# Elements of Computational Learning Theory

Dimitris Diochnos

University of Oklahoma
School of Computer Science

September 28, 2020
Norman, OK

# Outline

# Outline

# What is Machine Learning?

- **Machine learning** is the subfield of computer science that gives "computers the ability to learn without being explicitly programmed".

  - term coined by Arthur Samuel in 1959 while at IBM

- The study of algorithms that can learn from data.

# Another View of Machine Learning

- Learning from historical data to make decisions about unseen data.

- Traditional Programming

  Data $\longrightarrow$
  Program $\longrightarrow$ [Computer] $\longrightarrow$ Output

- Machine Learning

  Data $\longrightarrow$
  Output $\longrightarrow$ [Computer] $\longrightarrow$ Program

# When is Machine Learning a Good Idea?

- Situations where ...
    - humans can not describe how they do a task
        - character recognition

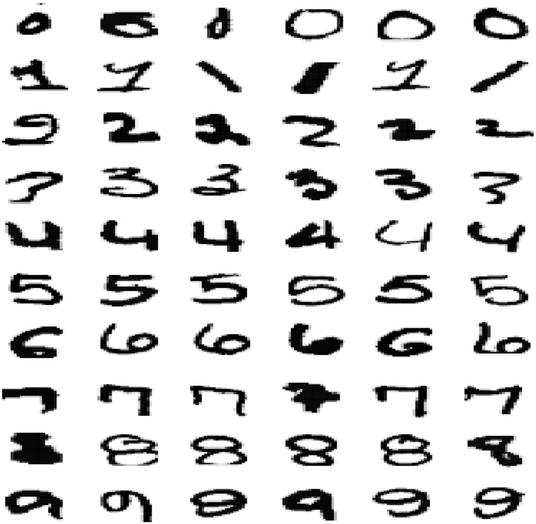
# When is Machine Learning a Good Idea?

- Situations where ...
  - humans can not describe how they do a task
    - character recognition
  - the desired function changes frequently
    - recommend stock transactions

# When is Machine Learning a Good Idea?

- Situations where ...
  - humans can not describe how they do a task
    - character recognition
  - the desired function changes frequently
    - recommend stock transactions
  - each user needs a customized function $f$
    - email spam / ham
    - email importance (perhaps delete without presenting?)
    - recommendations on Amazon

# Can you write a program that recognizes these digits?



# What Machine Learning Does

- Class A

- Class B

# What Machine Learning Does

- Class A

- Class B

- Want to be able to generalize the classification to unseen data.

http://ciml.info/ (Credit: Hal Daumé III)

# Classify These

(Credit: Hal Daumé III)

# Let's See ...

# Let's See ...

- Bird vs non-bird

# Let's See ...

- Bird vs non-bird
- Flies vs not-flies

# Let's See ...

- Bird vs non-bird
- Flies vs not-flies

- We need **bias** in order to be able to generalize to unseen data.

# No Free-Lunch Theorems

### Theorem 1

*Let $\mathcal{F}$ be the set of all possible Boolean functions on n variables. Let $Acc_G(L)$ be the (generalization) accuracy of L on non-training examples. Then, for any consistent learner L, it holds*

$$\frac{1}{|\mathcal{F}|} \cdot \sum_{\mathcal{F}} Acc_G(L) = 1/2 \,.$$

# No Free-Lunch Theorems

### Theorem 1

*Let $\mathcal{F}$ be the set of all possible Boolean functions on $n$ variables. Let $Acc_G(L)$ be the (generalization) accuracy of $L$ on non-training examples. Then, for any consistent learner $L$, it holds*

$$\frac{1}{|\mathcal{F}|} \cdot \sum_{\mathcal{F}} Acc_G(L) = 1/2\,.$$

### Proof Sketch.

Let $S$ be the set of training examples.
Let $f \in \mathcal{F}$ such that $Acc_G(f) = \frac{1}{2} + \delta$.
Then, $\exists f' \in \mathcal{F}$ such that $Acc_G(f') = \frac{1}{2} - \delta$.
To see why, note that we can have an $f' \in \mathcal{F}$ that satisfies:

$$\begin{cases} (\forall x \in S)(f'(x) = f(x)) \\ (\forall x \notin S)(f'(x) = \neg f(x)) \end{cases} \quad \square$$

# No Free-Lunch Theorems

**Theorem 2**

*Let $\mathcal{F}$ be the set of all possible Boolean functions on $n$ variables. Let $Acc_G(L)$ be the (generalization) accuracy of $L$ on non-training examples. Then, for any consistent learner $L$, it holds*

$$\frac{1}{|\mathcal{F}|} \cdot \sum_{\mathcal{F}} Acc_G(L) = {}^1\!/_2\,.$$

**Corollary 3**

*For any two learners $L_1, L_2$, if there exists a learning problem $P$ such that $Acc_G(L_1) > Acc_G(L_2)$, then there exists another learning problem $P'$ such that $Acc_G(L_1) < Acc_G(L_2)$.*

# Outline

## Some Basic Questions

- What kind of concepts are easy or hard to learn?

# Some Basic Questions

- What kind of concepts are easy or hard to learn?
- How many examples are enough?

## Some Basic Questions

- What kind of concepts are easy or hard to learn?
- How many examples are enough?
- How many examples are necessary?

# Some Basic Questions

- What kind of concepts are easy or hard to learn?
- How many examples are enough?
- How many examples are necessary?
- Which algorithm will we use to process the examples?

# Some Basic Questions

- What kind of concepts are easy or hard to learn?
- How many examples are enough?
- How many examples are necessary?
- Which algorithm will we use to process the examples?
  - Does it matter which algorithm we select?

# Some Basic Questions

- What kind of concepts are easy or hard to learn?
- How many examples are enough?
- How many examples are necessary?
- Which algorithm will we use to process the examples?
  - Does it matter which algorithm we select?
- How frequently will our solution make mistakes during prediction?

## Some Basic Questions

- What kind of concepts are easy or hard to learn?
- How many examples are enough?
- How many examples are necessary?
- Which algorithm will we use to process the examples?
    - Does it matter which algorithm we select?
- How frequently will our solution make mistakes during prediction?
    - How confident are we about such a claim?

# The Main Goal of Computational Learning Theory

Find a good approximation of a function with high probability

# Computational Learning Theory

**Goal** (Good Approximation with High Probability)
There is a function c over a space $X$. One wants to come up (in a reasonable amount of time) with a function h such that h is a *good approximation* of c with *high probability*.

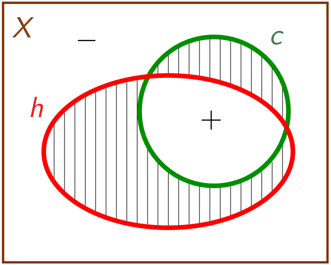
## Description 1 (Parameters and Terminology)

- $X$: *Instance Space* (*say*, $\{0, 1\}^n$) $\mathcal{Y}$: *Labels* (*say*, $\{+, -\}$)
- $c \in \mathcal{C}$: *Target concept belonging to a concept class*
- $h \in \mathcal{H}$: *Hypothesis belonging to a hypothesis class*
- *Good Approximation: Small Risk (Error)* $\varepsilon$
- *High Probability: Confidence* $1 - \delta$
- *Reasonable Amount of Time: Polynomial in* $n, {}^{1}/_{\varepsilon}, {}^{1}/_{\delta}$, *size*($c$)

- Realizability assumption: $(\forall c \in \mathcal{C})(\exists h \in \mathcal{H})(\forall x \in X)\,[h(x) = c(x)]$
($\mathcal{H}$ is at least as expressive as $\mathcal{C}$; we will see an example later)

# Probably Approximately Correct (PAC) Learning

- There is an *arbitrary, unknown* distribution $\mathcal{D}$ over $X$.
- Learn from *poly* $\left(\frac{1}{\varepsilon}, \frac{1}{\delta}\right)$ many examples $(x, \mathrm{c}(x))$, where $x \sim \mathcal{D}$.
- $\mathrm{Risk}_{\mathcal{D}}(\mathrm{h}, \mathrm{c}) = \mathrm{Pr}_{x\sim\mathcal{D}}(\mathrm{h}(x) \neq \mathrm{c}(x))$.




