Complexity: beyond space and time

AARON GEELON SO | AUGUST 7, 2019

Abstract

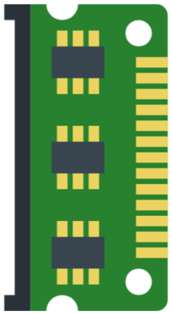
Background and roadmap

Goal: pique your curiosity in trade offs and techniques that may enable faster, less memory-intensive, and less data-intensive computing. The principal way we investigate this is through randomized algorithms.

Outline:

Anatomy of a Computation

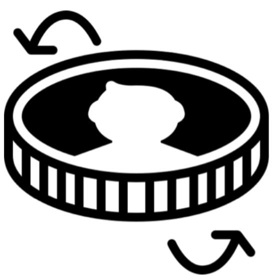
Resources and constraints



SPACE

TIME

DATA



RANDOMNESS

FAILURE

CORRECTNESS

COMMUNICATION

Tradeoff Analysis

Autonomous vehicles

Time: how quickly must the car be able to make decisions?

Space: how much memory can be practically given to a car?

Data: how much real-world training data is required?

Randomness: how much truly independent randomness is needed to provide guarantees?

Communication: if cars need to talk with each other to make decisions, how much is needed?

Failure: how unlikely do we want a car to fail to make the correct decision?

Correctness: what is the error tolerance when driving between lane lines?

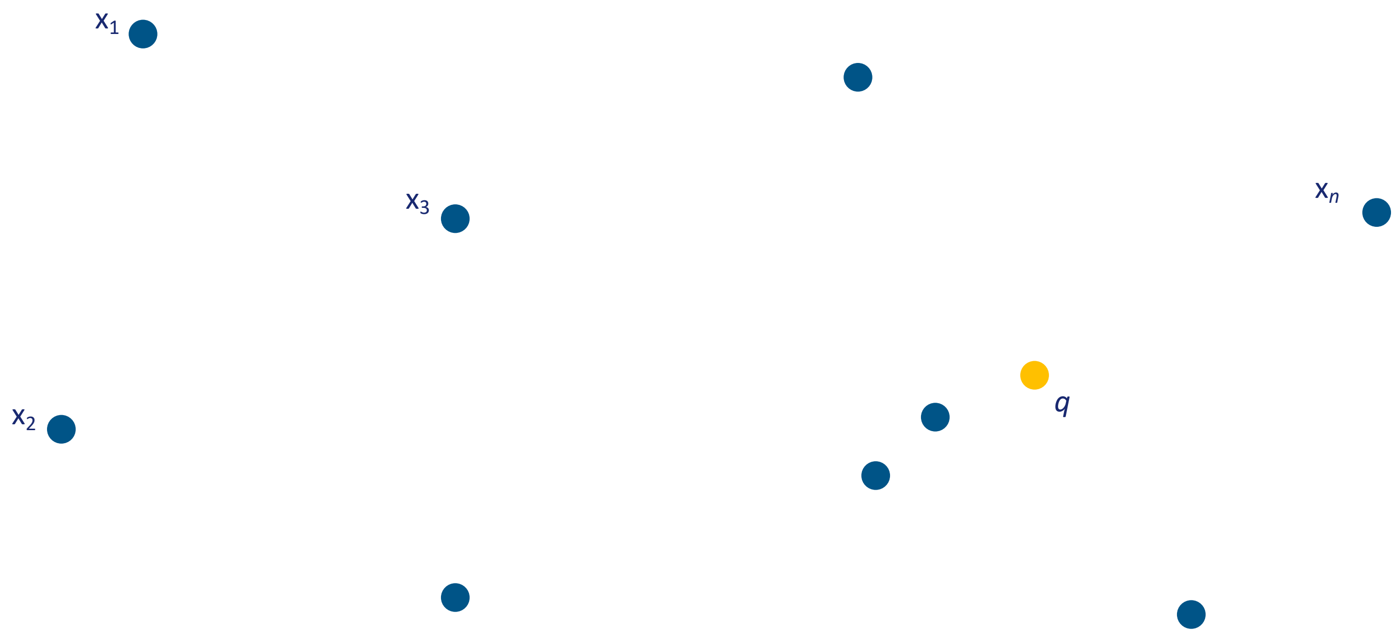
Anatomy of a Computation

Aspects of a viable computation summarized

- **Space:** amount of physical memory
- **Time:** speed of the computation
- **Data:** information required from the real world
- **Communication:** information shared/revealed to world
- **Correctness:** degree of accuracy demanded
- **Failure:** probability of arbitrary failure tolerated
- **Randomness:** access to random bits in the world

Space and Time Tradeoff

A simple example



Objective: There are *n* homes (blue) in the database; given a new home *q* (yellow), we want to query the database to determine which home is the nearest/most comparable.

Nearest Neighbor Search

Problem statement

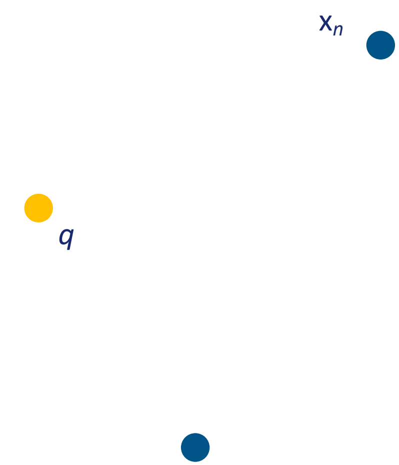
Formal Statement:

- let $X = \{x_1, \ldots, x_n\} \subset \mathbf{R}^d$ be the database with n homes
- let $Q = \{q_1, \ldots, q_m\} \subset \mathbf{R}^d$ be the set of all possible queries

Objective: Given a query $q \in Q$, compute:

$$\text{NearestNeighbor}(q) \equiv \text{argmin}\ d(x_i, q),$$

where d is some notion of distance between homes, say the standard Euclidean distance in $\mathbf{R}^d$.



Nearest Neighbor Search

Naïve solution 1: linear search over database

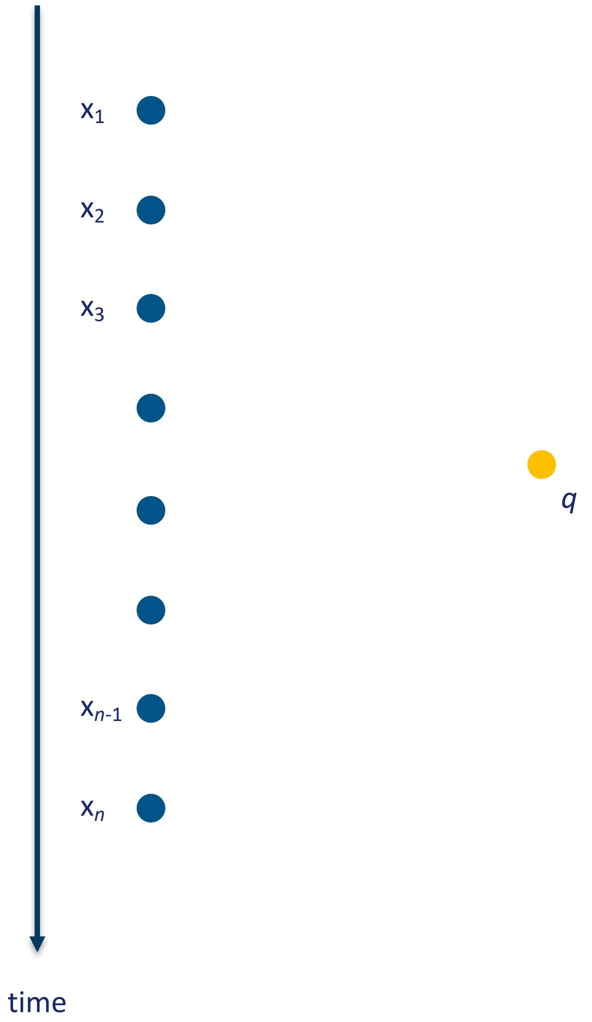
Algorithm (nearest neighbor linear search):

For each of the *n* homes x_i in the database:

- compute the distance $d(x_i, q)$
- return the closest home x_{i*}

Analysis:

- space complexity: O(dn)
- time complexity: O(dn)



Nearest Neighbor Search

Naïve solution 2: precomputing answers

Algorithm (nearest neighbor precomputed):

Preprocessing step:

- for each possible query q_j, compute the nearest home and store result in an array

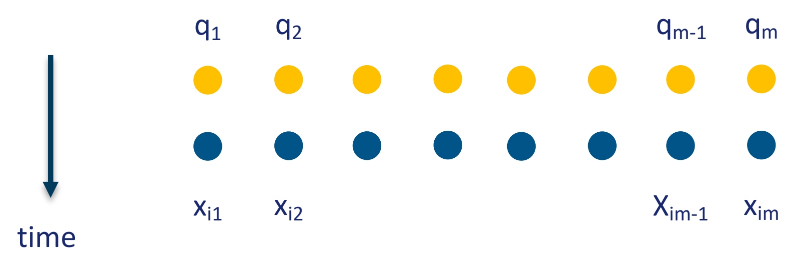
At query time, given *q*:

- lookup the precomputed answer

Analysis:

