Homework 5 Template

Use this template to record your answers for Homework 5. Add your answers using LaTeXand then save your document as a PDF to upload to Gradescope. You are required to use this template to submit your answers. **You should not alter this template in any way** other than to insert your solutions. You must submit all 15 pages of this template to Gradescope. Do not remove the instructions page(s). Altering this template or including your solutions outside of the provided boxes can result in your assignment being graded incorrectly. You may lose points if you do not follow these instructions.

You should also save your code as a .py or .zip file and upload it to the **separate** Gradescope coding assignment. Remember to mark all teammates on **both** assignment uploads through Gradescope.

Instructions for Specific Problem Types

On this homework, you must fill in (a) blank(s) for each problem; please make sure your final answer is fully included in the given space. **Do not change the size of the box provided.** For short answer questions you should **not** include your work in your solution. Only provide an explanation or proof if specifically asked. Otherwise, your assignment may not be graded correctly, and points may be deducted from your assignment.

Fill in the blank: What is the course number?

10-703

Problem 0: Collaborators

Enter your team's names and Andrew IDs in the boxes below. If you do not do this, you may lose points on your assignment.

Name 1:	Eu Jing Chua	Andrew ID 1:	eujingc
Name 2:		Andrew ID 2:	
Name 3:		Andrew ID 3:	

Problem 1: Model-based Reinforcement Learning with PETS (80 pts)

1.1 Planning with Cross Entropy Method (CEM) [20 pts]

1.1.1 CEM implementation (5 pts)

Success rate is 0.86

1.1.2 Plan with random actions (5 pts)

Success rate is 0.80
The performance is slightly worse than that observed in CEM.

1.1.3 MPC implementation; Comparison of MPC in environment w/o noise (10 pts)

Method	Pushing2D	Pushing2DNoisyControl
CEM	0.86	0.48
CEM + MPC	0.96	0.70

We can see that in both environments, CEM with MPC performs better than just CEM. This is especially evident in the noisy control environment, where CEM with MPC greatly outperforms just CEM.

This makes sense as with MPC we are more reactive to the environment, especially when it is stochastic. This is because we plan every time-step and only use the first action in the best plan found, instead of all the actions like in pure CEM. If our resulting state deviates from what was predicted in the plan, we will replan anyway to pick the best action given the new state, unlike pure CEM where we stick to our established plan until we hit the plan horizon.

However, the con with MPC is that it is much slower than pure CEM. In pure CEM, we only have to search for an action sequence once every plan horizon number of time-steps. However with MPC, we search for an action sequence every time-step instead.

1.2 Probabilistic Ensemble and Trajectory Sampling (PETS) [60 pts]

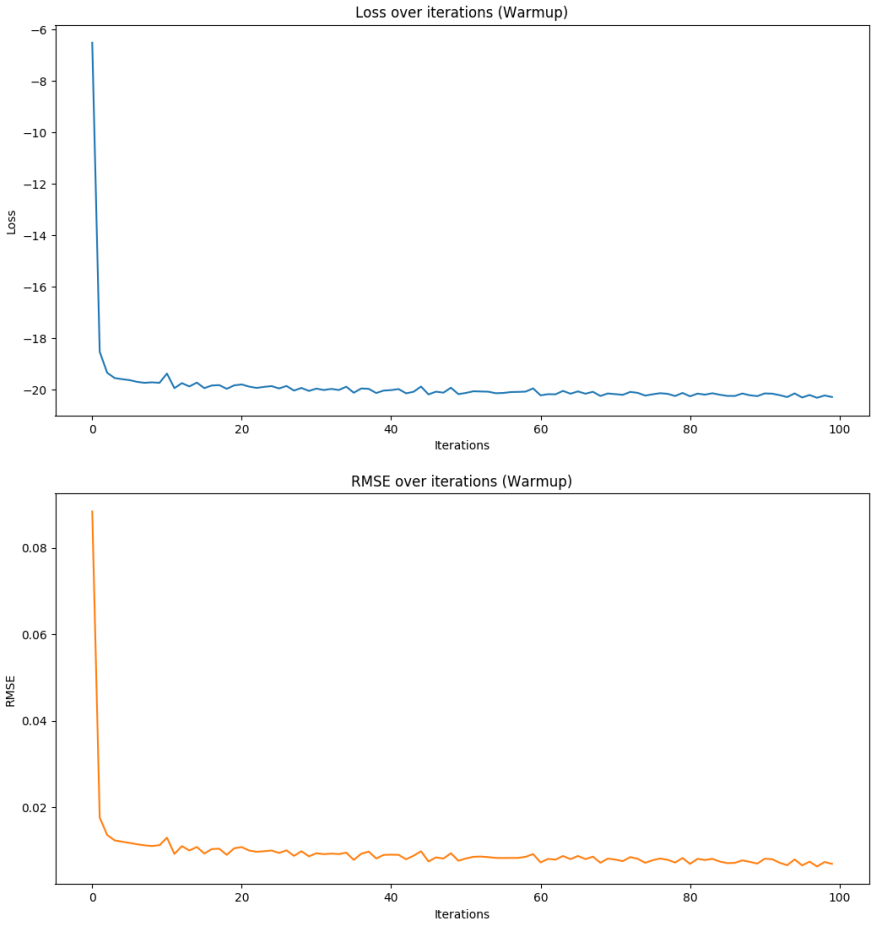
1.2.1 Derive the Loss of a Probabilistic Model (5 pts)