**Goal 1 ([Valiant, 1984])**

$$\Pr\left(Risk_{\mathcal{D}}(h, c) \leq \varepsilon\right) \geq 1 - \delta\ .$$

# Typical Functions Used for Learning

Monotone Conjunctions/Monomials (Boolean AND of some variables chosen from $\{x_1, x_2, \ldots, x_n\}$)

e.g., $c = x_2 \wedge x_5 \wedge x_8$ (sometimes simply write $c = x_2x_5x_8$)

- $|\mathcal{H}| = 2^n$.
- Sometimes we may need to include the `FALSE` function (e.g., for VC-dimension arguments) even if such a function can not be represented by combining variables. In these cases $|\mathcal{H}| = 2^n + 1$.

# Typical Functions Used for Learning

Monotone Conjunctions/Monomials (Boolean AND of some variables chosen from $\{x_1, x_2, \ldots, x_n\}$)

e.g., $c = x_2 \wedge x_5 \wedge x_8$ (sometimes simply write $c = x_2 x_5 x_8$)

- $|\mathcal{H}| = 2^n$.
- Sometimes we may need to include the `FALSE` function (e.g., for VC-dimension arguments) even if such a function can not be represented by combining variables. In these cases $|\mathcal{H}| = 2^n + 1$.

---

Conjunctions/Monomials (allow negated variables)

e.g., $c = x_2 \wedge \overline{x_5} \wedge x_8$ $\quad\quad$ ($c = x_2 \overline{x_5} x_8$)

- $|\mathcal{H}| = 3^n + 1$. (including the constant `FALSE` function.)
- `FALSE` function can be represented: e.g., $c' = x_1 \wedge \overline{x_1}$.

# Typical Functions Used for Learning

Monotone Conjunctions/Monomials (Boolean AND of some variables chosen from $\{x_1, x_2, \ldots, x_n\}$)

e.g., $c = x_2 \wedge x_5 \wedge x_8$ (sometimes simply write $c = x_2x_5x_8$)

- $|\mathcal{H}| = 2^n$.
- Sometimes we may need to include the `FALSE` function (e.g., for VC-dimension arguments) even if such a function can not be represented by combining variables. In these cases $|\mathcal{H}| = 2^n + 1$.

---

Conjunctions/Monomials (allow negated variables)

e.g., $c = x_2 \wedge \overline{x_5} \wedge x_8$ $\qquad$ $(c = x_2\overline{x_5}x_8)$

- $|\mathcal{H}| = 3^n + 1$. (including the constant `FALSE` function.)
- `FALSE` function can be represented: e.g., $c' = x_1 \wedge \overline{x_1}$.

---

Halfspaces $\quad$ e.g., $c = sgn(w_0 + w_1 \cdot x_1 + w_2 \cdot x_2 + \ldots + w_n \cdot x_n)$

$$sgn(z) = \begin{cases} +1 & \text{, if } z > 0 \\ -1 & \text{, if } z \leq 0 \end{cases}$$

# Why These Functions Used as Toy Examples?

- Exhibit **bias**.
- (Monotone) conjunctions is one of the most basic ways of selecting/combining features/constraints in a prediction mechanism.
- Building blocks for richer classes of functions that are less understood; e.g., general DNF formulae.
  (e.g., learning monotone DNF formulae over the uniform distribution is an open problem.)
- Directly or indirectly, applications to logic, circuit complexity, etc.
- Typical benchmarks as they usually provide interesting, but non-trivial insights of the definitions, the bounds that we should expect to get, etc.
- Can also be useful in contexts of other disciplines (see next slide).

# Why These Functions Used as Toy Examples?

- Exhibit **bias**.
- (Monotone) conjunctions is one of the most basic ways of selecting/combining features/constraints in a prediction mechanism.
- Building blocks for richer classes of functions that are less understood; e.g., general DNF formulae.
  (e.g., learning monotone DNF formulae over the uniform distribution is an open problem.)
- Directly or indirectly, applications to logic, circuit complexity, etc.
- Typical benchmarks as they usually provide interesting, but non-trivial insights of the definitions, the bounds that we should expect to get, etc.
- Can also be useful in contexts of other disciplines (see next slide).
- We will start with PAC learning (general) conjunctions.

# Finding All Common Properties of a Set of Objects

Let $X = \{0,1\}^6$ and $c = x_1 \wedge \overline{x}_3 \wedge x_4$.

- Request $m$ examples and look at the positive ones.
- Delete the variables that are falsified by the positive examples.

*A Study of Thinking* [Bruner, Goodnow, Austin, 1956]

# Finding All Common Properties of a Set of Objects

Let $X = \{0,1\}^6$ and $c = x_1 \wedge \overline{x}_3 \wedge x_4$.

- Request $m$ examples and look at the positive ones.
- Delete the variables that are falsified by the positive examples.

*A Study of Thinking* [Bruner, Goodnow, Austin, 1956]

| example | hypothesis h |
|---|---|
| | $x_1 \wedge \overline{x}_1 \wedge x_2 \wedge \overline{x}_2 \wedge x_3 \wedge \overline{x}_3 \wedge x_4 \wedge \overline{x}_4 \wedge x_5 \wedge \overline{x}_5 \wedge x_6 \wedge \overline{x}_6$ |
| $((110011), +)$ | $x_1 \wedge x_2 \wedge \overline{x}_3 \wedge \overline{x}_4 \wedge x_5 \wedge x_6$ |
| $((010011), +)$ | $x_2 \wedge \overline{x}_3 \wedge \overline{x}_4 \wedge x_5 \wedge x_6$ |
| $((010111), +)$ | $x_2 \wedge \overline{x}_3 \wedge x_5 \wedge x_6$ |

# Finding All Common Properties of a Set of Objects

Let $X = \{0, 1\}^6$ and $c = x_1 \wedge \overline{x}_3 \wedge x_4$.

- Request $m$ examples and look at the positive ones.
- Delete the variables that are falsified by the positive examples.

*A Study of Thinking* [Bruner, Goodnow, Austin, 1956]

| example | hypothesis h |
|---|---|
| | $x_1 \wedge \overline{x}_1 \wedge x_2 \wedge \overline{x}_2 \wedge x_3 \wedge \overline{x}_3 \wedge x_4 \wedge \overline{x}_4 \wedge x_5 \wedge \overline{x}_5 \wedge x_6 \wedge \overline{x}_6$ |
| $((110011), +)$ | $x_1 \wedge x_2 \wedge \overline{x}_3 \wedge \overline{x}_4 \wedge x_5 \wedge x_6$ |
| $((010011), +)$ | $x_2 \wedge \overline{x}_3 \wedge \overline{x}_4 \wedge x_5 \wedge x_6$ |
| $((010111), +)$ | $x_2 \wedge \overline{x}_3 \wedge x_5 \wedge x_6$ |

- Is such an algorithm good for PAC learning?
  - YES, provided $m$ is large enough.
  - Creates a consistent hypothesis:
    - Predicts correct label for each training example.

# How Many Examples are Enough?

**Theorem 4 (PAC Learning of Finite Concept Classes)**

*Let $\mathcal{H}$ contain a* ***finite amount*** $|\mathcal{H}|$ *of functions. For every distribution $\mathcal{D}$, drawing $m \geq \frac{1}{\varepsilon} \cdot \left(\ln|\mathcal{H}| + \ln\frac{1}{\delta}\right)$ examples are* ***enough*** *to guarantee that* ***any consistent*** *hypothesis $h$ satisfies* $\Pr\left(Risk_{\mathcal{D}}(h, c) \leq \varepsilon\right) \geq 1 - \delta$ .