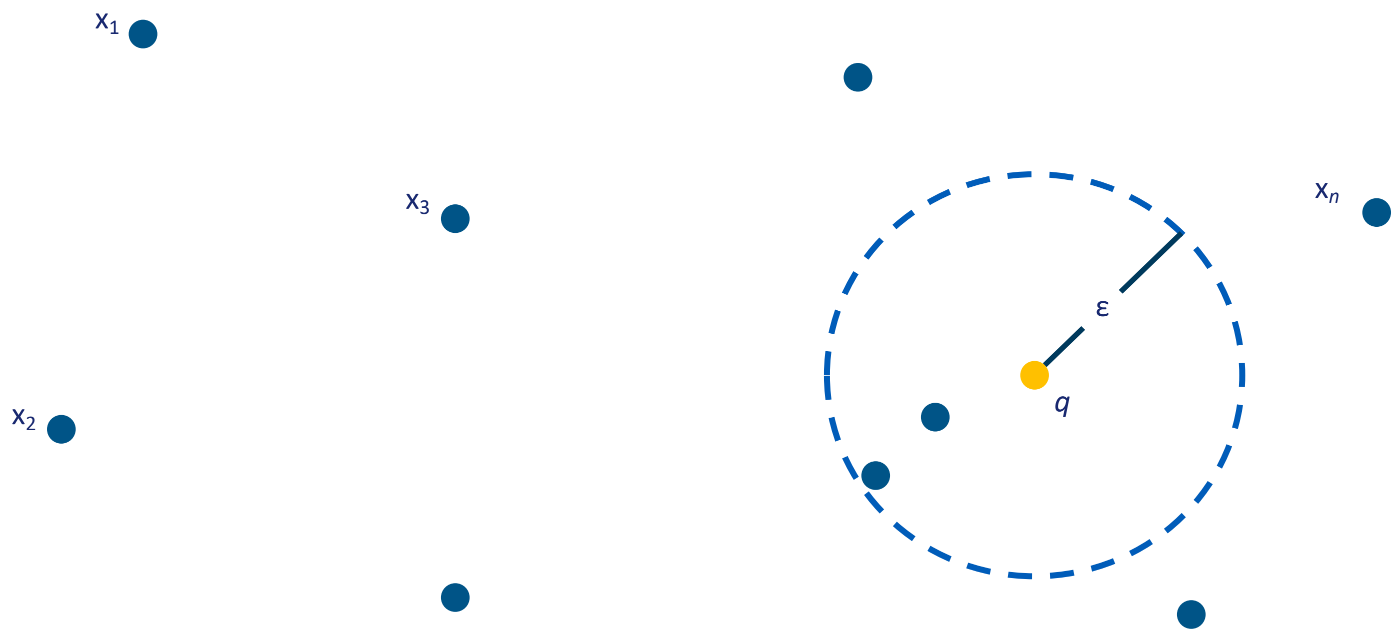
- space complexity: O(dm)
- time complexity*: O(1)

*the preprocessing time complexity is O(nm)



Other Tradeoffs

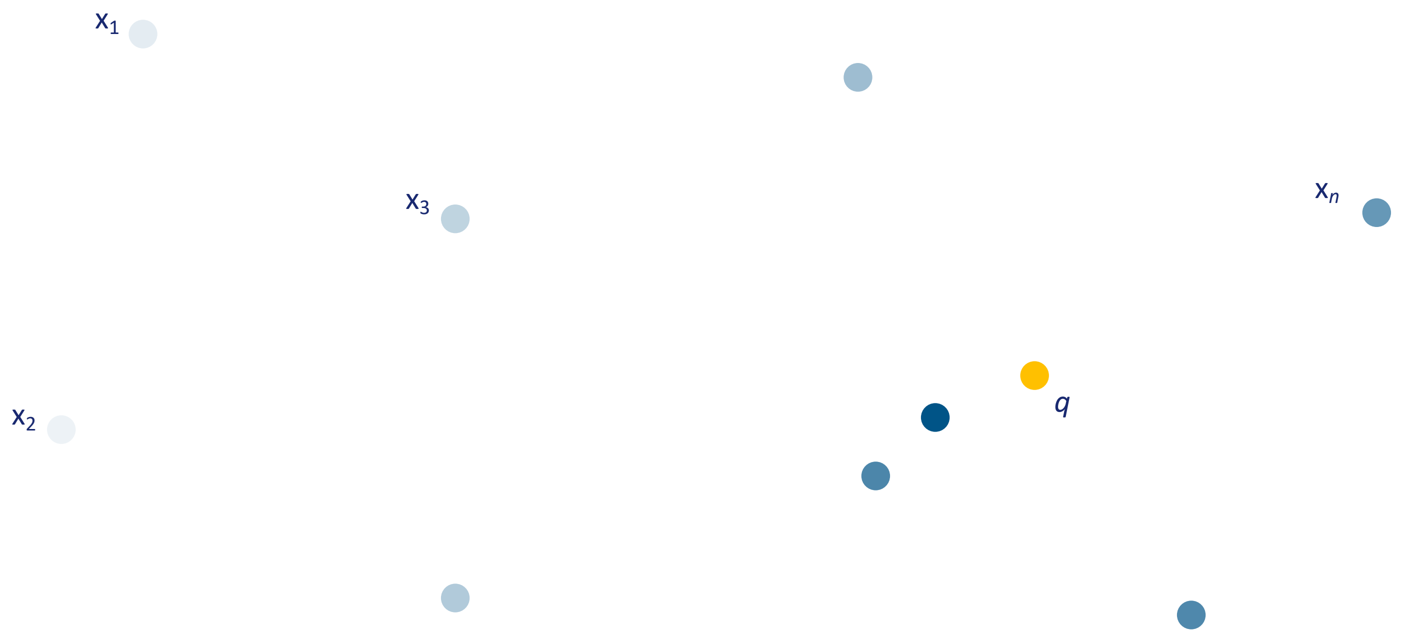
Approximations



If we are able to be error tolerant, we can relax our notion of correctness to allow for an approximate nearest neighbor.

Other Tradeoffs

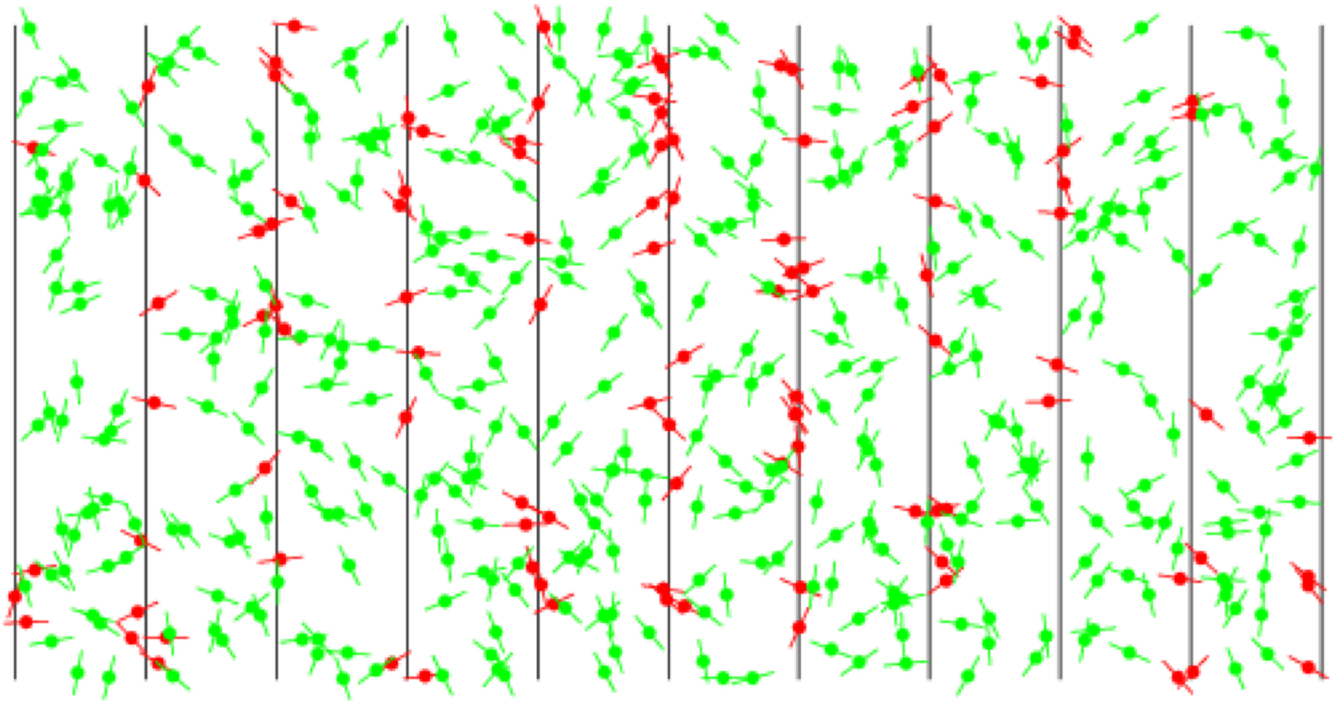
Failure probability



If we are failure tolerant, it might not matter if our algorithm returns an arbitrarily incorrect answer (e.g. x_2 in this case) some small δ fraction of the time.

Randomized Algorithms

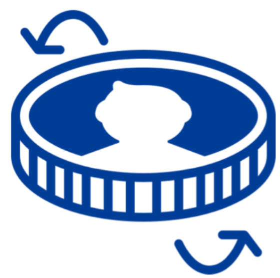
Introduction



Buffon's needle problem: $\Pr[\text{needle crosses line}] = \frac{2}{\pi}$

Resource Tradeoff

Communication, randomness, and failure



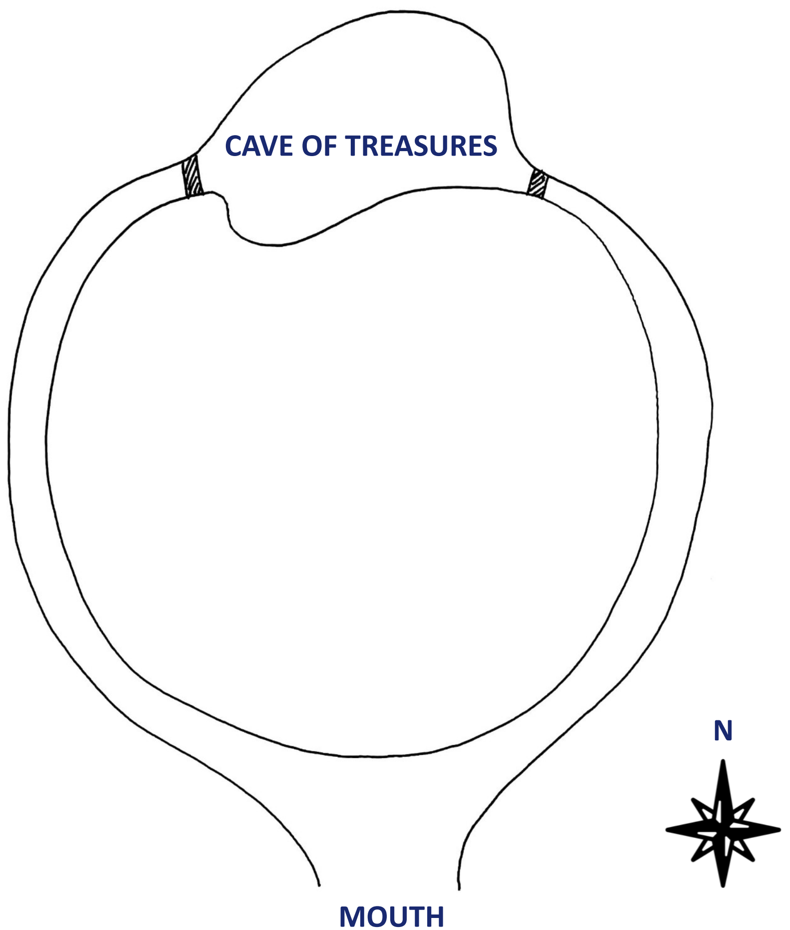
The Strange Cave of Ali Baba

Warm-up problem

Setup: We need to prove to Ali Baba that we can access the Cave of Treasures.

