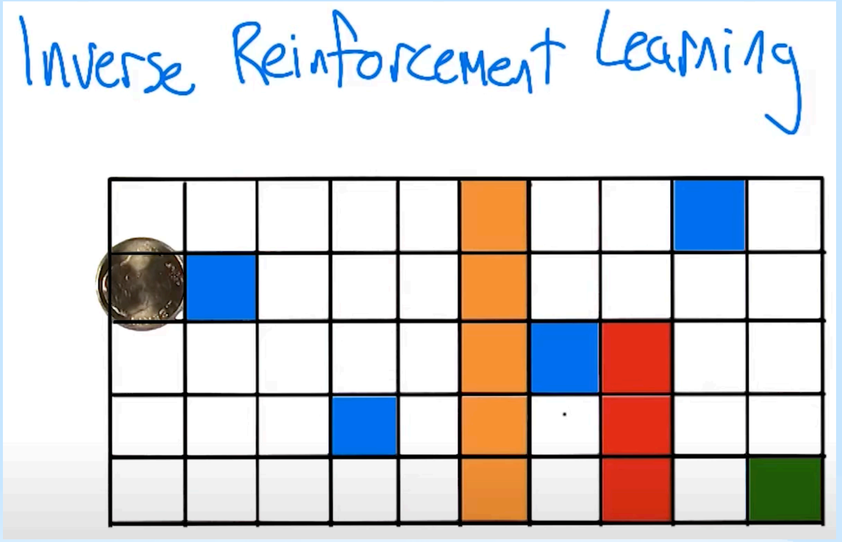
while True:

1. **Q = r + γ max Q(a I s) ≈ E[G]**
2. **Berechne *gierige* Policy: nimm da wo beste Rewards π = argmax(a, Q(a))**
3. **Berechne Wahrscheinlichkeit für die beobachteten Trajektorien (D)**
4. **Update Reward Schätzung per Gradient über Differenz zur Vorhersage**

Reward modelliert als Linear Layer:

$$R(s) = w^\top \phi(s)$$

$$\hat{\theta} = \operatorname*{argmax}_\theta P(D|M, f, \theta)$$



3. **Likelihood der beobachteten Trajektorien:**

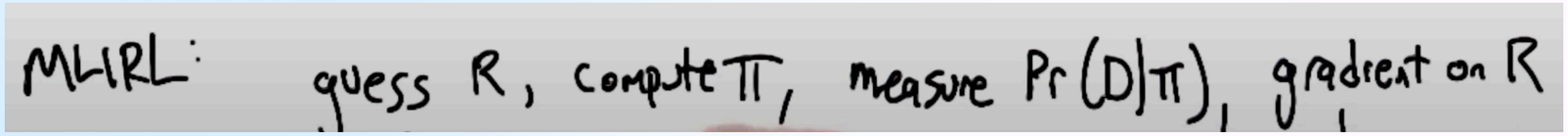
$$\mathcal{L}(\theta) = \prod_{i=1}^{N} \prod_{(s,a)\in\tau_i} \pi_\theta(a|s)$$

$$\log \mathcal{L}(\theta) = \sum_{i=1}^{N} \sum_{(s,a)\in\tau_i} \log \pi_\theta(a|s)$$

4. **Gradientaufstieg im Rewardparameterraum:**

$$\theta \leftarrow \theta + \alpha \cdot \nabla_\theta \log \mathcal{L}(\theta)$$

Der Gradient durchdringt $R_\theta(s) \to Q(s,a) \to \pi(a|s) \to \log \mathcal{L}$



https://github.com/Silviatulli/contrastive_examples_MLIRL **TODO: Simplify Example!**

https://cs.brown.edu/people/mlittman/theses/babes.pdf Michael Littman

Inverse Reinforcement Learning (IRL)

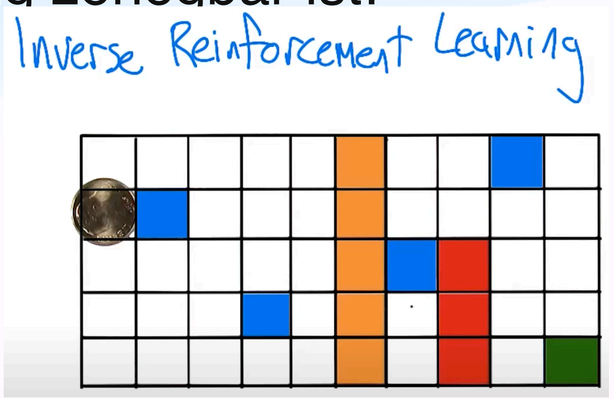
Frage:

Kann man aus dem Verhalten Rückschlüsse auf intrinsische / extrinsische rewards ziehen?

D.h. Belohnung von der Umgebung / Belohnung von uns selber ("custom")

Im Realen Leben: Ja schon irgendwie?