# How Many Examples are Enough?

**Theorem 4 (PAC Learning of Finite Concept Classes)**

*Let $\mathcal{H}$ contain a* ***finite amount*** $|\mathcal{H}|$ *of functions. For every distribution $\mathcal{D}$, drawing $m \geq \frac{1}{\varepsilon} \cdot \left(\ln|\mathcal{H}| + \ln\frac{1}{\delta}\right)$ examples are* ***enough*** *to guarantee that* ***any consistent*** *hypothesis $h$ satisfies $\Pr\left(Risk_{\mathcal{D}}(h, c) \leq \varepsilon\right) \geq 1 - \delta$ .*

**Proof Idea.**

- Consider all the hypotheses that make mistakes frequently.

# How Many Examples are Enough?

**Theorem 4 (PAC Learning of Finite Concept Classes)**

*Let $\mathcal{H}$ contain a* ***finite amount*** *$|\mathcal{H}|$ of functions. For every distribution $\mathcal{D}$, drawing $m \geq \frac{1}{\varepsilon} \cdot \left(\ln|\mathcal{H}| + \ln\frac{1}{\delta}\right)$ examples are* ***enough*** *to guarantee that* ***any consistent*** *hypothesis $h$ satisfies $\Pr\left(Risk_{\mathcal{D}}(h, c) \leq \varepsilon\right) \geq 1 - \delta$ .*

**Proof Idea.**

- Consider all the hypotheses that make mistakes frequently.
- As we are seeing more and more training examples, hypotheses that make mistakes frequently should make at least one mistake in the entire sequence of the training examples.

# How Many Examples are Enough?

**Theorem 4 (PAC Learning of Finite Concept Classes)**

*Let $\mathcal{H}$ contain a* ***finite amount*** *$|\mathcal{H}|$ of functions. For every distribution $\mathcal{D}$, drawing $m \geq \frac{1}{\varepsilon} \cdot \left(\ln|\mathcal{H}| + \ln\frac{1}{\delta}\right)$ examples are* ***enough*** *to guarantee that* ***any consistent*** *hypothesis $h$ satisfies $\Pr\left(Risk_{\mathcal{D}}(h, c) \leq \varepsilon\right) \geq 1 - \delta$ .*

**Proof Idea.**

- Consider all the hypotheses that make mistakes frequently.
- As we are seeing more and more training examples, hypotheses that make mistakes frequently should make at least one mistake in the entire sequence of the training examples.
- Draw enough many training examples to guarantee with high probability that all such hypotheses that make mistakes frequently, will also make at least one mistake in the training examples that we have drawn.

# How Many Examples are Enough?

**Theorem 4 (PAC Learning of Finite Concept Classes)**

*Let $\mathcal{H}$ contain a **finite amount** $|\mathcal{H}|$ of functions. For every distribution $\mathcal{D}$, drawing $m \geq \frac{1}{\varepsilon} \cdot \left(\ln|\mathcal{H}| + \ln\frac{1}{\delta}\right)$ examples are **enough** to guarantee that **any consistent** hypothesis $h$ satisfies* $\Pr\left(Risk_{\mathcal{D}}(h, c) \leq \varepsilon\right) \geq 1 - \delta$ .

**Proof Idea.**

- Consider all the hypotheses that make mistakes frequently.
- As we are seeing more and more training examples, hypotheses that make mistakes frequently should make at least one mistake in the entire sequence of the training examples.
- Draw enough many training examples to guarantee with high probability that all such hypotheses that make mistakes frequently, will also make at least one mistake in the training examples that we have drawn.
- Return some hypothesis that never made a mistake. □

# How Many Examples are Enough?

**Theorem 5 (PAC Learning of Finite Concept Classes)**

*Let $\mathcal{H}$ contain a **finite amount** $|\mathcal{H}|$ of functions. For every distribution $\mathcal{D}$, drawing $m \geq \frac{1}{\varepsilon} \cdot \left(\ln|\mathcal{H}| + \ln\frac{1}{\delta}\right)$ examples are **enough** to guarantee that **any consistent** hypothesis $h$ satisfies $\Pr\left(Risk_{\mathcal{D}}\left(h, c\right) \leq \varepsilon\right) \geq 1 - \delta$ .*

Proof.

Call a hypothesis $h$ **bad** if $\mathrm{Risk}_{\mathcal{D}}\left(\mathrm{h}, \mathrm{c}\right) > \varepsilon$.

# How Many Examples are Enough?

**Theorem 5 (PAC Learning of Finite Concept Classes)**

*Let $\mathcal{H}$ contain a **finite amount** $|\mathcal{H}|$ of functions. For every distribution $\mathcal{D}$, drawing $m \geq \frac{1}{\varepsilon} \cdot \left(\ln |\mathcal{H}| + \ln \frac{1}{\delta}\right)$ examples are **enough** to guarantee that **any consistent** hypothesis $h$ satisfies $\Pr\left(Risk_{\mathcal{D}}\left(h, c\right) \leq \varepsilon\right) \geq 1 - \delta$ .*

Proof.

Call a hypothesis $h$ **bad** if $\mathrm{Risk}_{\mathcal{D}}\left(\mathrm{h}, \mathrm{c}\right) > \varepsilon$. Then for such a bad $h$,

$$\Pr\left(h \text{ is consistent with the first training example}\right) \quad < \quad (1 - \varepsilon)$$

# How Many Examples are Enough?

**Theorem 5 (PAC Learning of Finite Concept Classes)**

*Let $\mathcal{H}$ contain a **finite amount** $|\mathcal{H}|$ of functions. For every distribution $\mathcal{D}$, drawing $m \geq \frac{1}{\varepsilon} \cdot \left(\ln|\mathcal{H}| + \ln\frac{1}{\delta}\right)$ examples are **enough** to guarantee that **any consistent** hypothesis $h$ satisfies $\Pr\left(Risk_{\mathcal{D}}(h, c) \leq \varepsilon\right) \geq 1 - \delta$ .*

Proof.

Call a hypothesis $h$ **bad** if $\mathrm{Risk}_{\mathcal{D}}(\mathrm{h}, \mathrm{c}) > \varepsilon$. Then for such a bad $h$,

$$\begin{aligned} \Pr(h \text{ is consistent with the first training example}) &< (1-\varepsilon) \\ \Pr(h \text{ is consistent with all } m \text{ training examples}) &< (1-\varepsilon)^m \end{aligned}$$

# How Many Examples are Enough?

### Theorem 5 (PAC Learning of Finite Concept Classes)

*Let $\mathcal{H}$ contain a **finite amount** $|\mathcal{H}|$ of functions. For every distribution $\mathcal{D}$, drawing $m \geq \frac{1}{\varepsilon} \cdot \left(\ln |\mathcal{H}| + \ln \frac{1}{\delta}\right)$ examples are **enough** to guarantee that **any consistent** hypothesis $h$ satisfies $\Pr\left(Risk_{\mathcal{D}}\left(h, c\right) \leq \varepsilon\right) \geq 1 - \delta$ .*

### Proof.

Call a hypothesis $h$ **bad** if $\mathrm{Risk}_{\mathcal{D}}\left(\mathrm{h}, \mathrm{c}\right) > \varepsilon$. Then for such a bad $h$,

$$\begin{aligned} \Pr\left(h \text{ is consistent with the first training example}\right) &< (1-\varepsilon) \\ \Pr\left(h \text{ is consistent with all } m \text{ training examples}\right) &< (1-\varepsilon)^m \end{aligned}$$

Let $h_1, h_2, \ldots, h_k$ be all the $k$ hypotheses from $\mathcal{H}$ that are bad. For each such bad hypothesis $h_i$ with $i \in \{1, \ldots, k\}$, consider the bad event

$$B_i \equiv h_i \text{ is consistent with all } m \text{ training examples}\,.$$

$\Pr\left(B_1 \vee \ldots \vee B_k\right) \leq \sum_{i=1}^{k} \Pr\left(B_i\right) < k \cdot (1-\varepsilon)^m \leq |\mathcal{H}|\,(1-\varepsilon)^m \leq |\mathcal{H}| \cdot e^{-\varepsilon \cdot m}$. □

# How Many Examples are Enough?

**Theorem 6 (PAC Learning of Finite Concept Classes)**

*Let $\mathcal{H}$ be a hypothesis space s.t. $|\mathcal{H}| < \infty$. For every distribution $\mathcal{D}$, drawing $m \geq \frac{1}{\varepsilon} \cdot \left( \ln|\mathcal{H}| + \ln\frac{1}{\delta} \right)$ examples are* ***enough*** *to guarantee that* ***any consistent*** *hypothesis $h$ satisfies $\Pr\left(Risk_{\mathcal{D}}(h, c) \leq \varepsilon\right) \geq 1 - \delta$ .*

- Theorem resolves the statistical question on $m$
  - How many examples are enough?

