Why Learning Theory?

- How can we tell if our learning algo will do a good job in future (test time)?

Experimental results
Theoretical analysis

Why theory?
- Can only run a limited number of experiments.
- Experiments rarely tell us what will go wrong.

Want to deploy our learning algorithms on Mars.

Using learning theory, we can make formal statements/give guarantees on:
- Expected performance ("generalization") of a learning algorithm on test data.
- Number of examples required to attain a certain level of test accuracy.
- Hardness of learning problems in general.

"Theory is the first term in the Taylor series expansion of Practice" - T. Cover.

Machine Learning (CS771A)
Introduction to Learning Theory
Why Learning Theory?

- How can we tell if our learning algo will do a good job in future (test time)?
  - Experimental results
Why Learning Theory?

- How can we tell if our learning algo will do a good job in future (test time)?
  - Experimental results
  - Theoretical analysis
Why Learning Theory?

- How can we tell if our learning algo will do a good job in future (test time)?
  - Experimental results
  - Theoretical analysis
- Why theory?
Why Learning Theory?

- How can we tell if our learning algo will do a good job in future (test time)?
  - Experimental results
  - Theoretical analysis

- Why theory?
  - Can only run a limited number of experiments..

Using learning theory, we can make formal statements/give guarantees on expected performance ("generalization") of a learning algorithm on test data, number of examples required to attain a certain level of test accuracy, and hardness of learning problems in general.

"Theory is the first term in the Taylor series expansion of Practice" - T. Cover
Why Learning Theory?

- How can we tell if our learning algo will do a good job in future (test time)?
  - Experimental results
  - Theoretical analysis

- Why theory?
  - Can only run a limited number of experiments.
  - Experiments rarely tell us what will go wrong

Using learning theory, we can make formal statements/give guarantees on:

- Expected performance ("generalization") of a learning algorithm on test data
- Number of examples required to attain a certain level of test accuracy
- Hardness of learning problems in general

"Theory is the first term in the Taylor series expansion of Practice" - T. Cover
Why Learning Theory?

- How can we tell if our learning algo will do a good job in future (test time)?
  - Experimental results
  - Theoretical analysis

- Why theory?
  - Can only run a limited number of experiments..
  - Experiments rarely tell us what will go wrong
  - Want to deploy our learning algorithms on Mars
Why Learning Theory?

- How can we tell if our learning algo will do a good job in future (test time)?
  - Experimental results
  - Theoretical analysis

- Why theory?
  - Can only run a limited number of experiments..
  - Experiments rarely tell us what will go wrong
  - Want to deploy our learning algorithms on Mars

- Using learning theory, we can make formal statements/give guarantees on
Why Learning Theory?

- How can we tell if our learning algo will do a good job in future (test time)?
  - Experimental results
  - Theoretical analysis

- Why theory?
  - Can only run a limited number of experiments..
  - Experiments rarely tell us what will go wrong
  - Want to deploy our learning algorithms on Mars

- Using learning theory, we can make formal statements/give guarantees on
  - Expected performance (“generalization”) of a learning algorithm on test data
Why Learning Theory?

- How can we tell if our learning algo will do a good job in future (test time)?
  - Experimental results
  - Theoretical analysis

- Why theory?
  - Can only run a limited number of experiments..
  - Experiments rarely tell us what will go wrong
  - Want to deploy our learning algorithms on Mars

- Using learning theory, we can make formal statements/give guarantees on
  - Expected performance ("generalization") of a learning algorithm on test data
  - Number of examples required to attain a certain level of test accuracy
Why Learning Theory?

- How can we tell if our learning algo will do a good job in future (test time)?
  - Experimental results
  - Theoretical analysis

- Why theory?
  - Can only run a limited number of experiments..
  - Experiments rarely tell us what will go wrong
  - Want to deploy our learning algorithms on Mars

- Using learning theory, we can make formal statements/give guarantees on
  - Expected performance ("generalization") of a learning algorithm on test data
  - Number of examples required to attain a certain level of test accuracy
  - Hardness of learning problems in general
Why Learning Theory?

- How can we tell if our learning algo will do a good job in future (test time)?
  - Experimental results
  - Theoretical analysis

- Why theory?
  - Can only run a limited number of experiments..
  - Experiments rarely tell us what will go wrong
  - Want to deploy our learning algorithms on Mars

- Using learning theory, we can make formal statements/give guarantees on
  - Expected performance ("generalization") of a learning algorithm on test data
  - Number of examples required to attain a certain level of test accuracy
  - Hardness of learning problems in general

“Theory is the first term in the Taylor series expansion of Practice” - T. Cover
A hypothesis class $\mathcal{H}$ is a set of functions/hypotheses (assume finite for now).
Hypothesis Class, Training and True Error

- A hypothesis class $\mathcal{H}$ is a set of functions/hypotheses (assume finite for now)
- The learning algorithm, given training data, learns a hypothesis $h \in \mathcal{H}$

The 0-1 true error (also called the expected error) of $h$

$$L_P(h) = \mathbb{E}_{(x,y) \sim P} \left[ I(h(x) \neq y) \right]$$

The true error, in general, is much worse than the training error

We want to know how much worse it is... without doing experiments.
Hypothesis Class, Training and True Error

- A hypothesis class $\mathcal{H}$ is a set of functions/hypotheses (assume finite for now).
- The learning algorithm, given training data, learns a hypothesis $h \in \mathcal{H}$.
- Assume $h$ is learned using a sample $D$ of $N$ i.i.d. training examples $(x_n, y_n)_{n=1}^N$ drawn from $P(x, y)$; (also denoted as $D \sim P^N$).

