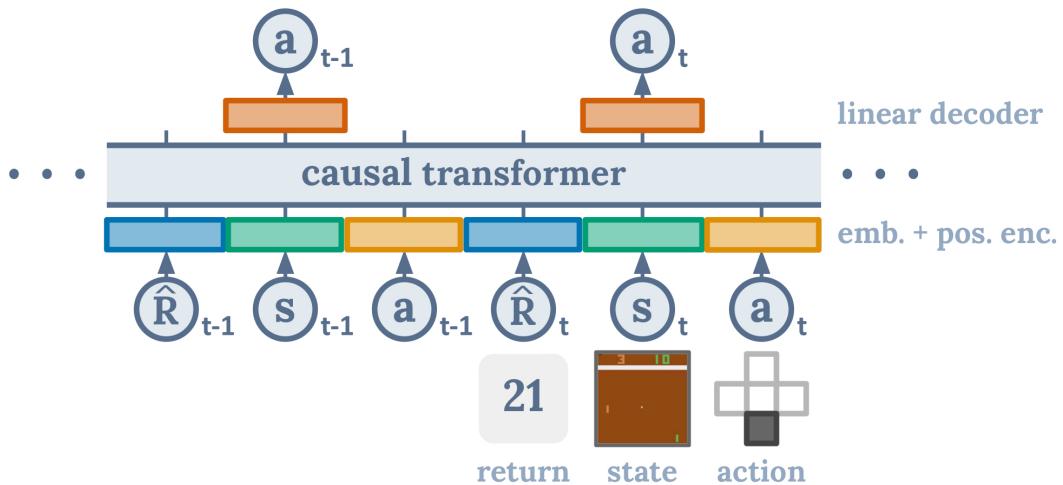




Decision Transformer: Reinforcement Learning via Sequence Modeling-hoho



论文试图解决什么问题？

将transformer用于强化学习的决策上。因为transformer可以带来如下优点：

1. 解决credit-assignment，通过self-attention
2. 避免未来折扣奖励（因为传统的RL使用折扣奖励的方法会引起undesirable short-sighted behaviors）
3. 使得训练更加稳定。

这是否是一个新的问题？

是一个新问题。之前将transformer引入到RL，依然使用的是actor-critic的方法去优化模型。而本文直接将actor-critic这种架构替换为transformer，即将RL的马尔科夫决策过程用transformer建模。

这篇文章要验证一个什么科学假设？

提出Decision Transformer，验证其是否可以在offline RL方法上进行策略优化。

有哪些相关研究？如何归类？谁是这一课题在领域内值得关注的研究员？

相关研究如下：

1. offline RL，本文将DT用于offline RL的研究中
2. RL中的监督学习
3. credit assignment研究，已经有很多相关研究通过状态分配（state-association）来解决credit-assignment，即学习一种可以使reward function分解到特定的状态中
4. 有条件语言生成（Conditional language generation）
5. 注意力与transformer。

论文中提到的解决方案之关键是什么？

本文希望预测的动作是基于未来的回报，所以每个时间步上的reward定义为return-to-go的累积回报： $R_t = \sum_{t'=t}^T r_{t'}'$ (hoho：这里不用加上个折扣因子么？)

最终数据形式为： $\tau = (R_1, s_1, a_1, R_2, s_2, a_2, \dots, R_T, s_T, a_T)$

每个时间步要输入R,s,a这三个token，通过线性映射转为embedding（如果s为图像，则通过CNN去转）

使用GPT模型

训练：每次采集一批长度为 K 的序列，输入为下图 st 及以前长度为 K 的这个序列，输出为一个动作，希望这个动作和训练集中的动作相近，以有监督学习模式来训练（自回归）。

测试：测试的时候，需要在最开始额外估计一个策略能达到的最大 return，作为第一步的 reward-to-go，然后依次使用训练好的 Transformer 去生成相应的动作。每次得到新的奖励之后，就从前一步的 reward-to-go 中减去，从而得到下一步需要输入的 reward-to-go。