# How Many Examples are Enough?

**Theorem 6 (PAC Learning of Finite Concept Classes)**

*Let $\mathcal{H}$ be a hypothesis space s.t. $|\mathcal{H}| < \infty$. For every distribution $\mathcal{D}$, drawing $m \geq \frac{1}{\varepsilon} \cdot \left(\ln|\mathcal{H}| + \ln\frac{1}{\delta}\right)$ examples are **enough** to guarantee that **any consistent** hypothesis $h$ satisfies* $\Pr\left(Risk_{\mathcal{D}}(h, c) \leq \varepsilon\right) \geq 1 - \delta$ .

- Theorem resolves the statistical question on $m$
  - How many examples are enough?
- We still need to resolve the computational question (algorithm returning $h$)

# How Many Examples are Enough?

**Theorem 6 (PAC Learning of Finite Concept Classes)**

*Let $\mathcal{H}$ be a hypothesis space s.t. $|\mathcal{H}| < \infty$. For every distribution $\mathcal{D}$, drawing $m \geq \frac{1}{\varepsilon} \cdot \left(\ln|\mathcal{H}| + \ln\frac{1}{\delta}\right)$ examples are **enough** to guarantee that **any consistent** hypothesis $h$ satisfies $\Pr\left(Risk_{\mathcal{D}}\left(h, c\right) \leq \varepsilon\right) \geq 1 - \delta$ .*

- Theorem resolves the statistical question on $m$
  - How many examples are enough?
- We still need to resolve the computational question (algorithm returning $h$)
- PAC learning conjunctions, with $\varepsilon = 1/100$ and $\delta = 0.05$.
  - $|\mathcal{H}| = 3^n + 1 \quad \leq \quad 3^n + 3^n + 3^n = 3^{n+1}$

# How Many Examples are Enough?

**Theorem 6 (PAC Learning of Finite Concept Classes)**

*Let $\mathcal{H}$ be a hypothesis space s.t. $|\mathcal{H}| < \infty$. For every distribution $\mathcal{D}$, drawing $m \geq \frac{1}{\varepsilon} \cdot \left(\ln|\mathcal{H}| + \ln\frac{1}{\delta}\right)$ examples are **enough** to guarantee that **any consistent** hypothesis $h$ satisfies $\Pr\left(Risk_{\mathcal{D}}\left(h, c\right) \leq \varepsilon\right) \geq 1 - \delta$ .*

- Theorem resolves the statistical question on $m$
  - How many examples are enough?
- We still need to resolve the computational question (algorithm returning $h$)
- PAC learning conjunctions, with $\varepsilon = 1/100$ and $\delta = 0.05$.
  - $|\mathcal{H}| = 3^n + 1 \quad \leq \quad 3^n + 3^n + 3^n = 3^{n+1}$
  - Theorem imples that when $n = 20$ ($2^{20}$ truth assignments in $\{0, 1\}^{20}$) only $m = 2607$ examples are enough! (less than 0.2% of $|X|$)

# Can we Learn a Disjunction of $k \geq 2$ Conjunctions?

- Say $k = 3$. Then a function looks like
  $(x_1 \wedge x_5) \vee (\overline{x_2} \wedge x_4 \wedge x_7) \vee (x_3 \wedge \overline{x}_4 \wedge \overline{x_5} \wedge x_7 \wedge \overline{x_8})$.
- Then, $|\mathcal{C}| \leq (3^n + 1) \cdot (3^n + 1) \cdot (3^n + 1) \leq 3^{n+1} \cdot 3^{n+1} \cdot 3^{n+1} = 3^{3n+3}$.
- The previous theorem implies $m = \left\lceil \frac{1}{\varepsilon} \cdot \ln\left(\frac{3^{3n+3}}{\delta}\right)\right\rceil = \left\lceil \frac{3n+3}{\varepsilon} \cdot \ln\left(\frac{3}{\delta}\right)\right\rceil$ training examples are more than enough for PAC learning the class.

So the question becomes:

Is there an algorithm for efficiently PAC learning such functions?
The answer is quite surprising!

# Can we Learn a Disjunction of $k \geq 2$ Conjunctions?

- Say $k = 3$. Then a function looks like
  $(x_1 \wedge x_5) \vee (\overline{x_2} \wedge x_4 \wedge x_7) \vee (x_3 \wedge \overline{x}_4 \wedge \overline{x_5} \wedge x_7 \wedge \overline{x_8})$.
- Then, $|\mathcal{C}| \leq (3^n + 1) \cdot (3^n + 1) \cdot (3^n + 1) \leq 3^{n+1} \cdot 3^{n+1} \cdot 3^{n+1} = 3^{3n+3}$.
- The previous theorem implies $m = \left\lceil \frac{1}{\varepsilon} \cdot \ln\left(\frac{3^{3n+3}}{\delta}\right) \right\rceil = \left\lceil \frac{3n+3}{\varepsilon} \cdot \ln\left(\frac{3}{\delta}\right) \right\rceil$ training examples are more than enough for PAC learning the class.

So the question becomes:

Is there an algorithm for efficiently PAC learning such functions?
The answer is quite surprising!

- Assuming $NP \neq RP$, we can not do that efficiently if we use $\mathcal{H} = \mathcal{C}$. (proper learning)

# Can we Learn a Disjunction of $k \geq 2$ Conjunctions?

- Say $k = 3$. Then a function looks like
  $(x_1 \wedge x_5) \vee (\overline{x_2} \wedge x_4 \wedge x_7) \vee (x_3 \wedge \overline{x}_4 \wedge \overline{x_5} \wedge x_7 \wedge \overline{x_8})$.
- Then, $|\mathcal{C}| \leq (3^n + 1) \cdot (3^n + 1) \cdot (3^n + 1) \leq 3^{n+1} \cdot 3^{n+1} \cdot 3^{n+1} = 3^{3n+3}$.
- The previous theorem implies $m = \left\lceil \frac{1}{\varepsilon} \cdot \ln\left(\frac{3^{3n+3}}{\delta}\right) \right\rceil = \left\lceil \frac{3n+3}{\varepsilon} \cdot \ln\left(\frac{3}{\delta}\right) \right\rceil$ training examples are more than enough for PAC learning the class.

So the question becomes:

Is there an algorithm for efficiently PAC learning such functions?
The answer is quite surprising!

- Assuming $NP \neq RP$, we can not do that efficiently if we use $\mathcal{H} = \mathcal{C}$. (proper learning)
- However, we can PAC learn $\mathcal{C}$ efficiently if we use a larger class of functions as our hypothesis class $\mathcal{H}$.

# The Complexity Class *RP*

**Randomized Polynomial (*RP*) time.** Complexity class of problems for which a non-deterministic Turing machine:

- runs in poly-time w.r.t. the input size,
- if the correct answer is *NO* it returns *NO*,
- if the correct answer is *YES* it returns *YES* with probability $p \geq 1/2$.

(a *YES* answer is always correct!)

