Homework 4

Due: Thursday, February 20, 2025 at 12:00pm **(Noon)**

Written Assignment

Problem 1: Bias-Complexity Tradeoff

(10 points)

a. Explain why the No Free Lunch Theorem requires us to introduce bias into our machine learning models.

   Solution: The No Free Lunch Theorem states that every learning algorithm will perform poorly over some data distribution. Intuitively, this is because the learning algorithm can find a function that performs very well on the observed training data but very poorly on the unobserved testing data. To limit the ability of learning algorithms to choose these poorly performing functions, we use our prior knowledge to exclude functions that we believe are unrealistic from our hypothesis class. This exclusion introduces bias into the learning process, but it can help avoid unexpectedly poor performance on test data.

b. While training a machine learning model, Steve notices that the model's error remains large even after many training iterations. Steve determines that this error results from overfitting, and he proposes changing the complexity of the hypothesis class to reduce the error. Will changing the complexity of the hypothesis class successfully reduce the error? If so, how should the complexity be changed? If not, why not? Explain your reasoning.

   Solution: If Steve reduces the complexity of the hypothesis class, this approach will reduce the error. Reducing the complexity of the hypothesis class gives the learning algorithm fewer options for optimally fitting the training data. That is, this approach will reduce overfitting because the learning algorithm will be forced to fit the training data with a less complex hypothesis.

c. After tinkering with his model, Steve eventually manages to prevent his model from overfitting. However, Steve notices that his model's error remains large because his model is now underfitting the data. To decrease underfitting, Steve decides to collect more training data. Will collecting more training data successfully reduce the error? Explain your reasoning. *Hint: Consider if collecting more training data affects approximation and/or estimation error, and if so, explain how.*

   Solution: Collecting more training data will not reduce the error because the training data is already too complex for the hypothesis class to fit well. More training data is useful for reducing estimation error, but it is not useful for reducing approximation error. A better approach to reduce approximation error would be to increase the size/complexity of the hypothesis class.

d. Tuning hyperparameters to strike a balance between underfitting and overfitting is a critical part of the machine learning process. Going too far in either direction can often lead to unintended consequences. What are some ways that overfitting or underfitting could produce unfairness in a learned model?

   Solution: Ethical question- Any reasonable answer satisfying the rubric requirements should get full credit.

Problem 2: Assumptions & Guarantees

(10 points)

a. In lecture, we made several assumptions before proving that an ERM solution is PAC learnable. What are these assumptions?

Solution: The assumptions that we need to make are:

- **I.I.D Data**: We assume that we draw independently and identical data (or that $\mathcal{D}^2(x, y) = \mathcal{D}(x)\mathcal{D}(y)$)
- **Finite Hypothesis Class**: We assume that $|\mathcal{H}| < \infty$ (although not necessary when we get to VC Dimension)
- **Realizability**: $\exists h \in \mathcal{H}$ s.t. $L_{\mathcal{D}}(h) = 0$.

b. The HTAs want to produce a model to predict the final grade of a student based off of their Homework 6 score. During the sampling process, the HTAs select the next example based on how similar it was to the previous example. Why does the ERM solution not perform well at test time? Explain in terms of the assumptions above.

Solution: This breaks the assumptions that the data is sampled i.i.d. This doesn't guarantee an ERM solution because the training process changes the probability that all examples will be seen, so that the learned hypothesis might only perform well on a subset of the dataset, rather than all the examples.

c. Steve wants to produce a model to predict the grams of sugar in a cake based on its characteristics (grams of flour, calories and number of eggs). He trains his model on a dataset of **muffins** and then uses it to make predictions about **cakes**. Why does the ERM solution not perform well at test time?

Solution: This fails because the train and test distributions are not the same. The hypothesis that is produced would perform well in predicting the sugar content of muffins, but would significantly underestimate the sugar content of cakes. Since the distributions are different, we have no guarantees that the true risk of the ERM hypothesis is close to zero.

Problem 3: PAC Learning of Partial Orderings

(20 points)

Background Information

A *partial ordering* (denoted $\preceq$) is a binary relation over a set X that satisfies the following properties:

- *Reflexivity*: $\forall s \in X, \quad s \preceq s$
- *Antisymmetry*: $\forall s, t \in X,$ if $s \preceq t, t \preceq s$, then $s = t$
- *Transitivity*: $\forall r, s, t \in X,$ if $r \preceq s, s \preceq t$, then $r \preceq t$

A *total ordering* is a partial ordering with the additional guarantee that any two elements of the set A are comparable, that is, for any two $a, b \in A$, either $a \preceq b$ or $b \preceq a$ is true. In the case that $\preceq$ is a partial ordering and neither $a \preceq b$ nor $b \preceq a$ are true, we say that a and b are *incomparable* under the partial ordering $\preceq$.