The 0-1 training error (also called the empirical error) of $h$

$$L_D(h) = \frac{1}{N} \sum_{n=1}^{N} I(h(x_n) \neq y_n)$$

The 0-1 true error (also called the expected error) of $h$

$$L_P(h) = E_{(x, y) \sim P}[I(h(x) \neq y)]$$

The true error, in general, is much worse than the training error. We want to know how much worse it is... without doing experiments.
A hypothesis class $\mathcal{H}$ is a set of functions/hypotheses (assume finite for now)

The learning algorithm, given training data, learns a hypothesis $h \in \mathcal{H}$

Assume $h$ is learned using a sample $\mathcal{D}$ of $N$ i.i.d. training examples $(x_n, y_n)_{n=1}^N$ drawn from $P(x, y)$; (also denoted as $\mathcal{D} \sim P^N$)

The 0-1 training error (also called the empirical error) of $h$

$$L_D(h) = \frac{1}{N} \sum_{n=1}^N \mathbb{I}(h(x_n) \neq y_n)$$

The true error, in general, is much worse than the training error

We want to know how much worse it is.. without doing experiments
Hypothesis Class, Training and True Error

- A **hypothesis class** $\mathcal{H}$ is a set of functions/hypotheses (assume finite for now)
- The learning algorithm, given training data, learns a hypothesis $h \in \mathcal{H}$
- Assume $h$ is learned using a sample $\mathcal{D}$ of $N$ i.i.d. training examples $(x_n, y_n)_{n=1}^N$ drawn from $P(x, y)$; (also denoted as $\mathcal{D} \sim P^N$)
- The 0-1 training error (also called the **empirical error**) of $h$
  \[
  L_{\mathcal{D}}(h) = \frac{1}{N} \sum_{n=1}^N \mathbb{I}(h(x_n) \neq y_n)
  \]
- The 0-1 true error (also called the **expected error**) of $h$
  \[
  L_P(h) = \mathbb{E}_{(x, y) \sim P}[\mathbb{I}(h(x) \neq y)]
  \]
Hypothesis Class, Training and True Error

- A hypothesis class $\mathcal{H}$ is a set of functions/hypotheses (assume finite for now)
- The learning algorithm, given training data, learns a hypothesis $h \in \mathcal{H}$
- Assume $h$ is learned using a sample $\mathcal{D}$ of $N$ i.i.d. training examples $(x_n, y_n)^N_{n=1}$ drawn from $P(x, y)$; (also denoted as $\mathcal{D} \sim P^N$)
- The 0-1 training error (also called the empirical error) of $h$
  \[ L_{\mathcal{D}}(h) = \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}(h(x_n) \neq y_n) \]
- The 0-1 true error (also called the expected error) of $h$
  \[ L_P(h) = \mathbb{E}_{(x,y) \sim P}[\mathbb{I}(h(x) \neq y)] \]
- The true error, in general, is much worse than the training error
Hypothesis Class, Training and True Error

- A hypothesis class $\mathcal{H}$ is a set of functions/hypotheses (assume finite for now)
- The learning algorithm, given training data, learns a hypothesis $h \in \mathcal{H}$
- Assume $h$ is learned using a sample $D$ of $N$ i.i.d. training examples $(x_n, y_n)_{n=1}^N$ drawn from $P(x,y)$; (also denoted as $D \sim P^N$)
- The 0-1 training error (also called the empirical error) of $h$
  \[ L_D(h) = \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}(h(x_n) \neq y_n) \]
- The 0-1 true error (also called the expected error) of $h$
  \[ L_P(h) = \mathbb{E}_{(x,y) \sim P}[\mathbb{I}(h(x) \neq y)] \]
- The true error, in general, is much worse than the training error
  - We want to know how much worse it is...
  - .. without doing experiments
Case 1: Zero Training Error

- Assume some $h \in \mathcal{H}$ can achieve zero training error

$$P_{D \sim P_N}(L_D(h) = 0 \cap L_P(h) > \epsilon) \leq (1 - \epsilon)^N$$

Let's call $L_D(h) = 0 \cap L_P(h) > \epsilon$ as "$h$ is bad"

Consider $K$ hyp. $\{h_1, \ldots, h_K\}$. Prob. that at least one of these is bad

$$P_{D \sim P_N}(\text{"$h_1$ is bad"} \cup \ldots \cup \text{"$h_K$ is bad"}) \leq K(1 - \epsilon)^N$$

Since $K \leq |\mathcal{H}|$, $K$ can be replaced by the size of set $\mathcal{H}$

$$P_{D \sim P_N}(\exists h: \text{"$h$ is bad"}) \leq |\mathcal{H}|(1 - \epsilon)^N$$

Uniform Convergence
Case 1: Zero Training Error

- Assume some \( h \in \mathcal{H} \) can achieve zero training error
- Assume its true error \( L_P(h) > \epsilon \)
Case 1: Zero Training Error

- Assume some $h \in \mathcal{H}$ can achieve zero training error
- Assume its true error $L_P(h) > \epsilon$
- Probability of $h$ being correct on a single training example $\leq 1 - \epsilon$
Case 1: Zero Training Error

- Assume some \( h \in \mathcal{H} \) can achieve zero training error
- Assume its true error \( L_P(h) > \epsilon \)
- Probability of \( h \) being correct on a single training example \( \leq 1 - \epsilon \)
- Probability of \( h \) having zero error on any training set of \( N \) examples

\[
P_{\mathcal{D} \sim \mathcal{P}_N}(L_D(h) = 0 \cap L_P(h) > \epsilon) \leq (1 - \epsilon)^N
\]
Case 1: Zero Training Error

- Assume some $h \in \mathcal{H}$ can achieve zero training error
- Assume its true error $L_P(h) > \epsilon$
- Probability of $h$ being correct on a single training example $\leq 1 - \epsilon$
- Probability of $h$ having zero error on any training set of $N$ examples

\[ P_{D \sim P_N}(L_D(h) = 0 \cap L_P(h) > \epsilon) \leq (1 - \epsilon)^N \]

- Let’s call $L_D(h) = 0 \cap L_P(h) > \epsilon$ as “$h$ is bad”
Case 1: Zero Training Error

- Assume some \( h \in \mathcal{H} \) can achieve zero training error
- Assume its true error \( L_P(h) > \epsilon \)
- Probability of \( h \) being correct on a single training example \( \leq 1 - \epsilon \)
- Probability of \( h \) having zero error on any training set of \( N \) examples
  \[
P_{D \sim P_N}(L_D(h) = 0 \cap L_P(h) > \epsilon) \leq (1 - \epsilon)^N
\]
- Let’s call \( L_D(h) = 0 \cap L_P(h) > \epsilon \) as “\( h \) is bad”
- Consider \( K \) hyp. \( \{h_1, \ldots, h_K\} \). Prob. that \textbf{at least one} of these is bad
  \[
P_{D \sim P_N}(\text{“} h_1 \text{ is bad”} \cup \ldots \cup \text{“} h_K \text{ is bad”}) \leq K(1 - \epsilon)^N \quad \text{(using union bound)}
\]
**Case 1: Zero Training Error**

