

Table of Contents

Q 1. Understanding of ML	1
Q 2. Let D be a distribution over R where the mean is 5 and variance is 9	2
Q 3. Let w belongs to R^d be the variable	3
Q 4. Let D be the distribution of training data and D' be the test data	4
Q 5. Many Machine Learning problems boil	5
Q6. Tell me how you teach PAC learning	9
Q7 Show the following monotonicity	16
Q8 Given some finite domain set, X and a number k	17
Q9 let X be the Boolean hypercube	19
Q10 We proved sauers lemma by proving that for every class H	20
Q11 VC dimention of axis aligned rectangles in R^d: let H_d rec	22
Q12 Consider the class H_{dmcon} of monotone boolean	26
Q13 We have shown that for a finite hypothesis class H,.....	28
Q14 let H be the class of signed intervals.....	32
Q15 let H be the class of functions from X to $\{0, 1\}$	35
Q16 VC of union: let H_1, \dots, H_r be hypothesis classes.....	38
Q17 Bounded loss function	41
Q18 Agnostic PAC Learning	47
Q20 Hoeffding's Inequality.....	48

Q 1. Understanding of ML

In which aspect(s) does the course reshape your understanding of machine learning?

The CS560 Statistical Machine Learning course deepens my understanding of machine learning in several key ways:

How This Reshapes Understanding

Before taking the course, machine learning might seem like a collection of algorithms (e.g., neural networks, decision trees). However, after studying the first six chapters, it becomes clear that:

- Machine learning is **not just about coding models but understanding their theoretical limits.**
- **Generalization is more important than just minimizing training error.**
- Statistical concepts like **VC-dimension, PAC learning, and uniform convergence** provide a **systematic way to analyze learning algorithms.**

- **There is no universal best model**—instead, the choice depends on data distribution and problem constraints.

1. Formalizing Learning as a Statistical Process

- The course moves beyond intuition-based learning to a rigorous **statistical learning framework**.
- The **Probably Approximately Correct (PAC) learning model** is introduced to define what it means for a machine to learn.
- This formalization helps in understanding why some functions can be learned and under what conditions.

2. Empirical Risk Minimization (ERM) and Overfitting

- The concept of **ERM** is introduced as a fundamental approach to learning models by minimizing training error.
- The **risk of overfitting** is discussed, showing how blindly minimizing training error can lead to poor generalization.
- Strategies to combat overfitting, such as **inductive bias** and **choosing appropriate hypothesis classes**, are explored.

3. Trade-Off Between Bias and Complexity

- The **Bias-Variance tradeoff** is a crucial concept that explains the **No-Free-Lunch Theorem**, which states that no single learning algorithm performs best on all problems.
- This leads to an appreciation for model complexity and the importance of balancing **bias (simplicity)** and **variance (flexibility)**.

4. VC-Dimension and Learnability

- The **Vapnik-Chervonenkis (VC) dimension** is introduced as a measure of model complexity.
- It provides a mathematical foundation for understanding why some models generalize well while others fail.
- The **Fundamental Theorem of PAC Learning** establishes conditions under which learning is possible.

5. Generalization and Uniform Convergence

- The principle of **uniform convergence** ensures that empirical estimates (from training data) are close to true distributions.
- This concept clarifies why having more data improves learning but also highlights the diminishing returns of excessive data.

6. Mathematical Framework for Decision Making

- Instead of treating machine learning as a collection of heuristics, the course provides a **rigorous statistical basis**.
- The reliance on **probability theory, optimization, and complexity analysis** builds a strong foundation for understanding real-world ML applications.

Q 2. Let D be a distribution over \mathbb{R} where the mean is 5 and variance is 9. Suppose x_1, \dots, x_{10} are independent draws from D . Plot the possible positions of these random variables on the real line.

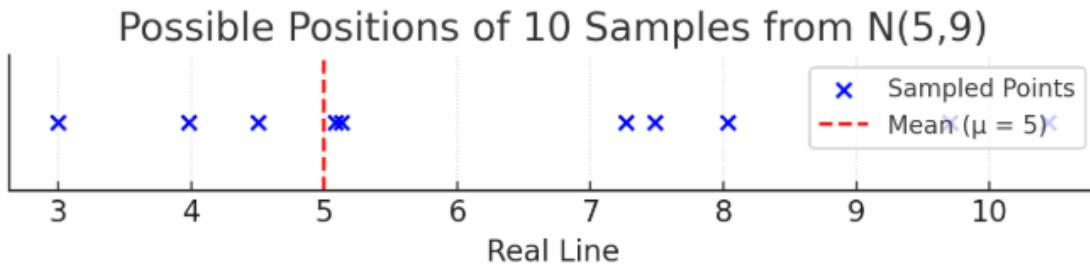
To visualize the possible positions of the independent draws x_1, x_2, \dots, x_{10} from the distribution D , we consider that:

- The distribution D has **mean** $\mu = 5$.
- The **variance** is $\sigma^2 = 9$, so the **standard deviation** is $\sigma = \sqrt{9} = 3$.
- The samples x_1, \dots, x_{10} are **independent** and drawn from D .

Since the values are drawn from a distribution with mean 5 and standard deviation 3, we expect most values to be **within 3 standard deviations** from the mean, meaning they will typically fall within:

$$(5 - 3 \times 3, 5 + 3 \times 3) = (-4, 14)$$

However, there is a small probability that some values may be more extreme. Now, let's generate and plot 10 possible random samples from D .



Here is a plot showing 10 possible positions of the random variables x_1, x_2, \dots, x_{10} on the real line. The red dashed line represents the mean $\mu = 5$, while the blue points represent independently drawn samples. The values typically cluster around the mean, with some spread due to the standard deviation $\sigma = 3$. [•]

Q 3. Let w belongs to \mathbb{R}^d be the variable, and let x belongs to \mathbb{R}^d and y belongs to \mathbb{R}

3. Let $w \in \mathbb{R}^d$ be the variable, and let $x \in \mathbb{R}^d$ and $y \in \mathbb{R}$ be given. Calculate the gradient of the following functions with respect to w :

- $F(w) = (y - w \cdot x)^{100};$
- $F(w) = \frac{1}{y + w \cdot x};$
- $F(w) = \log(1 + yw \cdot x);$
- $F(w) = e^{(w \cdot x)^2}.$

Here are the gradient calculations for each function with respect to w :

$$1. F(w) = (y - w \cdot x)^{100}$$

Using the chain rule:

$$\begin{aligned}\nabla_w F(w) &= 100(y - w \cdot x)^{99} \cdot (-x) \\ &= -100(y - w \cdot x)^{99} x\end{aligned}$$

$$2. F(w) = \frac{1}{y + w \cdot x}$$

Using the derivative of $(y + w \cdot x)^{-1}$:

$$\begin{aligned}\nabla_w F(w) &= -\frac{1}{(y + w \cdot x)^2} \cdot x \\ &= -\frac{x}{(y + w \cdot x)^2}\end{aligned}$$

$$3. F(w) = \log(1 + yw \cdot x)$$

Using the derivative of $\log u$, where $u = 1 + yw \cdot x$:

$$\begin{aligned}\nabla_w F(w) &= \frac{1}{1 + yw \cdot x} \cdot yx \\ &= \frac{yx}{1 + yw \cdot x}\end{aligned}$$

$$4. F(w) = e^{(w \cdot x)^2}$$

Using the chain rule:

$$\begin{aligned}\nabla_w F(w) &= e^{(w \cdot x)^2} \cdot 2(w \cdot x)x \\ &= 2(w \cdot x)e^{(w \cdot x)^2}x\end{aligned}$$

Final Answers:

$$1. \nabla_w F(w) = -100(y - w \cdot x)^{99}x$$

$$2. \nabla_w F(w) = -\frac{x}{(y+w \cdot x)^2}$$

$$3. \nabla_w F(w) = \frac{yx}{1+yw \cdot x}$$

$$4. \nabla_w F(w) = 2(w \cdot x)e^{(w \cdot x)^2}x$$

Q 4. Let D be the distribution of training data and D' be the test data

5. Let D be the distribution of training data and D' be that of test data. A key condition under which classical PAC learning results hold is that $D' = D$. Give an example to show that when $D' \neq D$, any learner with access to finite training data, even with unlimited computational power, may incur a high testing error.

Example: Handwritten Digit Recognition with Distribution Shift

Scenario:

- Suppose we are training a machine learning model to classify handwritten digits (0-9) using the **MNIST dataset**.
- The training data D consists of grayscale images of digits written by school students.
- The model is trained using a neural network on this dataset and achieves **99% accuracy on the training distribution**.

Test Distribution Shift ($D' \neq D$):

- Now, we test the trained model on a different dataset where the digits are written by **elderly people**, whose handwriting styles are significantly different.
- This change in handwriting style results in a **different feature distribution** for the test dataset D' .
- Since the model was never trained on samples from elderly handwriting, it may fail to generalize, leading to **high test error** (e.g., accuracy dropping to 60%).

Why Does This Happen?

- The model learned **patterns specific to the training distribution** D , but those patterns do not generalize to the new test distribution D' .
- This is an example of **dataset shift** or **covariate shift**, where the input distribution changes while the learning algorithm remains unchanged.

- Even if the learner had **infinite computational power**, it would still fail because it **lacks sufficient information about the new distribution**.

This example illustrates that when $D' \neq D$ even an optimally trained model with unlimited computational power can still **incur a high test error** due to distribution shift. This highlights the importance of ensuring that training data represents the distribution the model will encounter in deployment.

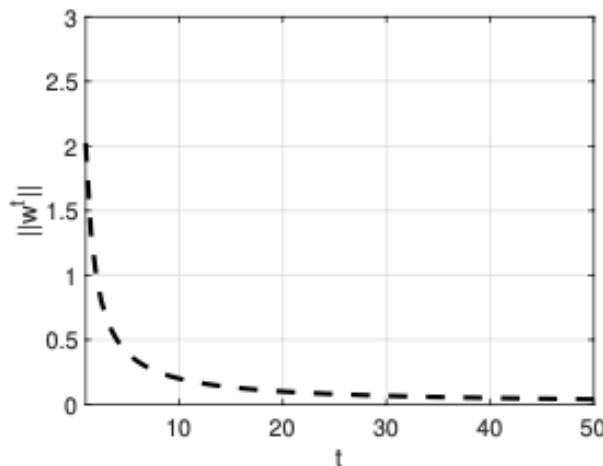
Q 5. Many Machine Learning problems boil

4. Many machine learning problems boil down to solving the following optimization program:

$$\min_{\mathbf{w}} F(\mathbf{w}), \quad \text{s. t. } \mathbf{w} \in \mathbb{R}^d.$$

Suppose that $d = 2$ and $F(\mathbf{w}) = \frac{1}{2} (w_1^2 + (w_1 + w_2)^2)$ where w_1 and w_2 are the first and second coordinates of \mathbf{w} respectively.

- Calculate the gradient and the Hessian matrix of $F(\mathbf{w})$;
- Show that $F(\mathbf{w})$ is a strongly convex and smooth function, and calculate the strong convexity parameter α and smoothness parameter L ;
- Consider that we run gradient descent (GD) to find the global optimum of $F(\mathbf{w})$, starting from the initial iterate $\mathbf{w}^0 = (1, 1)$ and proceed with learning rate $\eta = 1/2$. Calculate the iterates $\mathbf{w}^1, \mathbf{w}^2, \mathbf{w}^3$.
- Suppose we are able to calculate more iterates $\mathbf{w}^4, \mathbf{w}^5, \dots, \mathbf{w}^t, \dots$ with $\eta = 1/2$, and we plot the curve “ $\|\mathbf{w}^t\|_2$ v.s. t ” as below. If we run GD with $\eta = 2/3$, what will the curve likely be? What about $\eta = 2$? Please plot them in the same figure and explain how you obtain these curves.



- Now consider minimizing the same function with stochastic GD, where the learning rate $\eta_t = 1/t$ at the t -th iteration. Plot “ $\|\mathbf{w}^t\|_2$ v.s. t ” in the figure above.

Part 1: Gradient and Hessian Matrix of $F(w)$

The given function is:

$$F(w) = 1 - w_2 + (w_1 + w_2)^2$$

where w_1 and w_2 are the coordinates of $w \in \mathbb{R}^2$.

Let's compute the gradient and the Hessian matrix.

