A Discussion on Learning Theory

Dylan Chou,
dvchou

What is Machine Learning Theory, or Computational Learning Theory? *Computational Learning Theory* is a field focused on understanding the resources, such as data and information, required to learn various tasks using some computational process. The importance in learning theory comes from providing guarantees of algorithms or determining how confident we are in the predictions of a learner, for instance. These notes will cover well-posed learning problems by defining learners and classifiers and presenting paradigms of learning. Then we look into PAC learning where we define the *probably approximately correct* learning model along with some theorems with respect to the error and accuracy of hypothesis functions. It's later defined what it means for a hypothesis space to be PAC learnable. We also explore the minimum number of training examples to achieve a specified error and accuracy along with criticisms to PAC learning.

1 Well-Posed Learning Problems

Definition 1.1 (Learner and Classifier) A *learner* is a computer program P_L that receives input data in the form of $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where we have n input, output pairs $\mathbf{x}_i, y_i$ and $\mathbf{x}_i$ is a feature vector of $dim(\mathbf{x}_i)$ variables that may be useful in predicting the corresponding outcome y_i, and outputs a classifier. The *classifier* is a program P_C that takes in a new feature vector of size $dim(\mathbf{x}_i)$, the same size as the feature vectors it was trained on and outputs a prediction y_i based on input $\mathbf{x}_i$.

Definition 1.2 (A Well-Posed Learning Problem). A *well-posed* learning problem is defined as $\langle T, M, E\rangle$, where T is a task, M is a measure of performance from the learner, and E is experience. An algorithm or computer program P *learns* if its performance at task T, measured by performance measure M, improves with experience E.

This was Tom Mitchell's definition that he presented in 1997.

Example 1.3 (Well-Posed Learning Example 1). Consider the problem of classifying spam emails. Then task T = classifying spam emails. M = hamming loss or the proportion of emails incorrectly predicted by the classifier. *Note that other measures could be the number of correctly predicted emails or precision and recall of the predictions.* E = process of being exposed to labeled spam or non-spam emails being classified.

Example 1.4 (Well-Posed Learning Example 2) When designing autonomous vehicles, the task T = driving on roads without the presence of a human driver. M = average distance driven before the autonomous vehicle errs based on human judgement. E = process of being exposed to images received in sequential order when the vehicle is driven by a human driver.

Example 1.5 (Well-Posed Learning Example 3). In facial recognition systems, the task T is to predict whose face is present in an image. The face in an image is associated with a person. M = number of faces accurately predicted. E = process of passing in image data of faces for the recognition system to identify wide swaths of faces.

Example 1.6 (Well-Posed Learning Example 4) In the setting of chess, the task T is to play and win a game of chess against an opponent. M = proportion of chess games won by the program. E = process of training the program to play against itself, where the learner is updated as more games are played.

Definition 1.7 (Inductive Bias). The *inductive bias* of a learning algorithm is the set of assumptions that a learner uses to carry out predictions. Inductive bias is also known as the hypothesis class, which could be a type of function.

Example 1.8 (Inductive Bias Example 1). Suppose that we have an axis-aligned rectangle in $\mathbb{R}^2$ that classifies examples inside of it as "positive" examples while points outside of it are "negative" examples. The hypothesis class or inductive bias here would be the set of axis-aligned rectangles in $\mathbb{R}^2$.

Definition 1.9 (Generalizability). When a learner *memorizes*, it commits to memory the exact data it is given. Namely, the learner takes in data $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$ and the stored data becomes part of the classifier. The learner in Algorithm 1 stores the input data while the classifier in Algorithm 2 either returns the output from memory or returns a random output:

Algorithm 1.

```
procedure $p_L(D)$
store D
```

Algorithm 2.

```
procedure $p_C(x)$
    if $(\exists x_i \in D)(x_i == x)$ then
        return $y_i$
    else
        return rand($y \in Y$)
```

When a learner *generalizes*, it's able to use the data it is given to formulate answers on unseen examples of task T, unlike the memorizer that doesn't know what to return on unseen examples and instead gives back a random y.

Definition 1.10 (Representation). The *representation* of the learner needs to be expressed in a language understandable by a computer. A learner takes in an input vector of n training examples $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$, where $\mathbf{x}_i = (x_{i_1}, ..., x_{i_d})$ can be a vector of continuous, discrete, or both types of d variable quantities. Assuming the feature vectors are all the same length and there's no vector with missing values, we can view the n input training examples as an input design matrix $X : n \times d$ where each row is a training example and each column corresponds to one of the d variables.