But, we don't want to tell him the password nor show the treasures within.

Can we convince Ali Baba?

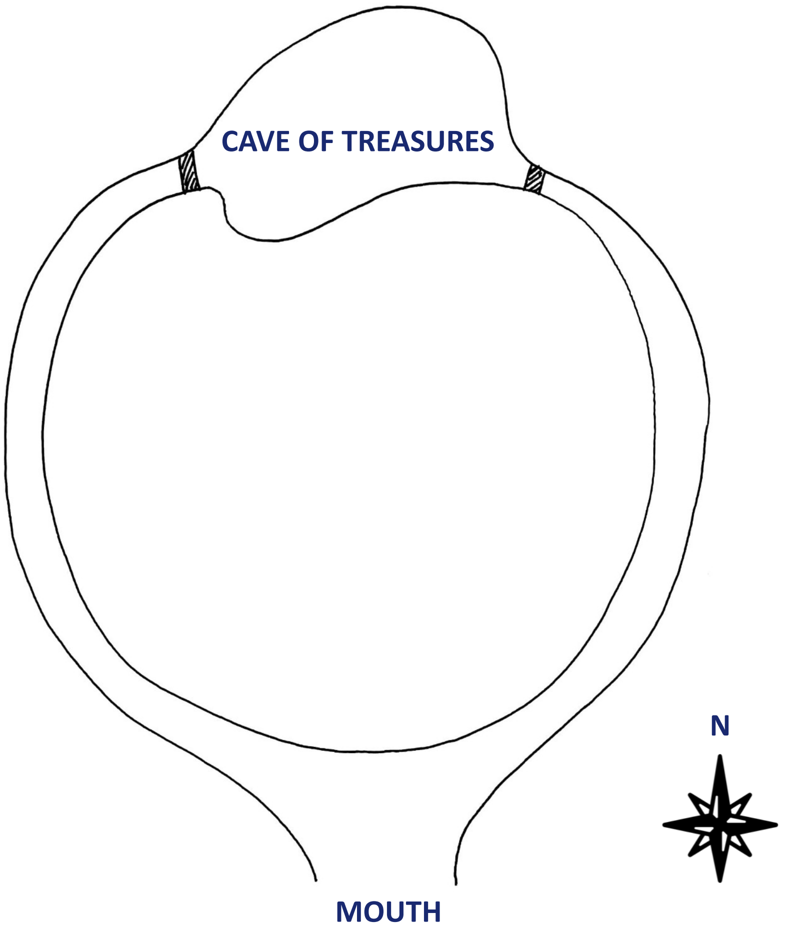


The Strange Cave of Ali Baba

Warm-up problem

Solution: we proceed into the depths of the cave, and tell Ali Baba to stand at its mouth.

Ali Baba flips a coin to choose right or left at random. He shouts into the cave, telling us to exit from either the right/left branch.



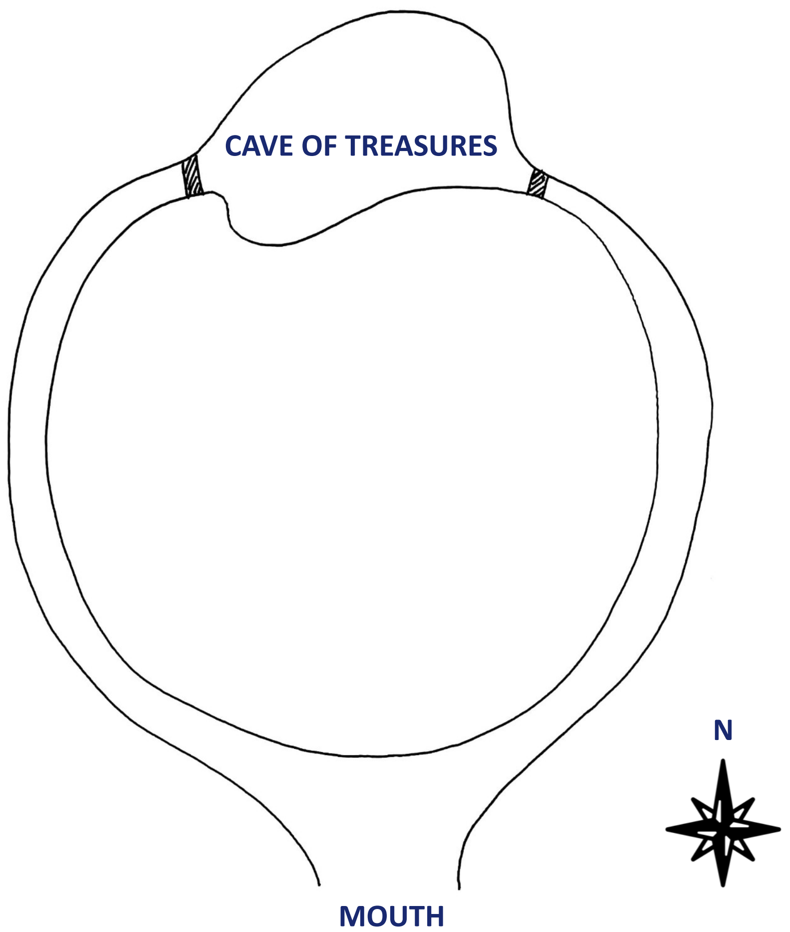
The Strange Cave of Ali Baba

Warm-up problem

Question: if we didn't actually know the password to traverse the cave, what is the probability that we get lucky and happen to be in the correct branch to begin with?

Answer: the probability that this *protocol* fails (i.e. we convince Ali Baba that we know the password even when we didn't) is:

$$\Pr[\text{failure}] = 0.5$$



The Strange Cave of Ali Baba

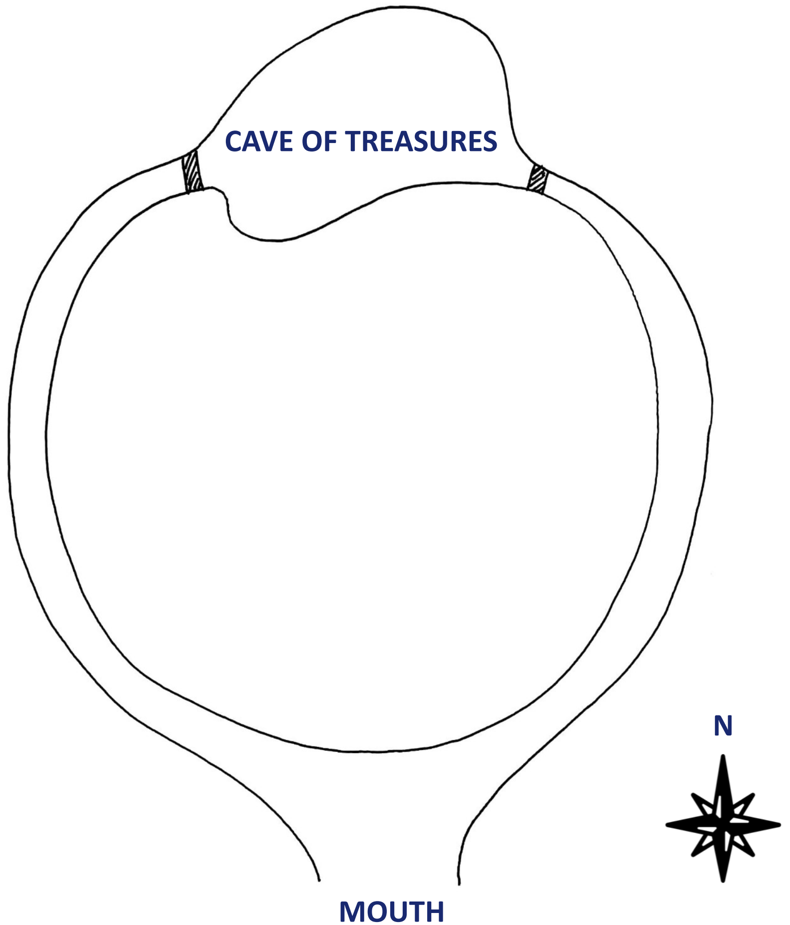
Warm-up problem

Question: can we *boost* the success rate?

Answer: if we repeat the protocol k times,

$$\Pr[\text{failure}] = 0.5^k$$

which can be made astronomically small.



The Strange Cave of Ali Baba

Moral of the story

Boosting technique: if we have a random algorithm that succeeds $(1 - \delta)$-fraction of the time, we can construct another random algorithm that succeeds $(1 - \delta^k)$-fraction of the time by performing k independent trials.

Tradeoffs: by expending more resources (e.g. in Ali Baba's cave: randomness, communication), we can decrease the rate of failure of our algorithm.

The Strange Cave of Ali Baba

Epilogue

Zero-knowledge proofs: a field in cryptography where a party can prove to another party that a statement is true without revealing the content/witness of the proof.

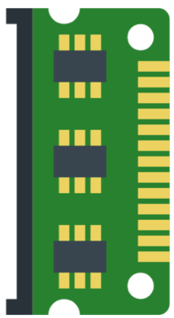
Applications:

- verify user identity without transmitting password [BM92]
- verify authenticity of nuclear warheads without revealing weapons design for arms control agreement [P+16]

Acknowledgments: the example of the Strange Cave of Ali Baba was taken from [Q+98].