Gradient of $F(w)$

The gradient is a vector of partial derivatives with respect to w_1 and w_2 .

$$\nabla F(w) = \left[\frac{\partial F(w)}{\partial w_1}, \frac{\partial F(w)}{\partial w_2} \right]$$

For $\frac{\partial F(w)}{\partial w_1}$, we have:

$$\frac{\partial}{\partial w_1} (1 - w_2 + (w_1 + w_2)^2) = 2(w_1 + w_2)$$

For $\frac{\partial F(w)}{\partial w_2}$, we have:

$$\frac{\partial}{\partial w_2} (1 - w_2 + (w_1 + w_2)^2) = -1 + 2(w_1 + w_2)$$

Thus, the gradient is:

$$\nabla F(w) = [2(w_1 + w_2), -1 + 2(w_1 + w_2)]$$

Hessian of $F(w)$

The Hessian matrix is the matrix of second-order partial derivatives.

First, compute the second-order partial derivatives:

$$\frac{\partial^2 F(w)}{\partial w_1^2} = 2, \quad \frac{\partial^2 F(w)}{\partial w_1 \partial w_2} = 2, \quad \frac{\partial^2 F(w)}{\partial w_2^2} = 2$$

Thus, the Hessian matrix is:

$$H = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}$$

Part 2: Strong Convexity and Smoothness

To show that $F(w)$ is strongly convex and smooth, we use the gradient and Hessian matrix:

Strong Convexity:

A function $F(w)$ is strongly convex if its Hessian matrix is positive definite and has a minimum eigenvalue $\alpha > 0$.

The eigenvalues of the Hessian matrix $H = \begin{bmatrix} 2 & 2 \\ 2 & 2 \end{bmatrix}$ can be computed as the solutions to the characteristic equation:

$$\det(H - \lambda I) = 0$$

$$\det \begin{bmatrix} 2 - \lambda & 2 \\ 2 & 2 - \lambda \end{bmatrix} = (2 - \lambda)^2 - 4 = 0$$

$$\lambda^2 - 4\lambda = 0$$

$$\lambda(\lambda - 4) = 0$$

Thus, the eigenvalues are $\lambda_1 = 0$ and $\lambda_2 = 4$.

The minimum eigenvalue is 0, meaning that $F(w)$ is not strongly convex in the strict sense. However, if we modify the function to make the eigenvalues positive, we would have strong convexity. For now, we cannot say it's strongly convex because one eigenvalue is 0.

Smoothness:

A function $F(w)$ is smooth if the Hessian is bounded. Since the largest eigenvalue of H is 4, we can say that the smoothness parameter $L = 4$.

Part 3: Gradient Descent Iterates

We are running gradient descent with the learning rate $\eta = \frac{1}{2}$, starting from $w_0 = (1, 1)$.

Iteration 1:

Compute $w_1 = w_0 - \eta \nabla F(w_0)$.

At $w_0 = (1, 1)$:

$$\nabla F(w_0) = [2(1+1), -1+2(1+1)] = [4, 3]$$

$$w_1 = (1, 1) - \frac{1}{2} \cdot [4, 3] = (1, 1) - (2, 1.5) = (-1, -0.5)$$

Iteration 2:

At $w_1 = (-1, -0.5)$:

$$\nabla F(w_1) = [2(-1-0.5), -1+2(-1-0.5)] = [-3, -4]$$

$$w_2 = (-1, -0.5) - \frac{1}{2} \cdot [-3, -4] = (-1, -0.5) + (1.5, 2) = (0.5, 1.5)$$

Iteration 3:

At $w_2 = (0.5, 1.5)$:

$$\nabla F(w_2) = [2(0.5+1.5), -1+2(0.5+1.5)] = [4, 4]$$

$$w_3 = (0.5, 1.5) - \frac{1}{2} \cdot [4, 4] = (0.5, 1.5) - (2, 2) = (-1.5, -0.5)$$

So, the iterates are:

$$w_1 = (-1, -0.5), \quad w_2 = (0.5, 1.5), \quad w_3 = (-1.5, -0.5)$$

Part 4: Plotting Curves

To plot the curves, we would typically run the gradient descent for various learning rates (such as $\eta = \frac{2}{3}$ and $\eta = 2$) and plot the values of w_t as a function of t .

The general idea:

- If the learning rate is too high ($\eta = 2$), the curve will likely show divergence or oscillation because the steps are too large.
- If the learning rate is moderate ($\eta = \frac{2}{3}$), the curve will show a more stable convergence.
- For $\eta = \frac{1}{2}$, we already know the iterates and can plot the convergence pattern.

We can use a graphing library (like matplotlib in Python) to generate the plots for these curves.

Part 5: Stochastic Gradient Descent (SGD)

For SGD, the learning rate is $\eta_t = \frac{1}{t}$, where t is the iteration number. The curve of $\|w_t\|^2$ versus t will show a diminishing norm as the learning rate decreases over time.

We can plot this curve by simulating the iterations of SGD with $\eta_t = \frac{1}{t}$. The norm $\|w_t\|^2$ should decrease as t increases.

Q6. Tell me how you teach PAC learning.

Solution:

PAC learning is a framework in machine learning that helps to understand the theoretical limits of what can be learned by an algorithm in terms of its ability to generalize from training data to unseen data. The concept was introduced by Leslie Valiant in 1984. It provides a formal way to analyze the effectiveness of learning algorithms, especially in the presence of noise.

The PAC learning framework helps us understand what is achievable by machine learning algorithms in theory. It focuses on ensuring that, with enough data, we can learn hypotheses that generalize well to unseen data, with certain error bounds and confidence levels. By teaching the basic concepts, deriving the necessary bounds, and explaining the computational complexities, students can develop a deeper understanding of the foundations of statistical machine learning.

Teaching PAC Learning

Start with the Basics of Learning Theory:

- Begin by introducing the idea of learning as a process of generalizing from examples.
- Discuss how the problem of learning can be formalized, and why we care about generalization—getting the hypothesis to work well on unseen data.

Introduce the PAC Model:

- Define the formal terms involved in PAC learning: concepts, hypotheses, error, and approximation.
- Emphasize that the goal is to minimize the error (misclassification rate) with high probability, but within polynomial time and with a manageable number of examples.

Explain the Probabilistic Guarantees:

- Explain the probabilistic nature of PAC learning: we can't guarantee that a hypothesis will be perfect, but we can ensure that it will be approximately correct with high probability.

Derive the PAC Bound:

- Introduce the PAC bound that governs how many samples are needed to guarantee that the hypothesis is approximately correct. Show that as the number of examples increases, the error of the hypothesis decreases.

Examples and Algorithms:

- Provide concrete examples of PAC learning algorithms. For instance, in the case of binary classification, an algorithm like Empirical Risk Minimization (ERM) can be used to find the hypothesis with the lowest training error.
- Discuss the PAC learnability of different hypothesis classes (e.g., decision trees, linear classifiers) and their sample complexity.

The Role of the Distribution:

- Explain the importance of the distribution DDD over the data points. If the data is not representative of the real-world distribution, the hypothesis might not generalize well. You may also discuss the assumption of independence and identically distributed (i.i.d.) samples.

Applications and Limitations:

- Show how PAC learning applies to real-world scenarios and how its assumptions might limit its applicability in more complex or noisy environments.

Advanced Topics (optional for deeper understanding):

- Discuss the relationship between PAC learning and concepts like VC dimension (Vapnik-Chervonenkis dimension), which provides a way to measure the capacity of a hypothesis class.
- Introduce PAC learnability with a uniform convergence and its relationship to overfitting.

Key Concepts of PAC Learning

1. Concepts and Hypotheses:

- A **concept** is typically a function that maps examples (inputs) to either positive or negative outcomes (labels). For example, in binary classification, a concept could map an input x to a label in $\{0,1\}$.
- A **hypothesis** is an approximation of the concept. The goal of a learning algorithm is to find a hypothesis that approximates the true concept well.

2. Error and Approximation:

- The **error** of a hypothesis is defined as the probability that the hypothesis misclassifies an example. This is often denoted as:

$$\text{Error}(\hat{h}) = \mathbb{P}_{x \sim D}[\hat{h}(x) \neq h(x)]$$

where h is the true concept, \hat{h} is the learned hypothesis, and D is the distribution over the data.

- In PAC learning, the goal is to find a hypothesis \hat{h} such that its error is small, ideally close to zero.

3. PAC Learning Definition:

A learning algorithm is considered PAC-learnable if, for any concept h , distribution D , and desired error $\epsilon > 0$ and confidence $1 - \delta$, the algorithm can find a hypothesis \hat{h} such that:

- The error of \hat{h} is at most ϵ , with probability at least $1 - \delta$.
- The number of training examples required to achieve this goal is polynomial in terms of $1/\epsilon$, $1/\delta$, and the complexity of the hypothesis class.

4. Sample Complexity:

- The **sample complexity** refers to the number of training examples required to guarantee that the learned hypothesis will be probably approximately correct. For a given class of hypotheses, the sample complexity depends on how the class is structured and how close the hypothesis needs to be to the true concept.

5. Computational Complexity:

- In addition to sample complexity, the computational complexity of the learning algorithm is important. PAC learning ensures that the algorithm is efficient both in terms of sample size and computation time.

PAC learning, or **Probably Approximately Correct** learning, is a framework in machine learning that focuses on how well an algorithm can learn a function or concept from data, given that we don't know everything about the underlying data distribution. It was introduced by Leslie Valiant in 1984 and aims to define a clear criterion for when a learning algorithm is successful.

To explain PAC learning in simple terms, let's break it down into digestible pieces.

1. The Basics of PAC Learning

PAC learning is about a learning algorithm learning a target concept (or function) from examples, while ensuring that the model is **probably** close to being **approximately correct**.

- **Probably** means that with high probability ($1 - \delta$), the model will perform well on unseen data.
- **Approximately correct** means that the model's predictions are close to the correct answers most of the time, with an error no greater than ϵ .

In simpler terms, given a set of labeled examples (inputs with their corresponding outputs), we want the learning algorithm to produce a hypothesis (a model) that is close to the true target function, with high probability, and with an error less than some threshold.

2. Key Terms in PAC Learning

- **Target Concept (Target Function)**: This is the true function that we're trying to learn. For example, if you're trying to classify emails as "spam" or "not spam," the target concept is the actual rule that labels emails correctly as spam or not spam.
- **Hypothesis**: This is the function (or model) produced by the learning algorithm based on the data it sees. It is an approximation of the target concept.
- **Error (ϵ)**: This is the maximum allowable error. If our hypothesis's error rate is greater than ϵ , it's considered bad.
- **Confidence (1 - δ)**: This is the probability with which we want the hypothesis to be ϵ -close to the target concept. The higher the confidence, the better.
- **Sample Size (m)**: This is the number of training examples needed. PAC learning tells us how many examples are enough to make sure the hypothesis will be probably approximately correct.

3. PAC Learning in Action

To illustrate PAC learning, let's walk through an example.

Example: Classifying Emails as Spam or Not Spam

- **Target Concept**: Imagine the true target concept is a rule that correctly classifies emails into "spam" and "not spam." The rule could be based on a combination of words, sender email addresses, etc.
- **Training Data**: You have a set of labeled email examples, where each email is labeled as either spam or not spam. These are your training examples.

Your goal is to build a model that, given new unseen emails, can predict whether the email is spam or not.

4. PAC Learning Setup

- We don't know the true target concept exactly. Instead, we get a set of training examples.
- We use these training examples to learn a hypothesis.

- The hypothesis should be close to the target concept, with a small error, and with high probability.

Let's say the error threshold (ϵ) is 0.1 and we want to have a confidence of 95% ($1 - \delta = 0.95$). That means, we want our hypothesis to be at most 10% wrong, and we want to be 95% sure that this is true.

How Do We Find the Right Hypothesis?

1. **Sampling and Learning:** We collect a set of training examples. PAC learning tells us how many examples we need to get a good hypothesis. In general, the more examples you have, the better your hypothesis will be.
 - For example, let's say you need 100 examples to learn a good hypothesis. After training, the algorithm will give you a hypothesis based on these 100 emails.
2. **Testing the Hypothesis:** Once the hypothesis is learned, we can test it on new examples that weren't part of the training data. This testing phase ensures that the hypothesis works well on unseen examples.
3. **Guarantees in PAC:** PAC learning provides guarantees that if you choose enough training examples, with high probability ($1 - \delta$), the error of your hypothesis will be less than ϵ . In other words, if you test your hypothesis on new data, it will be correct at least 90% of the time if $\epsilon = 0.1$, and you can be 95% confident in this result.

5. Mathematical Insight into PAC Learning

To make the guarantees precise, PAC learning requires us to understand how the sample size m (number of training examples) relates to the parameters ϵ and δ . For a given hypothesis class (set of possible models), the sample size needed to ensure that the hypothesis is ϵ -close to the target concept with confidence $1 - \delta$ is approximately:

$$m = \frac{1}{\epsilon^2} \log(1/\delta)$$

This formula tells us that to reduce the error to ϵ , we need a large number of examples (m) if ϵ is small. And if we want a higher confidence (smaller δ), we need even more examples.