Definition 1.11 (Evaluation). An evaluation, or objective, function quantifies how much better one classifier p_C is than another $p_{C'}$.

Example 1.12 (Evaluation Example 1). Suppose that we are given n images and want to classify whether an image is that of a cat or not. Given that we already had a learner p_L output classifier p_C, we would evaluate the predictions of the n images based on some expression of the *loss* for each image. Loss $l(y, \widehat{y}) = \begin{cases} 0 & y = \widehat{y} \\ 1 & \text{otherwise} \end{cases}$. This loss would be on a single image and is known as zero-one loss. When we sum over losses of each of the n images, we can average the losses in some way by the expectation $E_{(\mathbf{x},y)\sim D}[l(y, f(\mathbf{x}))]$. Here we assume that the data comes from some data generating process D and the expectation expression would be the *expected loss*.

Definition 1.13 (Optimization). Among the many possible classifiers used to solve a task T, we're often interested in the highest performing classifier based on an evaluation function. *Optimization* is the procedure of reaching an optimal classifier through techniques such as Newton's Method, Gradient Descent, or Greedy Search. To achieve an optimal or sub-optimal classifier, suppose that our learner produces a classifier that's based on a set of k parameters: $param_1, param_2, ...param_k$. We could plot the objective function that can be expressed with respect to these k parameters $f(param_1, param_2, ...param_k)$ and update a starting guess of parameters by subtracting parameters by some fraction of their gradient, following the path of steepest descent (to minimize *expected loss* of the classifier).

2 PAC Learning

Definition 2.1 (PAC Learning Model Setting). In the *probably approximately correct* learning model, we have an input space X with a data-generating process D, or distribution, that is unknown but fixed. The learner receives a set S of m training instances $x_1, ..., x_m$ that are sampled from D. A *concept class* C is a set of *concepts* c, which are mappings $c : X \to Y$ that take in a set of examples X and returns a binary outcome in $Y = \{0, 1\}$ or $Y = \{+, -\}$ to classify each example. A *target function* c_t, or some denote as c^*, is a function that labeled the input training data $\mathbf{x}_i \in X$. In other words, a target function is the know-all oracle that would return the correct corresponding y_i for any $\mathbf{x}_i$. We assume that our *target function* is in the set of concept classes C. Then our training examples are given as the set $\{(\mathbf{x}_i, c_t(\mathbf{x}_i))\}$, where c_t is the oracle or target function used to label the outcome of each input $\mathbf{x}_i$.

We want the PAC learner to perform well whatever the distribution D is. D is arbitrary. But D would be the same, so if the true distribution doesn't put much weight in specific regions, the learner won't pay as much attention to those.

The PAC learner only sees input-output behavior of the target concept. The time to write down representation of concept is a lower bound on runtime. **Representation Scheme** of concept class is function $\mathbb{R} : \Sigma^* \to C$. To understand concept size, we take mapping $\Sigma^* \to N$, where N is a natural number. Σ is an alphabet of symbols.

The *true error* of a hypothesis function h is the error: $err(h) = Pr_{\mathbf{x}\sim p^*(x)} \cdot [c_t(\mathbf{x}) \neq h(\mathbf{x})]$, where $\mathbf{x}$ are values from the population data. This is unlike the data D, which is a sample from the population $p^*(x)$. The true error is also known as *expected risk*. The *training* error is measurable on the data we are given: $\widehat{err}(h) = P_{\mathbf{x}\sim D}(c_t(\mathbf{x}) \neq h(\mathbf{x}))$. In the $+/-$ classification setting, the training error would simplify to be $\frac{1}{n}\sum_{i=1}^{n} \mathbb{I}(y_i \neq h(\mathbf{x}_i))$ given that $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$.

Definition 2.2 (PAC Criterion). The PAC criterion states that a PAC learner will yield a hypothesis function $h \in H$ that is approximately correct $R(h) \approx 0$ with high probability $p(R(h) \approx 0) \approx 1$: $P(|R(h) - \widehat{R}(h)| \leq \epsilon) \geq 1 - \delta$, where $\epsilon, \delta > 0$ are considered to be small.