Formell? Nur mit Zusatzannahmen, da **R = R_in + R_ex** beliebig zerlegbar ist!



$$\mathbb{P}[a_t \mid s_t] = \frac{\exp\{\eta Q_w^*(s,a)\}}{\sum_{a'} \exp\{\eta_Q Q_w^*(s,a')\}}$$

*Boltzmann (**softmax**) policy from Q-function*

value iteration

Inverse Reinforcement Learning (IRL)

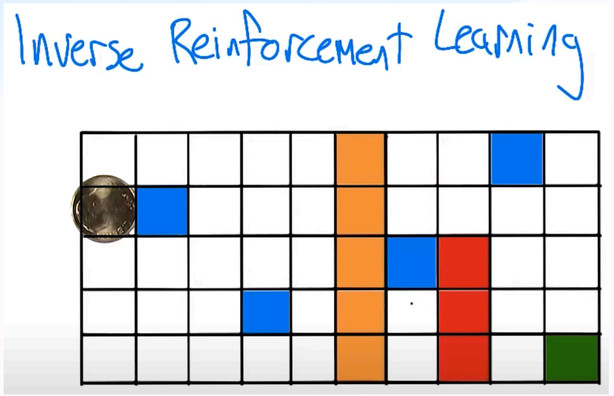
Feature

Reward Funktion Vereinfachung:

Statt beliebige Reward Funktion R(s,a) zu schätzen,

machen wir die Annahme, dass der Reward die einfache *'linear parametrisierte'* Form hat:

R(s) = w * ϕ(s)



ϕ Feature z.B. grün gibt Belohnung

Diese werden automatisch gelernt (irgendwelche Muster)

=> Reward als lineares Netzwerk!

Inverse Reinforcement Learning (IRL)

1. Feature Matching / Max-Margin IRL (Ng & Russell, 2000)

- assume **linear** reward model: R(s) = **theta^T phi(s) = w^T·phi(s)** // Linear NN
- Goal: Match feature expectations between **expert** and **learner**.
- Optimization:

$$\max_{\theta} \min_{\pi \in \Pi} \theta^\top (\mu_E - \mu_\pi)$$

where $\mu_\pi = \mathbb{E}_\pi \left[\sum_t \gamma^t \phi(s_t) \right]$

$$R(s) = w^\top \phi(s)$$

μ is the **expected feature count vector** (or **feature expectation**) under policy π

$\phi(s_t) \in \mathbb{R}^d$: feature vector for state s_t $\phi : \mathcal{S} \to \mathbb{R}^d$ (state features)

$\gamma \in [0, 1)$: discount factor or $\phi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^d$ (state-action features)

💡 **features** sind wie **latent embeddings**

Inverse Reinforcement Learning (IRL)

Maximum Entropy Inverse Reinforcement Learning

https://nbviewer.org/github/qzed/irl-maxent/blob/master/notebooks/maxent.ipynb

Maximizing the log-likelihood is equivalent to minimizing the Kullback-Leibler divergence $D_{KL}(p^E(\tau) \,\|\, p(\tau \mid \omega))$

$$
\begin{aligned}
\arg\max_p \quad & H(p) \\
\text{subject to} \quad & \mathbb{E}_{\pi^L}[\phi(\tau)] = \mathbb{E}_{\pi^E}[\phi(\tau)], && \text{(feature-expectation matching)} \\
& \sum_\tau p(\tau) = 1, \quad \forall \tau : p(\tau) > 0 && \text{(probability constraints)}
\end{aligned}
$$

The Principle of Maximum Entropy

The general idea of Maximum Entropy Inverse Reinforcement Learning is based on **feature-expectation** matching [Abbel & Ng 2004]: The expected *visitation-frequency* of features in the *demonstrations* provided by the *expert* should equal the expected visitation-frequency of features visited by the agent following the recovered reward function. In other words: The features describing the states of our world should be, in expectance, visited equally often by the agent following our recovered reward function and by the expert demonstrating (nearly) optimal behavior. In equations, we can express this as the expectation of features ϕ over trajectories derived from the reward R we want to recover.

$$\mathbb{E}_{\pi^L}[\phi(\tau)] = \sum_{\tau} p_{\pi^L}(\tau)\phi(\tau)$$

"unsere Policy π^L soll sich so verhalten wie beobachtete 'Experten' Policy π" :

$$\begin{aligned} \arg\max_p \quad & H(p) \\ \text{subject to} \quad & \mathbb{E}_{\pi^L}[\phi(\tau)] = \mathbb{E}_{\pi^E}[\phi(\tau)], && \text{(feature-expectation matching)} \\ & \sum_{\tau} p(\tau) = 1, \quad \forall \tau : p(\tau) > 0 && \text{(probability constraints)} \end{aligned}$$

The Principle of Maximum Entropy

But why should we maximize the entropy?

We now know the parameterized form of the distribution, so how can we optimize the parameters? The answer is fairly straightforward: We can maximize the log-likelihood over all demonstrated trajectories $\mathcal{D} = \{\tau_i\}_{i=1}^{N}$, i.e.

$$\omega^* = \arg\max_{\omega} \mathcal{L}(\omega) = \arg\max_{\omega} \sum_{\tau \in \mathcal{D}} \log p(\tau \mid \omega),$$

using gradient ascent (or any other gradient-based optimization technique) as this function is convex [Ziebart et al. 2008; Osa et al. 2018]. A small side-note: Maximizing the log-likelihood is equivalent to minimizing the Kullback-Leibler divergence $D_{KL}(p^E(\tau) \,\|\, p(\tau \mid \omega))$ [Bishop 2006],

Inverse Reinforcement Learning (IRL)

```
def compute_feature_expectations(demos):
    features = torch.zeros(dim_features)
    for traj in demos:
        for t, (s, a) in enumerate(traj):
            features += (gamma ^ t) * phi(s)
    return features / len(demos)
```

Aufgabe

Bei einem unserer Beispiele:

Agenten **normal trainieren** und speichern

Danach in neuer Datei Agenten laden und

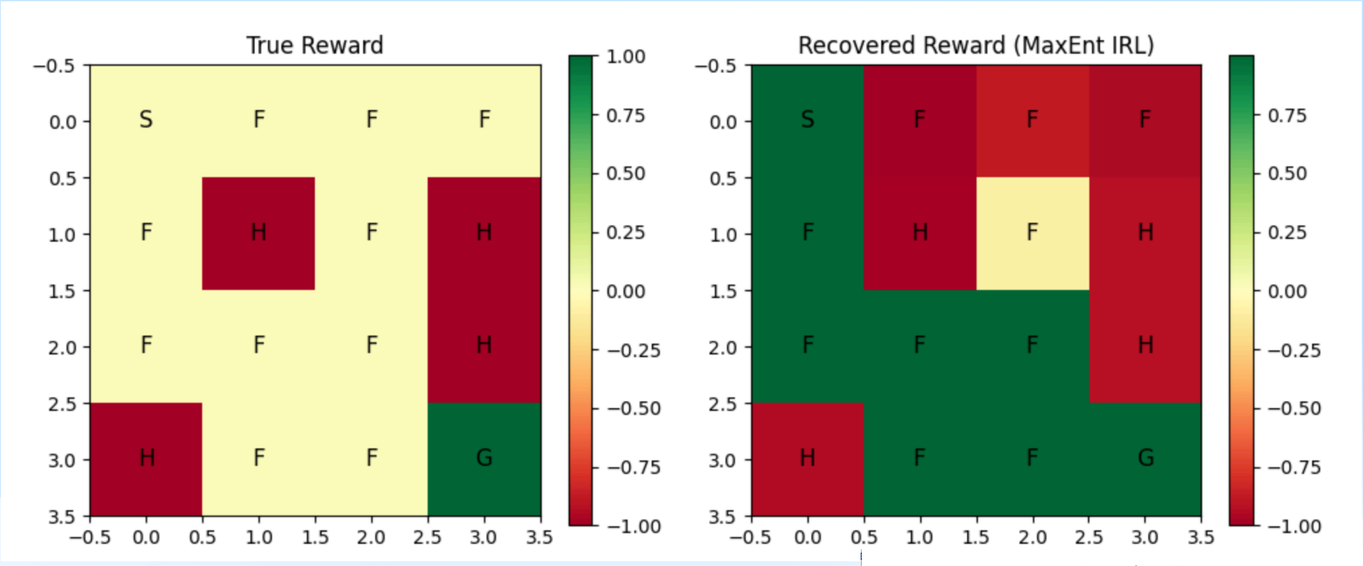
Ein paar Spiele spielen = **Daten** Trajektorien **erzeugen** lassen D=[T...]

Ziel: diese Daten in ein IRL Algorithmen füttern.

Am Ende die geschätzten Rewards visualisieren (möglichst hoch beim Geschenk!).

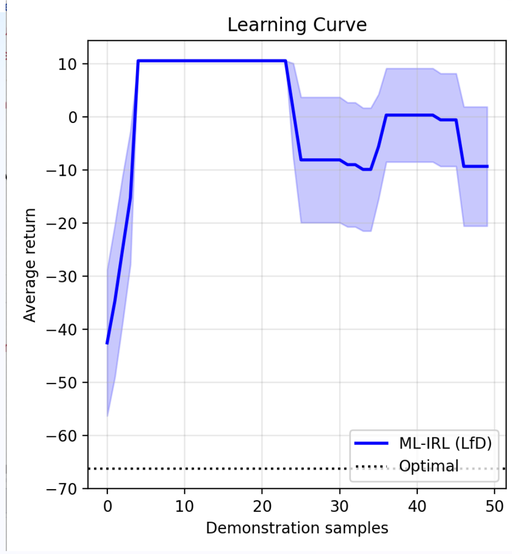
Aufgabe

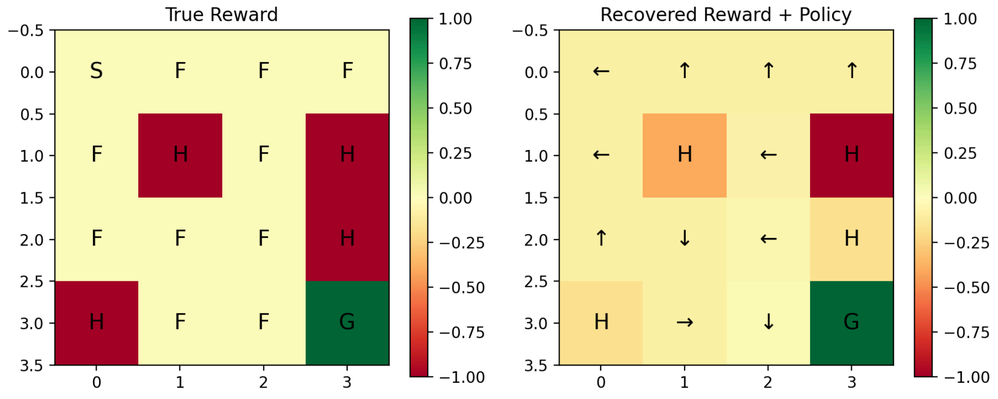
frozen-maxent-irl.py ("der Weg ist das Ziel")



vs frozen-maxlike-mlirl.py

```
Maximum Likelihood IRL
Wie im Video:
```





Inverse Reinforcement Learning (IRL)

```
# Estimate feature expectations for given policy via rollouts
def sample_policy_feature_expectations(policy_fn, n_episodes=20):
    features = torch.zeros(dim_features)
    for _ in range(n_episodes):
        s, _ = env.reset()
        done, t = False, 0
        while not done:
            a = policy_fn(s)
            s, _, terminated, truncated, _ = env.step(a)
            done = terminated or truncated
            features += (gamma ** t) * phi(s)
            t += 1
    return features / n_episodes
```