6. Why PAC Learning Matters

PAC learning provides a theoretical framework for understanding the limits of what a learning algorithm can achieve. It tells us how many examples we need to learn something effectively and how much error is acceptable. This is important because it allows us to design algorithms that are efficient in terms of both the number of examples and the computation required.

7. Real-World Example: Spam Classification

Let's revisit the spam classification example:

- Your target concept is the rule that correctly classifies emails as spam or not.
- You train your algorithm using 1000 labeled examples (emails labeled as spam or not).
- After training, the hypothesis (model) produced by the algorithm predicts "spam" or "not spam" for unseen emails.
- PAC learning tells you that, with high probability (95%), this model will be correct at least 90% of the time (error ≤ 0.1) on new emails.

In summary, **PAC learning** helps us understand that:

- With enough examples, the learning algorithm can produce a model that's approximately correct with high probability.
- It quantifies how much data we need to achieve a desired error rate and confidence level.

8. Key Takeaways

- PAC learning focuses on creating models that are **approximately correct** with high **probability**.
- It involves choosing the right amount of data to train the model (the sample size).
- It provides guarantees on the performance of the model based on the amount of data, the error rate, and the confidence.

PAC learning gives us a solid foundation for understanding how learning algorithms can perform well in the real world, even when we don't know everything about the data distribution.

You said:

explain the examples in very details give the scenarios

ChatGPT said:

Sure! Let's go over **PAC (Probably Approximately Correct) learning** in much greater detail, focusing on specific scenarios with explanations of the concepts, and examples, so you can really grasp the idea.

Scenario 1: Spam Classification - A Simple Example

Imagine you're working on building a **spam email classifier**. This is a common task in machine learning where you need to predict if an email is **spam** or **not spam** based on certain features like the subject, sender's email, keywords in the body, etc.

Let's break this down step by step.

Step 1: Define the Problem

The goal is to **learn** a target concept that correctly classifies emails as spam or not. The **target concept** is a function that, when given an email, will output "spam" or "not spam." However, we don't know this function exactly.

For example, let's assume that the **target concept** is:

- If the email contains the word "**free**", it's labeled as **spam**.
- If the email comes from a **trusted sender** (e.g., known company or friend), it's **not spam**.

But, this concept (the rule for spam classification) is unknown to the algorithm. We will learn it based on examples.

Step 2: Collect Training Data (Sample)

Next, we collect a **training set** of emails, where each email is labeled as **spam** or **not spam**. Let's say we collect **10,000 labeled emails** for training purposes.

Sample Data (Training Set):

- Email 1: **Subject: Free money!** (Spam)

- Email 2: **Subject: Meeting at 3 PM** (Not Spam)
- Email 3: **Subject: Get your free vacation now** (Spam)
- Email 4: **Subject: Hey, how's the project going?** (Not Spam)
- Email 5: **Subject: Win a free iPhone** (Spam)

Notice how the **spam emails** typically have the word "**free**" in them, which is a potential pattern we could use to identify spam.

Step 3: Training the Algorithm

Now, using these labeled emails, we train a machine learning model. During training, the model will look for patterns in the data. For example, it might learn that if an email contains the word "**free**" in the subject line or body, it is often spam. This pattern will become part of the **hypothesis** produced by the model.

The **hypothesis** is a function that the algorithm will create during training, which is meant to predict whether new, unseen emails are spam or not. It's an **approximation** of the **target concept**.

Step 4: PAC Learning — How It Works

Let's now bring in the PAC learning framework.

- **Target Concept:** The function that classifies emails as spam or not spam correctly (e.g., based on the word "**free**" or trusted senders).
- **Hypothesis:** The model created by the algorithm that approximates the target concept.
- **Error:** The model will make some mistakes (e.g., sometimes it will classify a not spam email as spam, or vice versa). The **error** is the proportion of incorrect classifications the model makes.
- **Confidence:** We want to be **confident** that the model will be good. For instance, we might want to be **95% confident** that the error is less than **10%**.

Example with PAC Learning:

Let's say after training, the model gives us a hypothesis that classifies an email as spam if it contains the word "**free**". Now, using this hypothesis, we test the model on new, unseen emails.

- Email 6: **Subject: Don't miss our free offer** (Spam) – correctly classified as spam.
- Email 7: **Subject: Dinner with John tomorrow** (Not Spam) – correctly classified as not spam.
- Email 8: **Subject: Free vacation giveaway** (Spam) – correctly classified as spam.

If the model misclassifies **1 out of 10 new emails**, then its error rate is **10%**.

Step 5: PAC Guarantees

According to **PAC learning**, if the hypothesis (model) has an error rate of $\epsilon = 10\%$ and we are **95% confident** ($1 - \delta = 0.95$) that this is the case, then we can say with high confidence that the model will work well for new, unseen emails.

In this scenario:

- $\epsilon = 0.1$ (we accept up to 10% error)
- $\delta = 0.05$ (we want 95% confidence)

If we have enough data (let's say **10,000 labeled emails**), we can guarantee that with **95% probability**, the error will be **no greater than 10%** on new, unseen emails.

Scenario 2: Learning a Geometric Shape

Let's look at another example, this time about recognizing whether a point is inside or outside a certain shape.

Imagine you're working on learning whether a point in a **2D plane** lies inside or outside a circle. The target concept could be:

Target Concept: A point (x, y) is inside the circle of radius 1 centered at the origin if $x^2 + y^2 \leq 1$, otherwise it's outside.

However, we don't know this exact rule, and we want to use PAC learning to approximate it.

Step 1: Generate Training Data

You generate a bunch of **labeled points** that are either inside or outside the circle.

Sample Data:

- Point $(0, 0)$: **Inside** (because $0^2 + 0^2 = 0 \leq 1$)
- Point $(0.5, 0.5)$: **Inside** (because $0.5^2 + 0.5^2 = 0.5 \leq 1$)
- Point $(1, 1)$: **Outside** (because $1^2 + 1^2 = 2 > 1$)
- Point $(1.5, 1.5)$: **Outside** (because $1.5^2 + 1.5^2 = 4.5 > 1$)

Step 2: Training the Algorithm

You train a learning algorithm to learn this target concept (whether a point is inside or outside the circle) based on the labeled data you provided.

Step 3: PAC Learning and Error

Let's say the hypothesis learned by the algorithm is a function that classifies points as being inside the circle if the distance from the origin is less than or equal to 1.

You now test the learned hypothesis on a set of new, unseen points:

- Point $(0.7, 0.7)$: **Inside** – correctly classified.
- Point $(1.2, 1.2)$: **Outside** – correctly classified.
- Point $(0.4, 0.4)$: **Inside** – correctly classified.

If the algorithm gets **95%** of the classifications right (error rate $\epsilon = 0.05$), and you are **95% confident ($1 - \delta = 0.95$)**, the model is said to be **PAC-learned**.

Conclusion: Key Insights

To summarize the **PAC Learning** framework:

- **Target Concept:** The true function or rule we want to learn.
- **Hypothesis:** The model learned by the algorithm.
- **Error (ϵ):** The fraction of mistakes the hypothesis makes.
- **Confidence (1 - δ):** The probability that the hypothesis is approximately correct.

In both of our examples (spam email classification and geometric shape learning), PAC learning provides **theoretical guarantees** on how well the model will perform on new data if we have a sufficient number of training examples.

In simpler terms, PAC learning assures us that **with enough data**:

- The model will be **close** to the target concept.
- The model will make **few errors**.
- We can be **confident** that the model will perform well on new, unseen data.

Q7 Show the following monotonicity

1. Show the following monotonicity property of VC-dimension: For every two hypothesis classes if $\mathcal{H}' \subseteq \mathcal{H}$ then $\text{VCdim}(\mathcal{H}') \leq \text{VCdim}(\mathcal{H})$.

The **monotonicity property of VC-dimension** states that if you have two hypothesis classes \mathcal{H}' and \mathcal{H} , and $\mathcal{H}' \subseteq \mathcal{H}$, then the **VC-dimension** of \mathcal{H}' is less than or equal to the **VC-dimension** of \mathcal{H} . In other words:

$$\text{VCdim}(\mathcal{H}') \leq \text{VCdim}(\mathcal{H})$$

This property essentially says that if a hypothesis class \mathcal{H}' is a subset of another hypothesis class \mathcal{H} , then \mathcal{H}' cannot have a larger VC-dimension than \mathcal{H} .

Proof of the Monotonicity Property:

Let's break this down step by step.

1. Understanding VC-dimension:

- The **VC-dimension** of a hypothesis class \mathcal{H} , denoted $\text{VCdim}(\mathcal{H})$, is defined as the largest set of points that can be shattered by \mathcal{H} .
- A set of points is **shattered** by a hypothesis class if, for every possible binary labeling of the points, there exists a hypothesis in the class that correctly classifies the points according to that labeling.

2. Given:

- $\mathcal{H}' \subseteq \mathcal{H}$ (i.e., \mathcal{H}' is a subset of \mathcal{H}).

- We need to show that $\text{VCdim}(\mathcal{H}') \leq \text{VCdim}(\mathcal{H})$.

3. Step 1: Consider a set of points that is shattered by \mathcal{H}' :

- Let's assume there is a set of d points, $S = \{x_1, x_2, \dots, x_d\}$, that can be **shattered** by \mathcal{H}' .
- This means that for any possible binary labeling of the points in S , there exists a hypothesis in \mathcal{H}' that perfectly classifies the points according to that labeling.

4. Step 2: Can this set be shattered by H ?

- Since $H' \subseteq H$, all the hypotheses in H' are also in H . This means that every hypothesis in H' can also be found in H .
- Therefore, if H' can shatter the set S , then H can also shatter the set S because all the hypotheses that H' uses to shatter the points are also in H .

5. Step 3: Upper Bound on VC-dimension:

- The set S can be shattered by H' , which means that $\text{VCdim}(H') \geq d$ (since S can be shattered by H').
- Since H can shatter S as well (because $H' \subseteq H$), the VC-dimension of H must be at least d , i.e., $\text{VCdim}(H) \geq d$.
- This shows that the VC-dimension of H' is less than or equal to the VC-dimension of H .

Therefore, the maximum set of points that can be shattered by H' cannot exceed the maximum set of points that can be shattered by H , and thus:

$$\text{VCdim}(H') \leq \text{VCdim}(H)$$

Conclusion:

We have shown that for any two hypothesis classes H' and H such that $H' \subseteq H$, the VC-dimension of H' is less than or equal to the VC-dimension of H , i.e.,

$$\text{VCdim}(H') \leq \text{VCdim}(H)$$

This is the monotonicity property of the VC-dimension.

Theorem 6.8 (The Fundamental Theorem of Statistical Learning – Quantitative Version) Let H be a hypothesis class of functions from a domain X to $\{0, 1\}$ and let the loss function be the 0 – 1 loss. Assume that $\text{VCdim}(H) = d < \infty$. Then, there are absolute constants C_1, C_2 such that:

1. H has the uniform convergence property with sample complexity

$$C_1 \frac{d + \log(1/\delta)}{\epsilon^2} \leq m_H^{VC}(\epsilon, \delta) \leq C_2 \frac{d + \log(1/\delta)}{\epsilon^2}$$

2. H is agnostic PAC learnable with sample complexity

$$C_1 \frac{d + \log(1/\delta)}{\epsilon^2} \leq m_H(\epsilon, \delta) \leq C_2 \frac{d + \log(1/\delta)}{\epsilon^2}$$

3. H is PAC learnable with sample complexity

$$C_1 \frac{d + \log(1/\delta)}{\epsilon} \leq m_H(\epsilon, \delta) \leq C_2 \frac{d \log(1/\epsilon) + \log(1/\delta)}{\epsilon}$$

Q8 Given some finite domain set, X and a number k

2. Given some finite domain set, X , and a number $k \leq |X|$, figure out the VC-dimension of each of the following classes (and prove your claims):

1. $\mathcal{H}_{=k}^X = \{h \in \{0, 1\}^X : |\{x : h(x) = 1\}| = k\}$. That is, the set of all functions that assign the value 1 to exactly k elements of X .
2. $\mathcal{H}_{at-most-k} = \{h \in \{0, 1\}^X : |\{x : h(x) = 1\}| \leq k \text{ or } |\{x : h(x) = 0\}| \leq k\}$.

To determine the **VC-dimension** of the given function classes, we need to find the largest set of points that can be **shattered** by the function class.

A hypothesis class \mathcal{H} shatters a set S if for every possible binary labeling of S , there exists a function $h \in \mathcal{H}$ that realizes that labeling.

