7 Randomization Basics

10 June 2025

Prof. Dr. Sebastian Wild

CS627 (Summer 2025)
Philipps-Universität Marburg
version `2025-06-11 10:04`

Outline

7 Randomization Basics

7.1 Motivation

Computational Lottery?

- If we are faced with solving an NP-hard problem and known smart algorithms are too slow, we likely have to compromise on what "solving" means.
- Classical algorithms are *always* and *exactly* correct.

⇝ Here: Let's compromise on "always", i. e., allow algorithms to occasionally **fail**!

Computational Lottery?

- If we are faced with solving an NP-hard problem and known smart algorithms are too slow, we likely have to compromise on what "solving" means.
- Classical algorithms are *always* and *exactly* correct.

⇝ Here: Let's compromise on "always", i. e., allow algorithms to occasionally **fail**!

ϟ A *deterministic* algorithm A that fails on input x will ***always*** fail for x.

  - ⇝ What if we require a solution for such an input x? We get **nothing** from A!
  - Must use a form of *nondeterminism*.

Computational Lottery?

- If we are faced with solving an NP-hard problem and known smart algorithms are too slow, we likely have to compromise on what "solving" means.
- Classical algorithms are *always* and *exactly* correct.

⇝ Here: Let's compromise on "always", i. e., allow algorithms to occasionally **fail**!

ϟ A *deterministic* algorithm A that fails on input x will ***always*** fail for x.

  - ⇝ What if we require a solution for such an input x? We get **nothing** from A!
  - Must use a form of *nondeterminism*.

- ***Randomization:*** Use *random bits* to guide computation.

⇝ *Instead of always failing on some rare inputs, we rarely fail on any input.*

can make this arbitrarily rare

Why Could Randomization Help?

- Main intuitive reason: (can be) much easier to be 99.999999% correct than 100%
  How can this manifest itself?
  - **Faster and simpler algorithms**
    Random choice can allow to sidestep tricky edge cases
  - We can use **fingerprinting** (a.k.a. checksums) hashing
    Cheap surrogate question, mostly correct, but sometimes wrong.
  - Protect against **adversarial inputs**
    We make our (algorithm's) behavior unpredictable, so it us harder to exploit us.

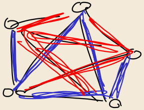
Why Could Randomization Help?

- Main intuitive reason: (can be) much easier to be 99.999999% correct than 100%
  How can this manifest itself?
  - **Faster and simpler algorithms**
    Random choice can allow to sidestep tricky edge cases
  - We can use **fingerprinting** (a.k.a. checksums)
    Cheap surrogate question, mostly correct, but sometimes wrong.
  - Protect against **adversarial inputs**
    We make our (algorithm's) behavior unpredictable, so it us harder to exploit us.

- Also: *probabilistic method* for proofs
  - Goal: Prove existence of discrete object with some property
  - Idea: Design randomized algorithm to find one
  - $\rightsquigarrow$ If algorithm succeeds with prob. > 0, object must exist!

Ramsey theory

complete graph on n vertices



Claim:

$\exists$ monochromatic clique of size $\geq R(n)$

$R(n) \approx \lg n$

Average-Case Analysis vs. Randomized Algorithms

Average-Case Analysis

- algorithm is **deterministic**
  same input, same computation

Randomized Algorithm (here)

- algorithm is **not** deterministic
  same input, potentially different comp.

Average-Case Analysis vs. Randomized Algorithms

Average-Case Analysis

- algorithm is **deterministic**
  same input, same computation
- input is chosen according to some **probability distribution**

Randomized Algorithm (here)

- algorithm is **not** deterministic
  same input, potentially different comp.
- input is chosen **adversarially** (worst-case inputs)

  oblivious adversary
  (can't see random bits)

Average-Case Analysis vs. Randomized Algorithms

Average-Case Analysis

- algorithm is **deterministic**
  same input, same computation
- input is chosen according to some **probability distribution**
- cost given as expectation over inputs

Randomized Algorithm (here)

- algorithm is **not** deterministic
  same input, potentially different comp.
- input is chosen **adversarially** (worst-case inputs)
- cost given as expectation over random choices of algorithm

Average-Case Analysis vs. Randomized Algorithms

Average-Case Analysis

- algorithm is **deterministic**
  same input, same computation
- input is chosen according to some **probability distribution**
- cost given as expectation over inputs

Randomized Algorithm (here)

- algorithm is **not** deterministic
  same input, potentially different comp.
- input is chosen **adversarially** (worst-case inputs)
- cost given as expectation over random choices of algorithm

example: sorting by first shuffle

Confusingly enough, the analysis (technique) is often the same!