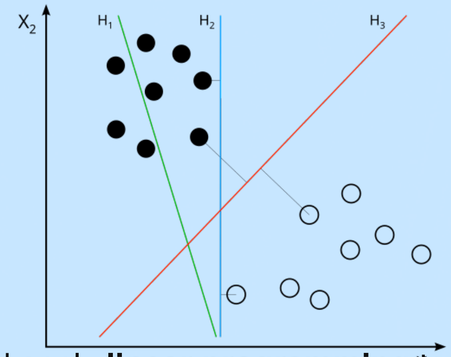
Inverse Reinforcement Learning (IRL)

```
# Simple policy: softmax over reward
def make_policy(theta):
    def policy_fn(s):
        state_feat = phi(s)
        scores = torch.stack([torch.dot(theta, phi(s)) for a in range(num_actions)])
        return torch.distributions.Categorical(logits=scores).sample().item()
    return policy_fn

# Main training loop
expert_demos = [...]  # list of trajectories: [(s0,a0), (s1,a1), ...]
mu_E = compute_feature_expectations(expert_demos)

for iteration in range(100):
    policy_fn = make_policy(theta)
    mu_pi = sample_policy_feature_expectations(policy_fn)
    loss = -torch.dot(theta, mu_E - mu_pi)  # Max-Margin IRL objective
```

Inverse Reinforcement Learning (IRL)



Key Algorithms and Techniques Used in Inverse Reinforcement Learning

1. **Max Margin** Methods

Cornerstone in inverse reinforcement learning. They utilize **Markov Decision Processes MDP** to model environments and apply **linear programming** to differentiate between optimal and less optimal actions. Consequently, they ensure AI systems make decisions closely aligned with expert behavior.

    1. **SVM** Support Vector Machine / Lagrange multipliers

2. **Bayesian** Methods 2007

Bayesian methods bring **prior knowledge** into the inverse reinforcement learning equation. They adapt well to complex or uncertain environments. Additionally, these techniques refine the reward structure, enhancing learning outcomes. Consequently, Bayesian methods address the challenges of dynamic environments.

$$P(R \mid \mathcal{D}_E) \propto P(\mathcal{D}_E \mid R) P(R)$$

3. **Maximum Entropy** Methods 2008

Maximum entropy methods focus on diversifying AI behaviors. They are perfect for environments with **continuous or variable spaces**. These methods use entropy to **encourage exploration,** helping AI navigate uncertainties more effectively. Therefore, maximum entropy methods are essential in developing robust machine learning algorithms. => ME Classifier / Shannon **E(p)=∑Log(p)*p**

$$P(\tau \mid R) \propto \exp(R(\tau)) = \exp\left(\sum_t R(s_t, a_t)\right)$$

4. **Apprenticeship** Learning 2004

Apprenticeship (Praktikanten) learning is a **specialized** branch of IRL. It aims to **replicate** expert behavior within AI systems and utilizes insights from inverse reinforcement learning to surpass or match expert performance. Consequently, apprenticeship learning enhances AI's ability to learn from human actions, which significantly contributes to artificial intelligence applications. Reward vereinfacht modelliert über lineare '**Feature**'

$$R(s) = w^\top \phi(s)$$

Inverse Reinforcement Learning (IRL)

Aspects	Inverse Reinforcement Learning	Traditional Reinforcement Learning
Goal	Infers the reward function from observed behavior	Maximizes a known reward function
Approach	Learns from expert demonstrations	Learns through trial and error
Output	Reward function	Optimal policy
Application	Complex environments where the reward is not clear	Environments with a well-defined reward
Data Utilization	Utilizes observed behavior to infer rewards	Utilizes rewards directly for learning
Dependency	Depends on the quality of expert demonstrations	Depends on the quality and clarity of the reward function
Adaptability	Can adapt to changes in the environment by reinterpreting behaviors	Requires redefinition of reward functions for new environments

Pause

Beispiel: Ein autonomes Fahrzeug (Rolle B in Abb. 1) muss sicher in einen überfüllten Autobahnzubringer einfahren und modelliert dazu das Verhalten eines vorausfahrenden Fahrzeugs (Rolle A). Anhand historischer Trajektorien (z. B. aus dem NGSIM-Datensatz [3]) lassen sich typische Präferenzen bezüglich Sicherheit und Geschwindigkeit eines Fahrers in Rolle B rekonstruieren.

NICHT:

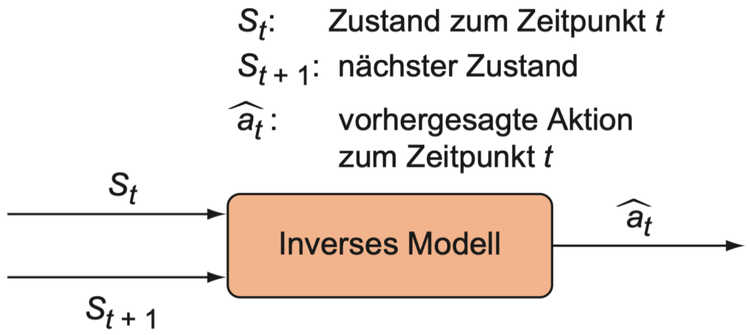


Bild 8.7 Das inverse Modell nimmt zwei aufeinanderfolgende Zustände und versucht vorherzusagen, welche Aktionen ergriffen wurden.