- Assume some $h \in \mathcal{H}$ can achieve zero training error
- Assume its true error $L_P(h) > \epsilon$
- Probability of $h$ being correct on a single training example $\leq 1 - \epsilon$
- Probability of $h$ having zero error on any training set of $N$ examples

$$ P_{D \sim P_N}(L_D(h) = 0 \cap L_P(h) > \epsilon) \leq (1 - \epsilon)^N $$

- Let’s call $L_D(h) = 0 \cap L_P(h) > \epsilon$ as “$h$ is bad”
- Consider $K$ hyp. $\{h_1, \ldots, h_K\}$. Prob. that at least one of these is bad

$$ P_{D \sim P_N}(“h_1 \text{ is bad” } \cup \ldots \cup “h_K \text{ is bad”) } \leq K(1 - \epsilon)^N \quad \text{(using union bound)} $$

- Since $K \leq |\mathcal{H}|$, $K$ can be replaced by the size of set $\mathcal{H}$

$$ P_{D \sim P_N}(\exists h : “h \text{ is bad”) } \leq |\mathcal{H}|(1 - \epsilon)^N \quad \text{(Uniform Convergence)} $$
Case 1: Zero Training Error

Using \((1 - \epsilon) < e^{-\epsilon}\), we get:

\[
P_{D \sim P_N}(\exists h: “h is bad”) \leq |\mathcal{H}|e^{-N\epsilon}
\]
Case 1: Zero Training Error

- Using \((1 - \epsilon) < e^{-\epsilon}\), we get:

  \[ P_{D \sim P_N}(\exists h: "h is bad") \leq |H| e^{-N\epsilon} \]

- Suppose \(|H| e^{-N\epsilon} = \delta\). Then for a given \(\epsilon\) and \(\delta\)

  \[ N \geq \frac{1}{\epsilon} (\log |H| + \log \frac{1}{\delta}) \]
Case 1: Zero Training Error

- Using \((1 - \epsilon) < e^{-\epsilon}\), we get:

\[
P_{D \sim P_N}(\exists h: "h is bad") \leq |\mathcal{H}|e^{-N\epsilon}
\]

- Suppose \(|\mathcal{H}|e^{-N\epsilon} = \delta\). Then for a given \(\epsilon\) and \(\delta\)

\[
N \geq \frac{1}{\epsilon}(\log |\mathcal{H}| + \log \frac{1}{\delta})
\]

.. gives the min. number of training ex. to ensure that there is a “bad” \(h\) with probability at most \(\delta\) (or no bad \(h\) with probability at least \(1 - \delta\))
Case 1: Zero Training Error

- Using \( (1 - \epsilon) < e^{-\epsilon} \), we get:

\[
P_{D \sim P_N} (\exists h: "h is bad") \leq |\mathcal{H}| e^{-N\epsilon}
\]

- Suppose \(|\mathcal{H}| e^{-N\epsilon} = \delta\). Then for a given \(\epsilon\) and \(\delta\)

\[
N \geq \frac{1}{\epsilon} (\log |\mathcal{H}| + \log \frac{1}{\delta})
\]

.. gives the min. number of training ex. to ensure that there is a “bad” \(h\) with probability at most \(\delta\) (or no bad \(h\) with probability at least \(1 - \delta\))

- Essentially, gives a condition that \(h\) will be probably (with probability \(1 - \delta\)) and approximately (with error \(\epsilon\)) correct, given at least these many examples
Case 1: Zero Training Error

- Using \((1 - \epsilon) < e^{-\epsilon}\), we get:

\[ P_{D \sim P_N}(\exists h: \text{“h is bad”}) \leq |\mathcal{H}| e^{-N\epsilon} \]

- Suppose \(|\mathcal{H}| e^{-N\epsilon} = \delta\). Then for a given \(\epsilon\) and \(\delta\)

\[
N \geq \frac{1}{\epsilon} (\log |\mathcal{H}| + \log \frac{1}{\delta})
\]

.. gives the min. number of training ex. to ensure that there is a “bad” \(h\) with probability at most \(\delta\) (or no bad \(h\) with probability at least \(1 - \delta\))

- Essentially, gives a condition that \(h\) will be probably (with probability \(1 - \delta\)) and approximately (with error \(\epsilon\)) correct, given at least these many examples

- Framework of “Probably and Approximately Correct” (PAC) Learning
Case 1: Zero Training Error

- Using $(1 - \epsilon) < e^{-\epsilon}$, we get:

$$P_{D \sim P_N}(\exists h : "h \text{ is bad"}) \leq |\mathcal{H}| e^{-N\epsilon}$$

- Suppose $|\mathcal{H}| e^{-N\epsilon} = \delta$. Then for a given $\epsilon$ and $\delta$,

$$N \geq \frac{1}{\epsilon} \left( \log |\mathcal{H}| + \log \frac{1}{\delta} \right)$$

.. gives the min. number of training ex. to ensure that there is a “bad” $h$ with probability at most $\delta$ (or no bad $h$ with probability at least $1 - \delta$)

- Essentially, gives a condition that $h$ will be probably (with probability $1 - \delta$) and approximately (with error $\epsilon$) correct, given at least these many examples

- Framework of “Probably and Approximately Correct” (PAC) Learning

- Likewise, given $N$ and $\delta$, with probability $1 - \delta$, the true error

$$L_P(h) \leq \frac{\log |\mathcal{H}| + \log \frac{1}{\delta}}{N}$$
Definition: An algorithm $A$ is an $(\epsilon, \delta)$-PAC learning algorithm if, for all distributions $D$: given samples from $D$, the probability that it returns a “bad hypothesis” $h$ is at most $\delta$, where a “bad” hypothesis is one with test error rate more than $\epsilon$ on $D$. 
PAC Learnability and Efficient PAC Learnability

**Definition:** An algorithm \( A \) is an \((\epsilon, \delta)\)-PAC learning algorithm if, for all distributions \( D \): given samples from \( D \), the probability that it returns a “bad hypothesis” \( h \) is at most \( \delta \), where a “bad” hypothesis is one with test error rate more than \( \epsilon \) on \( D \).

