Between Chapters 1 and 2

Chapter 2 of the textbook starts discussing the statistical properties of learning algorithms. Unfortunately, the textbook material jumps in difficulty a LOT between chapters 1 and 2. The purpose of these notes is to help "fill the gap." In particular, the textbook introduces the idea of *generalization bounds* and *infinite hypothesis classes* in chapter 2. Infinite hypothesis classes are fairly abstract and can be a bit tricky to understand. This set of notes introduces generalization bounds for *finite hypothesis classes*, which are more concrete and easier to understand.

Problem 1. For each hypothesis class below, draw a picture of what a "typical" hypothesis looks like and write the number of elements M in the hypothesis class.

$$\mathcal{H}_{\text{binary}} = \left\{ \mathbf{x} \mapsto +1, \mathbf{x} \mapsto -1 \right\}$$

$$\mathcal{H}_{\text{axis}} = \left\{ \mathbf{x} \mapsto \operatorname{sign}(x_i) : i \in [d] \right\}$$

$$\mathcal{H}_{\text{axis2}} = \left\{ \mathbf{x} \mapsto \sigma \operatorname{sign}(x_i) : \sigma \in \{+1, -1\}, i \in [d] \right\}$$

$$\mathcal{H}_{\text{multiaxis2}} = \left\{ \mathbf{x} \mapsto \operatorname{sign}\left(\sum_{j=1}^{d} \sigma_j \operatorname{sign}(x_j) \right) : \sigma_i \in \{+1, -1\}, i \in [d] \right\}$$

$$\mathcal{H}_{\text{multiaxis3}} = \left\{ \mathbf{x} \mapsto \operatorname{sign}\left(\sum_{j=1}^{d} \sigma_j \operatorname{sign}(x_j) \right) : \sigma_i \in \{+1, 0, -1\}, i \in [d] \right\}$$

Problem 2. One simple idea for selecting a hypothesis in the hypothesis class is to select the best hypothesis on the training data. That is, select

$$g = \arg\min_{h \in \mathcal{H}} E_{\text{in}}(h). \tag{1}$$

The *Try Everything Algorithm* (TEA) is a simple algorithm for computing Formula 1 above. The pseudo-code is shown below:

```
g = None
g_score = -infinity
for h in H:
    h_score = E_in(h)
    if h_score > g_score:
        g = h
        g_score = h_score
```

What is the runtime of the TEA algorithm?

Problem 3. Why does it not make sense to run the TEA algorithm on the perceptron hypothesis class?

HINT: This problem is "trivial" in the sense that it follows immediately from definitions and the runtime bound above.

Theorem 1 (Finite Hypothesis Class Generalization)**.** Let $\mathcal{H}$ be a hypothesis class of size M, let g be an arbitrary hypothesis in $\mathcal{H}$ (in particular, g is allowed to be the result of the TEA algorithm), and let N be the size of the dataset. Then we have that for all $\epsilon > 0$,

$$\mathbb{P}[|E_{\text{in}}(g) - E_{\text{out}}(g)| \geq \epsilon] \leq 2M \exp(-2\epsilon^2 N).$$

This implies that with probability at least $1 - \delta$,

$$E_{\text{out}}(g) \leq E_{\text{in}}(g) + \sqrt{\frac{1}{2N} \log \frac{2M}{\delta}}.$$

Proof on pages 40-41 of the textbook. It is an immediate consequence of the Hoeffding inequality and the union bound. Observe the similarities/differences between the equations above and the Hoeffding inequality; this should help you memorize both results.

Problem 4. Graph the in-sample error, out-of-sample error, and the generalization error as a function of the hypothesis class size M.

This should be done informally/at an "intuition level"; the next two problems explore these trade-offs more formally. This is very similar to Figure 2.3 on page 59 of the textbook.

Problem 5. You have a dataset with 100 dimensions and 500 data points. Due to the small amount of training data, you have decided not to split the dataset into training and testing datasets, and instead you will use the entire dataset for training. Your particular application requires that your final model have a generalization error less than 0.1 with probability at least 0.99.

1. Which of the finite hypothesis classes are guaranteed to meet your generalization requirements according to the FHCG Theorem?

2. Of the hypothesis classes that you CAN select, which one SHOULD you select?

Problem 6. Given two finite hypothesis classes $\mathcal{H}_1$ and $\mathcal{H}_2$, we can construct a third hypothesis class

$$\mathcal{H}_3 = \mathcal{H}_1 \cup \mathcal{H}_2. \tag{2}$$

1. Let g_i be the result of the TEA algorithm on hypothesis class $\mathcal{H}_i$. How does the in-sample error of g_3 compare to the in-sample errors of g_1 and g_2?

2. How does the bound on the generalization error from the FHCG theorem for $\mathcal{H}_3$ compare to the generalization error of $\mathcal{H}_1$ and $\mathcal{H}_2$?

Problem 7. Recall that the Perceptron hypothesis class is defined as

$$\mathcal{H}_{\text{perceptron}} = \left\{ \mathbf{x} \mapsto \operatorname{sign}(\mathbf{w}^T \mathbf{x}) : \mathbf{w} \in \mathbb{R}^d \right\}. \tag{3}$$

In practice, computers cannot efficiently implement arbitrary precision real number arithmetic, and so all perceptron implementations use a finite precision approximation.[1] Let b be the total number of bits of precision used to represent a real number.

1. Write a generalization bound for the finite precision version of the perceptron in terms of b, d, and N.

2. Based on your bound above, if you double the bits of precision b, how much more data N will you need to achieve the same generalization error?

[1]Python (and most other programming languages) natively use 64bit IEEE754 floating point numbers. Modern CPUs have native support for 32 bit floating numbers as well, and they are often used in numeric computations because they are about twice as fast because there are half as many bits that need to be computed. Recently, NVIDIA has begun supporting 16bit, 8bit, and even 4bit floating point numbers. These "low precision reals" are most commonly used to speed up computations or reduce memory requirements, but this problem shows that these low precision reals can also result in statistical improvements in the generalization error. The use of low precision floating point numbers is often called *quantization*.

Problem 8. Dimensionality reduction is a common technique for improving the computational and statistical performance of an algorithm. One simple form of dimensionality reduction is to multiply all the datapoints by a random matrix. Formally, let $A : d' \times d$ be a random matrix where each entry is sampled from the uniform distribution on $[-1, 1]$. Then each new data point $\mathbf{x}'$ is the d' dimensional vector defined to be

$$\mathbf{x}' = A\mathbf{x}. \tag{4}$$

1. Complete the table below describing the effect of this transform on the runtime and generalization error of our finite hypothesis classes.

hypothesis class	runtime of TEA	generalization error
$\mathcal{H}_{\text{binary}}$		
$\mathcal{H}_{\text{axis}}$		
$\mathcal{H}_{\text{axis2}}$		
$\mathcal{H}_{\text{multiaxis2}}$		
$\mathcal{H}_{\text{multiaxis3}}$		

2. Above, you should see either a linear or exponential improvement in all cases. Why does it not make sense to always perform this dimensionality reduction with a very small d'? (That is, what important quantity not shown in the table above will get worse as d' gets smaller?)