1. **Model-based RL** RL learning aims at related tasks of optimising the value function, the policy and the behaviour based on the reward signal. The rewards is used only locally (in simple RL algorithms), such that models are used in order to use the experience of the agents in the environment more efficiently. Model learning faces analogous challenges: The model should be correct (w.r.t. the data available at a given time), it should not or overfit but generalise beyond the data, but not over-generalise and it should focus on relevant data (i.e. on regions of high reward or high regret) although “relevance” might change during learning. Asymptotically, the model is (ideally) on the one hand perfect and on the other hand not needed. Discuss the interaction of the aspects of the learning the action and the aspects of learning a model. It might make sense to remember here the Actor-Critic architecture.

This problem is to mainly to discuss the role of models in RL. It is obvious the a wrong model can be harmful if the agent does not explore beyond the model issues. As mentioned, the model cannot be correct with respect to the temporality of the agent (Niels Bohr: “Prediction is very difficult, especially about the future.”), because the agent improves beyond its past experiences. The model can, in a minimal sense, be used to remember all previous examples and if this is done in an off-policy algorithm then there is no problem with actions that turn out to be “wrong”. For continuous cases this might not be sufficient: Limited numbers of basis function need to be moved or local descriptions in terms of action-dependent state transitions may not be reliably integrable into trajectories (think of a pendulum near the upright position).

2. **Dyna-Q** The nonplanning method looks particularly poor in Figure 8.6 because it is a one-step method; a method using eligibility traces would do better. Do you think an eligibility trace method could do as well as the Dyna method? Explain why or why not. [from Sutton&Barto]

![Figure 8.6: Policies found by planning and nonplanning Dyna-Q agents halfway through the second episode. The arrows indicate the greedy action in each state; no arrow is shown for a state if all of its action values are equal. The black square indicates the location of the agent.](image)

It can be useful to find simply a good $\lambda$ value rather than adapting a full model, because sometimes also the simulation may have its cost. Obviously, even the best $\lambda$ value may not apply everywhere in the environment in the same way. In the S&B book also the question of full vs. sample back-up is discussed. I have not mentioned this in the lecture, so here is an opportunity. It is not a very deep idea and can be mapped at the difference between eligibility traces (sample) and model (full), although obviously also in a model less than the full trajectory tree may be considered.

3. **Dyna-Q** Why did the Dyna agent with exploration bonus, Dyna-Q+, perform better in the first phase as well as in the second phase of the blocking and shortcut experiments? Careful inspection of Figure 8.8 reveals that the difference between Dyna-Q+ and Dyna-Q narrowed slightly over the first part of the experiment. What is the reason for this? [from Sutton&Barto]
You may remind the students first to the method of optimistic initialisation. Of course the answer depends on whether other exploration is used here, on any termination rules for the episode, and on the punishment per step. In most cases the “+” algorithm just explores better, while without “+” the algorithm tend to get stuck with earlier experiences. After the change the “+” version still explores, so it picks up what has changed.

4. **[Dyna-Q]** Dyna-Q+ uses exploration bonuses, e.g. of the form \( r + \kappa \sqrt{n} \). This is a kind of an intrinsic reward. Discuss advantages and problems in connection to intrinsic rewards. Consider also the aspect of evolutionary learning.

Intrinsic rewards are a bias. If the problem are typically such that a bias is useful, then it makes sense to use it. A slow and steady evolutionary algorithm can find out what bias is useful.

5. **[Dyna-Q]** Prioritised sweeping (see Sutton&Barto section 8.4 in 2nd ed.): While Dyna agents select state-action pairs uniformly at random from the previously experienced pairs, it might be more efficient to use a non-uniform probability distribution. Why? Which state-action pairs should be preferred? Discuss the role of a goal states in this context.

The idea is that “relevance” is not only about goals but about all changes (i.e. new information) in the reward/value structure. See also the questions 1 and 4 above.

6. **[MORL]**

(a) Under what conditions is a Pareto front not-connected?

The Pareto front of a \( D \) dimensional set of solutions must be a continuous function of each set of \( D-1 \) variables. Obviously, if one of these functions is multivalued, then the larger value dominates the smaller values. If a part of the Pareto front was parallel to an axis, then the values along this part would dominate each other in contradiction to the definition, but in this case would the front not be a continuous function w.r.t. the other variable(s). One can also say that the normal of the front must be strictly in the positive (hyper-)quadrant. For a different criterion see www.math.bas.bg/inres/MathBalk/MB-26/MB-26-399-407.pdf

(b) How can a MORL agent reach other connectivity components of the Pareto front? Can a scalarised MORL algorithm reach all points on the Pareto front by testing all combinations of weights in a weighted sum of the value function?