$$\mathcal{L}_{Gauss} = -\frac{1}{|D|}\sum_{i=1}^{|D|} \log p(s_{t+1}^{(i)} \mid s_t^{(i)}, a_t^{(i)})$$

$$= \frac{1}{2|D|}\sum_{i=1}^{|D|} d\log(2\pi) + \log(\det \mathbf{\Sigma}_\theta(s_t^{(i)}, a_t^{(i)})) +$$

$$(s_{t+1}^{(i)} - \mu_\theta(s_t^{(i)}, a_t^{(i)}))^T \mathbf{\Sigma}_\theta(s_t^{(i)}, a_t^{(i)})^{-1}(s_{t+1}^{(i)} - \mu_\theta(s_t^{(i)}, a_t^{(i)}))$$

1.2.2 Loss and RMSE of a single dynamic model (5 pts)



1.2.3 Planning on single model with random actions + MPC (5 pts)

Success rate is 0.58

1.2.4 Planning on single model with CEM+MPC (10 pts)

Success rate is 0.68

1.2.5 Discussion on the comparison between CEM and random actions (5 pts)

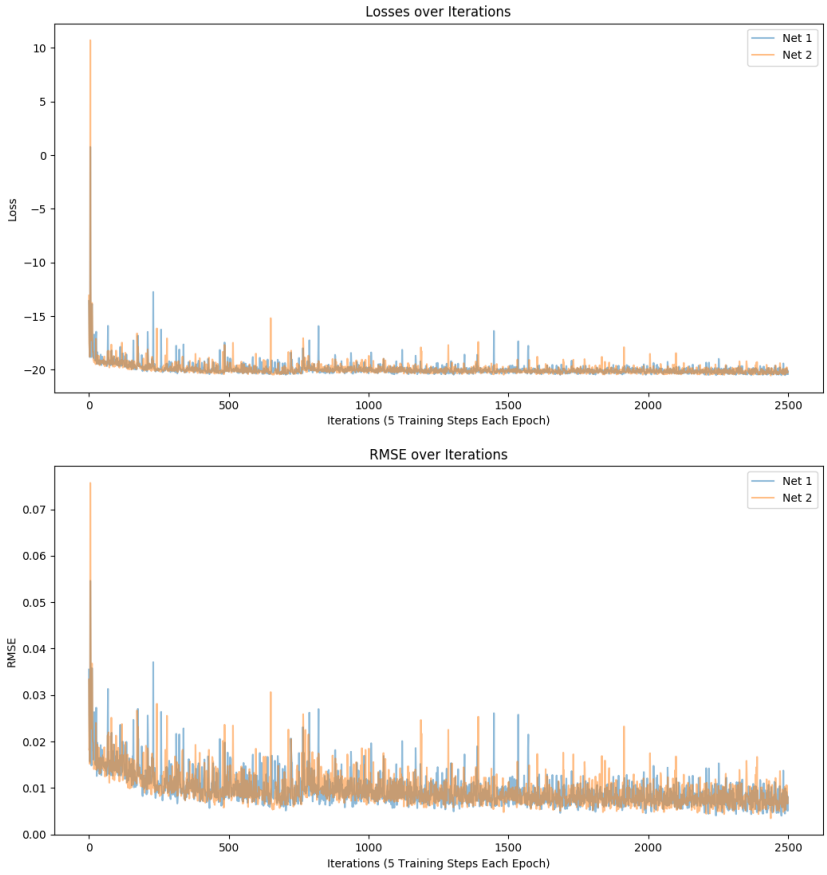
We can see that CEM performed slightly better than Random, both when using MPC and a trained dynamics model. MPC performs well with this model as it is reactive to differences between what the dynamics model predicted, and what the true dynamics actually produced. However, the policy can only succeed when the trained dynamics model well approximates the true dynamics.

1.2.6 Description of the implementation details (5 pts)

For most of the implementation, I have used the default hyperparameters. However, I have extra implementation details as follows:

1. For CEM, I lower bound the standard deviation from the elite population to 0.1 to prevent a collapse in exploration
2. For MBRL, instead of only training on the transitions collected over 1 episode and then discarding them (usually between 10 - 40), I keep a buffer of the 5000 most recently collected transitions and use this as D instead. This allowed much more consistent learning of the networks. I initially populate this buffer with transitions sampled from 100 episodes of a random policy.
3. TS1 is implemented just by randomly selecting one of the networks everytime we wish to predict the next state.