**Definition:** An algorithm \( A \) is an efficient \((\epsilon, \delta)\)-PAC learning algorithm if it is an \((\epsilon, \delta)\)-PAC learning algorithm with runtime polynomial in \( \frac{1}{\epsilon} \) and \( \frac{1}{\delta} \).
**Definition:** An algorithm \( A \) is an \((\epsilon, \delta)\)-PAC learning algorithm if, for all distributions \( D \): given samples from \( D \), the probability that it returns a “bad hypothesis” \( h \) is at most \( \delta \), where a “bad” hypothesis is one with test error rate more than \( \epsilon \) on \( D \).

**Definition:** An algorithm \( A \) is an **efficient** \((\epsilon, \delta)\)-PAC learning algorithm if it is an \((\epsilon, \delta)\)-PAC learning algorithm with runtime polynomial in \( \frac{1}{\epsilon} \) and \( \frac{1}{\delta} \)

- Note: a similar notion of an efficient \((\epsilon, \delta)\)-PAC learning algorithm holds in terms of the number of training examples required (polynomial in \( \frac{1}{\epsilon} \) and \( \frac{1}{\delta} \))
Case 2: Non-Zero Training Error

Given $N$ random variables $z_1, \ldots, z_N$, the empirical mean $\bar{z} = \frac{1}{N} \sum_{n=1}^{N} z_n$. Let's assume the true mean is $\mu_z$. Hoeffding's inequality says:

$$P\left( |\mu_z - \bar{z}| \geq \epsilon \right) \leq e^{-2N\epsilon^2}$$

Using the same result, for any single hypothesis $h \in H$, we have:

$$P\left( \text{L}_P(h) - \text{L}_D(h) \geq \epsilon \right) \leq e^{-2N\epsilon^2}$$

Using the union bound, we have:

$$P\left( \exists h: \text{L}_P(h) - \text{L}_D(h) \geq \epsilon \right) \leq |H|e^{-2N\epsilon^2}$$
Case 2: Non-Zero Training Error

- Given $N$ random variables $z_1, \ldots, z_N$, the empirical mean

$$
\bar{z} = \frac{1}{N} \sum_{n=1}^{N} z_n
$$
Case 2: Non-Zero Training Error

Given $N$ random variables $z_1, \ldots, z_N$, the empirical mean

$$\bar{z} = \frac{1}{N} \sum_{n=1}^{N} z_n$$

Let’s assume the true mean is $\mu_z$
Case 2: Non-Zero Training Error

- Given $N$ random variables $z_1, \ldots, z_N$, the empirical mean

\[
\bar{z} = \frac{1}{N} \sum_{n=1}^{N} z_n
\]

- Let’s assume the true mean is $\mu_z$

- **Hoeffding’s inequality** says:

\[
P(|\mu_z - \bar{z}| \geq \epsilon) \leq e^{-2N\epsilon^2}
\]
Case 2: Non-Zero Training Error

- Given $N$ random variables $z_1, \ldots, z_N$, the empirical mean
  \[
  \bar{z} = \frac{1}{N} \sum_{n=1}^{N} z_n
  \]

- Let’s assume the true mean is $\mu_z$

- **Hoeffding’s inequality** says:
  \[
  P(|\mu_z - \bar{z}| \geq \epsilon) \leq e^{-2N\epsilon^2}
  \]

- Using the same result, for any single hypothesis $h \in \mathcal{H}$, we have:
  \[
  P(L_P(h) - L_D(h) \geq \epsilon) \leq e^{-2N\epsilon^2}
  \]
Case 2: Non-Zero Training Error

- Given \( N \) random variables \( z_1, \ldots, z_N \), the empirical mean
  \[
  \bar{z} = \frac{1}{N} \sum_{n=1}^{N} z_n
  \]

- Let’s assume the true mean is \( \mu_z \).

- Hoeffding’s inequality says:
  \[
  P(|\mu_z - \bar{z}| \geq \epsilon) \leq e^{-2N\epsilon^2}
  \]

- Using the same result, for any single hypothesis \( h \in \mathcal{H} \), we have:
  \[
  P(L_P(h) - L_D(h) \geq \epsilon) \leq e^{-2N\epsilon^2}
  \]

- Using the union bound, we have:
  \[
  P(\exists h : L_P(h) - L_D(h) \geq \epsilon) \leq |\mathcal{H}|e^{-2N\epsilon^2}
  \]
Case 2: Non-Zero Training Error

- Suppose \(|H|e^{-2N\epsilon^2} = \delta\). Then for a given \(\epsilon\) and \(\delta\)

\[
N \geq \frac{1}{2\epsilon^2} (\log |H| + \log \frac{1}{\delta})
\]

.. gives the min. number of training ex. required to ensure that \(L_P(h) - L_D(h) \leq \epsilon\) with probability at least \(1 - \delta\)
Case 2: Non-Zero Training Error

Suppose \(|\mathcal{H}|e^{-2N\epsilon^2} = \delta\). Then for a given \(\epsilon\) and \(\delta\)

\[
N \geq \frac{1}{2\epsilon^2}(\log |\mathcal{H}| + \log \frac{1}{\delta})
\]

.. gives the min. number of training ex. required to ensure that \(L_P(h) - L_D(h) \leq \epsilon\) with probability at least \(1 - \delta\)

Note: Number of examples grows as square of \(1/\epsilon\) (note: \(\epsilon < 1\))

- In zero training error case, it grows linearly with \(1/\epsilon\)
- For given \(\epsilon, \delta\), the non-zero training error case requires more examples
Case 2: Non-Zero Training Error

- Suppose $|\mathcal{H}|e^{-2N\epsilon^2} = \delta$. Then for a given $\epsilon$ and $\delta$
  
  $$N \geq \frac{1}{2\epsilon^2} (\log |\mathcal{H}| + \log \frac{1}{\delta})$$

  This gives the min. number of training ex. required to ensure that $L_P(h) - L_D(h) \leq \epsilon$ with probability at least $1 - \delta$.