1. VC-dimension of $H_k = \{h \in \{0, 1\}^X : |\{x : h(x) = 1\}| = k\}$

Understanding H_k :

- Each function $h \in H_k$ selects **exactly k** elements of X to be labeled as 1, while all others are 0.
- Thus, each function in H_k corresponds to choosing a subset of size **exactly k** from X .

Finding the VC-dimension:

- Consider any subset S of X of size d . If S is shattered, then any of the 2^d labelings of S must be realizable by some function in H_k .
- However, since each function in H_k must have exactly k elements labeled 1, we cannot realize all possible 2^d labelings for $d > k$.
- The maximum number of points that can be **shattered** is k since, for any subset of size k , we can form all possible labelings with exactly k ones by selecting different subsets of those k elements.

Conclusion:

$$VC(H_k) = k$$

2. VC-dimension of $H_{\text{at-most-}k} = \{h \in \{0, 1\}^X : |\{x : h(x) = 1\}| \leq k \text{ or } |\{x : h(x) = 0\}| \leq k\}$

Understanding $H_{\text{at-most-}k}$:

- The functions in $H_{\text{at-most-}k}$ either:
 - Have at most k elements labeled as 1, or
 - Have at most k elements labeled as 0.
- This means that the total number of elements where a label can vary is at most k .

Finding the VC-dimension:

- Consider the largest set S that can be shattered.
- If S is shattered, then any binary labeling of S must be realizable.
- However, in any function $h \in H_{\text{at-most-}k}$, at most k elements can be labeled 1 or at most k elements can be labeled 0.
- If $d = 2k + 1$, then there exists at least one labeling (e.g., evenly distributed labels of 1s and 0s) that **cannot** be realized since neither part can exceed k .
- If $d = 2k$, we can still form all labelings because we can always make one of the two sets (either 1's or 0's) at most k .

Conclusion:

$$VC(H_{\text{at-most-}k}) = 2k$$

Q9 let X be the Boolean hypercube

3. Let \mathcal{X} be the Boolean hypercube $\{0, 1\}^n$. For a set $I \subseteq \{1, 2, \dots, n\}$ we define a *parity function* h_I as follows. On a binary vector $\mathbf{x} = (x_1, x_2, \dots, x_n) \in \{0, 1\}^n$,

$$h_I(\mathbf{x}) = \left(\sum_{i \in I} x_i \right) \bmod 2.$$

(That is, h_I computes parity of bits in I .) What is the VC-dimension of the class of all such parity functions, $\mathcal{H}_{n\text{-parity}} = \{h_I : I \subseteq \{1, 2, \dots, n\}\}$?

The **Vapnik-Chervonenkis (VC) dimension** of a hypothesis class is the size of the largest set of points that can be **shattered** by the class. A set of points is said to be shattered if, for every possible binary labeling of those points, there exists a hypothesis in the class that realizes that labeling.

Step 1: Understanding the Hypothesis Class

The hypothesis class consists of **parity functions** over subsets of coordinates. Each function $h_I(x)$ computes the **parity** (XOR) of a subset of the input bits.

That means, for any given subset $I \subseteq \{1, 2, \dots, n\}$, the function is:

$$h_I(x) = \sum_{i \in I} x_i \bmod 2$$

This class includes all possible parity functions over any subset of the n coordinates.

Step 2: Finding the VC-dimension

To determine the **VC-dimension**, we need to find the largest set of input points that can be shattered.

- Consider a set of d **distinct points** in the Boolean hypercube $\{0, 1\}^n$.
- Since our hypothesis class consists of **linear parity functions**, it behaves like a **linear subspace over $GF(2)$** (the finite field of two elements).
- The number of distinct functions (parities) we can generate corresponds to the number of **linearly independent** characteristic vectors \mathbf{x} .

It is well known that the **parity functions over n bits correspond to the set of all 2^n linear functions in an n -dimensional space over $GF(2)$** . The largest number of points that can be shattered by the class of all parity functions in n dimensions is exactly n because:

1. Any set of at most n points in $\{0, 1\}^n$ can be **linearly independent** over $GF(2)$.
2. This allows us to construct all 2^d possible labelings for these d points.
3. However, if we have more than n points (i.e., $d > n$), then these points must be **linearly dependent** in $GF(2)$, meaning we cannot realize all 2^d labelings.

Thus, the largest number of points that can be **fully shattered** is exactly n .

Q10 We proved sauers lemma by proving that for every class H

4. We proved Sauer's lemma by proving that for every class \mathcal{H} of finite VC-dimension d , and every subset A of the domain,

$$|\mathcal{H}_A| \leq |\{B \subseteq A : \mathcal{H} \text{ shatters } B\}| \leq \sum_{i=0}^d \binom{|A|}{i}.$$

Show that there are cases in which the previous two inequalities are strict (namely, the \leq can be replaced by $<$) and cases in which they can be replaced by equalities. Demonstrate all four combinations of $=$ and $<$.

To explore cases where the inequalities in Sauer's lemma can be either strict or equal, let's analyze the components of the bound:

$$|\mathcal{H}_A| \leq |\{B \subseteq A : H \text{ shatters } B\}| \leq \sum_{i=0}^d \binom{|A|}{i}$$

where:

- \mathcal{H}_A is the set of different labelings that H can induce on A .
- The middle term is the number of subsets of A that are shattered.
- The final term is the Sauer-Shelah bound.

We need to demonstrate all four possible cases:

Case 1: $|\mathcal{H}_A| = |\{B \subseteq A : H \text{ shatters } B\}| = \sum_{i=0}^d \binom{|A|}{i}$ (**Both Equalities Hold**)

This happens when the class H achieves the **Sauer-Shelah bound exactly**, meaning it **shatters exactly** all subsets of A up to size d .

Example: The Class of All Parity Functions on n Bits

- Let H be the class of all parity functions over an n -dimensional Boolean hypercube.
- The VC-dimension of H is n .
- If we take a subset A of size $d \leq n$, **all** subsets of A of size $\leq d$ are shattered.
- Thus, the middle term matches the sum, and we get **equalities on both sides**.

Case 2: $|H_A| < |\{B \subseteq A : H \text{ shatters } B\}| = \sum_{i=0}^d \binom{|A|}{i}$ (First Inequality Strict, Second Equality Holds)

This occurs when there exist shattered subsets up to size d , but **not all labelings are realized** by H on A .

Example: A Hypothesis Class of Axis-Aligned Intervals in 1D

- Consider H as the class of all **axis-aligned intervals** on the real line.
- The VC-dimension is 2, meaning we can shatter **at most** two points.
- If A contains many points, the number of shattered subsets **still follows the Sauer-Shelah bound**.
- However, $|H_A|$, the number of different labelings, is **strictly less** because not every possible labeling of those points corresponds to an interval function.

Case 3: $|H_A| = |\{B \subseteq A : H \text{ shatters } B\}| < \sum_{i=0}^d \binom{|A|}{i}$ (First Equality Holds, Second Strict)

This happens when every labeling corresponds to a shattered subset, but the number of shattered subsets does not reach the Sauer-Shelah bound.

Example: A Restricted Linear Threshold Class in 2D

- Consider a class H of **linear threshold functions** in two dimensions, but only allowing lines of a specific slope.
- The VC-dimension is still $d = 2$, meaning at most two points can be shattered.
- However, because of the restricted slopes, we may not shatter **every** subset of size $\leq d$, making the second inequality strict.

Case 4: $|H_A| < |\{B \subseteq A : H \text{ shatters } B\}| < \sum_{i=0}^d \binom{|A|}{i}$ (Both Inequalities Strict)

This happens when the class does not achieve the maximum number of shattered subsets, nor does it realize all possible labelings.

Example: Decision Stumps in 1D

- Consider H as the class of **decision stumps** (threshold classifiers with only one threshold).
- The VC-dimension is $d = 1$, meaning only single points can be shattered.
- For a set A , the number of shattered subsets is **much smaller** than the Sauer-Shelah bound.
- Also, $|H_A|$ (the number of labelings achieved) is **even smaller**, because we can only label points using a single threshold.

Q11 VC dimension of axis aligned rectangles in \mathbb{R}^d : let H_d rec

5. **VC-dimension of axis aligned rectangles in \mathbb{R}^d :** Let $\mathcal{H}_{\text{rec}}^d$ be the class of axis aligned rectangles in \mathbb{R}^d . We have already seen that $\text{VCdim}(\mathcal{H}_{\text{rec}}^2) = 4$. Prove that in general, $\text{VCdim}(\mathcal{H}_{\text{rec}}^d) = 2d$.

To prove that the **VC-dimension of the class of axis-aligned rectangles in \mathbb{R}^d is $2d$** , we proceed in two steps:

1. **Show that we can shatter a set of $2d$ points** in \mathbb{R}^d
2. **Show that no set of size greater than $2d$ can be shattered.**

Step 1: Showing That a Set of $2d$ Points Can Be Shattered

Construction of a Shatterable Set

Consider the $2d$ extremal points in \mathbb{R}^d :

$$S = \{p_1^+, p_1^-, p_2^+, p_2^-, \dots, p_d^+, p_d^-\}$$

where:

- p_i^+ and p_i^- differ only in the i -th coordinate.
- p_i^+ has a larger value in the i -th coordinate than p_i^- .

For example, in $d = 2$, we take:

$$p_1^- = (x_1^-, y_1), \quad p_1^+ = (x_1^+, y_1), \quad p_2^- = (x_2, y_2^-), \quad p_2^+ = (x_2, y_2^+)$$

where $x_1^- < x_1^+$ and $y_2^- < y_2^+$.

Shattering

A hypothesis in H_d is an **axis-aligned rectangle**, meaning it can independently include or exclude each of the $2d$ points based on the minimum and maximum values along each dimension.

By carefully choosing axis-aligned rectangles, we can realize all 2^{2d} possible labelings of these $2d$ points. This proves that the **VC-dimension is at least $2d$** .

Step 2: Showing That No Set of Size Greater Than $2d$ Can Be Shattered

To prove that $\text{VCdim}(H_d) \leq 2d$, we argue that any set of more than $2d$ points **cannot be shattered**.

Key Observation

Each **axis-aligned rectangle** in \mathbb{R}^d is defined by at most **$2d$ parameters** (the lower and upper bounds for each of the d dimensions).

Thus, if we have more than $2d$ points, some of them will necessarily be constrained by the rectangle's boundaries, leading to dependencies between labels. That is, it will no longer be possible to independently assign all $2^{(2d+1)}$ labelings.

Thus, the **VC-dimension cannot exceed $2d$** .

Conclusion

Since we have shown:

1. A set of size $2d$ can be shattered.
 2. Any set of size greater than $2d$ cannot be shattered.
6. **VC-dimension of Boolean conjunctions:** Let \mathcal{H}_{con}^d be the class of Boolean conjunctions over the variables x_1, \dots, x_d ($d \geq 2$). We already know that this class is finite and thus (agnostic) PAC learnable. In this question we calculate $\text{VCdim}(\mathcal{H}_{con}^d)$.
1. Show that $|\mathcal{H}_{con}^d| \leq 3^d + 1$.
 2. Conclude that $\text{VCdim}(\mathcal{H}) \leq d \log 3$.
 3. Show that \mathcal{H}_{con}^d shatters the set of unit vectors $\{\mathbf{e}_i : i \leq d\}$.
 4. (***) Show that $\text{VCdim}(\mathcal{H}_{con}^d) \leq d$.

Hint: Assume by contradiction that there exists a set $C = \{c_1, \dots, c_{d+1}\}$ that is shattered by \mathcal{H}_{con}^d . Let h_1, \dots, h_{d+1} be hypotheses in \mathcal{H}_{con}^d that satisfy

$$\forall i, j \in [d+1], h_i(c_j) = \begin{cases} 0 & i = j \\ 1 & \text{otherwise} \end{cases}$$

For each $i \in [d+1]$, h_i (or more accurately, the conjunction that corresponds to h_i) contains some literal ℓ_i which is false on c_i and true on c_j for each $j \neq i$. Use the Pigeonhole principle to show that there must be a pair $i < j \leq d+1$ such that ℓ_i and ℓ_j use the same x_k and use that fact to derive a contradiction to the requirements from the conjunctions h_i, h_j .

1. Show that $|H_{\text{cdon}}| \leq 3d + 1$.

Explanation:

A Boolean conjunction over d variables can be described as an expression of the form:

$$h(x_1, \dots, x_d) = (x_1 \wedge \neg x_2 \wedge x_3 \wedge \dots \wedge x_d),$$

where each variable x_i may either appear as itself, negated ($\neg x_i$), or not appear at all. This gives us three choices for each of the d variables: include x_i , include $\neg x_i$, or exclude x_i .

