# Ideas for Assignment 3

## 1 Sheepdog Bot 1

1. The easiest states to determine $T^*(state)$ is those states where the sheep is already at the center node and the sheepdog is not at the center node.
2. For any other given state state, express a formula for $T^*(state)$ in terms of the action the bot can make and the sheep can respond. For example, if the sheepdog is at (15, 16) and the sheep is at (15, 15), the sheepdog can move up, down, left, or right, and the sheep can move up, down, left, or right. If the sheepdog moves up, the sheep can move up, down, or left. If the sheepdog moves down, the sheep can move up, down, or right. If the sheepdog moves left, the sheep can move up, left, or right. If the sheepdog moves right, the sheep can move down, left, or right. Therefore, $T^*(state)$ is the average of $T^*(state)$ when the sheepdog moves up, down, left, or right.
3. If we could determine $T^*(state)$ for each state, the optimal action that the sheepdog should take is the action that minimizes $T^*(state)$.
4. Based on our computation, this number should be 49.5.
5. Intuitively, the sheep dog should start at (15, 16) due to symmetry. By calculation, the sheep dog should start at the position where its expected steps to catch the sheep is the smallest. The sheep dog should start at (15, 16) because the expected steps to catch the sheep is **remained to be filled**, which is the smallest among all the positions.
6. It seems that the simulated results take more steps than the expected steps. This is because the expected steps is the average of all the possible steps, but the simulated results is only one of the possible steps. Therefore, the simulated results may take more steps than the expected steps.

## 2 Sheepdog Bot 2

1. The input states can be represented as a 4-dimention interger array, where the first two elements are the coordinates of the sheep, and the last two elements are the coordinates of the sheepdog. I would like to use linear regression as the model. Input features like the relative state to the target should be useful. I would like to use MAE (mean absolute error) as the loss. The training algorithm should be gradient descent. Since we only need to know a subset of exact {T^*}, our model is much less than the size of the fully computed T^*. For instance, if we only use the center 16*16 grid, the size of the model is 16*16*16*16=65536, which is much less than 31*31*31*31=923521.
2. I can compare the predicted value of the model with the exact value of T^* using MAE error. Overfitting could be a problem so I can use regularization to prevent overfitting. For example, I can use L1 regularization, which adds a penalty term to the loss function, to prevent the model from overfitting.
3. No, it is not that reliable.

## 3 Sheepdog Bot 3

This bot utilizes the idea of machine learning. Since we have created and stored a large amount of data, we can solve this problem from a statistical perspective. We can use the data to train a model that can predict the next move of the sheepdog.

1. (a) Input space: the state of the game, which is a tuple of the positions of the sheep and the sheepdog.
   (b) Output space: the next move of the sheepdog, which is the direction of the sheepdog's movement.

    (c) Model space: a function that maps the input space to the output space. In particular, we can model this problem as a classification problem, and therefore we may choose tree-based classifiers, such as decision tree, random forest, and gradient boosting.

2. To assess whether the model suffers from overfitting, we can use cross-validation. In particular, we can use k-fold cross-validation, where we split the data into k subsets, and use one subset as the testing set and the rest as the training set. We can repeat this process k times, and then average the results to get a more accurate estimate of the model's performance. Overfitting of course is an issue, but we can use regularization to prevent overfitting. For example, we can use the L2 regularization, which adds a penalty term to the loss function, to prevent the model from overfitting.

3. Yes, it is reliable. The model is trained on a large amount of data, and therefore it is reliable.