Theorem 3. (No Free Lunch Theorem). For any $m \in \mathbb{N}$, consider the domain X of size $2m$. Consider an arbitrary algorithm A that outputs a hypothesis $h \in H$ given a data sample S. Then there exists a concept $f : X \to \{0, 1\}$ and distribution D such that

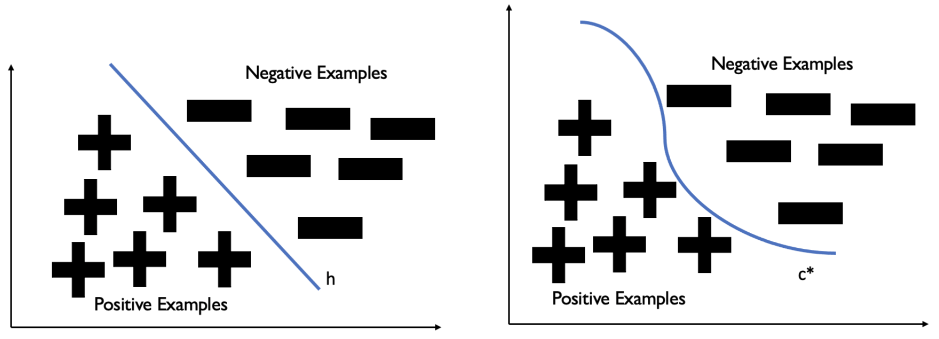
1. $err(f) = 0$
2. With probability at least $\frac{1}{10}$, $err(A(S)) \geq \frac{1}{10}$. *Note that* $\epsilon = \frac{1}{10}, \delta = \frac{1}{10}$ *here.*

Proof. WTS that for any learner, there exists some concept that will not learn well. Take D to be the uniform distribution over $\{x, f(x)\}$ examples. Consider the inequality for expected error: $Q = E_{f:X\to\{0,1\}}[E_{S\sim D^m}[err(A(S))]] \geq \frac{1}{4}$ where we sample m examples from D to form S. By Fubini's theorem, we can swap the two expectations and condition on the event that $x \in S$:
$Q = E_S[E_f[E_{x\in X}[A(S)(x) \neq f(x)]]]$
$= E_{S,x}[E_f[A(S)(x) \neq f(x)|x \in S]]P(x \in S) + E_{S,x}[E_f[A(S)(x) \neq f(x)|x \notin S]]P(x \notin S)$
by law of total probability.

$E_{S,x}[E_f[A(S)(x) \neq f(x)|x \in S]]P(x \in S) \geq 0$ because the chance that $x \in S$ may be 0 if the x is simply not in S. Note that $P(x \notin S) \geq \frac{1}{2}$ because we're sampling m examples from D^m in a domain of $2m$. And so $P(A(S)(x) \neq f(x)) = \frac{1}{2}$ for all $x \notin S$ because we are given that the true concept is chosen uniformly at random. Hence, we obtain $Q \geq \frac{1}{2}^2 = \frac{1}{4}$. By reverse Markov's inequality, $P(Q \geq \frac{1}{10}) \geq \frac{1/4-1/10}{1-1/10} \geq \frac{1}{10}$. The inequality is swapped by considering the random variable $\tilde{Q} = 1 - Q$ so that $E[Q \geq \frac{1}{10}] \geq \frac{E[Q]-\frac{1}{10}}{1-\frac{1}{10}}$. □

Exercise 2.3 (Hypothesis and Concept Functions). *A hypothesis function* h is an output from a learner p_C defined under definition 1.1. These functions are guesses at the true target function c_t that predict y_i for corresponding $\mathbf{x}_i$. The set of different hypothesis functions is the *hypothesis space* H where $h \in H$.

Solution: Are hypothesis and concept functions always the same? Not always. Suppose that the target or concept function c_t is a non-linear function that classifies $+$ or $-$ examples. If we fix our hypothesis space H of possible classifiers to be linear functions in the form of $y_i = \beta_0 + \beta_1 x_i$, our h hypothesis function would never be the same as the target function c^*, or c_t in other cases, because we're restricted to linear functions.



Exercise 2.4 (c_t oracle/concept/target function). Are there instances where c_t is known?
Solution: No. We assume in our learning model that the target function c_t is unknown because this would entail perfect classification.

Definition 2.5 (PAC Learnable). If N being some collection of training examples. If N is finite for some learning algorithm A, then hypothesis space H is *learnable*. If N is a polynomial function of $\frac{1}{\epsilon}$ and $\frac{1}{\delta}$ for some learning algorithm, H is *PAC learnable*.