But: Implications are quite different; randomization is much more versatile and robust.

7.2 Randomized Selection

Separation Example

- Before we introduce randomization more formally, let's see a successful example
- Here, not a "hard" problem, but a showcase where randomization makes something possible that is *provably*

Introductory Example – Quickselect

Selection by Rank

- **Given:** array $A[0..n)$ of numbers and number $k \in [0..n)$.
- **Goal:** find element that would be in position k if A was sorted (kth smallest element).

  but 0-based & counting dups

  - $k = \lfloor n/2 \rfloor \rightsquigarrow$ median; $k = \lfloor n/4 \rfloor \rightsquigarrow$ lower quartile
    $k = 0 \rightsquigarrow$ minimum; $k = n - \ell \rightsquigarrow$ ℓth largest

Introductory Example – Quickselect

Selection by Rank

- **Given:** array $A[0..n)$ of numbers and number $k \in [0..n)$.
- **Goal:** find element that would be in position k if A was sorted (kth smallest element).
  (but 0-based & counting dups)
  - $k = \lfloor n/2 \rfloor \rightsquigarrow$ median; $k = \lfloor n/4 \rfloor \rightsquigarrow$ lower quartile
    $k = 0 \rightsquigarrow$ minimum; $k = n - \ell \rightsquigarrow$ ℓth largest

```
procedure quickselect(A[0..n), k):
    l := 0;  r := n
    while r − l > 1
        b := random pivot from A[l..r)
        j := partition(A[l..r), b)
        if j ≥ k then r := j − 1
        if j ≤ k then l := j + 1
    return A[k]
```

- simple algorithm:
  determine rank of random element,
  recurse

$\rightsquigarrow$ $O(n)$ time in expectation (over random choices)

- worst case: $\Theta(n^2)$
- $O(n)$ also possible deterministically,
  but algorithms is more involved (median of medians)

A closer look at Selection

While all within $\Theta(n)$, we do get a strict separation for selecting the median.

Theorem 7.1 (Bent & John (1985))

Any **deterministic** comparison-based algorithm for finding the median of n elements uses at least $\mathbf{2n} - o(n)$ comparisons in the worst case. ◀

Proof omitted.

A closer look at Selection

While all within $\Theta(n)$, we do get a strict separation for selecting the median.

Theorem 7.1 (Bent & John (1985))

Any **deterministic** comparison-based algorithm for finding the median of n elements uses at least $2n - o(n)$ comparisons in the worst case.

Proof omitted.

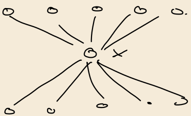
The following weaker result is easier to see:

Theorem 7.2 (Blum et al. (1973))

Any deterministic comparison-based algorithm for finding the median of n elements uses at least $n - 1 + (n-1)/2 \sim 1.5n$ comparisons in the worst case.

Proof: Two types of comparisons

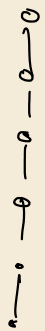
(1) certificate comparisons $n-1$



$n = 11$

always necessary for correct algorithm

sorting



n elements

n-1 comps

A Median Adversary

Proof (Theorem 7.2):

(2) "nonessential" comparisons

(not part of certificate)

in particular, comparisons between L and S

m = true median

$L = \{x : x > m\}$

$S = \{x : x < m\}$ $(|S| = |L|)$

Given a deterministic algorithm A, we (the adversary) try to answer comparison queries by A in the least useful way (for A)

Here: maintain elements in 3 sets, S, L and U (undecided)

initially all in U

query "$x \overset{?}{<} y$" if x and y not in same set, answer $S < U < L$

$x, y \in S$
$x, y \in L$ } arbitrary answer

$x, y \in U$ $x < y$, put x to S, y into L

$\Rightarrow$ created <u>one</u> non-essential cmp for A

remove 2 elements from U

$\Rightarrow \geq \frac{n-1}{2}$ non-essential comparisons $\square$

Randomized Selection

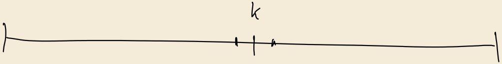
- Can prove: Randomized Quickselect uses in expectation $\sim (2\ln 2 + 2)n \approx 3.39n$ comparisons to find the median
- But we can do better!

