# Create batch dataset

The static dataset should be suboptimal, corrupted with exploratory variability (t cover the space to explore, avoid OOD inputs).

In current example projects, the static dataset is collected from one policy of a suboptimal agent: <https://de.mathworks.com/help/reinforcement-learning/ug/train-agent-offline-to-control-furuta-pendulum.html> .

In the case of this thesis, the static dataset derives from different suboptimal explorative runs, randomly chosen.

# Annotate batch dataset with reward

“However, there are some obstacles in applying offline RL to the real world: most algorithms require a reward signal which is often unavailable. To address this issue, we draw inspiration from the literature in inverse RL and reward modelling to learn a reward function.

When the reward signal is not readily available in the environment, it can be learnt. If demonstrations are given, the reward can be learnt either directly with inverse RL [21, 1], or indirectly with generative adversarial imitation learning (GAIL) [13]. If the end goal [8, 32] or reward values [5] are known for a subset of episodes, reward functions could be learnt with supervised learning.

When demonstrations, but no reward signal is available, policies could be learnt with imitation learning [22], and in particular, behavioral cloning (BC) [25, 26]. BC is a supervised learning technique that aims to imitate the exact actions observed in demonstrations.” *2020, Semi-supervised reward learning for offline reinforcement learning, Ksenia Konyushkova*

# Constrained input for control

“For multiple discrete actions, you need to calculate all possible combinations of discrete actions, and use these with rlFiniteSetSpec” <https://it.mathworks.com/matlabcentral/answers/482331-reinforcement-learning-toolbox-multiple-discrete-actions-for-actor-critic-agent-imageinputlayer-is>

“Moreover, this task becomes more demanding, when systems have bounds on the control input, which is referred to as actuator saturation. In the presence of actuator saturation, a given nominal controller may lose its performance or even make the system unstable unless the constraints are considered a priori in the design process.” 2022, *Optimal control of nonlinear systems with inputconstraints using linear time varying approximations Mehmet Itik*

“This example shows how to train a reinforcement learning (RL) agent with actions constrained using the Constraint Enforcement block. This block computes modified control actions that are closest to the actions output by the agent subject to constraints and action bounds.” <https://de.mathworks.com/help/reinforcement-learning/ug/train-reinforcement-learning-agent-with-constraint-enforcement.html>

“Some control applications require the controller to select control actions such that the plant states do not violate certain critical constraints. In many cases, the constraints are on plant states that the controller does not control directly. Instead, you define a constraint function that defines the constraint in terms of the control action signal.”

<https://de.mathworks.com/help/slcontrol/constraint-enforcement.html>

“Therefore, for the applicability of these algorithms, it is important that they execute policies that are guaranteed to perform at least as well as an existing baseline. We can think of the baseline either as a baseline value or the performance of a baseline policy. It is important to note that since the learning algorithms generate these polices from data, they are random variables, and thus, all the guarantees on their performance should be in high probability. This problem has been recently studied under the general title of safety w.r.t. a baseline in bandits and reinforcement learning (RL), in both offline Bottou et al. (2013); Thomas et al. (2015a,b); Swaminathan and Joachims (2015); Petrik et al. (2016) and online Mansour et al. (2015); Wu et al. (2016); Kazerouni et al. (2017); Katariya et al. (2019) settings.

…

Although we are confident that our algorithm will eventually learn a strategy that performs as well as the baseline, and possibly even better, we should control its exploration not to lose too many customers, as a result of providing them with unsatisfactory recommendations.” *2023, Constrained Exploration in Reinforcement Learning Evrard Garcelon*

# Offline RL Algo

## Off-policy Policy Gradient DRL

Algorithms:

1. Distributed Distributional Deterministic Policy Gradients <https://arxiv.org/abs/1804.08617>
2. “Offline RL is an active area of research and many algorithms have been proposed recently, e.g., BCQ [12] <https://github.com/sfujim/BCQ> <https://proceedings.mlr.press/v97/fujimoto19a/fujimoto19a.pdf> , MARWIL [33], BAIL [6], ABM [30] AWR [24], and CRR [34]. In our work, we adopt CRR due to its efficiency and simplicity. However, unlike standard offline RL, we do not observe the task rewards.” (see Chapter 2.) *2020, Semi-supervised reward learning for offline reinforcement learning, Ksenia Konyushkova*

Correct input to NN:

<https://it.mathworks.com/matlabcentral/answers/463693-reinforcement-learning-multiple-discrete-actions>

“In essence, offline reinforcement learning requires the

learning algorithm to derive a sufficient understanding of the dynamical system underlying the MDP

entirely from a fixed dataset, and then construct a policy that attains the largest possible

cumulative reward when it is actually used to interact with the MDP.

…

A more subtle but practically more important challenge with offline reinforcement learning is that,

at its core, offline reinforcement learning is about making and answering counterfactual queries.