Definition 2.6 (Efficiently PAC Learnable). Hypothesis space H is efficiently PAC learnable if it's PAC learnable and

1. The number of training examples that a learning algorithm A takes in is bounded by some polynomial of $n, \frac{1}{\epsilon}, and \frac{1}{\delta}$.
2. The algorithm runs in poly-time bounded by $n, \frac{1}{\epsilon}, \frac{1}{\delta}$.

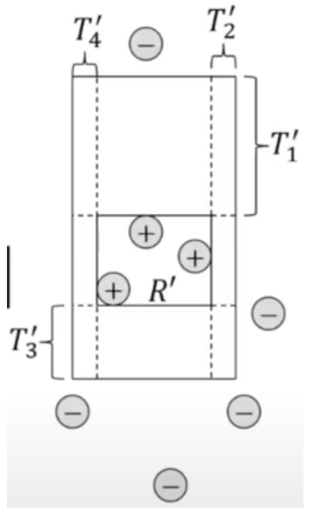
Theorem 4. Consider the scenario where we have two players: Player 1 and Player 2. Player 1 thinks of an interval $[a, b]$ on the real number line. Player 1 samples m examples forming S from D^m independent and identically distributed numbers $x_i \in \mathbb{R}$ from distribution D. Then Player 1 labels x_i with y_i if it's included in the interval $[a, b]$. Player 2 is then given the m intervals and needs to learn the original $[a, b]$ interval that Player 1 thought of. Intervals are PAC-learnable.

Proof. Assume the algorithm A and choose the interval $H = [a, b]$ where a, b are real numbers. Suppose we have a sample of m points $x_1, ..., x_m$ along with their corresponding labels of being in the interval H or not: $y_1, ..., y_m$. Define $a_1 = min\{x_i : c(x_i) = 1\}$ and $b_1 = max\{x_i : c(x_i) = 1\}$. Then our output would be $A(S) = [a_1, b_1]$.

Let $c = [a_0, b_0]$ be the target interval in H and D be a distribution over the domain R. Our chosen interval H is within the target interval c. Then the true error is
$P_{x \in D}(x \in [a_0, a_1) \cup (b_1, b_0]) \leq P_{x \in D}(x \in [a_0, a_1)) + P_{x \in D}(x \in (b_1, b_0])$ by the union bound. Define $[a_0, a_1^{'})$ to be the interval such that the chance of sampling a real number from $[a_0, a_1^{'})$ over D is $\epsilon/2$. If $[a_0, a_1^{'}) \subset [a_0, a_1)$, then there would be a positive example not included in the interval $[a_0, a_1^{'})$ and the chance of this occurring is $1 - \epsilon/2$ by construction. In the worst case, if none of our examples were in $[a_0, a_1^{'})$, this would occur with probability $(1 - \epsilon/2)^m$. Then $P[[a_0, a_1^{'}) \subset [a_0, a_1)] \leq (1 - \epsilon/2)^m$ assuming that all x_i were not in $[a_0, a_1^{'})$. The same case goes for B, where we construct an interval $(b_1^{'}, b_0]$ that has an $\epsilon/2$ chance of being sampled from. Then $P_{x \in D}[x \in [a_0, a_1) \cup (b_1, b_0]] \leq 2(1-\epsilon/2)^m$. For $2(1 - \epsilon/2)^m \leq \delta$, we would use $(1 - x) \leq e^{-x}$ and get $2(1 - \epsilon/2)^m \leq 2e^{-m\epsilon/2} \leq \delta$. We have that $log(2/\delta) \leq m\epsilon/2 \implies \frac{2}{\epsilon}log(\frac{2}{\delta}) \leq m$ and intervals are PAC-learnable. □

Theorem 5. Consider the learning algorithm where, given a training set, we return the tightest fit of an axis-aligned (horizontal-vertical) rectangle for positive examples. The concept class of axis-aligned rectangles over the Euclidean space $\mathbb{R}^2$ is PAC-learnable.