SoTA

Adversarial IRL (GAN-style)

1. GAIL (Ho & Ermon, 2016)

- Idee: **Vermeide Reward** Recovery – lerne direkt Policy via **adversarial imitation**.
- Verlust: $\min_{\pi} \max_{D} \mathbb{E}_{\pi}[\log D(s,a)] + \mathbb{E}_{\pi^E}[\log(1 - D(s,a))]$- Impliziter Reward: R(s,a) = -\log(1 - D(s,a))
- Stärken: stabil, skalierbar
- Schwächen: keine explizite Reward-Funktion

2. AIRL (Fu, Luo & Levine, 2018) — Adversarial IRL with Structured Discriminator

- Ziel: explizite, *disentangled* Reward-Funktion, transferfähig
- Reward-Struktur: $D(s,a,s') = \exp(f_\theta(s,a,s') + \gamma V(s') - V(s))$
- Output: $R(s,a) = f_\theta(s,a)$
- Vorteil gegenüber GAIL: echtes $R(s,a)$ rekonstruierbar

SoTA

3. DIRL (Discriminator-Reward IRL, 2020+)

- Architekturen zur Trennung von reward, policy und environment stochasticity
- Modularisierung von reward and dynamics learning
- Anwendbar auf multimodale Aufgaben und Transfer

II. Score-based / Density Ratio Methods

4. ValueDICE (Kostrikov et al., 2020)

- Formuliert IRL als **convex-concave saddle** point über density ratios
- Vermeidet instabiles adversariales Training
- Führt zu einheitlichem Optimierungsproblem:

\min_f \max_d \; \mathbb{E}{\pi_E}[f(s,a)] - \mathbb{E}{d}[e^{f(s,a)}]

- Effizient, besonders in offline-Settings

SoTA

5. IQ-Learn (Garg et al., NeurIPS 2021)

- Sample-efficient offline IRL
- Maximiere normalized state-action occupancy matching
- Policy iteration mit closed-form Q-function updates

6. *Dreamer* als Grundlage für IRL (Ha & Hafner, 2021+)

- Kombiniert IRL mit World Models (e.g., DreamerV3) **Reward predictor**

IV. Hierarchical / Meta IRL

7. MILO / MetaIRL (2020–2023)

- IRL über mehrere Aufgaben: Meta Inverse Learning of Objectives
- Ziel: Belohnungsfunktion generalisiert über Aufgaben
- Meta-Lernverfahren (z. B. MAML) zur schnellen Anpassung

https://link.springer.com/article/10.1007/s10462-021-10108-x#ref-CR128

Information und Entropie

Shannon-Information := -log(p) self Information (≠ Fischer Information)

p=1 => -log(1) = 0 keine 'Information' wenn ein sicheres Ereignis eintritt

p≈0 => -log(0) ≈ ∞ maximale 'Information' wenn ein seltenes Ereignis doch eintritt

"punktweise Überraschung"

Entropie:

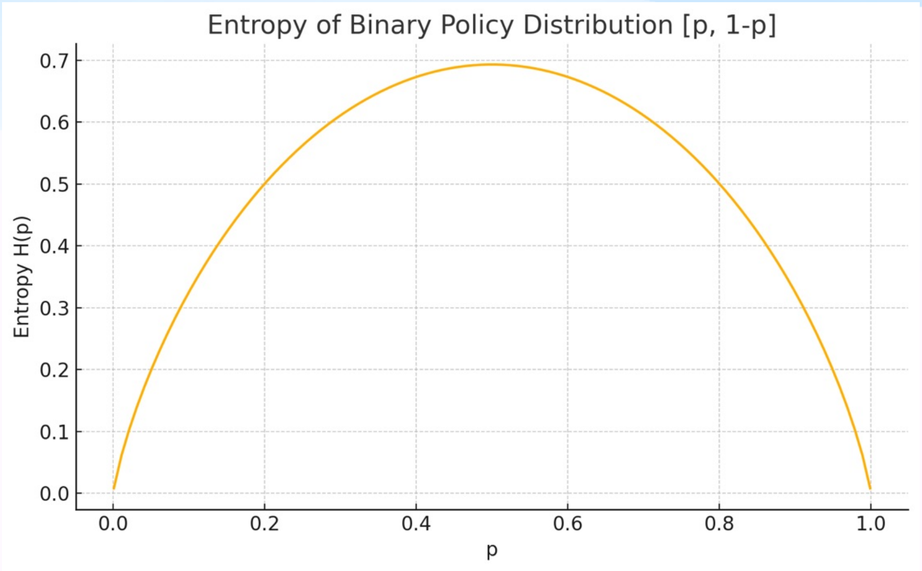
Entropie (mittlere Shannon-Information):

durchschnittliche Überraschung / Unordnung (einer Verteilung)

$H(p) = -\sum p_i * \log(p_i)$ $\sum p_i = 1$ Einheit “bit” oder “nat”

Bei Unfairer Münze die immer Kopf/Zahl ergibt, gibt es keine Überraschungen => H=0

Bei Fairer Münze die mal Kopf/Zahl ergibt, gibt ist Ergebnis nicht vorhersehbar "Überraschungsfaktor H=.7"