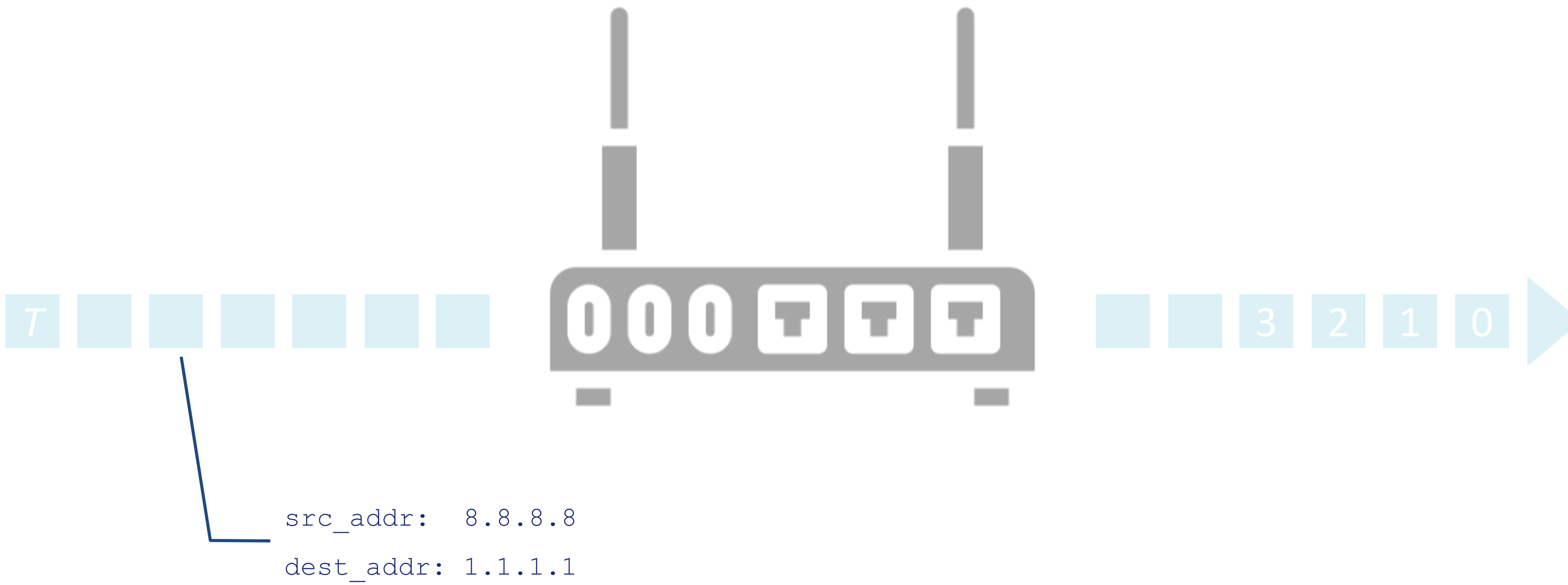
Resource Tradeoff

Space, correctness, and failure



Streaming and Sketching

Computation on massive data



Streaming and Sketching

Computation on massive data

IP Address	Number of Packets
1.2.3.4	42,320,564
10.2.1.78	2,301
53.23.0.0	576
…	
100.2.4.127	124,893,381

Frequency table: T packets stream through a router, originating from n IP addresses. This table shows the number of packets from each address.

Streaming and Sketching

Computation on massive data

Frequency vector: v =

Number of Packets
42,320,564
2,301
576
…
124,893,381

Objective: Compute the **second moment** of *v*:

$$F_2 = \sum_{i=1}^{n} v_i^2$$

The second moment can be used to compute variance, Gini's index of homogeneity, etc.

Streaming and Sketching

Naïve solution: store frequency vector

Data structure (dictionary):

For each of the n IP addresses, maintain a counter. After streaming T packets, compute F_2.

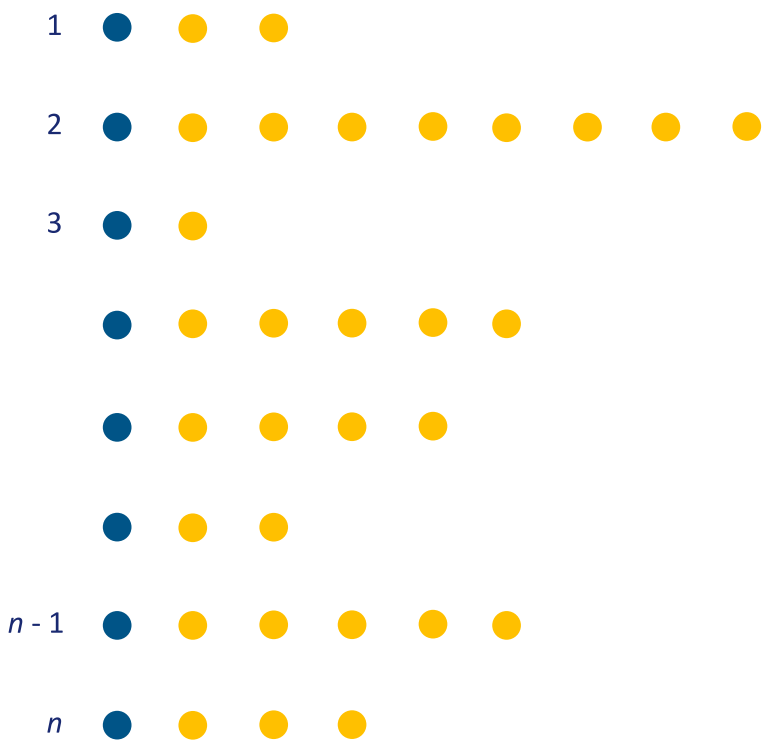
```
def count_packets():
    # initialize frequency table
    frequency_table = dict()

    # get first packet
    packet = get_packet()

    while packet is not None:
        # update frequency table
        if packet in frequency_table:
            frequency_table[packet] += 1
        else:
            frequency_table[packet] = 1

        # get next packet
        packet = get_packet()

    return frequency_table
```



Streaming and Sketching

Naïve solution: store frequency vector

Analysis:

- space complexity: O(n log T)

```python
def count_packets():
    # initialize frequency table
    frequency_table = dict()

    # get first packet
    packet = get_packet()

    while packet is not None:
        # update frequency table
        if packet in frequency_table:
            frequency_table[packet] += 1
        else:
            frequency_table[packet] = 1

        # get next packet
        packet = get_packet()

    return frequency_table
```

Question: can we use less space?

Answer: yes, we can **approximate** the true second moment **with high probability**.

Streaming and Sketching

Interlude 1: computing the first moment

Question: how much space is required if we want to compute the **first moment** of *v*, denoted by F_1?

$$F_1 = \sum_{i=1}^{n} v_i$$

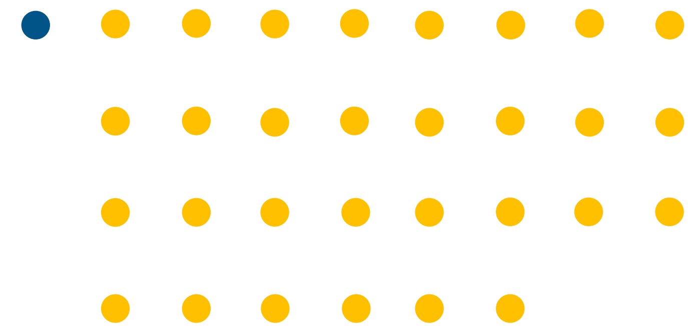
Answer: we just need a single counter

- space complexity: O(log T)

```python
def count_packets():
    # initialize counter
    num_packets = 0

    # stream packets
    while get_packet() is not None:
        num_packets += 1

    return num_packets
```



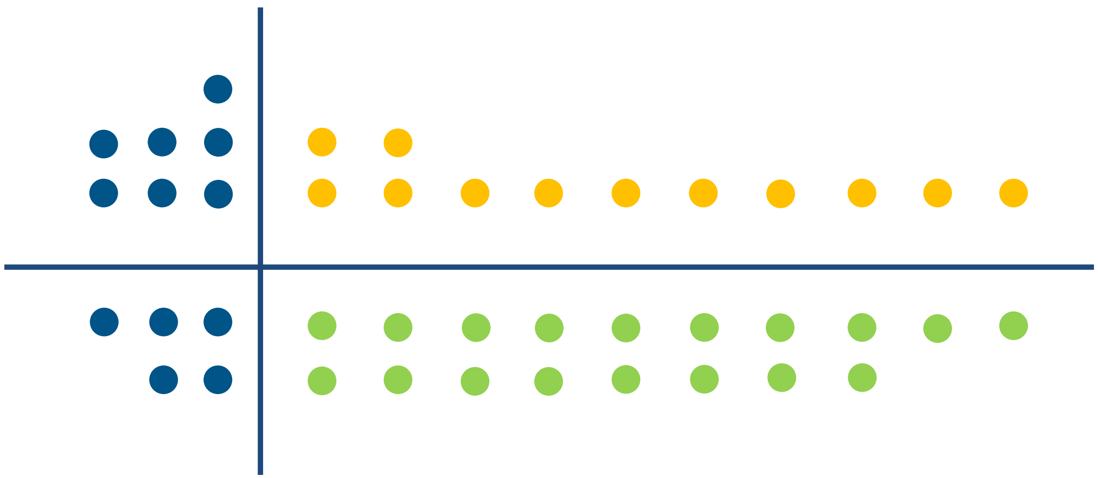
Let us now return to estimating F_2.

Streaming and Sketching

Tug-of-war solution: using randomness

Algorithm [AMS96]: $F_2 \equiv \sum_{i=1}^{n} v_i^2$

1. Assign each IP address into one of two ‘teams’ independently and uniformly at random.
2. Count the number of packets sent by the two teams respectively.
3. Square the difference in number of packets sent by the two teams.



$$\hat{F}_2 \equiv (\ \blacksquare\ -\ \blacksquare\ )^2$$

Streaming and Sketching

Tug-of-war solution: using randomness

Analysis: $F_2 \equiv \sum_{i=1}^{n} v_i^2$

For each IP address *i*, we randomly assign *i* to one of two teams: $r_i \in_R \{+1, -1\}$.