# The Complexity Class *RP*

**Randomized Polynomial (*RP*) time.** Complexity class of problems for which a non-deterministic Turing machine:

- runs in poly-time w.r.t. the input size,
- if the correct answer is *NO* it returns *NO*,
- if the correct answer is *YES* it returns *YES* with probability $p \geq 1/2$.

  (a *YES* answer is always correct!)
- For correct answer being *YES*, we get misleading $k$ consecutive *NO*'s in $k$ runs with probability $\leq 2^{-k}$.

  (Receiving a *YES* would change our evaluation.)

# The Complexity Class *RP*

**Randomized Polynomial (*RP*) time.** Complexity class of problems for which a non-deterministic Turing machine:

- runs in poly-time w.r.t. the input size,
- if the correct answer is *NO* it returns *NO*,
- if the correct answer is *YES* it returns *YES* with probability $p \geq 1/2$.

  (a *YES* answer is always correct!)
- For correct answer being *YES*, we get misleading $k$ consecutive *NO*'s in $k$ runs with probability $\leq 2^{-k}$.

  (Receiving a *YES* would change our evaluation.)
- Class ***co-RP***: *NO* is always correct; *YES* might be incorrect.
- It holds: $P \subseteq RP \subseteq NP$.

# The Complexity Class *RP*

**Randomized Polynomial (*RP*) time.** Complexity class of problems for which a non-deterministic Turing machine:

- runs in poly-time w.r.t. the input size,
- if the correct answer is *NO* it returns *NO*,
- if the correct answer is *YES* it returns *YES* with probability $p \geq 1/2$.

  (a *YES* answer is always correct!)
- For correct answer being *YES*, we get misleading $k$ consecutive *NO*'s in $k$ runs with probability $\leq 2^{-k}$.

  (Receiving a *YES* would change our evaluation.)
- Class ***co-RP***: *NO* is always correct; *YES* might be incorrect.
- It holds: $P \subseteq RP \subseteq NP$.

Alternative definition: In *RP* the NTM accepts a constant fraction of the computation paths. (In *NP* we only need one accepting path.) This immediately shows that $RP \subseteq NP$.

# The Complexity Class *RP*

**Randomized Polynomial (*RP*) time.** Complexity class of problems for which a non-deterministic Turing machine:

- runs in poly-time w.r.t. the input size,
- if the correct answer is *NO* it returns *NO*,
- if the correct answer is *YES* it returns *YES* with probability $p \geq 1/2$.

  (a *YES* answer is always correct!)
- For correct answer being *YES*, we get misleading $k$ consecutive *NO*'s in $k$ runs with probability $\leq 2^{-k}$.

  (Receiving a *YES* would change our evaluation.)
- Class ***co-RP***: *NO* is always correct; *YES* might be incorrect.
- It holds: $P \subseteq RP \subseteq NP$.

Alternative definition: In *RP* the NTM accepts a constant fraction of the computation paths. (In *NP* we only need one accepting path.) This immediately shows that $RP \subseteq NP$.

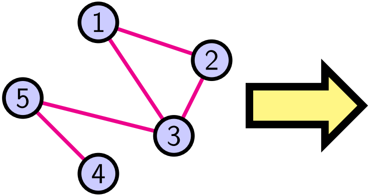
- Let us return to our problem now.

# An Intractability Result

**Theorem 7**

*If RP $\neq$ NP, the representation class of $k$-term DNF formulae is not efficiently PAC learnable for any $k \geq 2$.*

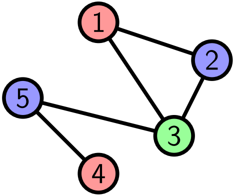
**Proof Idea:** Reduce Graph 3-Coloring problem to the problem of finding a consistent 3-term DNF formula with a sample $S_G = S_G^+ \cup S_G^-$.




$\langle(0,1,1,1,1),+\rangle$ $\langle(0,0,1,1,1),-\rangle$
$\langle(1,0,1,1,1),+\rangle$ $\langle(0,1,0,1,1),-\rangle$
$\langle(1,1,0,1,1),+\rangle$ $\langle(1,0,0,1,1),-\rangle$
$\langle(1,1,1,0,1),+\rangle$ $\langle(1,0,1,1,0),-\rangle$
$\langle(1,1,1,1,0),+\rangle$ $\langle(1,1,0,1,0),-\rangle$
$\langle(1,1,1,0,0),-\rangle$

- Positive examples encode the vertices of the given graph.
- Negative examples encode the edges of the given graph.
- Show: $G$ is 3-colorable *iff* $S_G$ is consistent with some 3-term DNF.

## $G$ is 3-colorable $\Rightarrow$ $S_G$ consistent with some 3-term DNF



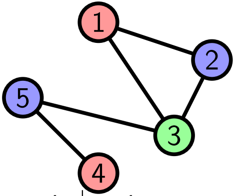

| node | color |
|---|---|
| 1 | *red* |
| 2 | *blue* |
| 3 | *green* |
| 4 | *red* |
| 5 | *blue* |

$$\begin{array}{ll} \langle(0,1,1,1,1),+\rangle & \langle(0,0,1,1,1),-\rangle \\ \langle(1,0,1,1,1),+\rangle & \langle(0,1,0,1,1),-\rangle \\ \langle(1,1,0,1,1),+\rangle & \langle(1,0,0,1,1),-\rangle \\ \langle(1,1,1,0,1),+\rangle & \langle(1,0,1,1,0),-\rangle \\ \langle(1,1,1,1,0),+\rangle & \langle(1,1,0,1,0),-\rangle \\ & \langle(1,1,1,0,0),-\rangle \end{array}$$

$$\Longrightarrow \left\{ \begin{array}{lcl} T_r & = & x_2 \wedge x_3 \wedge x_5 \\ T_b & = & x_1 \wedge x_3 \wedge x_4 \\ T_g & = & x_1 \wedge x_2 \wedge x_4 \wedge x_5 \end{array} \right. \Longrightarrow \varphi = T_r \vee T_b \vee T_g$$

- Consider a positive example $v(i) \in S_G^+$. Let $color(node\ i) = red$ (similar argument for other colors). Then, $T_r$ is a conjunction of non-red nodes, so $v(i)$ satisfies $T_r$ (and therefore $\varphi$).

# $G$ is 3-colorable $\Rightarrow$ $S_G$ consistent with some 3-term DNF




$$\begin{array}{ll}
\langle(0,1,1,1,1),+\rangle & \langle(0,0,1,1,1),-\rangle \\
\langle(1,0,1,1,1),+\rangle & \langle(0,1,0,1,1),-\rangle \\
\langle(1,1,0,1,1),+\rangle & \langle(1,0,0,1,1),-\rangle \\
\langle(1,1,1,0,1),+\rangle & \langle(1,0,1,1,0),-\rangle \\
\langle(1,1,1,1,0),+\rangle & \langle(1,1,0,1,0),-\rangle \\
 & \langle(1,1,1,0,0),-\rangle
\end{array}$$

| *node* | *color* |
|---|---|
| 1 | *red* |
| 2 | *blue* |
| 3 | *green* |
| 4 | *red* |
| 5 | *blue* |

$$\Longrightarrow \left\{ \begin{array}{lcl} T_r & = & x_2 \wedge x_3 \wedge x_5 \\ T_b & = & x_1 \wedge x_3 \wedge x_4 \\ T_g & = & x_1 \wedge x_2 \wedge x_4 \wedge x_5 \end{array} \right. \Longrightarrow \varphi = T_r \vee T_b \vee T_g$$

- Let $e(i,j) \in S_G^-$. A valid 3-coloring with nodes $i$ and $j$ connected by an edge implies that they have a different color. But $e(i,j)$ will falsify at least one of the variables in the term (say $T_r$) since at least one of the two nodes must have color other than red and is therefore included in the term $T_r$.

# $S_G$ consistent with some 3-term DNF $\Rightarrow$ $G$ is 3-colorable

Let $\varphi = \boldsymbol{T_r} \vee \boldsymbol{T_b} \vee \boldsymbol{T_g}$ be consistent with $S_G$.