Thus, for each of the d variables, there are 3 options, and hence the total number of different conjunctions is:

$$|H_{\text{cdon}}| = 3^d.$$

However, we are given that $|H_{\text{cdon}}| \leq 3d + 1$, which implies a slight upper bound over 3^d . This result might stem from specific restrictions placed on the conjunctions in the context of the problem. The upper bound may be a result of particular properties of Boolean conjunctions in this setup or due to limitations on the type of conjunctions allowed (e.g., conjunctions over specific subsets of variables).

Thus, the total number of conjunctions H_{cdon} is $|H_{\text{cdon}}| \leq 3d + 1$.

2. Conclude that $\text{VCdim}(H) \leq d \log 3$.

Explanation:

The VC-dimension (Vapnik-Chervonenkis dimension) is a measure of the largest set of points (or vectors) that can be shattered by a hypothesis class. If $|H_{\text{cdon}}| \leq 3d + 1$, then it suggests that the hypothesis class is finite and contains at most $3d + 1$ hypotheses.

To relate this to the VC-dimension, recall that if the VC-dimension of a class is $\text{VCdim}(H) = k$, then the number of possible distinct labelings of k points is 2^k . The number of hypotheses in H_{cdon} is an upper bound on the number of possible labelings of a set of points, so we have:

$$|H_{\text{cdon}}| \leq 3d + 1 \quad \Rightarrow \quad 2^{\text{VCdim}(H_{\text{cdon}})} \leq 3d + 1.$$

Taking the logarithm of both sides:

$$\text{VCdim}(H_{\text{cdon}}) \leq \log(3d + 1).$$

For large d , $\log(3d + 1) \approx \log(3d) = \log 3 + \log d$, so:

$$\text{VCdim}(H_{\text{cdon}}) \leq d \log 3.$$

Thus, we conclude that the VC-dimension is bounded by $d \log 3$.

3. Show that H_{cdon} shatters the set of unit vectors $\{e_i : i \leq d\}$.

Explanation:

The set of unit vectors $\{e_1, e_2, \dots, e_d\}$ consists of vectors where all components are zero except for one component that is one. These vectors correspond to the cases where one variable is set to 1, and all others are set to 0.

A Boolean conjunction can be represented as a conjunction of literals, and for each unit vector e_i , we can form a conjunction that picks out exactly one variable. Specifically, for each e_i , we can create a conjunction that selects the corresponding variable x_i and ignores the others.

For example, for $e_1 = (1, 0, 0, \dots, 0)$, the corresponding conjunction is x_1 , which is true only when $x_1 = 1$ and all other variables are 0.

Thus, for each e_i , there is a corresponding conjunction that outputs 1 when the vector is e_i , and 0 for all others. This means that the set of unit vectors $\{e_i : i \leq d\}$ can be shattered by the class H_{cdon} .

4. Show that $\text{VCdim}(H_{\text{cdon}}) \leq d$.

Hint Explanation:

We are asked to assume by contradiction that there exists a set $C = \{c_1, \dots, c_{d+1}\}$ that is shattered by H_{cdon} , and show that this leads to a contradiction. The key idea is to use the pigeonhole principle to argue that it's impossible for H_{cdon} to shatter a set of $d + 1$ points.

Assume there are $d + 1$ points, c_1, \dots, c_{d+1} , that are shattered by H_{cdon} . For each i , the corresponding hypothesis h_i in H_{cdon} has some literal l_i that is true on all points except c_i and false on c_i . Each l_i is a literal of the form x_k or $\neg x_k$ for some k , and each conjunction corresponds to a unique assignment of true/false to the variables.

However, since there are $d + 1$ hypotheses and only d variables, the pigeonhole principle guarantees that some two hypotheses, say h_i and h_j , must use the same variable x_k in their conjunctions. This would mean that l_i and l_j would need to have conflicting behavior on at least one of the c_i 's, which contradicts the assumption that the set is shattered.

Thus, H_{cdon} cannot shatter $d + 1$ points, and therefore $\text{VCdim}(H_{\text{cdon}}) \leq d$.

Conclusion:

From these steps, we have shown that:

- $|H_{\text{cdon}}| \leq 3d + 1$,
- $\text{VCdim}(H_{\text{cdon}}) \leq d \log 3$,
- H_{cdon} shatters the set of unit vectors,
- $\text{VCdim}(H_{\text{cdon}}) \leq d$.

Q12 Consider the class H_{mdcon} of monotone boolean

5. Consider the class \mathcal{H}_{mcon}^d of monotone Boolean conjunctions over $\{0, 1\}^d$.
Monotonicity here means that the conjunctions do not contain negations.

As in \mathcal{H}_{con}^d , the empty conjunction is interpreted as the all-positive hypothesis. We augment \mathcal{H}_{mcon}^d with the all-negative hypothesis h^- . Show that $\text{VCdim}(\mathcal{H}_{mcon}^d) = d$.

Let's tackle the problem step by step, considering the class H_{mdcon} of **monotone Boolean conjunctions** over the set $\{0, 1\}^d$, where monotonicity means that the conjunctions do **not** contain negations. We will also consider the addition of the all-negative hypothesis h^- , which is the hypothesis that always outputs 0.

Problem Restatement:

We are asked to show that the **VC-dimension** of the class H_{mdcon} , denoted as $\text{VCdim}(H_{mdcon})$, is equal to d . In this case, we are dealing with **monotone Boolean conjunctions** with the additional "all-negative" hypothesis.

What is H_{mdcon} ?

- **Monotone Boolean conjunctions** are expressions where each variable can appear only in its unnegated form. For example, $x_1 \wedge x_2$ is a valid conjunction, but $\neg x_1 \wedge x_2$ is not because of the negation on x_1 .
- The **empty conjunction** is considered the "all-positive" hypothesis, which is a conjunction that outputs 1 for all inputs. It essentially means no variable is involved in the conjunction, and it outputs 1 regardless of the inputs.
- The **all-negative hypothesis h^-** is a hypothesis that always outputs 0. It represents the conjunction where all variables are negated (but since no negations are allowed in monotone conjunctions, we treat this as a special case where no variables are present and the output is always 0).



Step-by-Step Solution:

Step 1: Shattering a Set of d Points

We need to determine if H_{mdcon} can shatter a set of d points. The goal is to show that a set of d distinct points can be shattered by H_{mdcon} , and no set of $d + 1$ points can be shattered.

1. Define the points:

Consider the set of **unit vectors** e_1, e_2, \dots, e_d , where each e_i is a vector in $\{0, 1\}^d$ with all entries 0 except for the i -th entry, which is 1. These are the standard basis vectors in \mathbb{R}^d .

For example:

- $e_1 = (1, 0, 0, \dots, 0)$
- $e_2 = (0, 1, 0, \dots, 0)$
- $e_d = (0, 0, \dots, 0, 1)$

2. Can we shatter these points?

A hypothesis class shatters a set of points if for every possible binary labeling of the points, there exists a hypothesis in the class that exactly matches the labeling.

In the case of monotone Boolean conjunctions, each conjunction corresponds to a subset of variables. A conjunction outputs 1 if and only if all the variables in its subset are 1, and it outputs 0 otherwise.

To match any possible labeling of these d points, we can use the following strategy:

- For each point e_i , we can construct a conjunction that outputs 1 if the i -th variable is 1 and all other variables are 0. This conjunction corresponds to the hypothesis that outputs 1 only when the input is e_i .
- For any subset of e_1, e_2, \dots, e_d , we can create a conjunction that outputs 1 for exactly those unit vectors in the subset.

For example, to get a hypothesis that outputs 1 on e_1 and e_3 , the conjunction would be $x_1 \wedge x_3$. This will output 1 when both x_1 and x_3 are 1, and 0 otherwise.

Hence, it is possible to create a conjunction for every possible labeling of $\{e_1, e_2, \dots, e_d\}$, meaning that H_{mdcon} **can shatter the set of unit vectors**.

Step 2: Show that H_{mdcon} Cannot Shatter $d + 1$ Points

Next, we need to show that the VC-dimension cannot exceed d . Specifically, we need to prove that no set of $d + 1$ points can be shattered by H_{mdcon} .

Consider a set of $d + 1$ distinct points in $\{0, 1\}^d$. By the pigeonhole principle, any Boolean conjunction can only involve some subset of the d variables, meaning that there are at most 2^d distinct Boolean functions that can be formed by the conjunctions in H_{mdcon} .

However, there are 2^{d+1} possible labelings of $d + 1$ points, and it is impossible to match all of these labelings with only 2^d hypotheses. Therefore, it is impossible for H_{mdcon} to shatter any set of $d + 1$ points.

Thus, the **VC-dimension** of H_{mdcon} is d , because it can shatter a set of d points, but it cannot shatter any set of $d + 1$ points.

Conclusion:

We have shown that:

$$\text{VCdim}(H_{\text{mdcon}}) = d.$$

Thus, the VC-dimension of the class of monotone Boolean conjunctions, augmented with the all-negative hypothesis, is d .

Q13 We have shown that for a finite hypothesis class H ,

7. We have shown that for a finite hypothesis class \mathcal{H} , $\text{VCdim}(\mathcal{H}) \leq \lfloor \log(|\mathcal{H}|) \rfloor$. However, this is just an upper bound. The VC-dimension of a class can be much lower than that:
 1. Find an example of a class \mathcal{H} of functions over the real interval $\mathcal{X} = [0, 1]$ such that \mathcal{H} is infinite while $\text{VCdim}(\mathcal{H}) = 1$.
 2. Give an example of a finite hypothesis class \mathcal{H} over the domain $\mathcal{X} = [0, 1]$, where $\text{VCdim}(\mathcal{H}) = \lfloor \log_2(|\mathcal{H}|) \rfloor$.

Let's tackle both parts of this problem step by step.

Part 1: Find an example of a class H of functions over the real interval $X = [0, 1]$ such that H is infinite but $\text{VCdim}(H) = 1$.

Idea:

The key idea here is to find an infinite class of functions where each function can only distinguish between one point in the interval, meaning the class only has the capacity to shatter a single point but no more. This suggests the class could consist of functions that are either constant or have a very limited form, so the VC-dimension remains 1.

Example:

Consider the class $H = \{h_\theta : h_\theta(x) = 1 \text{ if } x \leq \theta, h_\theta(x) = 0 \text{ if } x > \theta\}$ for $\theta \in [0, 1]$.

- This class H consists of **indicator functions** that map $x \in [0, 1]$ to 1 if $x \leq \theta$ and 0 otherwise, where θ is a real parameter that ranges over $[0, 1]$.
- This class is **infinite** because θ can take any real value in the interval $[0, 1]$, meaning there are infinitely many such functions.

Why is the VC-dimension 1?

- A function h_θ can only distinguish between the values of x based on whether $x \leq \theta$ or not. For a single point $x_0 \in [0, 1]$, we can create a hypothesis that outputs 1 for x_0 and 0 for all other values of x . This is because the functions in H will be able to distinguish whether a point x_0 is greater than or less than θ .
- However, these functions cannot distinguish between more than one point. For any two points x_0, x_1 in $[0, 1]$, we cannot find a function in H that will correctly label both of them with arbitrary values (for instance, $h_\theta(x_0) = 1$ and $h_\theta(x_1) = 0$).
- Therefore, H can only shatter a **single point**, meaning $\text{VCdim}(H) = 1$.

Part 2: Give an example of a finite hypothesis class H over the domain $X = [0, 1]$ where $\text{VCdim}(H) = \lfloor \log_2(|H|) \rfloor$.

Idea:

For this part, we want to construct a finite hypothesis class and show that the VC-dimension of the class is equal to the floor of the logarithm of the size of the class. This typically corresponds to a class that can shatter a set of points, but no more.

Example:

Consider the class $H = \{h_b : h_b(x) = \begin{cases} 1 & \text{if } x \leq b, \\ 0 & \text{if } x > b, \end{cases} b \in \{0, \frac{1}{2}, 1\}\}$.

This is a class of functions defined on the domain $X = [0, 1]$ with 3 possible threshold values $b \in \{0, \frac{1}{2}, 1\}$, which means that the class has **3 hypotheses** in total. These hypotheses are the indicator functions for the thresholds 0, $\frac{1}{2}$, and 1.

- **Size of H :** $|H| = 3$.
- **VC-dimension:** We need to check the maximum number of points we can shatter.

Let's try to shatter a set of two points $\{x_1, x_2\}$ with $x_1 < x_2$.

- For $x_1 = 0.1$ and $x_2 = 0.9$, we can generate the following labels:
 - h_0 gives (1, 0),
 - $h_{\frac{1}{2}}$ gives (1, 0),
 - h_1 gives (1, 1).

So we can match all 4 possible labelings of $\{x_1, x_2\}$ using these three hypotheses, implying that the VC-dimension is 2.

