Homework 4

Problem1:

(a) According to the No Free Lunch Theorem, the true error can be high even though there exists a function that classifies everything perfectly. In this case, our solution can be restrict the hypothesis class based on some prior knowledge of the distribution D and real mapping function f. However, as we restrict the hypothesis class, we bias the predictor toward a particular set of predictors. That's where the inductive bias comes.

(b) Yes. Since it's overfitting right now, the approximation error can be very small. Reducing the size of hypothesis class will not necessarily increase the approximation error. Meanwhile, the estimation error decreases as we reduce the size of hypothesis class. Therefore, we could reduce the error by reducing the size of hypothesis class.

(c) Collecting more training data will reduce the error. Because the estimation error decreases with m. The m doesn't change the approximation error. Therefore, collecting more training data will reduce the total error.

Problem2:

(a) Assumption1: The data that we sampled from D is independently and identically distributed.
Assumption2: The hypothesis class is finite.
Assumption3: The sample space is realizable.

(b) Because the data sampled by HTAs is not independently and identically distributed. One sample has some relation with previous sample. That's the reason why the ERM solution doesn't perform well.

(c) The training data and testing data obtained by Steve is not from the same distribution D. Even though the predictor can work well on baby animals, it does not necessarily perform well on adult animals.

Problem3:

(a) The lower bound of $|H|$ is that only one particular permutation is mapped to 1, while all others are mapped to 0. In this case, $|H| = n!$, which is the number of different permutations.

The upper bound of $|H|$ is to consider a partial ordering on a set of n elements as a DAG. The upper bound of $|H|$ becomes the number of different DAGs. Since it's hard to count the number of DAGs, we can choose the number of different

DGs instead. Apparently, the number of DGs can be a upper bound of number of DAGs.

There are n possible vertices in the graph. A edge can be considered as an ordered pair of vertices. Also, taking self-loops into consideration, there are n^2 possible edges. For each edge, we can include it as a directed edge going in the increasing direction, include it going in the decreasing direction, or not include it at all. Therefore, there are 3^{n^2} possible DGs, which is one upper bound of the $|H|$.

(b) To ensure the ERM solution is PAC, $m \leq \frac{log(|H|/\delta)}{\epsilon}$. From (a), we know that the upper bound of H is 3^{n^2}.

Therefore, $m \leq \frac{log3^{n^2} - log\delta}{\epsilon} = \frac{n^2 log3 - log\delta}{\epsilon}$, which is a polynomial in n

(c) **Step1**: Find each and every permutation that mapped to 1.

Step2: For each permutation obtained from Step1, generate all corresponding orders. For example, for a permutation ABC, generate (A,A),(A,B),(A,C),(B,B),(B,C),(C,C). That is to enumerate each element and all elements behind it including itself.

Step3: Use a hashset "partialOrdering" to store the generated order in Step2, use another hashset "result" to store the hypothesis that we are looking for.

Step4:If there is only one permutation that mapped to 1, store all the orders generated in Step2 in the "result" hashset. If there are more than 1 permutations that mapped to 1, we should check whether the order already exists in "partialOrdering" hashset before adding it to the hashset. If so, store this partial ordering in the "result" hashset.

Step5: The "result" hashset is the partial ordering or the hypothesis that we are looking for.

Time Complexity: Since there are $n!$ permutations. For each permutation, we need $O(n^2)$ to generate all orders in Step2. Since we use that hashset to store the order, the adding and checking existence operation is in $O(1)$ time complexity. Combining all steps above, the total Time Complexity is $O(n^2 * n!)$.

Correctness: The algorithm is a kind of brutal force. Analyze all permutations that mapped to 1, and generate the corresponding partial ordering. Then find the union of all partial ordering generated. Since the final partial ordering is subset of each partial ordering that generated from a permutation, the empirical risk must be 0.