流程看代码：

(1) (2) + 1/4

Algorithm 1 Decision Transformer Pseudocode (for continuous actions)

```
# R, s, a, t: returns-to-go, states, actions, or timesteps
# transformer: transformer with causal masking (GPT)
# embed_s, embed_a, embed_R: linear embedding layers
# embed_t: learned episode positional embedding
# pred_a: linear action prediction layer ↗

# main model
def DecisionTransformer(R, s, a, t):
    # compute embeddings for tokens
    pos_embedding = embed_t(t) # per-timestep (note: not per-token)
    s_embedding = embed_s(s) + pos_embedding
    a_embedding = embed_a(a) + pos_embedding
    R_embedding = embed_R(R) + pos_embedding

    # interleave tokens as (R_1, s_1, a_1, ..., R_K, s_K)
    input_embeds = stack(R_embedding, s_embedding, a_embedding)

    # use transformer to get hidden states
    hidden_states = transformer(input_embeds=input_embeds)

    # select hidden states for action prediction tokens
    a_hidden = unstack(hidden_states).actions

    # predict action
    return pred_a(a_hidden)

# training loop
for (R, s, a, t) in dataloader: # dims: (batch_size, K, dim)
    a_preds = DecisionTransformer(R, s, a, t)
    loss = mean((a_preds - a)**2) # L2 loss for continuous actions
    optimizer.zero_grad(); loss.backward(); optimizer.step()

# evaluation loop
target_return = 1 # for instance, expert-level return
R, s, a, t, done = [target_return], [env.reset()], [], [1], False
while not done: # autoregressive generation/sampling
    # sample next action
    action = DecisionTransformer(R, s, a, t)[-1] # for cts actions
    new_s, r, done, _ = env.step(action)

    # append new tokens to sequence
    R = R + [R[-1] - r] # decrement returns-to-go with reward
    s, a, t = s + [new_s], a + [action], t + [len(R)]
    R, s, a, t = R[-K:], ... # only keep context length of K
```

论文中的实验是如何设计的？

使用Atari评估离散环境空间，使用OpenAI Gym评估连续环境空间。

在两种强化学习方式上进行对比：

- TD-learning：对比当前的SOTA：CQL)
- Imitation learning：使用Behavior Cloning方式对比)

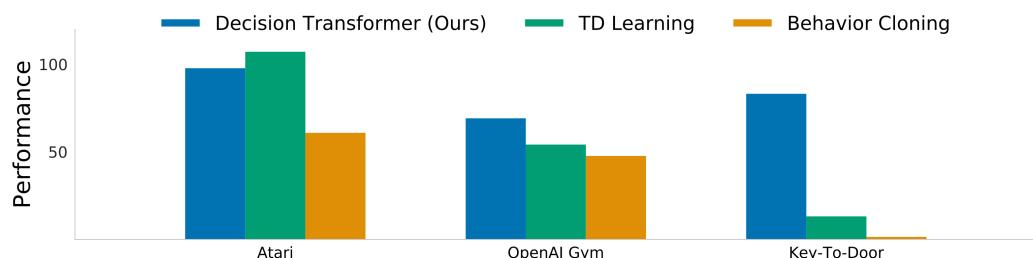
用于定量评估的数据集是什么？代码有没有开源？

强化学习不用数据集，直接在环境中评估。

代码 可参考 <https://github.com/kzl/decision-transformer>

论文中的实验及结果有没有很好地支持需要验证的科学假设？

- 总体实验结果：



在Atari上的对比数据：

Game	DT (Ours)	CQL	QR-DQN	REM	BC
Breakout	267.5 ± 97.5	211.1	17.1	8.9	138.9 ± 61.7
Qbert	15.4 ± 11.4	104.2	0.0	0.0	17.3 ± 14.7
Pong	106.1 ± 8.1	111.9	18.0	0.5	85.2 ± 20.0
Seaquest	2.5 ± 0.4	1.7	0.4	0.7	2.1 ± 0.3

Table 1: Gamer-normalized scores for the 1% DQN-replay Atari dataset. We report the mean and variance across 3 seeds. Best mean scores are highlighted in bold. Decision Transformer (DT) performs comparably to CQL on 3 out of 4 games, and outperforms other baselines in most games.