Then, our estimator is just:

$$\hat{F}_2 \equiv \left(\sum_{i=1}^{n} r_i v_i\right)^2 \equiv \sum_{i=1}^{n} r_i^2 v_i^2 + \sum_{i \neq j} r_i r_j v_i v_j$$

$$\equiv F_2 + \sum_{i \neq j} r_i r_j v_i v_j$$

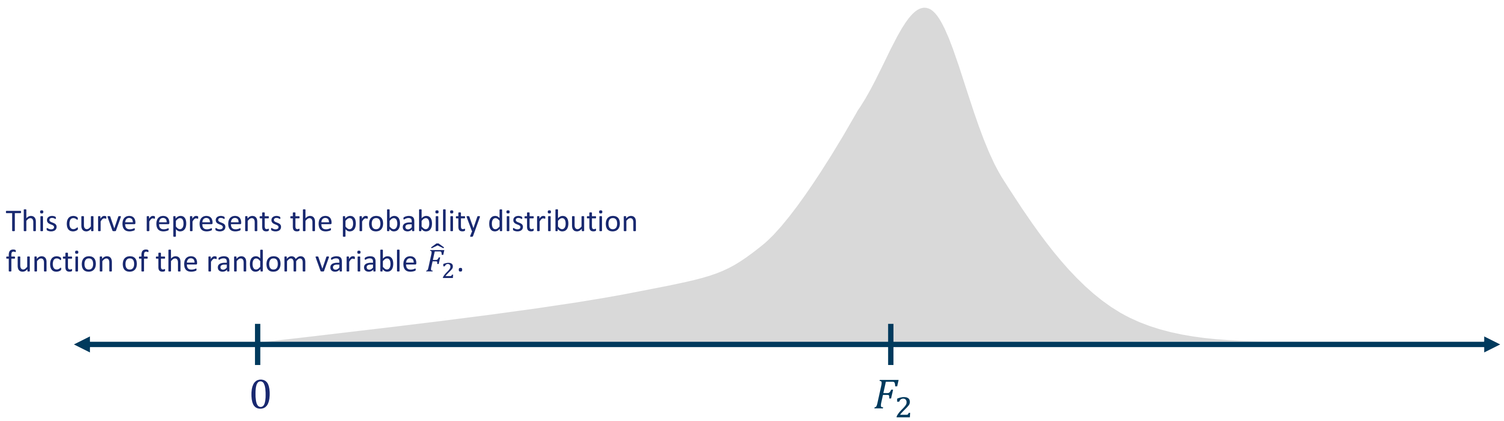
It follows that in expectation:

$$\mathbf{E}\left[\hat{F}_2\right] \equiv F_2 + \cancelto{0}{\sum_{i \neq j} v_i v_j \cdot \mathbf{E}\left[r_i r_j\right]}$$

Streaming and Sketching

Tug-of-war solution: using randomness

Stepping back: at this point, we've defined a process that yields a *random variable* $\hat{F}_2$ (i.e. our estimator), the mean of whose distribution is F_2.



Streaming and Sketching

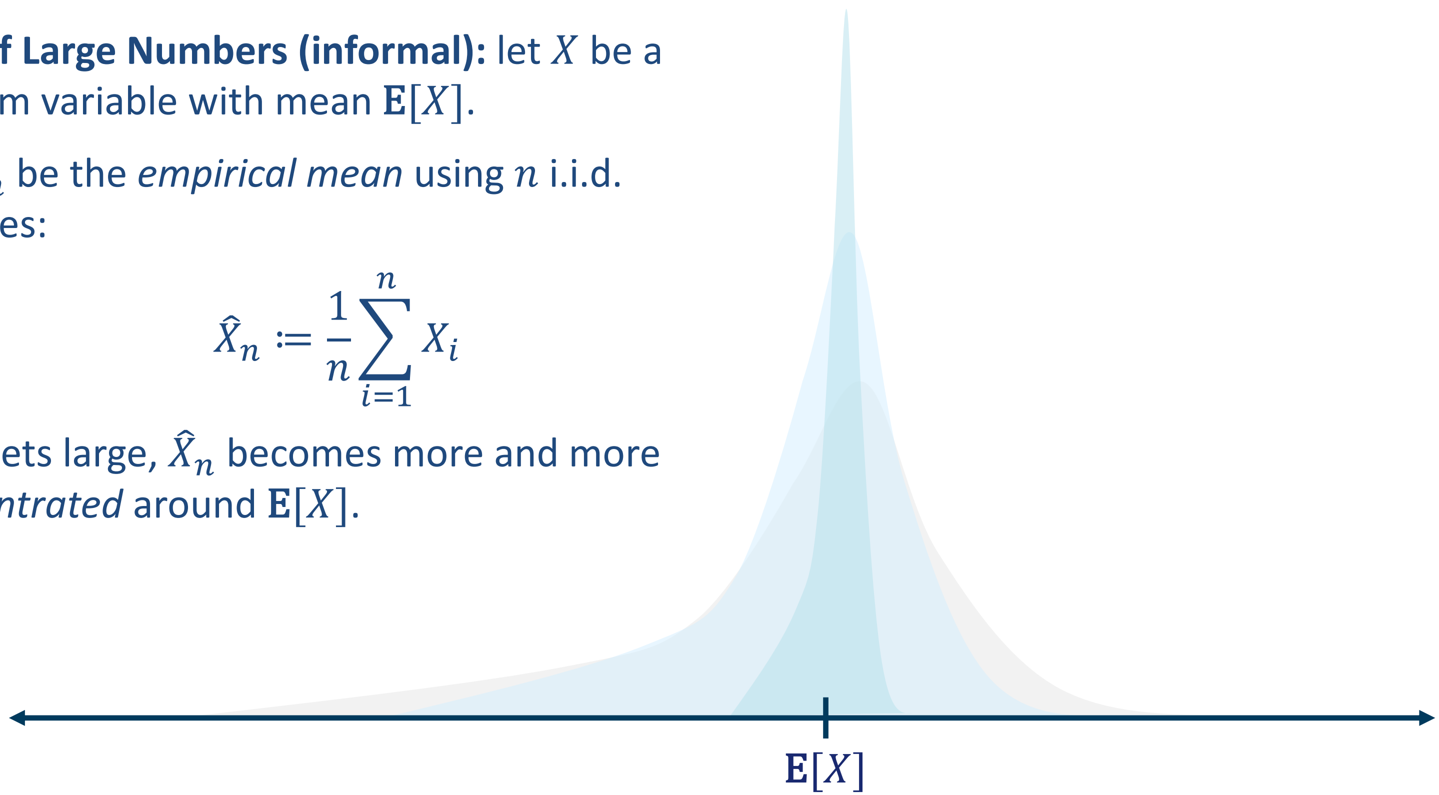
Interlude 2: the law of large numbers

Law of Large Numbers (informal): let X be a random variable with mean $\mathbf{E}[X]$.

Let $\hat{X}_n$ be the *empirical mean* using n i.i.d. samples:

$$\hat{X}_n := \frac{1}{n}\sum_{i=1}^{n} X_i$$

As n gets large, $\hat{X}_n$ becomes more and more *concentrated* around $\mathbf{E}[X]$.



Streaming and Sketching

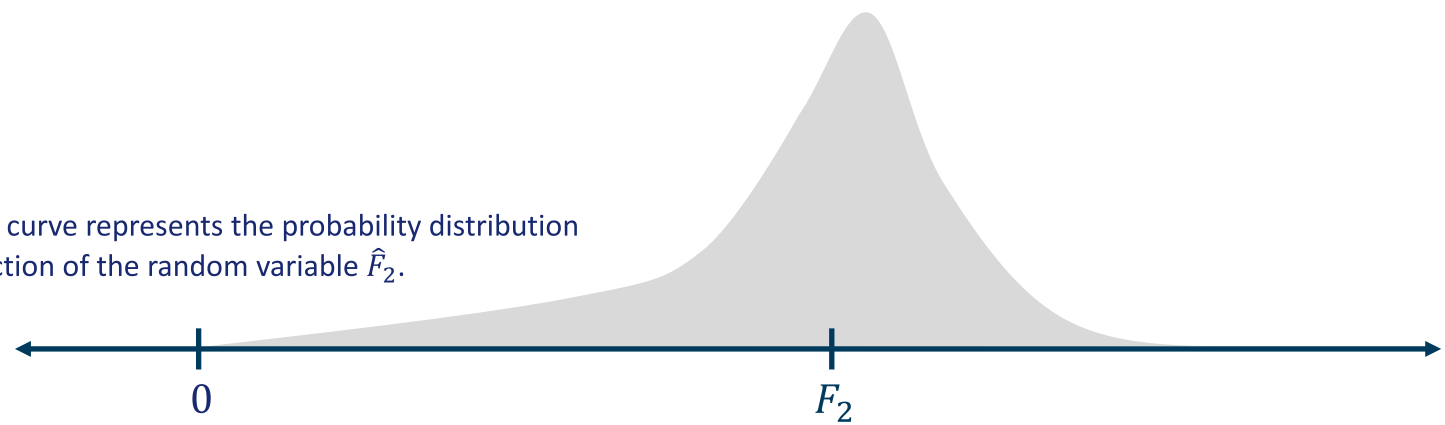
Tug-of-war solution: using randomness

Stepping back: at this point, we've defined a process that yields a *random variable* $\hat{F}_2$ (i.e. our estimator), the mean of whose distribution is F_2.

Intuition: even though our estimator $\hat{F}_2$ might not actually give a close approximation to F_2, if we estimate many times and take the mean, this mean can get very close to F_2.

Question: but how many times is "many times"?

This curve represents the probability distribution function of the random variable $\hat{F}_2$.