- Note: Number of examples grows as square of $1/\epsilon$ (note: $\epsilon < 1$)
  
  - In zero training error case, it grows linearly with $1/\epsilon$.
  
  - For given $\epsilon, \delta$, the non-zero training error case requires more examples.

- Likewise, given $N$ and $\delta$, with probability $1 - \delta$, the true error

  $$L_P(h) \leq L_D(h) + \sqrt{\frac{\log |\mathcal{H}| + \log \frac{1}{\delta}}{2N}}$$
Example: Decision Trees

- Let’s consider the hypothesis class of DTs with $k$ leaves
- Suppose data has $D$ binary features/attributes
Example: Decision Trees

- Let’s consider the hypothesis class of DTs with \( k \) leaves
- Suppose data has \( D \) binary features/attributes

\[
H_k = \text{Number of decision trees with } k \text{ leaves}
\]
\[
H_2 = 2
\]
\[
H_k = (\text{#choices of root attribute}) \times

\begin{align*}
&((\text{# left subtrees wth 1 leaf})^* (\text{# right subtrees wth } k-1 \text{ leaves})) \\
&+ ((\text{# left subtrees wth 2 leaves})^* (\text{# right subtrees wth } k-2 \text{ leaves})) \\
&+ \cdots \\
&+ ((\text{# left subtrees wth } k-1 \text{ leaves})^* (\text{# right subtrees wth 1 leaf}))
\end{align*}
\]

\[
H_k = n \sum_{i=1}^{k-1} H_i H_{k-i-1} = n^{k-1} C_{k-1} \quad (C_{k-1}: \text{Catalan Number})
\]
Example: Decision Trees

- Let’s consider the hypothesis class of DTs with \( k \) leaves

- Suppose data has \( D \) binary features/attributes

A loose bound (using Sterling’s approximation): \( H_k \leq D^{k-1}2^{2k-1} \)
Example: Decision Trees

- Let’s consider the hypothesis class of DTs with \( k \) leaves
- Suppose data has \( D \) binary features/attributes

A loose bound (using Sterling’s approximation): \( H_k \leq D^{k-1}2^{2k-1} \)
- Thus \( \log_2 H_k \leq (k - 1) \log_2 D + 2k - 1 \) (linear in \( k \))
Infinite Sized Hypothesis Spaces

- For the finite sized hypothesis class $\mathcal{H}$

$$L_P(h) \leq L_D(h) + \sqrt{\frac{\log |\mathcal{H}| + \log \frac{1}{\delta}}{2N}}$$

- What happens when the hypothesis class size $|\mathcal{H}|$ is infinite?
  - Example: the set of all linear classifiers
Infinite Sized Hypothesis Spaces

- For the finite sized hypothesis class $\mathcal{H}$

\[ L_P(h) \leq L_D(h) + \sqrt{\frac{\log |\mathcal{H}| + \log \frac{1}{\delta}}{2N}} \]

- What happens when the hypothesis class size $|\mathcal{H}|$ is infinite?
  - Example: the set of all linear classifiers

- The above bound doesn’t apply (it just becomes trivial)

- We need some other way of measuring the size of $\mathcal{H}$
Infinite Sized Hypothesis Spaces

- For the finite sized hypothesis class $\mathcal{H}$

$$L_P(h) \leq L_D(h) + \sqrt{\frac{\log |\mathcal{H}| + \log \frac{1}{\delta}}{2N}}$$

- What happens when the hypothesis class size $|\mathcal{H}|$ is infinite?
  - Example: the set of all linear classifiers

- The above bound doesn’t apply (it just becomes trivial)

- We need some other way of measuring the size of $\mathcal{H}$
  - One way: use the complexity $\mathcal{H}$ as a measure of its size
Infinite Sized Hypothesis Spaces

For the finite sized hypothesis class $\mathcal{H}$

$$L_P(h) \leq L_D(h) + \sqrt{\log |\mathcal{H}| + \log \frac{1}{\delta}}$$

What happens when the hypothesis class size $|\mathcal{H}|$ is infinite?

- Example: the set of all linear classifiers

The above bound doesn’t apply (it just becomes trivial)

We need some other way of measuring the size of $\mathcal{H}$

- One way: use the complexity $\mathcal{H}$ as a measure of its size
- .. enters the Vapnik-Chervonenkis dimension (VC dimension)
- VC dimension: a measure of the complexity of a hypothesis class
A set of points is **shattered** by a hypothesis class $\mathcal{H}$ if, no matter how the points are labeled, there exists some $h \in \mathcal{H}$ that can separate the points.
A set of points is **shattered** by a hypothesis class $\mathcal{H}$ if, no matter how the points are labeled, there exists some $h \in \mathcal{H}$ that can separate the points.

- Figure above: 3 points in 2D, $\mathcal{H}$: set of linear classifiers
VC Dimension: The Shattering Game

The concept of shattering is used to define the VC dimension of hypothesis classes.

Consider the following shattering game between us and an adversary:

We choose $d$ points in an input space, positioned however we want.

Adversary labels these $d$ points.

We find a hypothesis $h \in H$ that separates the points.

Note: Shattering just one configuration of $d$ points is enough to win.

The VC dimension of $H$, in that input space, is the maximum $d$ we can choose so that we always succeed in the game.
VC Dimension: The Shattering Game

The concept of shattering is used to define the VC dimension of hypothesis classes.

Consider the following shattering game between us and an adversary:

We choose $d$ points in an input space, positioned however we want. Adversary labels these $d$ points. We find a hypothesis $h \in H$ that separates the points.

Note: Shattering just one configuration of $d$ points is enough to win.

The VC dimension of $H$, in that input space, is the maximum $d$ we can choose so that we always succeed in the game.
VC Dimension: The Shattering Game

The concept of shattering is used to define the VC dimension of hypothesis classes.

Consider the following shattering game between us and an adversary:

- We choose $d$ points in an input space, positioned however we want.
VC Dimension: The Shattering Game

The concept of shattering is used to define the VC dimension of hypothesis classes.

Consider the following shattering game between us and an adversary:

1. We choose $d$ points in an input space, positioned however we want.
2. Adversary labels these $d$ points.

Note: Shattering just one configuration of $d$ points is enough to win.