Counterfactual queries are, intuitively, “what if” questions. Such queries require forming hypotheses

about what might happen if the agent were to carry out a course of action different from the one

seen in the data.” *Offline RL Tutorial, Sergey Levine*

## BC + Stitching

*“*We propose a model-based data augmentation strategy, Trajectory Stitching (TS), to improve the quality of sub-optimal historical trajectories. TS introduces unseen actions joining previously disconnected states: using a probabilistic notion of state reachability, it effectively ‘stitches’ together parts of the historical demonstrations to generate new, higher quality ones*” 2022, Model-based Trajectory Stitching for Improved Offline Reinforcement Learning, Hepburn and Montana*

## Monte Carlo Search Tree (Dynamical model needed step-by-step)

<https://www.analyticsvidhya.com/blog/2018/11/reinforcement-learning-introduction-monte-carlo-learning-openai-gym/>

## Sequence modelling for Environment solution – Beam search for planning

“However, we can also view reinforcement learning as analogous to a sequence generation problem, with the goal being to produce a sequence of actions that, when enacted in an environment, will yield a sequence of high rewards.

Taking this view to its logical conclusion, we begin by modeling the trajectory data provided to reinforcement learning algorithms with a Transformer architecture

In many model-based reinforcement learning methods, compounding prediction errors cause long-horizon rollouts to be too unreliable to use for control, necessitating either short-horizon planning or Dyna-style combinations of truncated model predictions and value functions. In comparison, we find that the Trajectory Transformer is a substantially more accurate long-horizon predictor than conventional single-step dynamics models.

Whereas the single-step model suffers from compounding errors that make its long-horizon predictions physically implausible, the Trajectory Transformer's predictions remain visually indistinguishable from rollouts in the reference environment.

We demonstrate the flexibility of this approach across long-horizon dynamics prediction, imitation learning, goal-conditioned RL, and offline RL” *2021, Offline Reinforcement Learning as One Big Sequence Modeling Problem, Janner, Levine* <https://bair.berkeley.edu/blog/2021/11/19/trajectory-transformer/>

“Offline reinforcement learning (RL) allows agents to learn effective, returnmaximizing policies from a static dataset. Three major paradigms for offlineRL are Q-Learning, Imitation Learning, and Sequence Modeling. A key open question is: which paradigm is preferred under what conditions? We study this question empirically by exploring the performance of representative algorithms — Conservative Q-Learning (CQL), Behavior Cloning (BC), and Decision Transformer (DT) — across the commonly used D4RL and ROBOMIMIC benchmarks.”*2023, Sequence Modeling is a Robust Contender for Offline Reinforcement Learning, Bhargava*

# Open-Loop Action Sequence Planning, Model based (not offline)

“The main difference between standard and single-step policy can be summed up as follows: where standard policies seeks the optimal set of actions yielding the largest possible reward, a single-step policy seeks the optimal mapping such that where denotes the network free parameters and is some input state (usually a vector of zeros) consistently fed to the agent for the optimal policy to eventually embody the transformation from to .” *2021, Single-step deep reinforcement learning for open-loop control of laminar and turbulent flows H. Ghraieb, J. Viquerat*

“A typical RL agent utilizes the following components for the purpose of learning and action selection: (i) state identification – the agent identifies its current state based on the sensory information received from its environment (e.g. visual cues); (ii) action selection – given its current state, and its knowledge about the environment, the agent evaluates possible actions then selects and executes one of them; (iii) learning – after executing an action, the agent enters a new state (e.g. receives new visual cues) and also receives a reward from the environment. Using this feedback, the agent improves its knowledge about the environment. This architecture is a closed-loop decisionmaking process because the action-selection process is guided by the current state of the agent, which is identified based on sensory inputs received from the environment. As we discussed in the previous section, action selection in sequence learning is not guided by environmental stimuli, and so does not require a state identification process. To explain sequence learning, therefore, we need to modify this framework.

Open-loop control:

A diagram of a system

Description automatically generated

Under certain conditions, it may be more beneficial for the agent to execute actions in a sequence without going through the action-selection process. … TODO read more *“2012, Habits, action sequences and reinforcement learning Amir Dezfouli and Bernard W. Balleine*

“In the open-loop setting, control and planning are the same because we are just finding a sequence of actions at an initial state (of an episode of a trajectory), and execute it with our eyes closed.” [*https://mandi-zhao.gitbook.io/deeprl-notes/model-based-methods/model-based-planning-and-model-based-predictive-control*](https://mandi-zhao.gitbook.io/deeprl-notes/model-based-methods/model-based-planning-and-model-based-predictive-control)

“In model-based reinforcement learning, action values are computed by mentally simulating the consequences of said actions within a representational map of the world that includes the states, the transition probabilities between them, and the rewards in each state — a ‘world model’, hence model-based reinforcement learning. In contrast, in model-free reinforcement learning, decisions are made without a world model. Instead, values for different actions are learned directly, through trial-and-error interaction with the environment every time an action is performed. ” 2020, Model-based decision making and model-free learning IN HUMANS, Drummond

Mention the risks of making the dynamical models, since small errors lead the system to diverge.

Agent can be LSTM

“Planning and learning may actually be combined, in a field which is known as model-based reinforcement learning. We define model-based RL as: ‘any MDP approach that i) uses a model (known or learned) and ii) uses learning to approximate a global value or policy function’”, *2022 Model-based Reinforcement Learning: A Survey, Moerland et al*

# RL Challenges

However, RL is considered sample inefficient, especially in the case of reward sparsity and large action spaces. <https://isl.anthropomatik.kit.edu/pdf/Retkowski2020.pdf>

# Offline RL libraries

<https://github.com/takuseno/d3rlpy>

<https://github.com/tinkoff-ai/CORL> <https://drive.google.com/file/d/18DfrEn6lqjCNHE3UFmAmGEUx-9chvApA/view>