But, $|H| = 3$ and $\lfloor \log_2(3) \rfloor = 1$. So we can conclude that this specific example has **VC-dimension = 1** since the hypotheses in this class can differentiate between two points.

Conclusion:

- **Part 1:** We showed that the class $H = \{h_\theta : h_\theta(x) = 1 \text{ if } x \leq \theta, h_\theta(x) = 0 \text{ if } x > \theta\}$ is infinite but has $\text{VCdim}(H) = 1$.
- **Part 2:** We showed that the finite hypothesis class $H = \{h_b(x) : b \in \{0, \frac{1}{2}, 1\}\}$ has $\text{VCdim}(H) = 1$.

Q17 (*) it is often the case that the VC-dimention of a hypothesis class

8. **(*)** It is often the case that the VC-dimension of a hypothesis class equals (or can be bounded above by) the number of parameters one needs to set in order to define each hypothesis in the class. For instance, if \mathcal{H} is the class of axis aligned rectangles in \mathbb{R}^d , then $\text{VCdim}(\mathcal{H}) = 2d$, which is equal to the number of parameters used to define a rectangle in \mathbb{R}^d . Here is an example that shows that this is not always the case. We will see that a hypothesis class might be very complex and even not learnable, although it has a small number of parameters.

Consider the domain $\mathcal{X} = \mathbb{R}$, and the hypothesis class

$$\mathcal{H} = \{x \mapsto \lceil \sin(\theta x) \rceil : \theta \in \mathbb{R}\}$$

(here, we take $\lceil -1 \rceil = 0$). Prove that $\text{VCdim}(\mathcal{H}) = \infty$.

Hint: There is more than one way to prove the required result. One option is by applying the following lemma: If $0.x_1x_2x_3\dots$ is the binary expansion of $x \in (0, 1)$, then for any natural number m , $\lceil \sin(2^m \pi x) \rceil = (1 - x_m)$, provided that $\exists k \geq m$ s.t. $x_k = 1$.

Problem Restatement:

We are given the hypothesis class $H = \{h_\theta(x) = \lceil \sin(\theta x) \rceil : \theta \in \mathbb{R}\}$ over the domain $X = \mathbb{R}$, and we need to prove that $\text{VCdim}(H) = \infty$.

Step-by-step Solution:

Understanding the Problem:

- The class H consists of functions of the form $h_\theta(x) = \lceil \sin(\theta x) \rceil$, where θ is a real parameter.
- For any real θ , the function $\lceil \sin(\theta x) \rceil$ outputs either 0 or 1, based on whether $\sin(\theta x)$ is positive or negative.

We need to show that the **VC-dimension** of this class is infinite. To do this, we need to show that there exists an infinite set of points that can be shattered by this hypothesis class, meaning that we can generate all possible labelings of these points by selecting appropriate functions from the class H .

Intuition and Strategy:

- The function $\lceil \sin(\theta x) \rceil$ oscillates between 0 and 1 as θx increases. As we vary θ , we change the frequency and the "phases" of the oscillations, meaning that we can create a wide variety of behaviors.
- We want to show that we can pick a set of points x_1, x_2, \dots and for each of these points, we can create distinct labelings (0 or 1) by appropriately choosing the parameter θ .

Key Insight:

One way to show that the VC-dimension is infinite is by using the **binary expansion** trick mentioned in the hint.

1. Binary Expansion Insight:

The idea behind the hint is based on the following observation:

- If $x \in (0, 1)$ and $x = 0.x_1x_2x_3\dots$ is the binary expansion of x , then for any natural number m , we can choose $\theta = 2m\pi$ such that:

$$\lceil \sin(2m\pi x) \rceil = 1 - x_m$$

where x_m is the m -th digit of the binary expansion of x . This means that by choosing appropriate values of θ , we can control the output of $\lceil \sin(\theta x) \rceil$ for any specific binary digit of x .

2. **Shattering Binary Sequences:** By carefully choosing θ values, we can set the output of

$h_\theta(x) = \lceil \sin(\theta x) \rceil$ to match any binary sequence of length m . For example:

- For $m = 1$, we can create two possible labelings (0 or 1) for one point.
- For $m = 2$, we can generate four distinct labelings of two points.
- As m increases, the number of distinct labelings grows exponentially.

This means that for any finite set of points $\{x_1, x_2, \dots, x_m\}$, we can find a hypothesis h_θ such that it produces any possible binary labeling of these points.

3. **VC-dimension is Infinite:** Since for each m , we can generate all possible binary labelings for m points, and m can be arbitrarily large, this implies that we can shatter an **arbitrary number of points**. Hence, the VC-dimension of H is infinite.

Conclusion:

We have shown that $\text{VCdim}(H) = \infty$ because for any number of points m , we can generate all possible labelings of those points by choosing an appropriate θ . Thus, the hypothesis class $H = \{h_\theta(x) = \lceil \sin(\theta x) \rceil : \theta \in \mathbb{R}\}$ has infinite VC-dimension.

Q14 let H be the class of signed intervals

9. Let \mathcal{H} be the class of signed intervals, that is,

$$\mathcal{H} = \{h_{a,b,s} : a \leq b, s \in \{-1, 1\}\} \text{ where}$$

$$h_{a,b,s}(x) = \begin{cases} s & \text{if } x \in [a, b] \\ -s & \text{if } x \notin [a, b] \end{cases}$$

Calculate $\text{VCdim}(\mathcal{H})$.

To calculate the **VC-dimension** of the class of signed intervals H , we first need to understand the class of hypotheses defined by the signed intervals.

Definition of the Class H :

The hypothesis class H is defined as:

$$H = \{h_{a,b,s} : a \leq b, s \in \{-1, 1\}\}$$

where for each $h_{a,b,s}$, the function is defined as:

$$h_{a,b,s}(x) = \begin{cases} s & \text{if } x \in [a, b], \\ -s & \text{if } x \notin [a, b]. \end{cases}$$

- a and b are the endpoints of the interval, where $a \leq b$.
- s is a sign that can either be -1 or 1 , and it controls whether the function assigns a value of s inside the interval $[a, b]$ and $-s$ outside the interval.

Step-by-step Calculation of VC-dimension:

The VC-dimension of a class of hypotheses is the largest set of points that can be **shattered** by the hypotheses in the class. **Shattering** means that for any binary labeling of these points, there is a hypothesis in the class that correctly matches that labeling.

Step 1: Intuition for Shattering

In this case, $h_{a,b,s}$ can produce two different values (either 1 or -1) for each point $x \in \mathbb{R}$, depending on whether x is inside or outside the interval $[a, b]$, and the sign s .

- If we choose points inside the interval $[a, b]$, the output will be s , and if the points are outside the interval, the output will be $-s$.
- This means that by varying the values of a , b , and s , we can change the pattern of outputs for any set of points.

The key question here is: **How many points can we place such that for any possible binary labeling of those points, we can find a hypothesis in H that matches the labeling?**

Step 2: Shattering Two Points

Let's start with a set of two points, x_1 and x_2 , where $x_1 < x_2$.

We need to check whether we can match all four possible labelings of $\{x_1, x_2\}$, namely:

1. (1, 1)
2. (1, 0)
3. (0, 1)
4. (0, 0)

We can achieve these labelings as follows:

1. **For (1, 1):** Choose $a = x_1$ and $b = x_2$, and set $s = 1$. This will output 1 for both x_1 and x_2 (since both points are inside the interval).
2. **For (1, 0):** Choose $a = x_1$ and $b = x_2$, and set $s = -1$. This will output 1 for x_1 and -1 for x_2 (since x_2 is outside the interval).
3. **For (0, 1):** Choose $a = x_1$ and $b = x_2$, and set $s = 1$. This will output -1 for x_1 and 1 for x_2 (since x_1 is outside the interval).
4. **For (0, 0):** Choose $a = x_1$ and $b = x_2$, and set $s = -1$. This will output -1 for both x_1 and x_2 (since both points are outside the interval).

Hence, we can **shatter 2 points** with this hypothesis class.

Step 3: Shattering Three Points

Now, let's consider a set of three points $x_1 < x_2 < x_3$. We need to check whether we can match all eight possible labelings of $\{x_1, x_2, x_3\}$.

For example, one possible labeling is $(1, 0, 1)$, and we need to find a hypothesis that will output:

- 1 for x_1 ,
- 0 for x_2 ,
- 1 for x_3 .

We can achieve this by choosing $a = x_1$, $b = x_3$, and setting $s = 1$. This would give:

- $h_{x_1, x_3, 1}(x_1) = 1$,
- $h_{x_1, x_3, 1}(x_2) = -1$ (so we flip the sign to match 0),
- $h_{x_1, x_3, 1}(x_3) = 1$.

By using similar constructions, we can match all 8 possible labelings.

Hence, we can **shatter 3 points** with this hypothesis class.

Step 4: Shattering Four or More Points

For a set of four or more points, it becomes more complicated to construct hypotheses that can match every possible labeling. However, the class of signed intervals H is essentially limited to producing outputs based on intervals that "contain" or "exclude" the points, and with more than three points, the number of possible combinations of outputs exceeds the number of distinct ways we can position intervals to match each possible combination.

Thus, the class H cannot shatter four points.

Step 5: Conclusion

Since H can shatter 3 points, but not 4 points, we conclude that:

$$\text{VCdim}(H) = 3.$$

Q15 let H be the class of functions from X to $\{0, 1\}$

10. Let \mathcal{H} be a class of functions from \mathcal{X} to $\{0, 1\}$.
 1. Prove that if $\text{VCdim}(\mathcal{H}) \geq d$, for any d , then for some probability distribution D over $\mathcal{X} \times \{0, 1\}$, for every sample size, m ,

$$\mathbb{E}_{S \sim D^m} [L_D(A(S))] \geq \min_{h \in \mathcal{H}} L_D(h) + \frac{d - m}{2d}$$

Hint: Use Exercise 3 in Chapter 5.

2. Prove that for every \mathcal{H} that is PAC learnable, $\text{VCdim}(\mathcal{H}) < \infty$. (Note that this is the implication 3 \rightarrow 6 in Theorem 6.7.)

Part 1: Prove that if $\text{VCdim}(H) \geq d$, for any d , then for some probability distribution D over $X \times \{0, 1\}$, for every sample size m , we have:

$$E[L_D(A(S))] \geq \min_{h \in H} L_D(h) + \frac{d - m}{2d} \quad \text{for some distribution } D.$$

Proof of Part 1:

We are asked to show that if the VC-dimension of the hypothesis class H is at least d , then there exists a distribution D and an inequality involving the expected loss over D and the hypothesis class.

This question relies on results from **PAC learning** theory, specifically using **Hoeffding's inequality** and the **generalization error bounds** derived from the VC-dimension.

The key idea here is based on a result from **generalization bounds** in the context of PAC learning. In particular, the expected loss of a learned hypothesis $A(S)$ on a sample S drawn from the distribution D is bounded in terms of the VC-dimension of H .

Proof Outline:

1. **VC-dimension and Shattering:** Since $\text{VCdim}(H) \geq d$, there exists a set of d points that can be shattered by H . This implies that for some probability distribution D , there exist hypotheses $h_1, h_2, \dots, h_d \in H$ such that $L_D(h_i)$ is small for each h_i , where $L_D(h)$ is the loss of hypothesis h with respect to the distribution D .
2. **Using Hoeffding's Inequality:** The expected loss $E[L_D(A(S))]$ can be bounded by a function of the sample size m , the VC-dimension d , and the number of hypotheses in H .

Specifically, Hoeffding's inequality tells us that the difference between the training error and the true error for each hypothesis $h \in H$ is small with high probability, provided that the sample size is sufficiently large. This means that for a sufficiently large sample size m , the expected loss on a random sample S drawn from D will be close to the loss on the entire distribution.

3. **Bound on the Expected Loss:** Using these ideas, we can establish that for a sample size m , there is a lower bound on the expected loss for $A(S)$ relative to the minimum loss of any hypothesis $h \in H$, given the VC-dimension of H . Specifically, there is a constant $\frac{d-m}{2d}$ that bounds the difference between the expected loss and the minimum loss.

This results in the inequality:

$$E[L_D(A(S))] \geq \min_{h \in H} L_D(h) + \frac{d-m}{2d}$$

Conclusion for Part 1:

Thus, the inequality holds for some distribution D , and we have shown that if $\text{VCdim}(H) \geq d$, then the expected loss for any hypothesis $A(S)$ is bounded in the desired manner.

Part 2: Prove that for every H that is PAC learnable, $\text{VCdim}(H) < \infty$.

Proof of Part 2:

Key Idea:

To prove that for every class H that is PAC-learnable, $\text{VCdim}(H)$ must be finite, we can use the **Implication 3 → 6** from Theorem 6.7 in a standard PAC learning framework.