We claim that the following coloring is valid:

- color node $i$ red if $v(i) \in S_G^+$ satisfies $T_r$.
- color node $i$ blue if $v(i) \in S_G^+$ satisfies $T_b$.
- color node $i$ green if $v(i) \in S_G^+$ satisfies $T_g$.
- (break ties arbitrarily if $v(i) \in S_G^+$ satisfies more than one term)

# $S_G$ consistent with some 3-term DNF $\Rightarrow$ $G$ is 3-colorable

Let $\varphi = \boldsymbol{T_r} \vee \boldsymbol{T_b} \vee \boldsymbol{T_g}$ be consistent with $S_G$.

We claim that the following coloring is valid:

- color node $i$ red if $v(i) \in S_G^+$ satisfies $T_r$.
- color node $i$ blue if $v(i) \in S_G^+$ satisfies $T_b$.
- color node $i$ green if $v(i) \in S_G^+$ satisfies $T_g$.
- (break ties arbitrarily if $v(i) \in S_G^+$ satisfies more than one term)

Since $\varphi$ is consistent with $S_G$, every $v(i) \in S_G^+$ satisfies some term $\Rightarrow$ every node is assigned a color.

# $S_G$ consistent with some 3-term DNF $\Rightarrow$ $G$ is 3-colorable

Let $\boldsymbol{\varphi} = \boldsymbol{T_r} \vee \boldsymbol{T_b} \vee \boldsymbol{T_g}$ be consistent with $S_G$.

We claim that the following coloring is valid:

- color node $i$ red if $v(i) \in S_G^+$ satisfies $T_r$.
- color node $i$ blue if $v(i) \in S_G^+$ satisfies $T_b$.
- color node $i$ green if $v(i) \in S_G^+$ satisfies $T_g$.
- (break ties arbitrarily if $v(i) \in S_G^+$ satisfies more than one term)

Since $\varphi$ is consistent with $S_G$, every $v(i) \in S_G^+$ satisfies some term $\Rightarrow$ every node is assigned a color.

- Suppose nodes $i$ and $j$ are assigned the same color (say red). Then both $v(i)$ and $v(j)$ satisfy term $T_r$. $\Rightarrow$ $x_i \notin T_r$ and moreover $\overline{x_i} \notin T_r$ because these two vectors satisfy $T_r$ and their $i$-th bit is 0 in one case and 1 in the other case.
- But $e(i,j)$ and $v(j)$ differ only in their $i$-th bit and if $v(j)$ satisfies $T_r$, so does $e(i,j)$. But then this means $e(i,j) \notin S_G^-$ since $\varphi$ is consistent with $S_G$. Therefore, $(i,j)$ is not an edge in $G$ as required.

# Why the Reduction is About *RP*?

- PAC learning should work for every small $\varepsilon$ and every small $\delta$.
- Work against this definition.
- If we have a sample $S$ of $m$ training examples (say, all distinct), a PAC learning algorithm should also be able to learn these $m$ examples to error $\varepsilon = \frac{1}{m+1}$ even when the distribution on these points is uniform; i.e., for every $(x, y) \in S$ it holds $\Pr_{x\sim\mathcal{D}}(x) = \frac{1}{m}$.

# Why the Reduction is About *RP*?

- PAC learning should work for every small $\varepsilon$ and every small $\delta$.
- Work against this definition.
- If we have a sample $S$ of $m$ training examples (say, all distinct), a PAC learning algorithm should also be able to learn these $m$ examples to error $\varepsilon = \frac{1}{m+1}$ even when the distribution on these points is uniform; i.e., for every $(x, y) \in S$ it holds $\Pr_{x \sim \mathcal{D}}(x) = \frac{1}{m}$.
- But then this means that the algorithm should create a **consistent** hypothesis with the training examples.

  (Otherwise the risk would be very large.)
- Per the PAC criterion, a consistent hypothesis will be created with high probability.
- This explains why we care about *RP*.

# Learning 3-Term DNF Formulae using 3-CNF Formulae

- We use the fact:

$$(u \wedge v) \vee (w \wedge z) = (u \vee w) \wedge (u \vee z) \wedge (v \vee w) \wedge (v \vee z)$$

- So, a 3-term DNF formula can be represented as a 3-CNF formula; i.e., a CNF formula where each clause has at most 3 literals.

$$T_1 \vee T_2 \vee T_3 = \bigwedge_{u \in T_1, v \in T_2, w \in T_3} (u \vee v \vee w)$$

- In general, this construction can take a $k$-term DNF formula and represent it with a $k$-CNF formula.
- Reduce the problem of learning a $k$-CNF formula to learning conjunctions:
  - For every triple $(u, v, w)$ over the original variables $\{x_1, \ldots, x_n\}$, create a variable $y_{u,v,w}$ corresponding to this triple.
  - Hence number of variables $y_{u,v,w}$ is at most $(2n)^3$, which is $O(n^3)$.
    (For k-term DNF the corresponding $y$'s will be $O(n^k)$ in total.)

## Learning 3-Term DNF Formulae using 3-CNF Formulae

- 3-CNF over $\{x_1, \ldots, x_n\}$ is equivalent to a 3-CNF over the new variables $\{y_{u,v,w}\}$.

So:

- A truth assignment $\sigma \in \{0,1\}^n$ corresponding to the variables $\{x_1, \ldots, x_n\}$ can be converted in time $O(n^3)$ to a truth assignment corresponding to the variables $\{y_{u,v,w}\}$.
- So, we can run our algorithm for learning conjunctions in polynomial time over the variables $\{y_{u,v,w}\}$.
    - Find-S may run in time $O(mn)$; for $m$ examples of bitsize $n$ each.
    - In the new setting: $n' \mapsto (2n)^3$ and $m' \approx O(n') = O(n^3)$.
- Once we are done learning, we can convert the solution that uses the variables $\{y_{u,v,w}\}$ back to $\{x_1, \ldots, x_n\}$ by simply expanding each variable $\{y_{u,v,w}\}$ to the clause $(u \vee v \vee w)$.

# Learning 3-Term DNF Formulae using 3-CNF Formulae

Finally, we need to argue that the solution that we compute indeed has low risk.

- Let $c$ be the target 3-CNF and $\mathcal{D}$ the target distribution over $\{0,1\}^n$.
- Let $c'$ be the target 3-CNF using the variables $\{y_{u,v,w}\}$ and $\mathcal{D}'$ the (induced) distribution over the assignments to the $\{y_{u,v,w}\}$ variables.
- We need to argue that if $h'$ has risk less than $\varepsilon$, so does $h$.
  - For $\sigma_1, \sigma_2 \in \{0,1\}^n$ with $\sigma_1 \neq \sigma_2$, it follows that we have $\sigma_1' \neq \sigma_2'$.
  - So, $h'(\sigma') \neq c'(\sigma') \Rightarrow$ there is a *unique preimage* $\sigma \in \{0,1\}^n$ such that $h(\sigma) \neq c(\sigma)$ and the weight of $\sigma$ under $\mathcal{D}$ is the same as that of $\sigma'$ under $\mathcal{D}'$.

(We have used the fact that *our algorithm learns under any distribution*.)

- For example, let $\mathcal{D}$ be the uniform distribution over $\{0,1\}^n$; i.e., each variable in the truth assignment is satisfied with probability $1/2$.
- Under $\mathcal{D}'$, a variable $y_{u,v,w}$ corresponding to the clause $(u \vee v \vee w)$ is satisfied with probability $7/8$. Similarly, $y_{u,u,u}$ is satisfied with probability $1/2$, or $y_{u,u,\overline{u}}$ is satisfied with probability $1$.

# How many examples are enough?

What if $|\mathcal{H}| = \infty$?

# Different Classifications and the Growth Function

- $\mathrm{x} = (x_1, x_2, \ldots, x_m)$ is a set of $m$ examples.