Streaming and Sketching

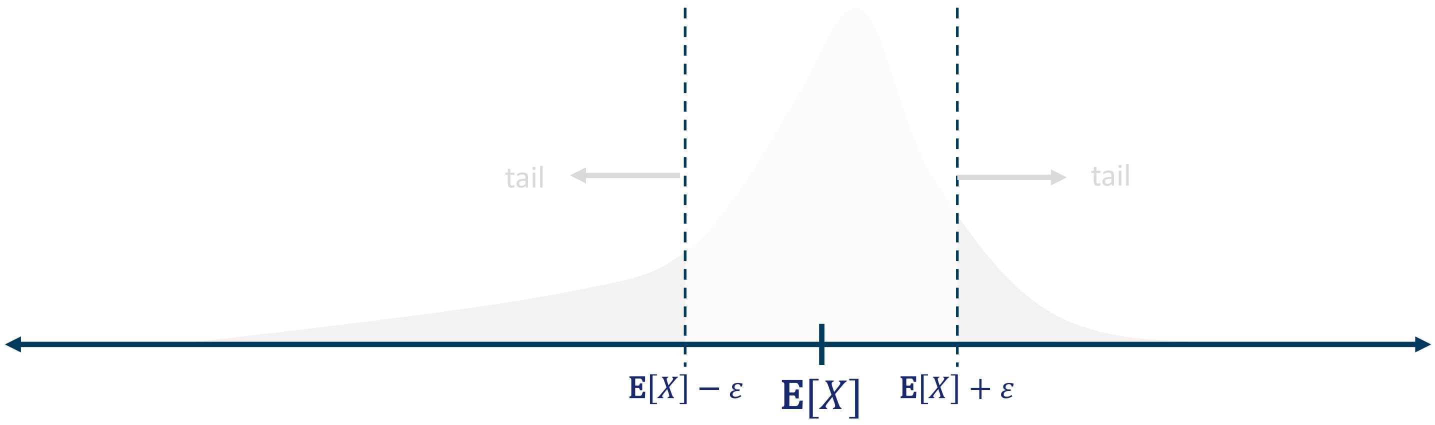
Interlude 3: concentration inequalities

Concentration inequality: describes how likely a random variable X will be close to its mean $\mathbf{E}[X]$.

A typical form of a concentration inequality:

$$\Pr[|X - \mathbf{E}[X]| \geq \varepsilon] < \delta(\varepsilon),$$

where δ is a function of ε.

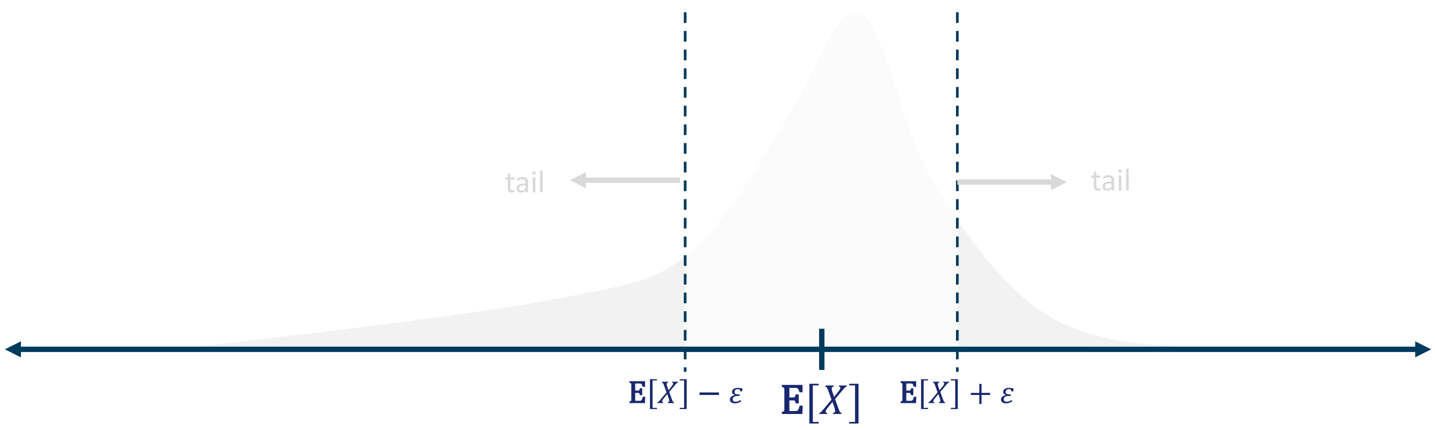


Streaming and Sketching

Interlude 3: concentration inequalities

Chebyshev's inequality: let $\bar{X}_n \coloneqq \frac{1}{n}(X_1 + \cdots + X_n)$ be the empirical mean of n independent trials of the random variable X. Then:

$$\Pr[|\bar{X}_n - \mathbf{E}[X]| \geq \varepsilon] \leq \frac{\mathrm{Var}[X]}{n\varepsilon^2}.$$

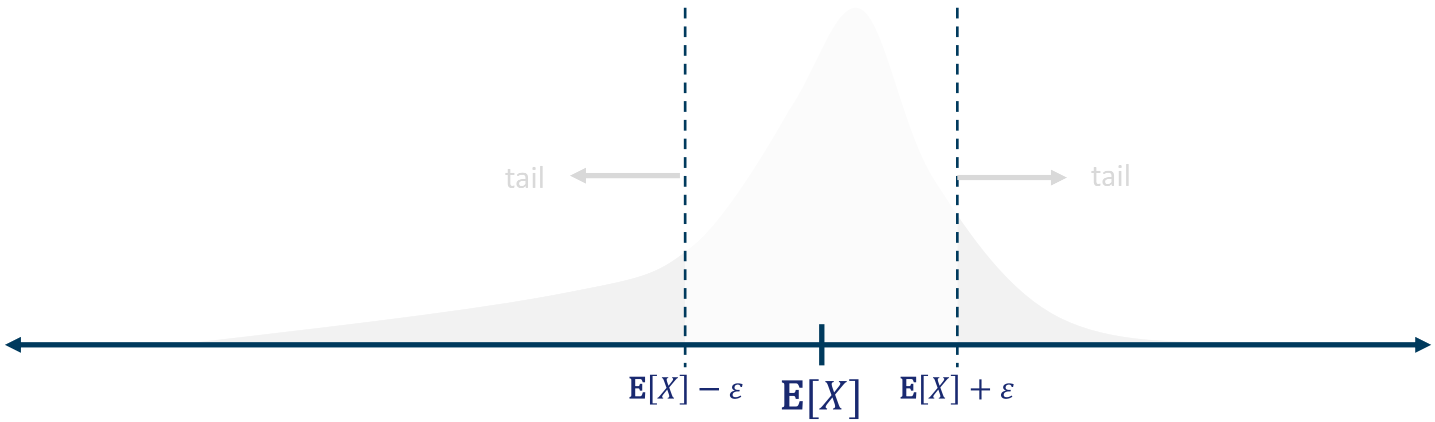


Streaming and Sketching

Completing the analysis

Theorem [AMS'98]. Let $\bar{F}_2$ be the average of $O\left(\frac{1}{\varepsilon^2}\log\frac{1}{\delta}\right)$ independent copies of the estimator $\hat{F}_2$. With probability $1 - \delta$:

$$(1 - \varepsilon)F_2 \leq \bar{F}_2 \leq (1 + \varepsilon)F_2.$$



Streaming and Sketching

In summary

Recall our notation:

- n is the number of distinct IP addresses that stream through router
- T is the number of IP packets that stream through
- ε parametrizes our error tolerance
- δ parametrizes our failure tolerance

	Space	Correctness	Failure probability
Naïve algorithm	$O(n \log T)$	Exact	0
Tug-of-war algorithm	$O\left(\frac{1}{\varepsilon^2} \log \frac{1}{\delta} \log T\right)$	$(1 \pm \varepsilon)$-factor approximation	δ

Streaming and Sketching

Moral of the story

Boosting technique: let A be an random algorithm that produces the correct in expectation. By running multiple independent copies and taking their mean, we obtain an estimator that becomes more concentrated around the mean.

- we can prove this using various concentration inequalities

Tradeoffs: by relaxing the problem (here: correctness and failure), we can significantly reduce the amount of space required.

Streaming and Sketching

Epilogue

Other important concentration inequalities:

Markov

Chebyshev

Hoeffding-Chernoff

Bernstein

Azuma

Bennett

McDiarmid

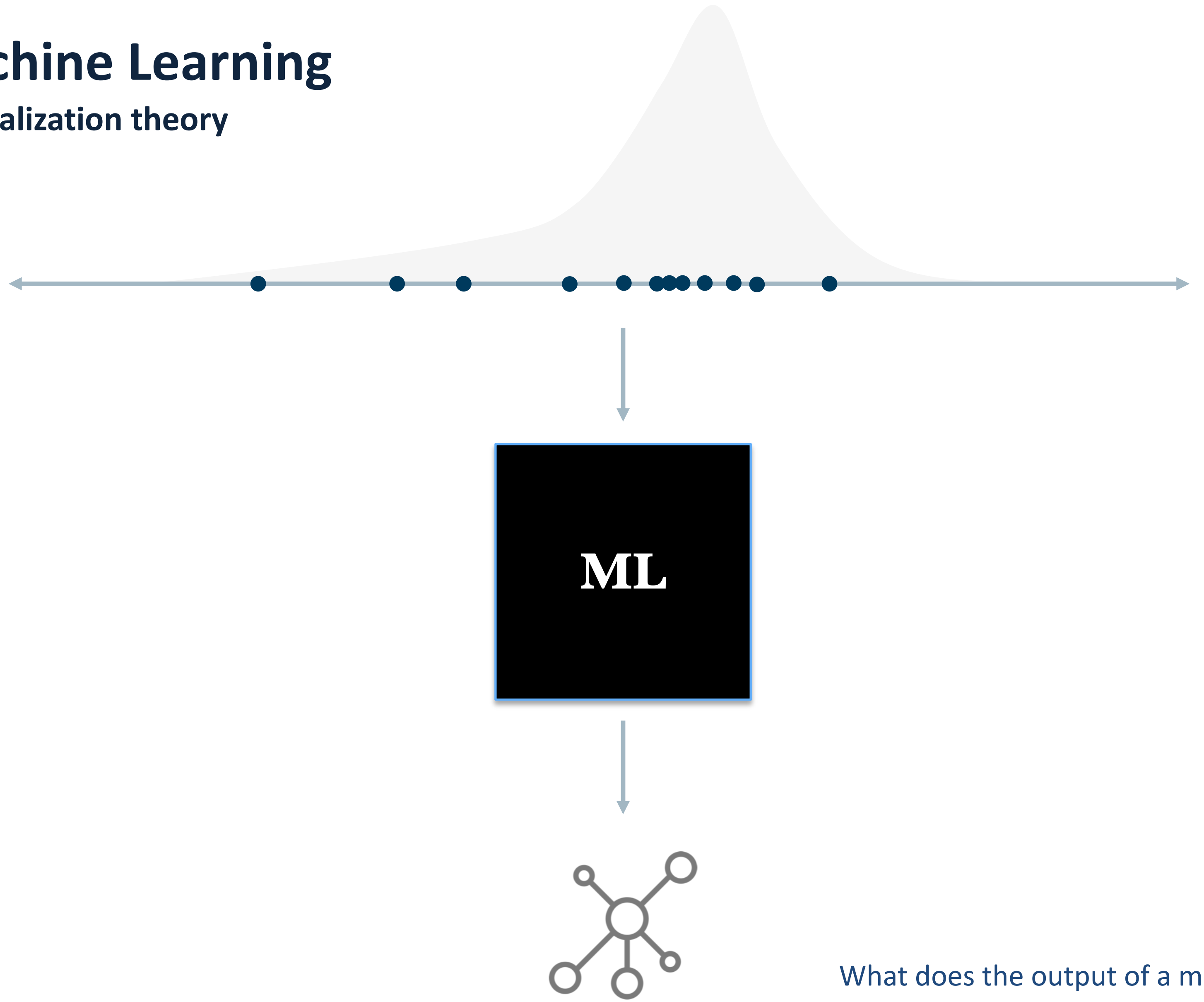
Talagrand

Resource Tradeoff

Data, correctness, and failure

Machine Learning

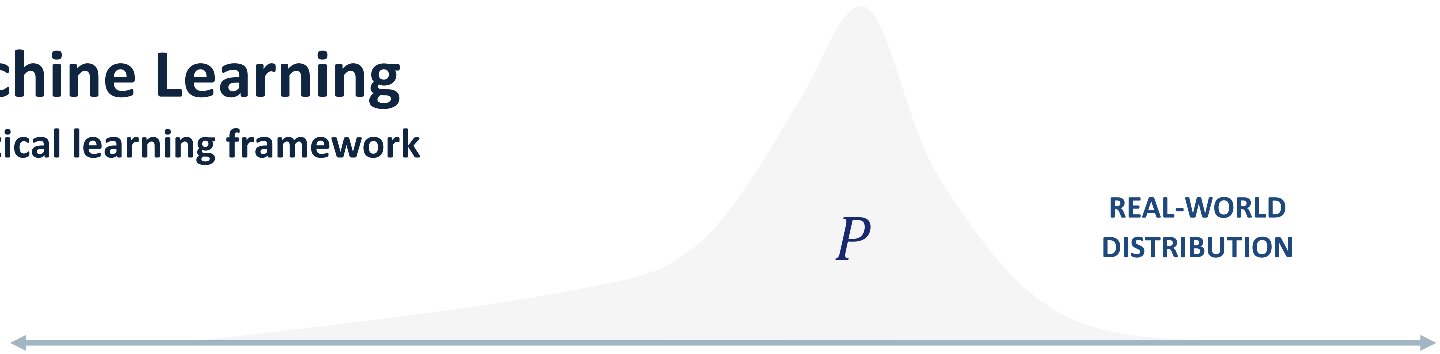
Generalization theory

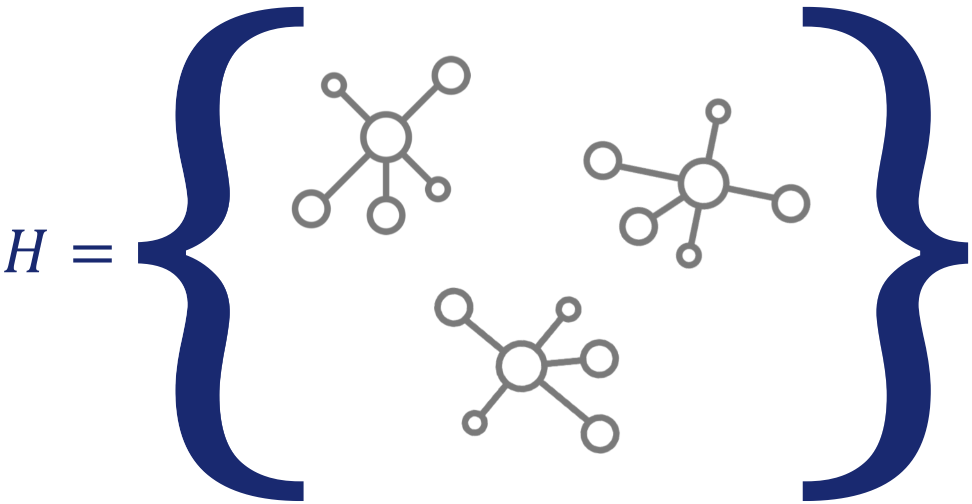


What does the output of a machine learning model even mean?

Machine Learning

Statistical learning framework





HYPOTHESIS CLASS

Model	Error
	**0.00623**
	0.48399
	0.24029

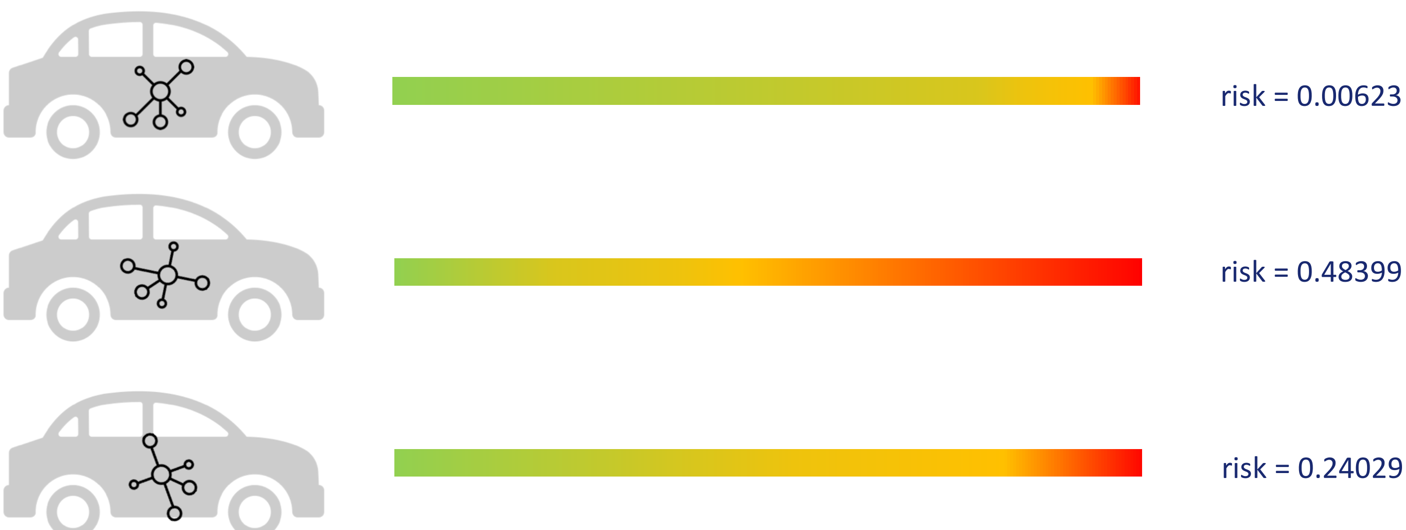
Machine Learning

Statistical learning framework

Reduction: learning becomes an *optimization* problem, of finding the model that minimizes the *risk*:

$$\arg\min_{h \in H} R(h)\,,$$

where the risk $R(h)$ of the model h is a measure of how bad it would perform when tested against real world scenarios (distributed according to P).



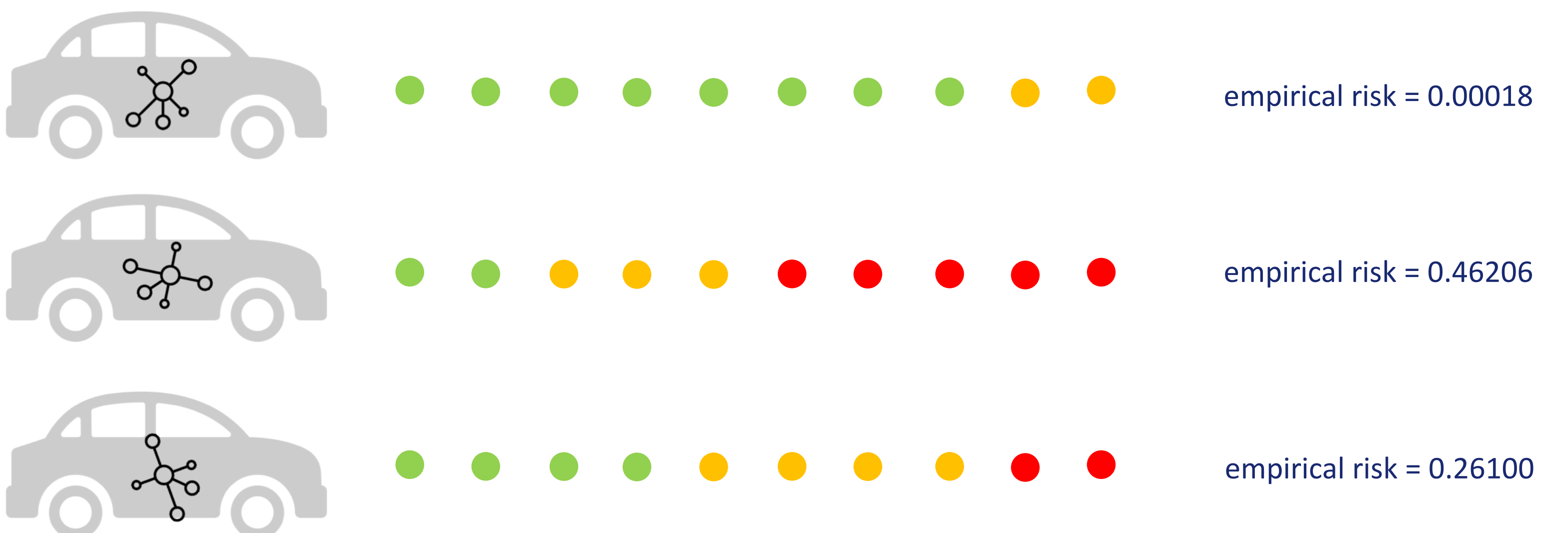
Machine Learning

Statistical learning framework

Estimation: we can't optimize directly over the real world distribution P; we look at training data drawn from P:

$$X_1, X_2, \ldots, X_n \sim P.$$

Then, we *estimate* the true risk R using an *empirical risk* $\hat{R}$.