Wird verwended z.B. in **SAC, PPO, KL-Divergenz, MaxEntropy IRL**

Fazit

Sehr junges Feld

Sehr viel Mathematik

Sehr viele Ansätze aus anderen Disziplinen

Für uns zu hoch? Aber in Kooperation mit Codex? <<

Agentic Coding:

Aufgabe: IRL Prompt für Codex/Claude Copilot/GPT nicht so gut

NEW SKILL: ***IDEAS and* agency *and speaking***

Aspekte

Max likelihood MLIRL

$$\max_{\theta} \mathcal{L}(\theta) = \prod_{(s,a)\in\mathcal{D}} \pi_\theta(a \mid s)$$

Max *margin* *MMIRL*

$$\max_{R} \gamma \quad \text{s.t.} \quad R^\top \mu_E \geq R^\top \mu_\pi + \gamma, \; \forall \pi$$

Max *entropy* *MEIRL*

Bayesian Features …

$$\omega^* = \arg\max_{\omega} \mathcal{L}(\omega) = \arg\max_{\omega} \sum_{\tau\in\mathcal{D}} \log p(\tau \mid \omega),$$

$$R(\tau) = \omega^\top \phi(\tau)$$

- **MLIRL:** Maximiert *Wahrscheinlichkeit* der Demos.
- **MMIRL:** Maximiert *Abstand* zwischen Experte und Alternativen.
- **MaxEnt IRL:** Maximiert *Entropie* unter den Lösungen, die die Demos erklären. Maximum Causal Entropy IRL?

Mimic the expert

Interact with other systems

Learn about a system

Pause

Further reading:

https://rail.eecs.berkeley.edu/deeprlcourse-fa17/f17docs/lecture_12_irl.pdf
https://ai.stanford.edu/~cbfinn/_files/bootcamp_inverserl.pdf
https://www.lesswrong.com/w/inverse-reinforcement-learning

Ausblick:
IRL über foundation model? LLM als baysscher schätzer? Weltwissen!
Ideal machen wir lieber alles über ein Neuronales Netz statt komischer Mathematik.

Inverse ≈ Imitation learning?

Imitation learning Oberbegriff

Tanz der Bienen (umvölken umvölkeln umwaiseln)

IRL ≈ IL? Nicht so ganz. Bei IL wird **Policy π** wird **direkt** gelernt
IRL **expert** vs IL **trainer / forced teacher** (z.B. mountain_car_trainer_policy.py oder über MSE)

=> IRL als spezielle Technik innerhalb **Imitation Learning!**

IL via **supervised learning:** *behavior cloning*, **off-policy**, DAGGER

IRL and AL are examples of imitation learning [Argall et al., **2009**] learn reward / policy

Behavior cloning (BC):

- Treats imitation as a supervised learning problem: train a mapping π : s -> a from expert demonstrations. (ohne RL Daten Sammlung)
- You only ever see expert state–action pairs (**demonstrations => supervised ML**)
- Simple, fast, data-efficient if demonstrations cover all relevant states.
- Weakness: distribution shift. Small errors in unfamiliar states compound, because the learner never saw corrections there.

DAgger (“Dataset Aggregation”, Ross, Gordon & Bagnell 2011) is a core imitation learning algorithm.
Unlike standard behavior cloning, which simply trains on expert demonstrations once, DAgger actively collects new data to avoid covariate shift.

Inverse ≈ Imitation learning?

IRL als Imitation learning mit *Model* (R)

=> mit Reward Model kann man dann besser **generalisieren**
=> eigene Policy statt nur zu kopieren

- *Imitation reicht* → **Supervised Learning**. Nachäffen
- *Warum wird so gehandelt?* → **IRL** *Verständnis!*

apprenticeship learning (AL)

Pause

Offene spannende Themen:

Novelty Exploration! Curiosity & **Surprise** < Karsten Fr. Ch8_book.ipynb
https://github.com/danijar/dreamerv3 < Karsten

Vertiefung Planning = Model-based
Dyna-Q (Sutton 1990) MuZero Transitionen in latenten Zuständen; plant ohne expliziten Umgebungszustand, MCTS, MPC, Dreamer

RL in **LLM** (RLHF / **reasoning** fine-tuning / Pokémon) : << Torsten
DeepSeek GRPO
AbsoluteZero < Christian
LLM als **hierarchischer** Feudaler "Chef" GPT Latent Schicht mit RL Koppeln. GPT als Datensammler, GPT als RL Trainer?
LLM AlfaTensor **AlfaEvolve** gamification (matrix mul als Spiel) via **ollama?** << Torsten
https://deepmind.google/discover/blog/alphaevolve-a-gemini-powered-coding-agent-for-designing-advanced-algorithms/

einfachster fast **SoTa** ?
Foundation Model RL? Dreamer? Free Downloadable Model Weights. Transfer Learning to other domains.
most **impressive example** a) running locally b) trainable locally c) tuneable locally

Graph Neural Networks
Aufmerksamkeitsmodelle << Tom Graph / Relational Attention key value query
priorisierten Wiederholung vs Novelty?
Shooting Algorithms model predictive control (MPC)

Beispiele Teil 3

Anwendungen von Inverse Reinforcement Learning (IRL):

1. Robotik

Ziel: Lernen von komplexem Verhalten durch Demonstration.

Beispiel: • **Ein Mensch führt einen Greifvorgang aus**

→ Roboter lernt den zugrunde liegenden Reward (z.B. Sanftheit, Effizienz) statt nur die Bewegungen.

2. Autonomes Fahren

Ziel: Rekonstruktion menschlicher Fahrintentionen.