Number of Classifications $\Pi_{\mathcal{H}}(\mathrm{x})$ of x by $\mathcal{H}$: Distinct vectors $(\mathrm{h}(x_1), \mathrm{h}(x_2), \ldots, \mathrm{h}(x_m))$ as h runs through $\mathcal{H}$.

- $\Pi_{\mathcal{H}}(\mathrm{x}) \leq 2^m$.

# Different Classifications and the Growth Function

- $\mathrm{x} = (x_1, x_2, \ldots, x_m)$ is a set of $m$ examples.

**Number of Classifications $\Pi_{\mathcal{H}}(\mathrm{x})$ of x by $\mathcal{H}$:** Distinct vectors $(\mathrm{h}(x_1), \mathrm{h}(x_2), \ldots, \mathrm{h}(x_m))$ as h runs through $\mathcal{H}$.

- $\Pi_{\mathcal{H}}(\mathrm{x}) \leq 2^m$.

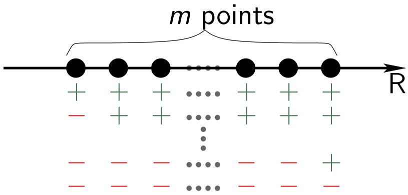
**Growth Function:** $\Pi_{\mathcal{H}}(m) = \max\{\Pi_{\mathcal{H}}(\mathrm{x}) \ : \ \mathrm{x} \in X^m\}$ .

### Example 8

Rays on a line:

$$\mathrm{h}_\vartheta(x) = \begin{cases} + & , \quad \text{if } x \geq \vartheta \\ - & , \quad \text{otherwise} \end{cases}$$

$\Pi_{\mathcal{H}}(m) = m + 1$ .



# The Vapnik-Chervonenkis Dimension

**Definition 9**

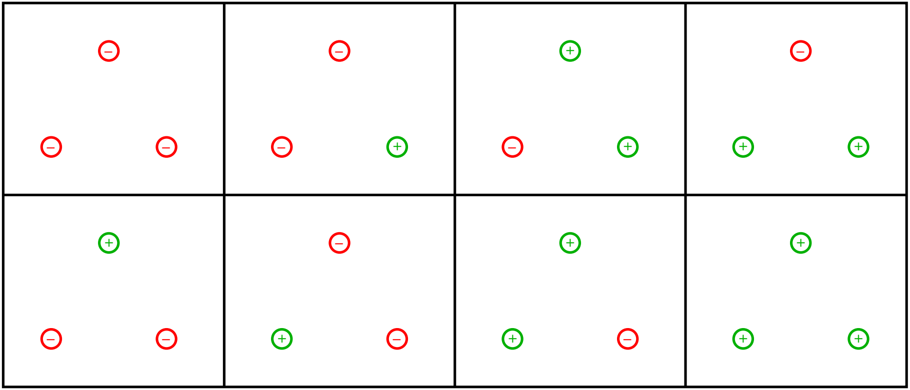
A sample x of size $m$ is *shattered* by $\mathcal{H}$, or $\mathcal{H}$ *shatters* x, if $\mathcal{H}$ can give all $2^m$ possible classifications of x.

**Definition 10 (*VC* dimension)**

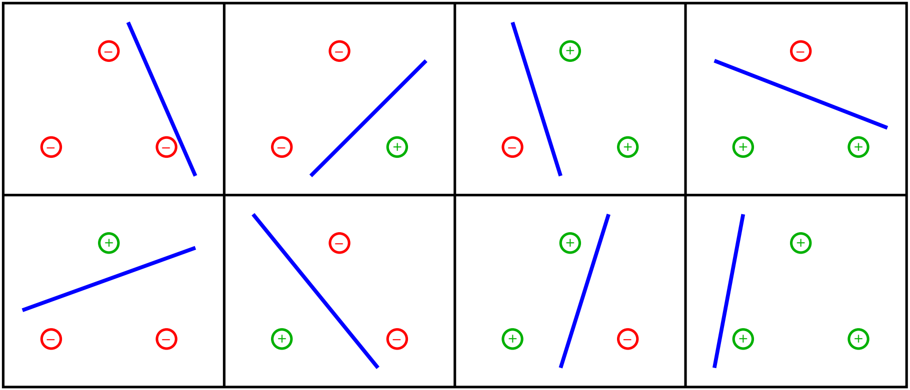
$$VC\text{-}dim(\mathcal{C}) = \max\{m \ : \ \Pi_{\mathcal{C}}(m) = 2^m\}$$

- Our ray example has $VC\text{-}dim(\text{Rays}) = 1$.
  - One point is shattered.
  - Two points are not shattered $(+, -)$
- Lower Bound $\Longrightarrow$ Explicit construction that achieves $2^m$.
- Upper Bound $\Longrightarrow$ For *any* sample x of length $m$ we can not achieve $2^m$.

# Configurations of 3 Points in 2D



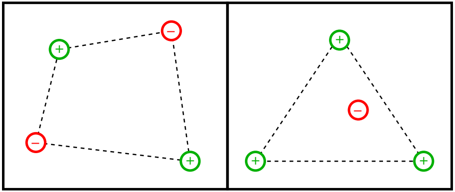
# Halfspaces Shatter 3 Points in 2D



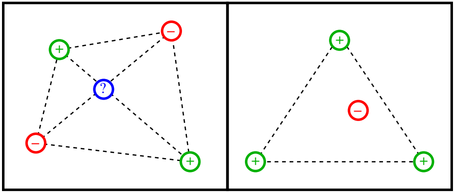

**Question 1**

*Can we shatter 4 points ?*

# Can Halfspaces Shatter 4 Points in 2D?



# Halfspaces can *not* Shatter 4 Points in 2D




### Theorem 11 (Radon)

*Any set of $d + 2$ points in $\mathbb{R}^d$ can be partitioned into two (disjoint) sets whose convex hulls intersect.*

### Corollary 12

- $VC\text{-}dim\,(HALFSPACES) = 3$ *in* $2$ *dimensions.*
- $VC\text{-}dim\,(HALFSPACES) = d + 1$ *in* $d \geq 1$ *dimensions.*

# Sauer's Lemma

**Lemma 13 (Sauer's Lemma)**

*Let $d \geq 0$ and $m \geq 1$ be given integers and let $\mathcal{H}$ be a hypothesis space with VC-dim$(\mathcal{H}) = d$. Then*

$$\Pi_{\mathcal{H}}(m) \leq 1 + \binom{m}{1} + \binom{m}{2} + \cdots + \binom{m}{d} = \Phi(d, m).$$

**Proposition 1**

*For all $m \geq d \geq 1$, $\Phi(d, m) < \left(\frac{em}{d}\right)^d$ .*

# VC Dimension: How Many Examples are Enough for Learning (Distribution Independently)?

**Theorem 14**

*Let $\mathcal{C}$ have finite VC-dim$(\mathcal{C}) = d \geq 1$ and moreover let $0 < \delta, \varepsilon < 1$. Then,*

$$m \geq \left\lceil \frac{4}{\varepsilon} \cdot \left( d \cdot \lg\left(\frac{12}{\varepsilon}\right) + \lg\left(\frac{2}{\delta}\right)\right)\right\rceil$$

*samples guarantee that any consistent hypothesis has small error with high probability (in the PAC-learning sense).*

- We still need an efficient algorithm to efficiently PAC-learn the class.

# VC Dimension: How Many Examples are Necessary for Learning (Distribution Independently)?

**Theorem 15**

*Any algorithm for PAC-learning a concept class of VC dimension $d$ with parameters $\varepsilon < 1/16$ and $\delta \leq 1/15$, must use*

$$m > \frac{d-1}{64\varepsilon}$$

*training examples in the worst case.*

# Proving the Lower Bound

Let $X = \{x_1, \ldots, x_d\}$ be shattered by $\mathcal{C}$.