Proof. Consider the graphic below where the target rectangle R is the "oracle" true function that labels the points. However, our algorithm returns the tightest fit of the positive examples R'. Let $w(E) = Pr_{x \sim D}[x \in E]$ for a region E. Then the $err(R) = w(R \setminus R')$ because the probability of the target rectangle classifying an example differently from the tightest fit rectangle is the probability that the example is in the region $R \setminus R'$. The region is where R' would classify examples as negative while R would classify them as positive in its interior. We consider breaking up the rectangle into four strips $T_1^{'}, T_2^{'}, T_3^{'}, T_4^{'}$ and the $err(R') \leq \sum_{i=1}^{4} w(T_i^{'})$ because the strips overlap. We estimate $P[w(T_i^{'})] \geq \epsilon/4$. WLOG, consider $T_1^{'}$. We can define T_1 to be a portion of the height of $T_1^{'}$ so $w(T_1^{'}) \geq \epsilon/4 \iff T_1 \subseteq T_1^{'} \iff x_1, ...x_m \notin T_1$. In turn, $P[w(R \setminus R') \geq \epsilon] \leq 4(1 - \epsilon/4)^m$. This is bounded by ϵ since the probability of a bad event is considered to be at most δ and probability of a good event is at least $1 - \delta$. Using $1 - x \leq e^{-x}$, we get $4(1 - \epsilon/4)^m \leq 4e^{-m\epsilon/4}$ to obtain $4e^{-m\epsilon/4} \leq \delta \implies 4/\delta \leq e^{m\epsilon/4} \implies log(\frac{4}{\delta}) \leq m(\epsilon/4) \implies m \geq (\frac{4}{\epsilon})log(\frac{4}{\delta})$. □



Theorem 6. All finite hypothesis classes/spaces H are PAC-learnable.

Proof. Take m examples to form S from D^m. An *empirical risk minimization* algorithm is defined to be one that returns the hypothesis function
$\widehat{h} = argmin_{h \in H} \widehat{R}(h) = argmin_{h \in H} E_{(\mathbf{x},y) \sim S}[l(h(x), y)]$ with the lowest training error. The expectation is the expected value of the loss between the classified $\widehat{y}$ using $h(x)$ and the true label y. Suppose that we have a hypothesis $\widehat{h}$ where $err(\widehat{h}) \geq \epsilon$. We will show that it's unlikely for our learner to output a classifier with greater error than ϵ. For all our m examples to be correctly classified and the hypothesis to be an empirical risk minimizer, $P(err_S(\widehat{h}) = 0) \leq (1-\epsilon)^m$ where each one of the m examples is not erred and this is also unlikely. For our empirical risk minimization algorithm A to return a "bad hypothesis" ($err(\widehat{h}) > \epsilon$), this would mean that $P(err(h_A) > \epsilon)$
$\leq P(\exists \widehat{h}, err(\widehat{h}) > \epsilon, err_S(\widehat{h}) = 0)$ which is the probability of there being at least one bad hypothesis.
$\leq \sum_{\widehat{h}: err(\widehat{h} > \epsilon)} P(err(\widehat{h}) = 0) \leq (1-\epsilon)^m |H|$ if we assume that every hypothesis is bad. This is a very loose upper bound, but we know that $(1-\epsilon)^m |H| \leq e^{-\epsilon m}|H| \leq \delta$. Moving around the terms, $\frac{|H|}{\delta} \leq e^{\epsilon m} \implies log(\frac{|H|}{\delta}) \leq \epsilon m \implies m \geq \frac{log(\frac{|H|}{\delta})}{\epsilon}$, so finite hypothesis spaces are PAC-learnable. □

Definition 2.7 (Sample Complexity). The *sample complexity* is the minimum number of training examples N where the PAC criterion is satisfied with respect to δ and ϵ. In other words, this is the minimum number of training examples to allow for a classifier to achieve less than ϵ error with high probability of at least $1 - \delta$.

Definition 2.8 (A Consistent Hypothesis). A hypothesis function $h \in H$ is consistent with the training data if $\widehat{R}(h) = 0$.

Definition 2.9 (A Consistent PAC Learner or Algorithm). An algorithm is consistent if for each ϵ, δ, there is a positive number of training examples N such that for any distribution p^*, we have that $P(|R(h) - \widehat{R}(h)| > \epsilon) < \delta$.

Definition 2.10 (Types of Problems for Sample Complexity Results). For sample complexity, we inspect four different possible problems defined by whether we expect the target function to be in the hypothesis space and the size of the hypothesis space: c_t can be realizable ($c_t \in H$) or agnostic ($c_t \in H$ or $c_t \notin H$). We can also have that $|H|$ is either finite or infinite. Below are the theorems defined for the sample complexities in the four possible cases with respect to c_t and $|H|$.