Beispiel: **Warum weicht ein Fahrer *jetzt* leicht zur Seite aus?**

3. Verhaltensanalyse

Ziel: Interpretieren oder bewerten von Handlungen in sozialen, ökonomischen oder biologischen Kontexten.

Beispiel: • **Tierverhalten: Welche Ziele verfolgt ein Tier (Futter, Deckung, Energieeffizienz)?**

4. Medizin / Rehabilitation

Ziel: Verstehen von Bewegungsmustern oder Therapieentscheidungen.

Beispiel:• **Aus Bewegungen rekonstruieren, welche Bewegungsziele Patienten verfolgen → adaptive Robotik für Reha.**

Vokabular

IRL inverse reinforcement learning
learning from demonstrations ≈ IRL
demonstrator / expert der Agent den wir beobachten
D "**D**emonstrationen" also eine Menge von **D**ata:
Trajektorien $\mathbf{D}^{\intercal}=(\tau_0, \ldots, \tau_n)$ $\tau=(\mathbf{s_0},\mathbf{a_0}, s_1, a_1, \ldots s_t)$ "Experten"
expert demonstrations (s.o.)
apprentice (us;) "Lehrling"
R(τ)=w*ϕ(τ)=∑w*ϕ(s) Return of Trajectory as sum over states in that trajectory
Entropy $H(p)=-\sum p_i * \log p_i$ misst ”Überraschung“ oder „Unordnung” einer Verteilung $\sum p_i=1$

offline IRL [12, 13, 14, 9, 15]. Different from standard IRL [16, 17, 18, 19, 20, 21]

distribution shift

Latent Space

In tiefen Neuronalen Netzen gibt es in den mittleren Schichten oft einen '**Bottleneck**' in dem die komplizierten Eingangsdaten komprimiert als ein Vektor abgebildet.

Autoencoder!

Latent Space = mittlere Schicht Vektor(raum)

Feature Vector mit Aspekten

Entscheidungen finden nicht mehr auf Pixelbasis sondern auf "abstrakter Situation"
z.B. Lunar Lander per Bilderkennung => Latent Space = (Position, Winkel, Geschwindigkeit)
400*300 pixel => Vektor mit 6 Einträgen extreme Komprimierung!

Abstrakte Entscheidungen: "flieg nach oben rechts" => Aktion: gib gas 30% Situationsabhängig
=> Skills

ChatGPT = 4096 "Großmutterneuron" => abstrakt verteilt

20 Questions 2^**20**
lebend? ja
pflanze? ja
europa? ja
wohnzimmer? ja
tannenbaum ja
akinator

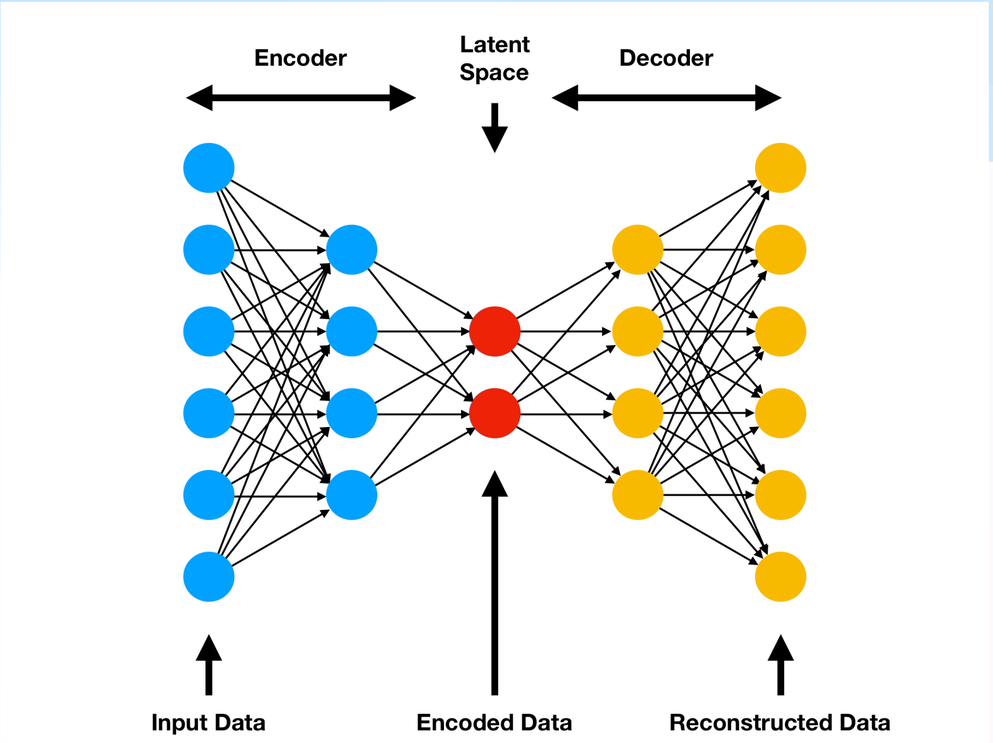
Latent Space

In tiefen Neuronalen Netzen gibt es in den mittleren Schichten oft einen '**Bottleneck**' in dem die komplizierten Eingangsdaten komprimiert als ein Vektor abgebildet.

Autoencoder!
Bottleneck Mittelschicht immer (fully connected) Perceptron

Latent Space = mittlere Schicht Vektor(raum)
≈
Feature Vector mit Aspekten
Feature Activation
• komprimierte abstrakte Information
als universelle "Währung"
zwischen Bildern, Text, ...