Example: The "$\leq$" relation is a total ordering on the real numbers. Any two numbers a, b may be compared such that either $a \leq b$ or $b \leq a$. Additionally, the "$\leq$" relation satisfies the conditions of reflexivity, antisymmetry and transitivity stated above.

Example: The "$\subseteq$" relation is a partial ordering but not total ordering. "$\subseteq$" satisfies reflexivity, transitivity and antisymmetry. However, given two sets A and B, there is no guarantee that either $A \subseteq B$ or $B \subseteq$ A.

Problem

Now consider the following:

- Let X be a set of n unique elements: $\{1, 2, 3, ..., n\}$.
- **Input Space:** The set S of all possible permutations of length n, composed the elements of X.
  For example, if X is a set of length $n = 5$, a permutation could look like $(3, 4, 2, 5, 1)$, and S would be the set of all different possible 5-element permutations.
- **Hypothesis Space (H):** The set of all hypotheses, where each hypothesis maps all n-element permutations to either 0 or 1. Each hypothesis represents a partial ordering. So, according to the hypothesis, a n-element permutation would be mapped to 1 if it abides by the partial ordering and to 0 otherwise. For example, if a hypothesis h is $1 \preceq 3$, and we have the permutation $(3, 1)$, the permutation does not abide by the hypothesis since 3 is before 1, reading left to right. Therefore, h would map $(3, 1)$ to 0.

Suppose that there is a hidden partial ordering that we do not have access to. We have a sample of permutations, and a label for each sample indicating whether the permutation is consistent with the true hidden partial ordering. Assume that the sample has both positively and negatively labelled permutations. We are trying to learn the rule from this sample.

Example: Let $\preceq$ be a partial ordering on the set $X = \{1, 2, 3\}$, defined such that $1 \preceq 2$ and 3 is incomparable to both $1, 2$. Let h be the hypothesis associated with $\preceq$; that is, h is the hypothesis which maps all permutations of X abiding by $\preceq$ to 1 and all permutations not abiding by $\preceq$ to 0.

Then, the permutations

$$(1, 2, 3), \qquad (1, 3, 2), \qquad (3, 1, 2)$$

abide by $\preceq$ (notice that 1 precedes 2 in all three and it does not matter where 3 is) and are labelled 1 by h. However,

$$(2, 1, 3), \qquad (2, 3, 1), \qquad (3, 2, 1)$$

do not abide by $\preceq$ since 2 precedes 1 in all three of these. Hence, they are labelled 0 by h.

a. Show that $|H|$ is between $n!$ and $3^{(n^2)}$. That is, show that the number of possible valid partial orders generated from n elements is bounded by $n!$ and $3^{(n^2)}$.
Hint: For the upper bound, it may be helpful to consider that $h \in H$ may be represented as a Directed Acyclic Graph (DAG). The nodes of the DAG can be thought of as the elements that make up the n-element permutations of the input space. Think about how the relations between nodes in a DAG are similar to the relations between elements within a partial ordering. This question may also therefore be approached by thinking about the number of valid DAGs with n nodes.

Solution: On the lower bound, we know that any total ordering will be a partial ordering as well. As a total ordering will order all elements within the set, it is essentially a permutation. There are $n!$ total orderings, so there are at least $n!$ partial orderings.

For the upper bound, think about each n element as a node in a DAG. There are (on the order of) n^2 pairs of nodes. (The exact number would be $\frac{n(n-1)}{2}$) Reflexivity may be handled separately by drawing an edge from every node to itself. Between any pair of elements i and j, you have three possibilities:

- A directed edge from i to j.
- A directed edge from j to i.
- No edge between i and j.