	Realizable	Agnostic
Finite $\lvert\mathcal{H}\rvert$	**Thm. 1** $N \geq \frac{1}{\epsilon}\left[\log(\lvert\mathcal{H}\rvert) + \log(\frac{1}{\delta})\right]$ labeled examples are sufficient so that with probability $(1-\delta)$ all $h \in \mathcal{H}$ with $\hat{R}(h) = 0$ have $R(h) \leq \epsilon$.	**Thm. 2** $N \geq \frac{1}{2\epsilon^2}\left[\log(\lvert\mathcal{H}\rvert) + \log(\frac{2}{\delta})\right]$ labeled examples are sufficient so that with probability $(1-\delta)$ for all $h \in \mathcal{H}$ we have that $\lvert R(h) - \hat{R}(h)\rvert \leq \epsilon$.
Infinite $\lvert\mathcal{H}\rvert$	**Thm. 3** $N{=}O(\frac{1}{\epsilon}\left[\mathsf{VC}(\mathcal{H})\log(\frac{1}{\epsilon}) + \log(\frac{1}{\delta})\right])$ labeled examples are sufficient so that with probability $(1-\delta)$ all $h \in \mathcal{H}$ with $\hat{R}(h) = 0$ have $R(h) \leq \epsilon$.	**Thm. 4** $N = O(\frac{1}{\epsilon^2}\left[\mathsf{VC}(\mathcal{H}) + \log(\frac{1}{\delta})\right])$ labeled examples are sufficient so that with probability $(1-\delta)$ for all $h \in \mathcal{H}$ we have that $\lvert R(h) - \hat{R}(h)\rvert \leq \epsilon$.

Exercise 2.11 (Realizable and Finite $|H|$). Suppose that we have H = class of conjunctions over $\mathbf{x} \in \{0,1\}^N$. The hypothesis functions can take on the form of, for instance, $h(\mathbf{x}) = x_1(1-x_2)...(x_n)$. If we fix the input vector, or "booleans" (can either be 0 or 1), to have length 10, then how many examples would suffice to achieve an error less than 0.1 with high probability of at least 99%?

Solution: We harp on Theorem 1 in the realizable and finite case that can provide the minimum number of labeled examples.
$N \geq \frac{1}{0.1}(log(3^{10}) + log(\frac{1}{0.01}))$
$= 10(10log(3) + log(100)) \approx 156$ because the hypothesis space of all possible hypothesis functions is the number of combinations of excluding, including x_i or including $1 - x_i$ for any given x_i in the conjunction. There would be 3^{10} possible hypothesis functions as each variable has three possibilities of being or not being in the hypothesis function, as described earlier.

Theorem 7. In the scenario where our hypothesis space H is finite and our learning problem is realizable, $N \geq \frac{1}{\epsilon}(log(|H|) + log(\frac{1}{\delta}))$, where N is the minimum number of examples to train our classifier and ensure that with high probability $1-\delta$, all $h \in H$ with $\widehat{err}(h) = 0$ have $err(h) \leq \epsilon$.

Proof. Suppose that we have N training examples forming S sampled from D^N. Assume that there are k bad hypotheses $\widehat{h}_1, ..., \widehat{h}_k$ where $err(\widehat{h}_i) > \epsilon$. Suppose that we pick one of these bad hypotheses and get the probability that hypothesis function h_i is an empirical risk minimizer (defined in Theorem 6) and has $P(err_S(\widehat{h}_i) = 0) \leq (1-\epsilon)^N$. Then the probability of at least one bad hypothesis with 0 training error is less than $k(1-\epsilon)^N$ over the k assumed bad hypotheses. But this is also less than $(1-\epsilon)^N|H|$ if we assumed every hypothesis function were a bad hypothesis. Using $(1-x) \leq e^{-x} \implies (1-\epsilon)^N|H| \leq e^{-\epsilon N}|H|$, which is to be less than δ. We solve for N: $\frac{|H|}{\delta} \leq e^{\epsilon N} \implies \frac{1}{\epsilon}log(\frac{|H|}{\delta}) \leq N \implies N \geq \frac{1}{\epsilon}(log(|H|) + log(\frac{1}{\delta}))$. Then there is a hypothesis where $err(\widehat{h}) > \epsilon$ and $err_S(\widehat{h}) = 0$ (a bad hypothesis) with probability δ. With probability $1-\delta$, for all $h \in H$, classifiers with true error $err(\widehat{h}) > \epsilon$ have training error $err_S(\widehat{h}) > 0$, or by contrapositive that all $h \in H$ with training error $err_S(\widehat{h}) = 0$ have true error $err(\widehat{h}) \leq \epsilon$. □

Definition 2.12 (When H is infinite). The minimum number of training examples to satisfy the PAC criterion is intuitive for finite hypothesis spaces. In the scenario where we have an infinite hypothesis space, we introduce the *VC-dimension*. VC-dimension of a classifier is the maximum number of points arranged so that it may shatter them. A set of points S is shattered by hypothesis space H if there are hypotheses in H that split S in all of $2^{|S|}$ possible ways. Namely, all possible ways of classifying points in S are achievable using concepts in H.