1.2.7 Loss and RMSE of the probabilistic ensemble model (10 pts)



1.2.8 Success percentage of CEM+MPC and random actions + MPC (10 pts)

Dynamics	Method	Rate
Model	Random + MPC	0.58
Model	CEM + MPC	0.68
GT	Random + MPC	0.90
GT	CEM + MPC	0.96

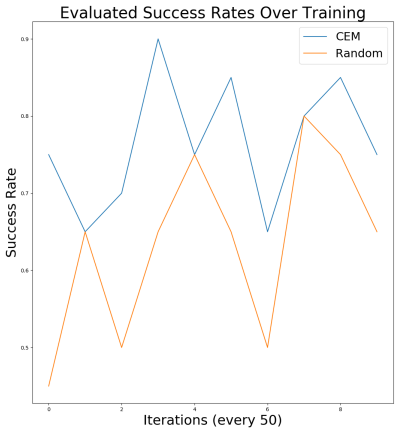


Figure 1: Left: Single Model and Ground Truth Dynamics. Right: PETS

From the above plot of success rates over the course of the MBRL algorithm, CEM+MPC tends to perform better than Random+MPC for most parts but not by a lot. There are times where they get similar success rates. In comparison to the single model case, we see a similar behavior where CEM+MPC outperforms Random+MPC, but also not by a lot. In the ground truth dynamics experiment, we see CEM+MPC almost performing perfectly, while Random+MPC is just a little behind.
This makes sense as the performance of the derived policies greatly depend on the quality of the underlying dynamics. CEM generally is better than Random, but only if the underlying dynamics model is a good one.

1.2.9 Limitation of MBRL (5 pts)

MBRL is limited by our dynamics model's capacity to model the true dynamics. If we fail to model the true dynamics well, then derived policies will never be good. In this case, we also need to have a good cost function which might take a lot of effort to design.
However, we would want to use MBRL in an environment where we want our models to be able to generalize and achieve different goals which are target states in the environment. This would be hard to achieve with policy-gradient methods.

Problem 2: Theoretical Questions (20 pts)

2.1 Deterministic vs Stochastic Model (10 pts)

This is not true for all MDPs. A counter-example would look like this:
Imagine we had an MDP where for each state and action taken (s, a), there is a very high-number of next-states s' with non-zero probability for the stochastic dynamics. The usual optimal policy would be the deterministic one that picks the action that leads to s' with the highest value function $V^*(s')$.
When we only consider deterministic dynamics for the same MDP, we want to know which of the many possible next-states we would like to assign probability 1. To have the optimal policy be the same, we would require the assignment to be the one that matches the above (s, a) to s' with the highest $V^*(s')$. However, this requires knowing $V^*(s')$ with the stochastic dynamics in the first place which defeats the purpose of finding a deterministic fit for the dynamics.
Otherwise, we would have to enumerate through all possible assignments of the next-states, which would be intractable too.
Therefore, this would be infeasible.

2.2 Aleatoric vs Epistemic Uncertainty (5 pts)

For measuring aleatoric uncertainty, we can measure the variance of the output distribution parameterized by the output of our networks, assuming the networks agree. For example when out networks parameterized Gaussian actions, they estimate the tightest Gaussian based on the actual transitions it has seen before. If there was high stochasticity for particular transitions that have been observed before, then the network would predict the a correspondingly high variance for the output Gaussian. If there was low stochasticity instead, then the network would predict a correspondingly low variance for the output Gaussian.

For measuring epistemic uncertainty, we can measure how different the output distributions are between the different networks. Based on the bootstrap way of training the different networks on different resamples of the data, the networks should be making similar predictions on data they have both seen before, and different predictions on data they have not seen before. This manifests as either close or far output distributions of the actions. Thus, we can see some sort of metric of measuring the distance between output distributions, such as KL or Jensen-Shannon divergence, or even just vector distance metrics on the raw distribution parameters of the network outputs, such as L2.

2.3 Failure Modes without Considering Uncertainty (5 pts)

If aleatoric uncertainty was not considered, our dynamics model would be a deterministic dynamics model. If the actual dynamics was stochastic, then we know a deterministic model will not induce the same optimal policy as the original optimal policy for the stochastic dynamics.
If epistemic uncertainty was not considered, then we would not know how well the current dynamics model really models the true dynamics. It would not be easy to identify a situation in which the model has not seen data from particular states and actions as uncertainty is not propagated. Thus, we might observe inconsistent performance when we occasionally encounter these areas we have not explored.

Feedback (1pts): You can help the course staff improve the course for future semesters by providing feedback. You will receive a point if you provide actionable feedback **for each of the following categories**.

What advice would you give to future students working on this homework?

Make sure to observe standard deviations across iterations in things like CEM, and to make sure they do not collapse to 0 as it would kill exploration.
Also, check on how much data the networks actually train on each iteration to make sure it is a sufficient amount that would not cause high variance gradient updates.
Finally, tensorflow probability was a great way to sanity check my loss function.

Time Spent (1pt): How many hours did you spend working on this assignment? Your answer will not affect your grade.

Alone	20
With teammates	0
With other classmates	0
At office hours	2