- Construct a pathological distribution that forces any algorithm to take many examples.
- $supp(\mathcal{D}) = X \Rightarrow$ w.l.o.g. $\mathcal{C} = \mathcal{C}(X)$, so $\mathcal{C}$ is a finite class, $|\mathcal{C}| = 2^d$.
- Choosing a $c$ from $\mathcal{C}$ is equivalent to tossing a fair coin $d$ times to determine the labeling on $X$.
- Suppose there is a learning algorithm $\mathcal{A}$ that uses at most $m = \left\lceil \frac{d-1}{64\varepsilon} \right\rceil$ training examples producing a hypothesis $h$.
- Want to show:
  $(\exists \mathcal{D} \text{ on } X)(\exists c \in \mathcal{C}) \left[\Pr_{S\sim\mathcal{D}^m}\left(\mathrm{Risk}_{\mathcal{D}}\left(\mathrm{h}, \mathrm{c}\right) > \varepsilon\right) > 1/15\right]$.

# Proving the Lower Bound

- Define $\mathcal{D}$ independently of $\mathcal{A}$:
$$\begin{cases} \Pr(x_1) & = & 1 - 16\varepsilon \\ \Pr(x_2) & = & \Pr(x_3) = \ldots = \Pr(x_d) = \frac{16\varepsilon}{d-1} \end{cases}$$
- Let $X' = \{x_2, x_3, \ldots, x_d\}$.
- Let $\mathrm{Risk}'_{\mathcal{D}}(h, c) = \Pr_{x \sim \mathcal{D}}(h(x) \neq c(x) \wedge x \in X')$.

Note that

$$\begin{aligned} \mathrm{Risk}_{\mathcal{D}}(h, c) &= \Pr_{x \sim \mathcal{D}}(h(x) \neq c(x)) \\ &\geq \Pr_{x \sim \mathcal{D}}(h(x) \neq c(x) \wedge x \in X') \\ &= \mathrm{Risk}'_{\mathcal{D}}(h, c) . \end{aligned}$$

- It is easier to prove $\Pr_{S \sim \mathcal{D}^m}\left(\mathrm{Risk}'_{\mathcal{D}}(\mathrm{h}, \mathrm{c}) > \varepsilon\right) > 1/15$.
  - But then the result follows from the above observation.

# Proving the Lower Bound

**Probabilistic argument:** Pick a random $c \in \mathcal{C}$ and show that $c$ is hard to learn for $\mathcal{A}$ with positive probability. This implies that there is at least one $c \in \mathcal{C}$ that is hard to learn for $\mathcal{A}$.

**Idea:** Argue that the sample $S$ containing $m$ iid examples from $\mathcal{D}$, will miss more than half of the points from $X'$.

- $h$ will be 'guessing' the labels for these points $\Rightarrow$ inevitable to have large risk under $\mathcal{D}$.
- Expected # of instances from $X'$ appearing in $S$:
  $\mu = \left[\frac{16\varepsilon}{d-1} \cdot (d-1)\right] \cdot \left(\frac{d-1}{64\varepsilon}\right) = \frac{d-1}{4}$.
- Markov $\Rightarrow$ Pr $\left(\text{\# of instances from } X' \text{ in } S \geq \frac{d-1}{2}\right) \leq \frac{\frac{d-1}{4}}{\frac{d-1}{2}} = 1/2$.
- Define the **bad event**

$$B \equiv S \text{ contains less than } \tfrac{d-1}{2} \text{ instances from } X' \,.$$

By the above,

$$\Pr_{S\sim\mathcal{D}^m}(B) = 1 - \Pr_{S\sim\mathcal{D}}\left(\text{\# instances from } X' \text{ in S} \geq \frac{d-1}{2}\right) \geq \frac{1}{2}\,. \qquad (1)$$

# Proving the Lower Bound

- $h$ is independent of $X' \setminus S$
- we pick $c \in \mathcal{C}$ at random

So, $h$ will make a mistake on each instance $x \in X' \setminus S$ with probability $1/2$.

- Each instance $x \in X' \setminus S$ contributes to $\text{Risk}'_{\mathcal{D}}(h, c)$ an amount of $\frac{1}{2} \cdot \frac{16\varepsilon}{(d-1)}$.
- When the bad event $B$ occurs, we have $|X' \setminus S| > \frac{d-1}{2}$.

This implies

$$\mathrm{E}_{c,S}\left[\text{Risk}'_{\mathcal{D}}(h, c) \ \mid \ B\right] > 4\varepsilon\,. \tag{2}$$

- By (1) and (2) we get a lower bound on $\mathrm{E}_{c,S}\left[\text{Risk}'_{\mathcal{D}}(h, c)\right]$:

$$\mathrm{E}_{c,S}\left[\text{Risk}'_{\mathcal{D}}(h, c)\right] \geq \mathrm{E}_{c,S}\left[\text{Risk}'_{\mathcal{D}}(h, c) \ \mid \ B\right] \cdot \Pr_S(B) > (4\varepsilon) \cdot (1/2) = 2\varepsilon\,.$$

(We used $\mathrm{E}[Y] = \sum_i \mathrm{E}[Y \mid A_i] \cdot \Pr(A_i)$, where $A_i$: finite or countable partition of the sample space.)

# Proving the Lower Bound

$\mathrm{E}_{c,S}\left[\mathrm{Risk}'_{\mathcal{D}}(h,c)\right] > 2\varepsilon \Longrightarrow (\exists c^\star \in \mathcal{C})\ \left[\mathrm{E}_S\left[\mathrm{Risk}'_{\mathcal{D}}(h,c^\star)\right] > 2\varepsilon\right]$.

- Take that $c^\star$ as the target concept.
- Show that $\mathcal{A}$ will be prone to produce an $h$ with large risk.

$\mathrm{Risk}'_{\mathcal{D}}(h,c) = \mathrm{Pr}_{x\sim\mathcal{D}}(h(x) \neq c(x) \wedge x \in X') \leq \mathrm{Pr}_{x\sim\mathcal{D}}(x \in X') = 16\varepsilon$.
So,

$$\mathrm{E}_S\left[\mathrm{Risk}'_{\mathcal{D}}(h,c) \;\middle|\; \mathrm{Risk}'_{\mathcal{D}}(h,c) > \varepsilon\right] \leq 16\varepsilon\,.$$

Therefore,

$$\begin{aligned}
2\varepsilon &< \mathrm{E}_S\left[\mathrm{Risk}'_{\mathcal{D}}(h,c)\right] \\
&= \mathrm{Pr}_S\left(\mathrm{Risk}'_{\mathcal{D}}(h,c) > \varepsilon\right)\cdot \mathrm{E}_S\left[\mathrm{Risk}'_{\mathcal{D}}(h,c) \;\middle|\; \mathrm{Risk}'_{\mathcal{D}}(h,c) > \varepsilon\right] \\
&\quad + (1 - \mathrm{Pr}_S\left(\mathrm{Risk}'_{\mathcal{D}}(h,c) > \varepsilon\right))\cdot \mathrm{E}_S\left[\mathrm{Risk}'_{\mathcal{D}}(h,c) \;\middle|\; \mathrm{Risk}'_{\mathcal{D}}(h,c) \leq \varepsilon\right] \\
&\leq \mathrm{Pr}_S\left(\mathrm{Risk}'_{\mathcal{D}}(h,c) > \varepsilon\right)\cdot(16\varepsilon) + (1 - \mathrm{Pr}_S\left(\mathrm{Risk}'_{\mathcal{D}}(h,c) > \varepsilon\right))\cdot(\varepsilon) \\
&= 15\varepsilon\cdot \mathrm{Pr}_S\left(\mathrm{Risk}'_{\mathcal{D}}(h,c) > \varepsilon\right) + \varepsilon\,.
\end{aligned}$$

In other words, $\mathrm{Pr}_S\left(\mathrm{Risk}'_{\mathcal{D}}(h,c) > \varepsilon\right) > \frac{1}{15}$.