The VC dimension of $H$, in that input space, is the maximum $d$ we can choose so that we always succeed in the game.
VC Dimension: The Shattering Game

The concept of shattering is used to define the VC dimension of hypothesis classes.

Consider the following shattering game between us and an adversary:

- We choose \( d \) points in an input space, positioned however we want.
- Adversary labels these \( d \) points.
- We find a hypothesis \( h \in \mathcal{H} \) that separates the points.

Note: Shattering just one configuration of \( d \) points is enough to win.

The VC dimension of \( \mathcal{H} \), in that input space, is the maximum \( d \) we can choose so that we always succeed in the game.
VC Dimension: The Shattering Game

The concept of shattering is used to define the VC dimension of hypothesis classes.

Consider the following shattering game between us and an adversary:

- We choose \( d \) points in an input space, positioned however we want.
- Adversary labels these \( d \) points.
- We find a hypothesis \( h \in \mathcal{H} \) that separates the points.
- Note: Shattering **just one configuration** of \( d \) points is enough to win.

The VC dimension of \( \mathcal{H} \), in that input space, is the maximum \( d \) we can choose so that we always succeed in the game.
VC Dimension: The Shattering Game

The concept of shattering is used to define the VC dimension of hypothesis classes

Consider the following shattering game between us and an adversary

- We choose \(d\) points in an input space, positioned however we want
- Adversary labels these \(d\) points
- We find a hypothesis \(h \in \mathcal{H}\) that separates the points
- Note: Shattering just one configuration of \(d\) points is enough to win

The VC dimension of \(\mathcal{H}\), in that input space, is the maximum \(d\) we can choose so that we always succeed in the game
VC Dimension: The Shattering Game

The concept of shattering is used to define the VC dimension of hypothesis classes.

Consider the following shattering game between us and an adversary:

- We choose \( d \) points in an input space, positioned however we want.
- Adversary labels these \( d \) points.
- We find a hypothesis \( h \in \mathcal{H} \) that separates the points.
- Note: Shattering just one configuration of \( d \) points is enough to win.

The VC dimension of \( \mathcal{H} \), in that input space, is the maximum \( d \) we can choose so that we always succeed in the game.
VC Dimension

VC dimension of linear classifiers in $\mathbb{R}^2 = 3$?

What about the VC dimension of linear classifiers in $\mathbb{R}^D$?

$\text{VC} = D + 1$

Recall: a linear classifier in $\mathbb{R}^D$ is defined by $D$ parameters

For linear classifiers, high $D$ $\Rightarrow$ high VC dimension $\Rightarrow$ high complexity

What about the VC dimension of 1-nearest neighbors?

Infinite. Why?

What about the VC dimension of SVM with RBF kernel?

Infinite. Why?

VC dimension intuition: How many points the hypothesis class can “memorize”
VC Dimension

VC dimension of linear classifiers in $\mathbb{R}^2 = 3$?

VC dimension of linear classifiers in $\mathbb{R}^3 = 4$?
VC Dimension

VC dimension of linear classifiers in $\mathbb{R}^2 = 3$?

VC dimension of linear classifiers in $\mathbb{R}^3 = 4$?

What about the VC dimension of linear classifiers in $\mathbb{R}^D$?

Recall: a linear classifier in $\mathbb{R}^D$ is defined by $D$ parameters.

For linear classifiers, high $D \Rightarrow$ high VC dimension $\Rightarrow$ high complexity.

What about the VC dimension of 1-nearest neighbors? Infinite. Why?

What about the VC dimension of SVM with RBF kernel? Infinite. Why?

VC dimension intuition: How many points the hypothesis class can “memorize”.
VC Dimension

VC dimension of linear classifiers in $\mathbb{R}^2 = 3$?

VC dimension of linear classifiers in $\mathbb{R}^3 = 4$?

What about the VC dimension of linear classifiers in $\mathbb{R}^D$?

$\text{VC} = D + 1$
VC Dimension

VC dimension of linear classifiers in $\mathbb{R}^2 = 3$?

VC dimension of linear classifiers in $\mathbb{R}^3 = 4$?

What about the VC dimension of linear classifiers in $\mathbb{R}^D$?

$VC = D + 1$

Recall: a linear classifier in $\mathbb{R}^D$ is defined by $D$ parameters

What about the VC dimension of 1-nearest neighbors? Infinite. Why?

What about the VC dimension of SVM with RBF kernel? Infinite. Why?

VC dimension intuition: How many points the hypothesis class can “memorize”
VC Dimension

VC dimension of linear classifiers in $\mathbb{R}^2 = 3$?

VC dimension of linear classifiers in $\mathbb{R}^3 = 4$?

What about the VC dimension of linear classifiers in $\mathbb{R}^D$?

$VC = D + 1$

Recall: a linear classifier in $\mathbb{R}^D$ is defined by $D$ parameters

For linear classifiers, high $D \Rightarrow$ high VC dimension $\Rightarrow$ high complexity
VC Dimension

VC dimension of linear classifiers in $\mathbb{R}^2 = 3$?

VC dimension of linear classifiers in $\mathbb{R}^3 = 4$?

What about the VC dimension of linear classifiers in $\mathbb{R}^D$?

$VC = D + 1$

Recall: a linear classifier in $\mathbb{R}^D$ is defined by $D$ parameters

For linear classifiers, high $D \Rightarrow$ high VC dimension $\Rightarrow$ high complexity

What about the VC dimension of 1-nearest neighbors?

Infinite. Why?
VC Dimension

VC dimension of linear classifiers in $\mathbb{R}^2 = 3$?

VC dimension of linear classifiers in $\mathbb{R}^3 = 4$?

What about the VC dimension of linear classifiers in $\mathbb{R}^D$?

$$VC = D + 1$$

Recall: a linear classifier in $\mathbb{R}^D$ is defined by $D$ parameters

For linear classifiers, high $D \Rightarrow$ high VC dimension $\Rightarrow$ high complexity

What about the VC dimension of 1-nearest neighbors?

Infinite. Why?
VC Dimension

VC dimension of linear classifiers in $\mathbb{R}^2 = 3$?

VC dimension of linear classifiers in $\mathbb{R}^3 = 4$?