Note that to show the VC-dimension of the hypothesis space H is d, we need to show:

1. *$\exists C$ s.t. $|C| = d$ can be shattered by H*
2. *$\forall C$ s.t. $|C| = d+1$ cannot be shattered by H.*

Exercise 2.13 (VC-dimension of Rectangles). What is the VC-dimension of the axis-aligned rectangle classifiers defined in Theorem 4?

Solution: We can find the VC-dimension by observing the maximum number of points that axis-aligned rectangles can shatter. $+/-$ points may be oriented in a diamond shape, where there's a point at the top, bottom, left and right. This arrangement would allow an axis-aligned rectangle to perfectly classify any assignment of points. However, after adding in another point in the middle, the rectangle would not be able to label the points if the four diamond points were positive, but a fifth middle point were negative. Then the fifth point is internal and we wouldn't be able to perfectly classify five points. The VC-dimension is 4.

Exercise 2.14 (VC-dimension of intervals along $\mathbb{R}$ number line). Suppose that we have the real number line and can only use one interval to classify points as $+$ if within the interval or $-$ if outside of it. What is the VC-dimension of the interval classifier?

Solution: We would be able to perfectly classify 2 points as an interval can encompass none, one of the points, or both (if positive) and accurately $+/-$ examples. However, if another point is added, we can have the configuration of $+, -, +$ and never perfectly classify points using an interval where the examples inside the interval are deemed $+$ and those outside are deemed $-$ because we have discontinuity between positive examples that are separated by a negative example. Then the VC-dimension of the interval classifier is 2.

Exercise 2.15 (Many Intervals). Suppose that we modified Exercise 2.14 so that we may classify using many intervals. What would be the VC-dimension of the multiple interval classifier?

Solution: We can perfectly classify any set of points in the interval. In the overfitted extreme case, we could encompass each positive example with its own interval and allow for perfect classification that way. Then the VC-dimension of the multiple interval classifier would be ∞.

Theorem 8. Let H be a hypothesis space of hypothesis functions $h : X \to \{0, 1\}$ where $VC(H) = \infty$ and we're using the $0 - 1$ loss defined in Example 1.12. Show that H is not PAC-learnable.

Proof. Assume for sake of contradiction that H is PAC learnable. Then there is an algorithm A s.t. $\forall \epsilon, \delta > 0$, $\exists M(\epsilon, \delta)$ (some expression of ϵ, δ) s.t. if $m > M(\epsilon, \delta)$, then for all distributions D,
$P_{S\sim D^m}(err_D(A(S)) > err_D(h^*)+\epsilon) < \delta$ where the hypothesis function h^* is the one that minimizes the true error err_D.

Assume for sake of contradiction that A exists. We select $\epsilon < \frac{1}{8}, \delta < \frac{1}{7}$ s.t. $m > M(\epsilon, \delta)$. Because the VC-dimension is infinite, there exists some $x_1, ...x_{2m} \in X$ that H shatters where X is the domain of the hypothesis function. By Theorem 3, we know there exists a distribution D over $x_1, ...x_{2m}$ and corresponding labels such that there is some function $f : X \to \{0, 1\}$ where $err(f) = 0$ and $P_{S\sim D^m}[err_D(A(S)) \geq \epsilon] > \delta \implies P_{S\sim D^m}[err_D(A(S)) \geq \frac{1}{8}] > \frac{1}{7}$. The distribution is supported only over the $2m$ points and it's shattered by H, so $err_D(h^*) = 0$ by the property that being shattered by H means perfect classification. This means that $P_{S\sim D^m}[err_D(A(S)) > err_D(h^*) + \epsilon] \geq P_{S\sim D^m}[err_D(A(S)) > \frac{1}{8}] > \frac{1}{7} > \delta$, which contradicts our original claim that $P_{S\sim D^m}(err_D(A(S)) > err_D(h^*) + \epsilon) < \delta$ by the definition of PAC-learnable. Then H with $VC(H) = \infty$ is not PAC-learnable. □

3 Criticisms of PAC Learning

PAC learning considers the *worst-case scenario* where the sample complexity of a problem is defined to be the minimum number of examples to satisfy the PAC criterion. However, this may overestimate the real complexity of a learning algorithm A. This is evident in the proofs in these notes where we often considered a very loose upper bound that could overestimate the expression $m > M(\epsilon, \delta)$ where m is the number of training examples. Consider the scenario where we try to predict learning curves, which is the ratio of accuracy to the number of samples. This is because PAC learning theory tends to overestimate the number of samples needed to attain some level of accuracy.