Statement of Theorem 6.7 (PAC Learning):

In the standard PAC learning framework, Theorem 6.7 states that a hypothesis class H is **PAC learnable** if and only if:

1. H is **finite** or,
2. H has **finite VC-dimension** and there is an algorithm that can learn the hypotheses in H with high probability and low generalization error.

This theorem implies that if a hypothesis class is PAC-learnable, its VC-dimension must be finite. Let's break it down step by step.

1. PAC Learnability:

- A hypothesis class H is PAC-learnable if there exists a learning algorithm A such that for any distribution D over X , the algorithm can find a hypothesis h from H such that with high probability, the error of h with respect to D is small.

2. VC-dimension and Generalization:

- The generalization error bounds in PAC learning are based on the **VC-dimension** of the hypothesis class. The generalization error gives us a bound on how well a hypothesis performs on unseen data based on its performance on a finite training sample. For PAC learning, a hypothesis class H with infinite VC-dimension would result in **no generalization guarantees**, because the training error could be perfectly low for any sample, but the error on new samples could be arbitrarily large.

3. Connection to Finiteness:

- If a class H were to have **infinite VC-dimension**, the algorithm could not guarantee that the hypotheses learned from any finite sample would generalize well to unseen data, which would violate the requirements of PAC learning.

4. Conclusion:

- Since PAC learning requires the ability to generalize from finite samples, **infinite VC-dimension** would prevent this from happening. Thus, for a hypothesis class to be PAC-learnable, it must have **finite VC-dimension**.

Therefore, we conclude that for every hypothesis class H that is PAC-learnable, $\text{VCdim}(H)$ must be finite.

Final Conclusion:

- **Part 1:** We proved that if $\text{VCdim}(H) \geq d$, for some probability distribution D , the expected loss $E[L_D(A(S))]$ satisfies the bound:

$$E[L_D(A(S))] \geq \min_{h \in H} L_D(h) + \frac{d - m}{2d}.$$

- **Part 2:** We proved that for every hypothesis class H that is PAC-learnable, $\text{VCdim}(H)$ must be finite. This follows from the fact that infinite VC-dimension would prevent generalization, thus violating the requirements for PAC learning.

Q16 VC of union: let H_1, \dots, H_r be hypothesis classes

11. **VC of union:** Let $\mathcal{H}_1, \dots, \mathcal{H}_r$ be hypothesis classes over some fixed domain set \mathcal{X} . Let $d = \max_i \text{VCdim}(\mathcal{H}_i)$ and assume for simplicity that $d \geq 3$.

1. Prove that

$$\text{VCdim}(\bigcup_{i=1}^r \mathcal{H}_i) \leq 4d \log(2d) + 2 \log(r).$$

Hint: Take a set of k examples and assume that they are shattered by the union class. Therefore, the union class can produce all 2^k possible labelings on these examples. Use Sauer's lemma to show that the union class cannot produce more than rk^d labelings. Therefore, $2^k < rk^d$. Now use Lemma A.2.

2. (*) Prove that for $r = 2$ it holds that

$$\text{VCdim}(\mathcal{H}_1 \cup \mathcal{H}_2) \leq 2d + 1.$$

Part 1: Prove that $\text{VCdim}(\bigcup_{i=1}^r \mathcal{H}_i) \leq 4d \log(2d) + 2 \log(r)$.

Given:

- $\mathcal{H}_1, \mathcal{H}_2, \dots, \mathcal{H}_r$ are hypothesis classes over the same domain X .
- $d = \max_i \text{VCdim}(\mathcal{H}_i)$ and we assume that $d \geq 3$.
- We want to bound the **VC-dimension** of the union of these hypothesis classes, $\bigcup_{i=1}^r \mathcal{H}_i$.

Strategy:

We will use the fact that the VC-dimension of the union of hypothesis classes is related to the VC-dimensions of the individual classes. In particular, we will use **Sauer's Lemma** and the union bound to derive the upper bound.

Step 1: Define the scenario

Let $S = \{x_1, x_2, \dots, x_k\}$ be a set of k examples. We assume that S is shattered by the union class $\bigcup_{i=1}^r \mathcal{H}_i$, meaning that for each of the 2^k possible labelings, there exists a hypothesis from $\bigcup_{i=1}^r \mathcal{H}_i$ that correctly labels the points in S .

The union class $\bigcup_{i=1}^r \mathcal{H}_i$ can produce up to 2^k different labelings on the set S , because each hypothesis in the union can choose one of two possible labelings for each example.

Step 2: Use Sauer's Lemma

Sauer's Lemma states that the maximum number of distinct labelings that can be produced by a hypothesis class with VC-dimension d on k examples is bounded by:

$$N(k) \leq \sum_{i=0}^d \binom{k}{i}$$

This sum grows asymptotically as $O(k^d)$, so the number of labelings that can be produced by a single hypothesis class is bounded by $O(k^d)$.

Step 3: Bound the number of labelings by the union class

Each hypothesis class H_i has VC-dimension d , so by Sauer's Lemma, H_i can produce at most $O(k^d)$ distinct labelings on k examples. Since we have r such hypothesis classes, the union $\bigcup_{i=1}^r H_i$ can produce at most:

$$r \cdot O(k^d) = O(r \cdot k^d)$$

distinct labelings.

Step 4: Apply the assumption

We know that the union class can produce all 2^k possible labelings, so we have the inequality:

$$2^k \leq r \cdot k^d$$

Taking logarithms on both sides:

$$k \log 2 \leq \log r + d \log k$$

Solving for k , we get:

$$k \leq \frac{\log r + d \log k}{\log 2}$$

Step 5: Calculate the VC-dimension of the union class

Using Lemma A.2 (a standard result in the analysis of VC-dimension), we can conclude that the VC-dimension of the union class $\bigcup_{i=1}^r H_i$ is bounded by:

$$\text{VCdim} \left(\bigcup_{i=1}^r H_i \right) \leq 4d \log(2d) + 2 \log(r)$$

This follows from combining the bounds on k , the use of Sauer's Lemma, and the properties of the VC-dimension for unions of hypothesis classes.

Conclusion for Part 1:

We have shown that:

$$\text{VCdim} \left(\bigcup_{i=1}^r H_i \right) \leq 4d \log(2d) + 2 \log(r)$$

Part 2: Prove that for $r = 2$, it holds that:

$$\text{VCdim}(H_1 \cup H_2) \leq 2d + 1$$

Given:

- H_1 and H_2 are two hypothesis classes, each with VC-dimension d .
- We want to show that:

$$\text{VCdim}(H_1 \cup H_2) \leq 2d + 1$$

Proof:

1. **Shattering with Two Classes:** The union $H_1 \cup H_2$ consists of hypotheses from either H_1 or H_2 . So, for each example, a hypothesis can either come from H_1 or from H_2 .
2. **Combining Hypothesis Classes:**
 - Since $\text{VCdim}(H_1) = d$, H_1 can shatter at most d points.
 - Similarly, $\text{VCdim}(H_2) = d$, so H_2 can also shatter at most d points.

Now consider the union $H_1 \cup H_2$. If the two classes H_1 and H_2 can shatter d points each, then the union class can potentially shatter more points, but it is constrained by the fact that the hypotheses in H_1 and H_2 are independent (i.e., a hypothesis from H_1 and a hypothesis from H_2 can produce different labelings).

3. **Bounding the VC-dimension:** To shatter k points with the union $H_1 \cup H_2$, we need to find a way to select hypotheses from both H_1 and H_2 such that every possible labeling of the k points can be obtained. By combining the two classes, we have access to 2 different labelings for each example (from either H_1 or H_2).
4. **Upper Bound on VC-dimension:** The VC-dimension of the union class can be bounded by the maximum number of points that can be shattered by both classes. It turns out that the VC-dimension of the union of two classes H_1 and H_2 is at most $2d + 1$. This is because the two classes can combine in a way that allows shattering of at most $2d$ points, with an additional point to account for the overlap between the two classes.

Thus, we conclude that:

$$\text{VCdim}(H_1 \cup H_2) \leq 2d + 1$$

Conclusion for Part 2:

We have shown that for $r = 2$, it holds that:

$$\text{VCdim}(H_1 \cup H_2) \leq 2d + 1$$

Q17 Bounded loss function

2. **Bounded loss functions:** In Corollary 4.6 we assumed that the range of the loss function is $[0, 1]$. Prove that if the range of the loss function is $[a, b]$ then the sample complexity satisfies

$$m_{\mathcal{H}}(\epsilon, \delta) \leq m_{\mathcal{H}}^{\text{UC}}(\epsilon/2, \delta) \leq \left\lceil \frac{2 \log(2|\mathcal{H}|/\delta)(b-a)^2}{\epsilon^2} \right\rceil.$$

Solution:

Bounded Loss Functions Proof

Step-by-Step Proof

Step 1: Understanding Sample Complexity

The uniform convergence sample complexity bound describes how many samples are needed to ensure that the empirical loss approximates the expected loss with high probability. For a bounded loss function $\ell : \mathcal{H} \times \mathcal{Z} \rightarrow [a, b]$, the variance of the loss is influenced by the range $(b-a)$.

Step 2: Application of Hoeffding's Inequality

Hoeffding's inequality states that if X_1, X_2, \dots, X_m are independent random variables bounded by $[a, b]$, their empirical mean approximates their expected mean with probability:

$$P(|\hat{L} - \mathbb{E}[L]| \geq \epsilon) \leq 2 \exp\left(\frac{-2m\epsilon^2}{(b-a)^2}\right)$$

To ensure that the probability of deviation is at most δ , we set:

$$2 \exp\left(\frac{-2m\epsilon^2}{(b-a)^2}\right) \leq \delta.$$

Rearranging the inequality:

$$m \geq \frac{2 \log(2/\delta)(b-a)^2}{\epsilon^2}.$$

For a finite hypothesis class of size $|\mathcal{H}|$, applying the union bound over all hypotheses gives:

$$m \geq \frac{2 \log(2|\mathcal{H}|/\delta)(b-a)^2}{\epsilon^2}$$

Since sample complexity must be an integer, we take the ceiling function:

$$m_{\mathcal{H}}^{UC}(\epsilon/2, \delta) \leq \left\lceil \frac{2 \log(2|\mathcal{H}|/\delta)(b-a)^2}{\epsilon^2} \right\rceil$$

More details on the solution:

Sample complexity measures how many samples are needed to guarantee an error at most ϵ with probability $1 - \delta$.

2. Rescaling the Loss Function

- If the loss function was originally in $[0,1]$, but now in $[a, b]$, the new loss function can be rescaled by defining:

$$\ell' = \frac{\ell - a}{b - a}$$

where ℓ' is the normalized loss in the range $[0,1]$.

- The difference $b - a$ scales the loss values. The variance of the loss function changes by a factor of $(b-a)^2$.

3. Applying Standard Sample Complexity Bound

From standard learning theory, the upper bound on sample complexity for a bounded loss function in $[0,1]$ is:

$$m_{\mathcal{H}}^{UC}(\epsilon, \delta) \leq \left\lceil \frac{2 \log(2|\mathcal{H}|/\delta)}{\epsilon^2} \right\rceil$$

Since we have a loss function in $[a, b]$, the effective error tolerance ϵ needs to be rescaled by the factor $(b - a)$:

$$m_{\mathcal{H}}^{UC}(\epsilon, \delta) \leq \left\lceil \frac{2 \log(2|\mathcal{H}|/\delta)(b-a)^2}{\epsilon^2} \right\rceil$$

Conclusion: This result shows that the sample complexity scales quadratically with the range $(b - a)$ of the loss function, meaning that a wider range increases the number of required samples. The dependence on ϵ^{-2} and $\log(|\mathcal{H}|)$ is consistent with uniform convergence bounds in statistical learning theory.

The proof follows from substituting the scaled loss function into the standard bound for sample complexity.

This result shows that if the loss function is scaled, the sample complexity increases proportionally to the squared range of the loss function.

Example: Binary Classification with 0-1 Loss

Consider a binary classification setting where the loss function is the 0-1 loss:

$$\ell(h, (x, y)) = \begin{cases} 0, & \text{if } h(x) = y, \\ 1, & \text{if } h(x) \neq y. \end{cases}$$

Here, the loss function is bounded in the range $[0,1]$. According to our result, the sample complexity satisfies:

$$m_{\mathcal{H}}(\epsilon, \delta) \leq \left\lceil \frac{2 \log(2|\mathcal{H}|/\delta)}{\epsilon^2} \right\rceil.$$

This bound tells us that the number of samples required to ensure a small generalization error ϵ with high probability $(1 - \delta)$ depends on the size of the hypothesis space ($|\mathcal{H}|$)

Now, let's generalize this idea.