A local search along the front could also continue through the boundaries that are not part of the front. This would required some additional mechanisms, as the local search must run on the inside of the front. Along the boundary between connectivity components of the front, the maximisation of one of the criteria would lead outside the front, i.e. the search gets stuck. In order to avoid this the criterion must be relaxed a bit rather than optimised, (i.e. when the algorithm gets stuck).

In a scalarised algorithm one can start with many linear combinations of the criteria and will find in this way points on the front as well as points which are not part of the front, i.e. which are dominated by front-points that cannot be found in this way.

Since both ways have their problems, we should either use a combination: Finding several points
7. **[MORL]** *How can policy gradient methods be adapted to the MORL problem?*
   For a scalarised approach, we can use the same weights also for the gradients w.r.t. the individual rewards. For a somewhat more general algorithms, see lecture slide Multi-Objective Policy Gradient (Parisi et al. 2014).

8. **[MORL]** *Instead of using MORL, it may be possible to provide the agent with appropriate state information, e.g. instead of a spatial state and rewards for (a) reaching a goal and (b) keeping batteries charged, we could design a state that contains both information about the battery level and about the spatial position. Compare the two variants.*
   Admittedly, the question is not very well posed, but may still help to understand MORL better. Whatever states we define, we still need a single reward for standard RL. One could add information that in some states one reward signal is used, in other states other rewards or, more generally, define for different states different combinations of the rewards. What combinations are useful cannot be inferred from the states, but is a question of domain knowledge. On the other hand for the MORL situation also the full state information is required.

9. **[MARL]** *Apply RL to the iterated prisoners dilemma. Discuss several scenarios: (i) Two prisoners, (ii) simultaneous plays between pairs randomly selected from many prisoners, (iii) prisoners are situated in a plane, plays with neighbours (iv) different state definitions in the group. Discuss also the effect of details of the RL algorithm.*

<table>
<thead>
<tr>
<th>column player</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>cooperate (C)</td>
<td>defect (D)</td>
</tr>
<tr>
<td>row</td>
<td>R (= 0.3)</td>
<td>S (= 0.0)</td>
</tr>
<tr>
<td>player</td>
<td>T (= 0.5)</td>
<td>P (= 0.1)</td>
</tr>
</tbody>
</table>


The first problem is to decide about the history, i.e. the state. In principle histories need to be infinitely long, which is practically not possible.

In this paper a number of cases are considered: First on agents plays against Tit-for-Tat. Surely enough the learning agent learns Tit-for-Tat, but it takes a number of iterations. This implies some complexity in the more general situations which may not be resolved even in long plays. If they play only against neighbours, the complexity a Tit-for-Tat region can slowly grow. If the learning agents are not equal, those with longer history or more precise representations (look-up table is more precise than function approximation) will have an advantage that they will use in order to get more reward.

If the agent use a meta-algorithm to change their state representation (i.e. can take an action that changes history length) then the dynamics becomes chaotic, i.e. histories will grow, but if they get longer than the learning speed allows, then agents with shorter histories can take over again. This is a nice problem to run some own simulations, many subquestions are still subject of current research.

10. **[Applications]** *Assume your are heading a large team of researches and technicians working on building a humanoid robot. Motivate your coworkers to use RL in various tasks related to the project. Consider hierarchical approaches, specify sub-tasks and required resources. Discuss alternatives. What options for hybrid algorithms might be interesting here?*
   This is an open-ended question. The project leader could first identify subtasks, where RL can be used in a standard form. This would yield a number of policies which can be used as options in a high-level
RL algorithm. Another approach would be to use MORL: The robot gets rewards for standing up, walking, doing dishes etc. and would choose the task were most reward is obtained. This should perhaps involve some type of R-learning where good performance at one reward, would make appear other types rewards more promising.

11. **Applications** Assume your are heading a large team of researches and technicians working in robot soccer. Motivate your coworkers to use RL in various tasks related to the project. Consider hierarchical approaches, specify sub-tasks and required resources. Discuss in particular team-level learning tasks and the role of models.
   This is an open-ended question. Similar to the previous, but here we should also consider after states and decentralised RL.