Machine Learning

Statistical learning framework

Empirical risk minimization: a theoretical algorithm for learning

1. Draw i.i.d. training data, $X_1, \ldots, X_n \sim P$
2. Compute empirical risks, returning model that minimizes estimated risk

Model	True Risk	Empirical Risk
	0.00623	**0.00018**
	0.48399	0.46206
	0.24029	0.26100

Machine Learning

Statistical learning framework

Trade off: with any estimation problem, we need to balance

- amount of data used to perform estimation: n
- error tolerance of the estimator: ε
- failure tolerance of the estimator: δ

This framework is called **probably approximately correct (PAC) learning**, and we say that an algorithm (ε, δ)-learns a hypothesis class using n samples if:

$$\Pr\left[R\left(\hat{h}\right) - R(h^*) \geq \varepsilon\right] \leq \delta \,,$$

where $\hat{h}$ is the model learned by the algorithm, and h^* is the "true" model.

Machine Learning

Statistical learning framework

Theorem [Fundamental Theorem of Learning Theory]. Let H be a hypothesis class of classifiers of size N. It is information-theoretically possible to (ε, δ)-learn H using n samples:

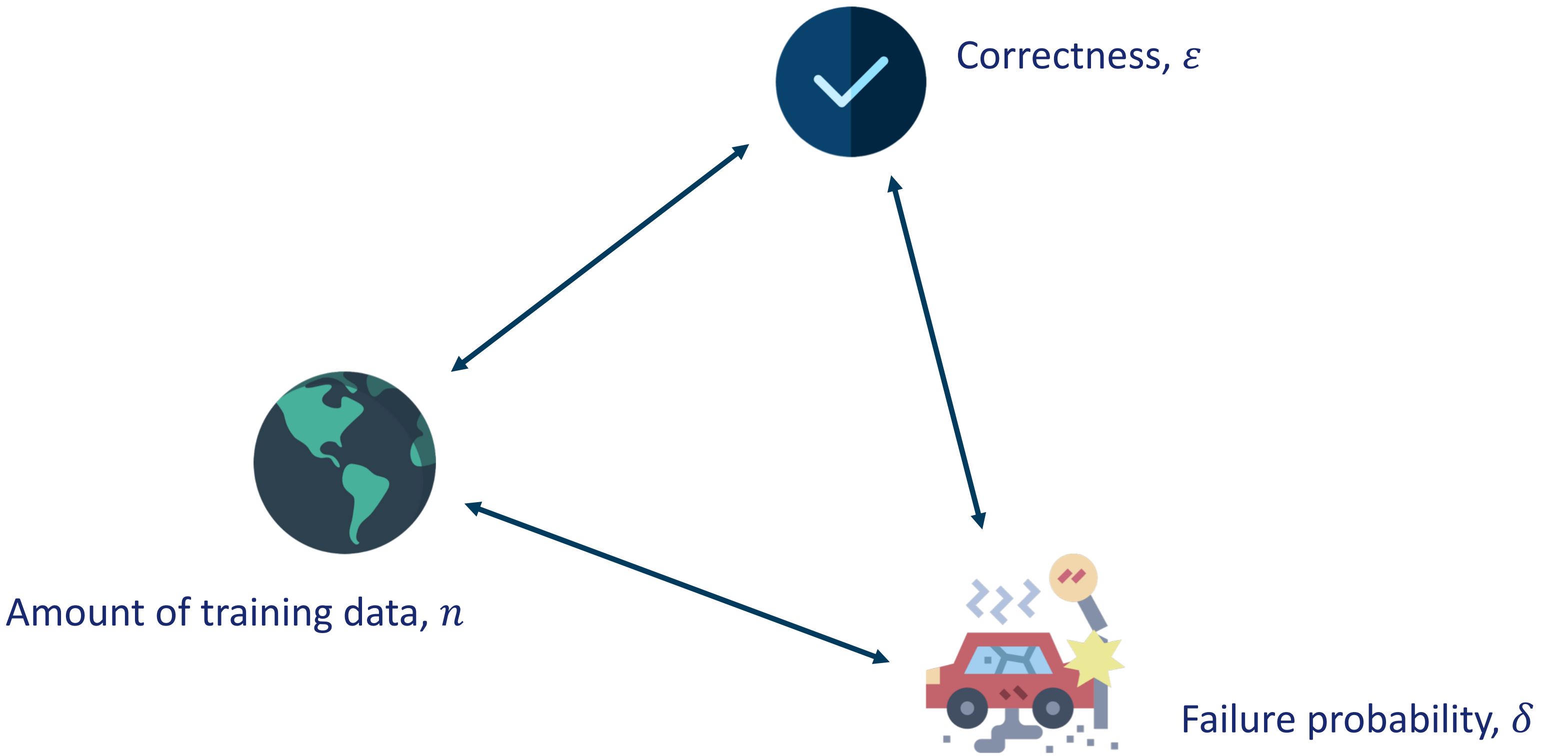
$$n = O\left(\frac{1}{\varepsilon^2} \log \frac{N}{\delta}\right).$$

- Note: even though it is possible to learn using n training points, whether a specific algorithm can/does achieve this is a separate matter

Machine Learning

Moral of the story

Statistical learning theory is a field that aims to put machine learning on solid theoretical ground, attempting to quantify the following trade offs:



Machine Learning

Epilogue and ongoing research

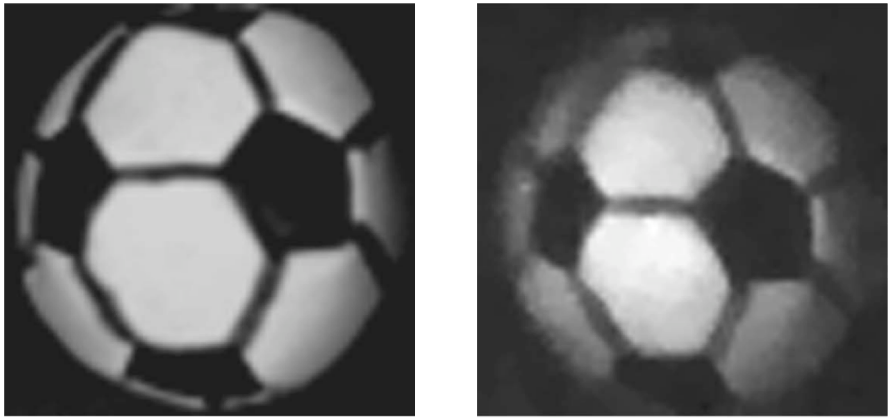
We have an upper bound n on the number training points we need to learn.

- It seems that the upper bounds we show about models like neural networks can't explain why they work so well (i.e. our upper bounds are not tight at all). How can we understand why neural networks **generalize** so well?
- For specific hypothesis classes, what are lower bounds on the amount of data needed (i.e. how little data is not enough data)?
- What if the learning algorithm not only learned on the data, but chose the data on which it learned? Then, can we reduce the amount of data required?

Additional Topics

Some teasers

1. **Johnson-Lindenstrauss:** fix any n points in **R**^{d}. Without looking at those points, I can produce a linear map that maps those points down from d to $O\left(\frac{\log n}{\varepsilon^2}\right)$ dimensions such that all pairwise distances are preserved up to a $(1 \pm \varepsilon)$-factor.
2. **Compressed sensing:** reconstruction of a signal using extremely few data points.



Left: image taken by a 64 x 64 pixel camera.

Right: image taken by a one-pixel camera using 1,600 shots.

References

Resource and recommendations

Zero-knowledge proofs

- [P+16] Philippe, Sébastien, et al. "A physical zero-knowledge object-comparison system for nuclear warhead verification." *Nature communications* 7 (2016): 12890.
- [BM92] Bellovin, Steven M., and Michael Merritt. "Encrypted key exchange: Password-based protocols secure against dictionary attacks." *Proceedings 1992 IEEE Computer Society Symposium on Research in Security and Privacy*. IEEE, 1992.
- **[Q+89]** Quisquater, Jean-Jacques, et al. "How to explain zero-knowledge protocols to your children." *Conference on the Theory and Application of Cryptology*. Springer, New York, NY, 1989.

Sketching/streaming

- **[A19]** Andoni, Alex. *Algorithms for Massive Data Lecture Notes*, 2019. [link]
- [AMS98] Alon, Noga, Yossi Matias, and Mario Szegedy. "The space complexity of approximating the frequency moments." *Journal of Computer and system sciences* 58.1 (1999): 137-147.

Machine learning

- **[SS14]** Shalev-Shwartz, Shai, and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.
- **[BBL05]** Boucheron, Stéphane, Olivier Bousquet, and Gábor Lugosi. "Theory of classification: A survey of some recent advances." *ESAIM: probability and statistics* 9 (2005): 323-375.
- **[Z+16]** Zhang, Chiyuan, et al. "Understanding deep learning requires rethinking generalization." *arXiv preprint arXiv:1611.03530* (2016).
- **[D11]** Dasgupta, Sanjoy. "Two faces of active learning." *Theoretical computer science* 412.19 (2011): 1767-1781.

Nearest neighbor search

- **[AR15]** Andoni, Alexandr, and Ilya Razenshteyn. "Optimal data-dependent hashing for approximate near neighbors." *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*. ACM, 2015.

Compressed sensing

- [A07] A, Richard G. "Compressive sensing." *IEEE signal processing magazine* 24.4 (2007).

References

Sources

Flaticons

- Memory, icon made by Smashicons from flaticon.com.
- Hourglass, icon made by Smashicons from flaticon.com.
- Globe, icon made by Flat Icons from flaticon.com.
- Cluster, icon made by Eucalyp from flaticon.com.
- Check, icon made by Freepik from flaticon.com.
- Crash, icon made by smalllikeart from flaticon.com.
- Coin, icon made by Smashicons from flaticon.com.
- Car, icon made by Freepik from flaticon.com.
- Router, icon made by Payungkead from flaticon.com.
- Graph network, icon made by Smashicons from flaticon.com.
- Car profile, icon made by Creaticca Creative Agency from flaticon.com.

Other images

- Buffon's needle problem: http://mathworld.wolfram.com/images/eps-gif/BuffonNeedleTosses_825.gif
- Soccer balls: https://www.ams.org/publicoutreach/math-history/hap7-pixel.pdf

Final Question

How many…

texed tech talks could a texed tech-talk talk if a texed tech talk could talk texed-tech-talks?