在Gym上的对比数据：

Dataset	Environment	DT (Ours)	CQL	BEAR	BRAC-v	AWR	BC
Medium-Expert	HalfCheetah	86.8 ± 1.3	62.4	53.4	41.9	52.7	59.9
Medium-Expert	Hopper	107.6 ± 1.8	111.0	96.3	0.8	27.1	79.6
Medium-Expert	Walker	108.1 ± 0.2	98.7	40.1	81.6	53.8	36.6
Medium-Expert	Reacher	89.1 ± 1.3	30.6	-	-	-	73.3
Medium	HalfCheetah	42.6 ± 0.1	44.4	41.7	46.3	37.4	43.1
Medium	Hopper	67.6 ± 1.0	58.0	52.1	31.1	35.9	63.9
Medium	Walker	74.0 ± 1.4	79.2	59.1	81.1	17.4	77.3
Medium	Reacher	51.2 ± 3.4	26.0	-	-	-	48.9
Medium-Replay	HalfCheetah	36.6 ± 0.8	46.2	38.6	47.7	40.3	4.3
Medium-Replay	Hopper	82.7 ± 7.0	48.6	33.7	0.6	28.4	27.6
Medium-Replay	Walker	66.6 ± 3.0	26.7	19.2	0.9	15.5	36.9
Medium-Replay	Reacher	18.0 ± 2.4	19.0	-	-	-	5.4
Average (Without Reacher)		74.7	63.9	48.2	36.9	34.3	46.4
Average (All Settings)		69.2	54.2	-	-	-	47.7

Table 2: Results for D4RL datasets^[3]. We report the mean and variance for three seeds. Decision Transformer (DT) outperforms conventional RL algorithms on almost all tasks.

说明：

1. Medium：使用接近专家策略三分之一分数的策略（所谓的medium policy）采样的数据；
2. Medium-Replay：也是使用medium policy，但是这里使用了reply buffer方法
3. Medium-Expert：使用medium policy和expert policy联合采样的数据

大部分的实验结果表明，Decision Transformer比当前最优的模型要好。

- 本文还证明了超参K（输入的时间步大小）的重要性：

Game	DT (Ours)	DT with no context ($K = 1$)
Breakout	267.5 ± 97.5	73.9 ± 10
Qbert	15.1 ± 11.4	13.6 ± 11.3
Pong	106.1 ± 8.1	2.5 ± 0.2
Seaquest	2.5 ± 0.4	0.6 ± 0.1

Table 5: Ablation on context length. Decision Transformer (DT) performs better when using a longer context length ($K = 50$ for Pong, $K = 30$ for others).

- 对于验证是否很好解决credit-assignment，使用Key-to-Door环境，结果如下：

Dataset	DT (Ours)	CQL	BC	%BC	Random
1K Random Trajectories	71.8%	13.1%	1.4%	69.9%	3.1%
10K Random Trajectories	94.6%	13.3%	1.6%	95.1%	3.1%

Table 6: Success rate for Key-to-Door environment. Methods using hindsight (Decision Transformer, %BC) can learn successful policies, while TD learning struggles to perform credit assignment.

这篇论文到底有什么贡献？

提出了新的进行offline RL学习的架构Decision Transformer，完全用transformer的架构进行强化学习的策略提升

下一步呢？有什么工作可以继续深入？

- 在online RL中transformer架构的试验
- 试试用来预测状态和未来回报
- 对于embedding状态，动作，回报的研究，譬如返回回报的概率分布而不是确定性的回报
- 在model-base RL上的应用
- 对于现实应用，需要研究transformer在MDP上引发的错误和负面结果。