What about the VC dimension of linear classifiers in $\mathbb{R}^D$?

$$VC = D + 1$$

Recall: a linear classifier in $\mathbb{R}^D$ is defined by $D$ parameters

For linear classifiers, high $D \Rightarrow$ high VC dimension $\Rightarrow$ high complexity

What about the VC dimension of 1-nearest neighbors?

Infinite. Why?

What about the VC dimension of SVM with RBF kernel?
VC Dimension

VC dimension of linear classifiers in $\mathbb{R}^2 = 3$?

VC dimension of linear classifiers in $\mathbb{R}^3 = 4$?

What about the VC dimension of linear classifiers in $\mathbb{R}^D$?

$$VC = D + 1$$

Recall: a linear classifier in $\mathbb{R}^D$ is defined by $D$ parameters.

For linear classifiers, high $D \Rightarrow$ high VC dimension $\Rightarrow$ high complexity.

What about the VC dimension of 1-nearest neighbors?

Infinite. Why?

What about the VC dimension of SVM with RBF kernel?

Infinite. Why?
VC Dimension

VC dimension of linear classifiers in $\mathbb{R}^2 = 3$?
VC dimension of linear classifiers in $\mathbb{R}^3 = 4$?
What about the VC dimension of linear classifiers in $\mathbb{R}^D$?

$$VC = D + 1$$

Recall: a linear classifier in $\mathbb{R}^D$ is defined by $D$ parameters
For linear classifiers, high $D \Rightarrow$ high VC dimension $\Rightarrow$ high complexity

What about the VC dimension of 1-nearest neighbors?
Infinite. Why?

What about the VC dimension of SVM with RBF kernel?
Infinite. Why?

VC dimension intuition: How many points the hypothesis class can “memorize”
Using VC Dimension in Generalization Bounds

Recall the PAC based Generalization Bound

\[ \text{ExpectedLoss}(h) \leq \text{TrainingLoss}(h) + \sqrt{\frac{\log |\mathcal{H}| + \log \frac{1}{\delta}}{2N}} \]

For hypothesis classes with infinite size (\(|\mathcal{H}| = \infty\)), but VC dimension \(d\):

\[ \text{ExpectedLoss}(h) \leq \text{TrainingLoss}(h) + \sqrt{d \left( \log \frac{1}{\delta} + \frac{1}{2N} \right)} \]

For linear classifiers, what does it imply?

Having fewer features is better (since it means smaller VC dimension)
Using VC Dimension in Generalization Bounds

Recall the PAC based Generalization Bound

$$\text{ExpectedLoss}(h) \leq \text{TrainingLoss}(h) + \sqrt{\frac{\log |\mathcal{H}| + \log \frac{1}{\delta}}{2N}}$$

For hypothesis classes with infinite size ($|\mathcal{H}| = \infty$), but VC dimension $d$:

$$\text{ExpectedLoss}(h) \leq \text{TrainingLoss}(h) + \sqrt{\frac{d(\log \frac{2N}{d} + 1) + \log \frac{4}{\delta}}{2N}}$$

For linear classifiers, what does it imply?

Having fewer features is better (since it means smaller VC dimension)
Using VC Dimension in Generalization Bounds

Recall the PAC based Generalization Bound

\[
\text{ExpectedLoss}(h) \leq \text{TrainingLoss}(h) + \sqrt{\frac{\log |\mathcal{H}| + \log \frac{1}{\delta}}{2N}}
\]

For hypothesis classes with infinite size (\(|\mathcal{H}| = \infty\)), but VC dimension \(d\):

\[
\text{ExpectedLoss}(h) \leq \text{TrainingLoss}(h) + \sqrt{d \left( \log \frac{2N}{d} + 1 \right) + \log \frac{4}{\delta}} \quad \frac{1}{2N}
\]

For linear classifiers, what does it imply?
Using VC Dimension in Generalization Bounds

Recall the PAC based Generalization Bound

\[ \text{ExpectedLoss}(h) \leq \text{TrainingLoss}(h) + \sqrt{\frac{\log |\mathcal{H}| + \log \frac{1}{\delta}}{2N}} \]

For hypothesis classes with infinite size (|\mathcal{H}| = \infty), but VC dimension \( d \):

\[ \text{ExpectedLoss}(h) \leq \text{TrainingLoss}(h) + \sqrt{\frac{d(\log \frac{2N}{d} + 1) + \log \frac{4}{\delta}}{2N}} \]

For **linear classifiers**, what does it imply?

Having fewer features is better (since it means smaller VC dimension)
Recall: VC dimension of an SVM with RBF kernel is infinite. Is it a bad thing?

Theorem (Vapnik, 1982):

- Given \( N \) data points in \( \mathbb{R}^D \):
  \[ X = \{ x_1, \ldots, x_N \} \]
  with \( ||x_n|| \leq R \)
- Define \( H_{\gamma} \): set of classifiers in \( \mathbb{R}^D \) having margin \( \gamma \) on \( X \)

The VC dimension of \( H_{\gamma} \) is bounded by:

\[
\text{VC}(H_{\gamma}) \leq \min\{D, \lceil 4R^2\gamma^2 \rceil \}
\]

Generalization bound for the SVM:

\[
\text{ExpectedLoss}(h) \leq \text{TrainingLoss}(h) + \sqrt{\text{VC}(H_{\gamma})} \left( \log \frac{2N}{\text{VC}(H_{\gamma})} + 1 \right) + \log \frac{4}{\delta} \frac{1}{2N}
\]

Large \( \gamma \) ⇒ small VC dim. ⇒ small complexity of \( H_{\gamma} \) ⇒ good generalization.
VC Dimension of Support Vector Machines

Recall: VC dimension of an SVM with RBF kernel is infinite. Is it a bad thing?

Not really. SVM’s large margin property ensures good generalization

**Theorem (Vapnik, 1982):**

- Given $N$ data points in $\mathbb{R}^D$: $X = \{x_1, \ldots, x_N\}$ with $\|x_n\| \leq R$
- Define $\mathcal{H}_\gamma$: set of classifiers in $\mathbb{R}^D$ having margin $\gamma$ on $X$

Large $\gamma$ ⇒ small VC dim. ⇒ small complexity of $\mathcal{H}_\gamma$ ⇒ good generalization
Recall: VC dimension of an SVM with RBF kernel is infinite. Is it a bad thing?