Some solutions to remedy the overestimation problem in PAC learning can come from distribution-specific models, a bayesian approach, or the "probability of mistake" that's an out-of-sample error where we inspect the probability that after training on m examples forming sample $S \sim D^m$, the algorithm errs on the next sample $S^{'} \sim D^m$.

4 Check Your Understanding

1. What is a well-posed learning problem? *Answer: A well-posed learning problem is defined by a task T, measure of performance M and experience E. The well-posed learning problem defines a way that a program can learn: an algorithm or computer program P learns if its performance at task T, measured by performance measure M, improves with experience E.*
2. Give five examples of well-posed learning problems. *Answer: A well-posed learning problem, as stated earlier in the notes, can be among the four examples (1.3,1.4,1.5,1.6). Another example can be part of speech (POS) tagging on textual data. The task would be T=tagging each word with a POS tag, M=proportion of words in a document correctly tagged, and E=the different sentences in documents that a learner tags and is exposed to.*
3. What could the representation of a learner be for data that contains all binary categorical variables, including the output? *Answer: One possibility can be to store the information from the data in a recursive tree datatype. Each node would contain a subset of the input data $D = \{(x_i, y_i)\}_{i=1}^n$. From a node, there would be two branches that denote the binary possibilities of a variable that the node splits on. SPS we have $x_i = 0$ as the left branch of the tree root. Then the new node would contain the subset of data where $x_i = 0$. The subsets of data at each node are defined by the branching in the tree.*
4. True/False: A hypothesis function is always the same as a concept function. *Answer: False. See Exercise 2.3.*
5. What does it mean for a hypothesis class to be PAC learnable? *Answer: See definition 2.5.*
6. True/False: All finite hypothesis classes H are PAC-learnable. *Answer: True. See the proof of Theorem 6.*
7. True/False. Axis-aligned rectangles are PAC-learnable. *Answer: True. We considered splitting up the region outside of the tightest fitting rectangle but inside the target rectangle into four regions and define an upper bound on the error of the tightest fitting rectangle as an expression of an example landing in any one of the four regions. We then used the inequality $1 - x \leq e^{-x}$ to complete the proof for Theorem 5.*
8. What is a consistent learner? *Answer: See definition 2.9*
9. What is the VC-dimension of an axis-aligned rectangle? *Answer: See Exercise 2.13*
10. When the size of the hypothesis space is infinite, how can we express the sample complexity in those situations? *Answer: We use the VC-dimension. See definition 2.11.*

11. What is an example of a hypothesis space H that is not PAC-learnable? *Answer: The multiple interval classifier defined in Exercise 2.15 is not PAC-learnable as the VC-dimension is ∞. For the proof of $VC(H) = \infty \implies$ not PAC-learnable, see Theorem 8.*

12. What is problematic about PAC learning? *Answer: PAC learning often overestimates the sample complexity, or minimum number of examples to attain the PAC criterion, as seen in the PAC-learnable proofs above where we had very loose bounds.*

5 Acknowledgements

1. The definition of Computational Learning Theory was derived from this talk.
2. Discussion on learners and classifiers was based on this review article
3. The definition of well-posed learning problems harps on Tom Mitchell's original definition here.
4. The definition of inductive bias was repurposed from here and here
5. Perspectives on what learning is composed uses ideas specified here.
6. Concept class definition from here
7. PAC Learning Model Definition was from Track C video 4.
8. The "No Free Lunch" Theorem from here.
9. Efficiently PAC learnable definition here.
10. Theorem 4 referenced from here.
11. Theorem 5 referenced from the Track C video 4.
12. Theorem 6 referenced from here.
13. Theorem 7 results from here.
14. Theorem 8 ideas harped from here.
15. Sample Complexity Results and the number of examples sufficiency is from this lecture.
16. Exercises on VC-dimension are from the Track C video 4.
17. Criticisms of PAC Learning came from results in the notes and here.