Intuitively this makes sense because in a partial ordering, you can either have a relationship between two elements or no relationship between two elements (e.g., if the partial ordering is subsets, comparing $\{a, b\}$ with $\{c, d\}$ would give us no relationship between two elements). Note that this collection of 3 states also enforces antisymmetry. This gives us a total of $3^{(n^2)}$ possible partial orderings. Note that this bound is not tight because it allows for cycles and other violations of transitivity; however, every partial ordering is indeed within this set.

b. Show that the amount of data needed is polynomial in n (provide the asymptotic bound using the Big-O notation), in order to ensure the ERM solution is PAC. *Hint: Consider the bounds for $|H|$ we proved in part (a).*

Solution:

The dependence on the hypothesis space is $log(|H|)$, which has an upper bound of $log(3^{(n^2)}) = O(n^2)$, so this is polynomial in n.

c. Describe an algorithm you could use to efficiently find an ERM solution (the true partial ordering, or DAG) for a given sample of observations. Be sure to explain the steps of your algorithm, prove the correctness of the algorithm and analyze its runtime. Note, we do not have a strict runtime requirement, but the algorithm should be in polynomial time. The algorithm itself can be written in pseudocode or in words. *Hint: The true hypothesis is within our hypothesis space. Consider the direction of the ordering between any two elements in every permutation with label 1.*

Solution: We begin by looking at the positive examples in the training set. For each arbitrary pair of nodes i, j, we check if we always have i before j, j before i, or if both orders exist, in all of the positive examples. Orders that are consistent throughout all positive examples create a solution corresponding partial order. This collection of constraints will always adhere to the definition of a partial order:

- If for all positive instances, $i \preceq j, j \preceq k$, it follows that $i \preceq k$ for all positive instances.
- It cannot be simultaneously true that $i \preceq j$ and $j \preceq i$ for any given instance.

Intuitively, given that all orders found work in every example, all such orders composed as a partial order work in each positive example.

We are given the true hypothesis is within our hypothesis space, so the ERM should be able to correctly classify all data in our sample. Our partial ordering is correct in positive classification by construction. We will now prove our partial ordering correctly classifies all negative instances.

Proof. Proof by contradiction: Suppose our ERM partial ordering, which is the true hypothesis, incorrectly classifies a negative instance as positive. The negative instance must therefore satisfy all ordering constraints that are shared among all of the positive instances. This indicates that there are no constraints that uniquely differentiate the set of positive instances from this negative instance. Thus, there exists a better partial ordering and the current ordering is not the true hypothesis. Therefore, our ERM partial ordering cannot incorrectly classify a negative instance. □

Constructing the ERM partial ordering is polynomial over the sample size $|S|$ and the size of the permutations n. Determining whether to include each potential i, j pairing in the partial ordering occurs in $|S|$: checking whether a particular instance satisfies the rule occurs in constant time, and there are at most $|S|$ positive instances to check. There are $O(n^2)$ rules to check, so the entire algorithm runs in $O(|S|n^2)$.

Possible solutions:

Solution 1: Constructing the ERM partial ordering is polynomial over the sample size $|S|$ and the size of the permutations n. First, preprocess the data by storing the index of each element in each permutation in a dictionary. This process takes $O(|S|n)$ time because there are $|S|$ permutations, each of length n. Next, determine whether to include each potential i, j pairing in the partial ordering. This process occurs in $O(|S|)$ time for each pairing: checking whether a particular instance satisfies the rule occurs in constant time because we can use the previously constructed dictionary, and there are at most $|S|$ positive instances to check. There are $O(n^2)$ total rules to check, so the entire algorithm runs in $O(|S|n + |S|n^2) = O(|S|n^2)$ time.

Solution 2: Constructing the ERM partial ordering is polynomial over the sample size $|S|$ and the size of the permutations n. Determining whether to include each potential i, j pairing in the partial ordering occurs in $O(|S|n)$ time: there are at most $|S|$ positive instances to check, and it takes $O(n)$ time to read through each positive instance to determine the order of i and j. There are $O(n^2)$ rules to check, so the entire algorithm runs in $O(|S|n^3)$ time.

Grading Breakdown

The grading breakdown for the assignment is as follows:

Problem 1	25%
Problem 2	25%
Problem 3	50%
Total	100%

Handing in

Your written assignment should be uploaded to Gradescope under "Homework 4". If you have questions on how to set up or use Gradescope, ask on Edstem! For this assignment, you should have written answers for Problems 1, 2, and 3.

Anonymous Grading

You need to be graded anonymously, so do not write your name anywhere on your handin.

Obligatory Note on Academic Integrity

Plagiarism — don't do it.

As outlined in the Brown Academic Code, attempting to pass off another's work as your own can result in failing the assignment, failing this course, or even dismissal or expulsion from Brown. More than that, you will be missing out on the goal of your education, which is the cultivation of your own mind, thoughts, and abilities. Please review this course's collaboration policy and if you have any questions, please contact a member of the course staff.