Latent heißt 'hidden' meint aber eine spezielle hidden Schicht.



Latent Space

Anwendungen:

...

Suche

Erzeugung aus Text

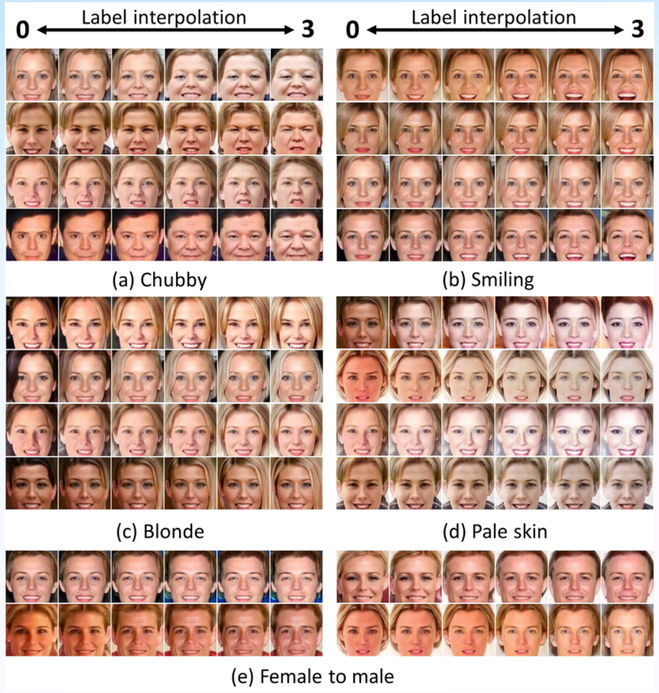
Style Transfer

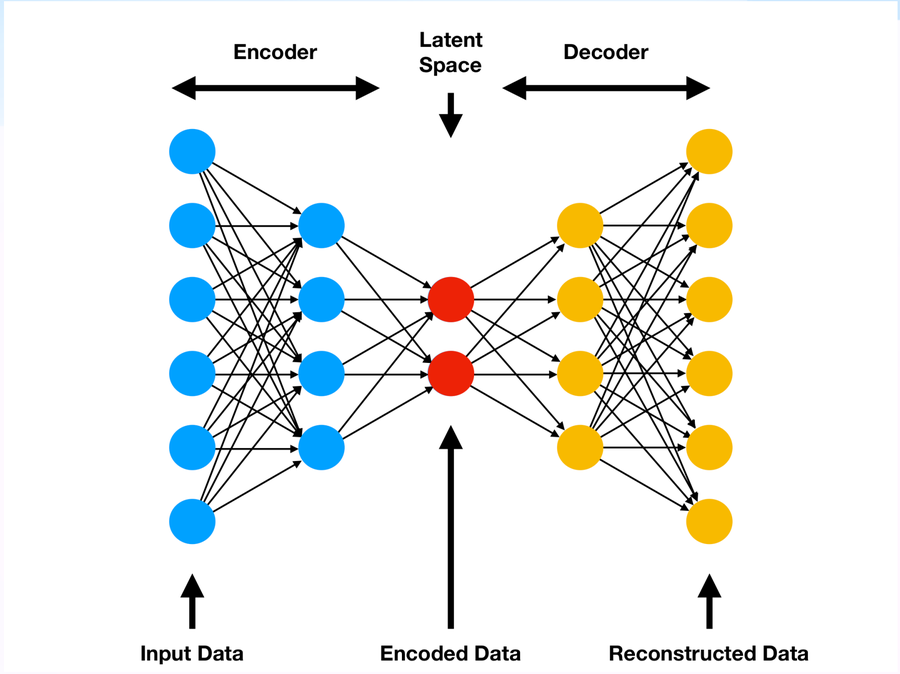
Feature Interpolation (lachend ja/nein)

RL: Statt policy auf pixeln zu suchen, suchen wir policy auf "Spielsituation"

state s => z komprimierter state Φ(s) phi für 'feature' 'phiture;)'

Modell	Parameter	d_{model} / hidden size
GPT-2 small	124M	768
GPT-2 medium	355M	1024
GPT-2 large	774M	1280
GPT-2 XL	1.5B	1600
GPT-3 small	125M	768
GPT-3 medium	350M	1024
GPT-3 large	760M	1536
GPT-3 XL	1.3B	2048
GPT-3 2.7B	2.7B	2560
GPT-3 6.7B	6.7B	4096
GPT-3 13B	13B	5140
GPT-3 175B	175B	12288
GPT-3.5	unknown	not public





Referate : Extra Wunsch Themen

Wiederholung

Algorithmen: **SAC, TD3, TD(λ),** S**H**ARSA (**h**igh level **a**ction) ≈ hierarchische actions

RL mit Sprachmodellen **LLM** (RLHF / **reasoning** DeepSeek / pokemon) **<<**
Aufmerksamkeitsmodelle
priorisierte Wiederholung
RL in Spielen?

- **virtual twin / factory.io / Kuka?** Enes
- Sparmodell? **BudgetEnv**? Custom Environment.
  mit baseline evaluieren?

- **reward shaping?**

https://gibberblot.github.io/rl-notes/single-agent/reward-shaping.html
https://arxiv.org/abs/2011.02669

einfachster State of the Art Algorithmus **SoTa ±** ?
most **impressive example** a) running locally b) trainable locally c) tuneable locally
RL in Spielen welche Spiele wurden (noch nicht) gelöst.