Not really. SVM’s large margin property ensures good generalization

Theorem (Vapnik, 1982):

• Given $N$ data points in $\mathbb{R}^D$: $\mathbf{X} = \{x_1, \ldots, x_N\}$ with $\|x_n\| \leq R$
• Define $\mathcal{H}_\gamma$: set of classifiers in $\mathbb{R}^D$ having margin $\gamma$ on $\mathbf{X}$

The VC dimension of $\mathcal{H}_\gamma$ is bounded by:

$$VC(\mathcal{H}_\gamma) \leq \min \left\{ D, \left\lceil \frac{4R^2}{\gamma^2} \right\rceil \right\}$$
Recall: VC dimension of an SVM with RBF kernel is infinite. Is it a bad thing?

Not really. SVM's large margin property ensures good generalization

**Theorem (Vapnik, 1982):**
- Given $N$ data points in $\mathbb{R}^D$: $X = \{x_1, \ldots, x_N\}$ with $||x_n|| \leq R$
- Define $\mathcal{H}_\gamma$: set of classifiers in $\mathbb{R}^D$ having margin $\gamma$ on $X$

The VC dimension of $\mathcal{H}_\gamma$ is bounded by:

$$
\text{VC}(\mathcal{H}_\gamma) \leq \min \left\{ D, \left\lceil \frac{4R^2}{\gamma^2} \right\rceil \right\}
$$

Generalization bound for the SVM:

$$
\text{ExpectedLoss}(h) \leq \text{TrainingLoss}(h) + \sqrt{\frac{\text{VC}(\mathcal{H}_\gamma)(\log \frac{2N}{\text{VC}(\mathcal{H}_\gamma)} + 1) + \log \frac{4}{\delta}}{2N}}
$$
Recall: VC dimension of an SVM with RBF kernel is infinite. Is it a bad thing?

Not really. SVM’s large margin property ensures good generalization.

**Theorem (Vapnik, 1982):**

- Given \( N \) data points in \( \mathbb{R}^D \): \( X = \{ x_1, \ldots, x_N \} \) with \( ||x_n|| \leq R \)
- Define \( \mathcal{H}_\gamma \): set of classifiers in \( \mathbb{R}^D \) having margin \( \gamma \) on \( X \)

The VC dimension of \( \mathcal{H}_\gamma \) is bounded by:

\[
VC(\mathcal{H}_\gamma) \leq \min \left\{ D, \left\lceil \frac{4R^2}{\gamma^2} \right\rceil \right\}
\]

Generalization bound for the SVM:

\[
\text{ExpectedLoss}(h) \leq \text{TrainingLoss}(h) + \sqrt{\frac{VC(\mathcal{H}_\gamma)(\log \frac{2N}{VC(\mathcal{H}_\gamma)} + 1) + \log \frac{4}{\delta}}{2N}}
\]

Large \( \gamma \) \( \Rightarrow \) small VC dim. \( \Rightarrow \) small complexity of \( \mathcal{H}_\gamma \) \( \Rightarrow \) good generalization.
Things to Remember..

- We care about the expected error, not the training error.
We care about the expected error, not the training error

Generalization bounds quantify the difference between these two errors

It has the following general form

\[ \text{ExpLoss}(h) \leq \text{TrainLoss}(h) + f(\text{model complexity, } N) \]

approaches 0 as \( N \to \infty \)
Things to Remember..

- We care about the expected error, not the training error.
- Generalization bounds quantify the difference between these two errors.
  - It has the following general form:
    \[
    \text{ExpLoss}(h) \leq \text{TrainLoss}(h) + f(\text{model complexity}, N) \]
    approaches 0 as \(N \to \infty\)
- Finite sized hypothesis spaces: \(\log |\mathcal{H}|\) is a measure of complexity.
Things to Remember..

- We care about the **expected error**, not the **training error**
- Generalization bounds quantify the difference between these two errors
  - It has the following general form
    \[
    \text{ExpLoss}(h) \leq \text{TrainLoss}(h) + f(\text{model complexity, } N)
    \]
    approaches 0 as \( N \to \infty \)

- Finite sized hypothesis spaces: \( \log |\mathcal{H}| \) is a measure of complexity
- Finite sized hypothesis spaces: VC dimension is a measure of complexity
Things to Remember..

- We care about the **expected error**, not the **training error**
- Generalization bounds quantify the difference between these two errors
  - It has the following general form
    \[
    \text{ExpLoss}(h) \leq \text{TrainLoss}(h) + f(\text{model complexity, } N)
    \]
    \[\text{approaches 0 as } N \to \infty\]
- Finite sized hypothesis spaces: \(\log |\mathcal{H}|\) is a measure of complexity
- Finite sized hypothesis spaces: VC dimension is a measure of complexity
- Often these bounds are loose for moderate values of \(N\)
Things to Remember..

- We care about the expected error, not the training error.
- Generalization bounds quantify the difference between these two errors.
  - It has the following general form:
    \[
    \text{ExpLoss}(h) \leq \text{TrainLoss}(h) + f(\text{model complexity, } N)
    \]
    approaches 0 as \( N \to \infty \)
- Finite sized hypothesis spaces: \( \log |\mathcal{H}| \) is a measure of complexity.
- Finite sized hypothesis spaces: VC dimension is a measure of complexity.
- Often these bounds are loose for moderate values of \( N \).
  - Tighter generalization bounds exist (often data-dependent; e.g., using complexity measures such as Radamacher Complexity).
Things to Remember..

- We care about the expected error, not the training error.
- Generalization bounds quantify the difference between these two errors.
  - It has the following general form:
    \[
    \text{ExpLoss}(h) \leq \text{TrainLoss}(h) + f(\text{model complexity}, N)
    \]
    approaches 0 as \( N \to \infty \)
- Finite sized hypothesis spaces: \( \log |\mathcal{H}| \) is a measure of complexity.
- Finite sized hypothesis spaces: VC dimension is a measure of complexity.
- Often these bounds are loose for moderate values of \( N \).
  - Tighter generalization bounds exist (often data-dependent; e.g., using complexity measures such as Radamacher Complexity).
  - But even loose bounds are often useful for understanding the basic properties of learning models/algorithms.