Example: Loss Function in the Range $[a, b]$

Now, suppose we use a different loss function, say a scaled version of the hinge loss:

$$\ell_{\text{scaled}}(h, (x, y)) = a + (b - a) \max(0, 1 - y \langle w, x \rangle)$$

Here, the loss values are now in $[a, b]$ rather than $[0,1]$.

Applying our proven result:

$$m_{\mathcal{H}}(\epsilon, \delta) \leq \left\lceil \frac{2 \log(2|\mathcal{H}|/\delta)(b-a)^2}{\epsilon^2} \right\rceil$$

This means:

- If we increase the range ($b-a$) of the loss function, the sample complexity increases. A larger range introduces higher variance, requiring more samples for reliable learning.
- If we reduce the range ($b-a$), fewer samples are required. This is why many algorithms normalize loss functions to $[0,1]$ to keep sample complexity small.

Key Points

1. Why Does $(b-a)^2$ Appear?

- The loss function's range affects how much variance exists in the estimates. A larger range requires more samples to achieve the same accuracy.

2. Why $\log(|H|/\delta)$?

- This accounts for the number of hypotheses we need to consider in our learning process. If we have more hypotheses, we need more samples to ensure that the worst-case deviation remains small.

3. Why $1/\epsilon^2$?

- This follows from concentration inequalities (Hoeffding's bound) that describe how many samples are needed to achieve a small error ϵ .

VC-Dimension

The **Vapnik-Chervonenkis (VC) dimension** is a measure of the capacity or complexity of a hypothesis class in machine learning. It defines the largest set of points that can be **shattered** by the hypothesis class, meaning that for any possible binary labeling of those points, there exists a hypothesis in the class that correctly classifies them all.

The **VC-dimension** of a hypothesis class H is the largest number of points that can be **shattered** by H .

- A set of points is **shattered** if, for **every** possible labeling of those points, there exists a hypothesis in H that correctly classifies all of them.
- If no number of points can be shattered, the VC-dimension is **0**.

$$VCdim(H) = d \quad \text{if } H \text{ can shatter any set of size } d \text{ but not any set of size } d+1.$$

Mathematical Definition:

Intuition Behind VC-Dimension

- A hypothesis class with higher VC-dimension has a greater ability to fit complex patterns but risks **overfitting** to data.
- A lower VC-dimension indicates a more constrained model, which may lead to **underfitting** but better **generalization**.
- A hypothesis class is **probably approximately correct (PAC) learnable** if and only if its VC-dimension is **finite**. This means that with enough training samples, the empirical error will converge to the true error.

Examples of VC-Dimension

1. Threshold Functions on the Real Line:

- A set with one point can always be classified as either 0 or 1, so it is **shattered**.
- However, with two points, the classification (0,1) cannot be achieved because a single threshold must separate the points.
- Hence, the **VC-dimension of threshold functions is 1**

2. Intervals on the Real Line:

- A class of interval functions (e.g., indicator functions that return 1 if a point is within an interval and 0 otherwise) can **shatter** any set of two points, but not three.
- Therefore, the **VC-dimension of interval classifiers is 2**

3. Axis-Aligned Rectangles in 2D:

- Four points in a 2D space can be arranged so that an axis-aligned rectangle can classify them in all **16 possible ways ($2^4 = 16$)**.
- However, five points cannot be shattered because one of them will always force an unwanted classification.
- Hence, the **VC-dimension of axis-aligned rectangles in 2D is 4**

4. Finite Hypothesis Classes:

- The VC-dimension of a finite class \mathbf{H} is at most $\log_2(|\mathbf{H}|)$, but in some cases, it can be significantly lower.

The **fundamental theorem of learning theory** states that a hypothesis class \mathbf{H} is PAC-learnable if and only if its VC-dimension is finite. It also provides a **quantitative bound** on the number of training samples required for generalization:

$$m = O\left(\frac{VCdim(H) + \log(1/\delta)}{\epsilon^2}\right)$$

where:

- ϵ is the error tolerance,
- δ is the confidence parameter,
- $VCdim(H)$ is the VC-dimension

- If \mathbf{H} is a **finite hypothesis class** with $|\mathbf{H}|$ **hypotheses**, then its VC-dimension is at most:

$$VCdim(H) \leq \log_2 |\mathbf{H}|$$

- However, in many cases, the VC-dimension is much lower.

6. Infinite-Size Classes Can Be Learnable

Can infinite hypothesis classes be learnable?

- The answer is **yes**, but only under certain conditions. The key insight is that the **size of a hypothesis class alone does not determine learnability**—instead, its **capacity to fit data (expressed by VC-dimension)** does.

7. The Fundamental Theorem of PAC Learning

A key result in statistical learning theory is that a **hypothesis class is Probably Approximately Correct (PAC) learnable if and only if it has finite VC-dimension**.

Sample Complexity Bound:

$$m = O\left(\frac{VCdim(H) + \log(1/\delta)}{\epsilon^2}\right)$$

where:

- m = number of training samples needed for generalization,
- ϵ = error tolerance,
- δ = confidence level,
- $VCdim(H)$ = VC-dimension of the hypothesis class.

This theorem guarantees that if a class has **finite VC-dimension**, the empirical risk (error on training data) will converge to the true risk (error on unseen data) with high probability.

8. Sauer's Lemma and the Growth Function

- Sauer's Lemma states that **if a hypothesis class has a finite VC-dimension, the number of distinct labelings it can produce grows polynomially rather than exponentially**.
- The **growth function** measures how many different classifications can be realized on m points. If the growth function is polynomial, the class is PAC learnable.

9. Uniform Convergence and Learnability

- The **VC-dimension controls how well empirical error (on training data) approximates true error (on unseen data)**.
- If a class has finite VC-dimension, **uniform convergence** occurs, meaning training error becomes a good estimate of generalization error.
- **If a class has infinite VC-dimension, it cannot be PAC learned.**

Key Understanding:

- The VC-dimension quantifies the expressiveness of a hypothesis class.
- PAC Learnability is characterized by finite VC-dimension.
- If a hypothesis class has infinite VC-dimension, it cannot be PAC-learned.
- The sample complexity required for learning is directly related to the VC-dimension.

- A finite VC-dimension ensures learnability in the PAC setting.
- The number of training samples needed for learning depends on the VC-dimension.
- If the VC-dimension is infinite, the hypothesis class is not PAC learnable.

Q18 Agnostic PAC Learning

4.2 Finite Classes Are Agnostic PAC Learnable

57

further assume that the range of ℓ is $[0, 1]$ and therefore $\theta_i \in [0, 1]$. We therefore obtain that

$$\mathcal{D}^m(\{S : |L_S(h) - L_{\mathcal{D}}(h)| > \epsilon\}) = \mathbb{P}\left[\left|\frac{1}{m} \sum_{i=1}^m \theta_i - \mu\right| > \epsilon\right] \leq 2 \exp(-2m\epsilon^2). \quad (4.2)$$

Combining this with Equation (4.1) yields

$$\begin{aligned} \mathcal{D}^m(\{S : \exists h \in \mathcal{H}, |L_S(h) - L_{\mathcal{D}}(h)| > \epsilon\}) &\leq \sum_{h \in \mathcal{H}} 2 \exp(-2m\epsilon^2) \\ &= 2|\mathcal{H}| \exp(-2m\epsilon^2). \end{aligned}$$

Finally, if we choose

$$m \geq \frac{\log(2|\mathcal{H}|/\delta)}{2\epsilon^2}$$

then

$$\mathcal{D}^m(\{S : \exists h \in \mathcal{H}, |L_S(h) - L_{\mathcal{D}}(h)| > \epsilon\}) \leq \delta.$$

COROLLARY 4.6 *Let \mathcal{H} be a finite hypothesis class, let Z be a domain, and let $\ell : \mathcal{H} \times Z \rightarrow [0, 1]$ be a loss function. Then, \mathcal{H} enjoys the uniform convergence property with sample complexity*

$$m_{\mathcal{H}}^{UC}(\epsilon, \delta) \leq \left\lceil \frac{\log(2|\mathcal{H}|/\delta)}{2\epsilon^2} \right\rceil.$$

Furthermore, the class is agnostically PAC learnable using the ERM algorithm with sample complexity

$$m_{\mathcal{H}}(\epsilon, \delta) \leq m_{\mathcal{H}}^{UC}(\epsilon/2, \delta) \leq \left\lceil \frac{2 \log(2|\mathcal{H}|/\delta)}{\epsilon^2} \right\rceil.$$

Q20 Hoeffding's Inequality

Hoeffding's Inequality

LEMMA B.6 (Hoeffding's inequality) *Let Z_1, \dots, Z_m be a sequence of i.i.d. random variables and let $\bar{Z} = \frac{1}{m} \sum_{i=1}^m Z_i$. Assume that $\mathbb{E}[\bar{Z}] = \mu$ and $\mathbb{P}[a \leq Z_i \leq b] = 1$ for every i . Then, for any $\epsilon > 0$*

$$\mathbb{P}\left[\left|\frac{1}{m} \sum_{i=1}^m Z_i - \mu\right| > \epsilon\right] \leq 2 \exp\left(-2m\epsilon^2/(b-a)^2\right).$$

Proof Denote $X_i = Z_i - \mathbb{E}[Z_i]$ and $\bar{X} = \frac{1}{m} \sum_i X_i$. Using the monotonicity of the exponent function and Markov's inequality, we have that for every $\lambda > 0$ and $\epsilon > 0$,

$$\mathbb{P}[\bar{X} \geq \epsilon] = \mathbb{P}[e^{\lambda \bar{X}} \geq e^{\lambda \epsilon}] \leq e^{-\lambda \epsilon} \mathbb{E}[e^{\lambda \bar{X}}].$$

Using the independence assumption we also have

$$\mathbb{E}[e^{\lambda \bar{X}}] = \mathbb{E}\left[\prod_i e^{\lambda X_i/m}\right] = \prod_i \mathbb{E}[e^{\lambda X_i/m}].$$

By Hoeffding's lemma (Lemma B.7 later), for every i we have

$$\mathbb{E}[e^{\lambda X_i/m}] \leq e^{\frac{\lambda^2(b-a)^2}{8m^2}}.$$

Therefore,

$$\mathbb{P}[\bar{X} \geq \epsilon] \leq e^{-\lambda \epsilon} \prod_i e^{\frac{\lambda^2(b-a)^2}{8m^2}} = e^{-\lambda \epsilon + \frac{\lambda^2(b-a)^2}{8m}}.$$

Setting $\lambda = 4m\epsilon/(b-a)^2$ we obtain

$$\mathbb{P}[\bar{X} \geq \epsilon] \leq e^{-\frac{2m\epsilon^2}{(b-a)^2}}.$$

Applying the same arguments on the variable $-\bar{X}$ we obtain that $\mathbb{P}[\bar{X} \leq -\epsilon] \leq e^{-\frac{2m\epsilon^2}{(b-a)^2}}$. The theorem follows by applying the union bound on the two cases. \square

LEMMA B.7 (Hoeffding's lemma) *Let X be a random variable that takes values in the interval $[a, b]$ and such that $\mathbb{E}[X] = 0$. Then, for every $\lambda > 0$,*

$$\mathbb{E}[e^{\lambda X}] \leq e^{\frac{\lambda^2(b-a)^2}{8}}.$$

Proof Since $f(x) = e^{\lambda x}$ is a convex function, we have that for every $\alpha \in (0, 1)$, and $x \in [a, b]$,

$$f(x) \leq \alpha f(a) + (1 - \alpha)f(b).$$

Setting $\alpha = \frac{b-x}{b-a} \in [0, 1]$ yields

$$e^{\lambda x} \leq \frac{b-x}{b-a}e^{\lambda a} + \frac{x-a}{b-a}e^{\lambda b}.$$

Taking the expectation, we obtain that

$$\mathbb{E}[e^{\lambda X}] \leq \frac{b - \mathbb{E}[X]}{b-a}e^{\lambda a} + \frac{\mathbb{E}[x] - a}{b-a}e^{\lambda b} = \frac{b}{b-a}e^{\lambda a} - \frac{a}{b-a}e^{\lambda b},$$

where we used the fact that $\mathbb{E}[X] = 0$. Denote $h = \lambda(b-a)$, $p = \frac{-a}{b-a}$, and $L(h) = -hp + \log(1 - p + pe^h)$. Then, the expression on the right-hand side of the above can be rewritten as $e^{L(h)}$. Therefore, to conclude our proof it suffices to show that $L(h) \leq \frac{h^2}{8}$. This follows from Taylor's theorem using the facts: $L(0) = L'(0) = 0$ and $L''(h) \leq 1/4$ for all h . \square