Randomized Selection

- Can prove: Randomized Quickselect uses in expectation $\sim (2\ln 2 + 2)n \approx 3.39n$ comparisons to find the median
- But we can do better!

```
procedure floydRivest(A[ℓ..r], k):
    n := r − ℓ
    if n < n0 return quickselect(A, k)
    s := 1/2 n^(2/3) // all numbers to be rounded
    sd := 1/2 √(ln(n) s(n − s)/n)
    S[0..s) := random sample from A
    k̂ := s k/n
    p := floydRivest(S, k̂ − sd)
    q := floydRivest(S, k̂ + sd)
    (i, j) := partition A around p0 and p1
    if i == k return A[i]
    if j == k return A[j]
    if k < i return floydRivest(A[ℓ..i), k)
    if k > j return floydRivest(A[j..r), k)
    return floydRivest(A[i..j), k)
```

- Variant of Quickselect with huge sample
- Analysis sketch:
  - partition costs $1.5n$ comparisons



Randomized Selection

- Can prove: Randomized Quickselect uses in expectation $\sim (2\ln 2 + 2)n \approx 3.39n$ comparisons to find the median
- But we can do better!

procedure floydRivest($A[\ell..r]$, k):
    $n := r - \ell$
    if $n < n_0$ **return** quickselect(A, k)
    $s := \frac{1}{2}n^{2/3}$ *// all numbers to be rounded*
    $sd := \frac{1}{2}\sqrt{\ln(n)s(n-s)/n}$
    $S[0..s)$:= random sample from A
    $\hat{k} := s\frac{k}{n}$
    p := floydRivest(S, $\hat{k} - sd$)
    q := floydRivest(S, $\hat{k} + sd$)
    (i, j) :=partition A around p_0 and p_1
    if $i == k$ **return** $A[i]$
    if $j == k$ **return** $A[j]$
    if $k < i$ **return** floydRivest($A[\ell..i)$, k)
    if $k > j$ **return** floydRivest($A[j..r)$, k)
    return floydRivest($A[i..j)$, k)

- Variant of Quickselect with huge sample
- Analysis sketch:
  - partition costs $1.5n$ comparisons
  - Everything on sample has cost $o(n)$
  - by the choice of parameters, with prob $1 - o(1)$:
    (a) $i < k < j$ after partition
    (b) $j - i = o(n)$
  - $\rightsquigarrow$ all recursive calls expected $o(n)$ cost

Randomized Selection

- Can prove: Randomized Quickselect uses in expectation $\sim (2\ln 2 + 2)n \approx 3.39n$ comparisons to find the median
- But we can do better!

```
procedure floydRivest(A[ℓ..r], k):
    n := r − ℓ
    if n < n_0 return quickselect(A, k)
    s := (1/2) n^(2/3)   // all numbers to be rounded
    sd := (1/2) sqrt(ln(n) s (n − s) / n)
    S[0..s) := random sample from A
    k̂ := s (k/n)
    p := floydRivest(S, k̂ − sd)
    q := floydRivest(S, k̂ + sd)
    (i, j) := partition A around p_0 and p_1
    if i == k return A[i]
    if j == k return A[j]
    if k < i return floydRivest(A[ℓ..i), k)
    if k > j return floydRivest(A[j..r), k)
    return floydRivest(A[i..j), k)
```

- Variant of Quickselect with huge sample
- Analysis sketch:
  - partition costs $1.5n$ comparisons
  - Everything on sample has cost $o(n)$
  - by the choice of parameters, with prob $1 - o(1)$:
    (a) $i < k < j$ after partition
    (b) $j - i = o(n)$
  - $\rightsquigarrow$ all recursive calls expected $o(n)$ cost

$\rightsquigarrow$ Randomized median selection with $1.5n + o(n)$ comparisons

$\rightsquigarrow$ Separation from deterministic case!

Power of Randomness

- Selection by Rank shows two aspects of randomization:
  - A simpler algorithm by avoiding edge cases (like an initial order giving bad pivots)
  - Protection against adversarial inputs
    (inputs constructed with knowledge about the algorithm)

    Here randomization provably more powerful than any thinkable deterministic algorithm!

    constant factor for #cmps

Power of Randomness

- Selection by Rank shows two aspects of randomization:
  - A simpler algorithm by avoiding edge cases (like an initial order giving bad pivots)
  - Protection against adversarial inputs
    (inputs constructed with knowledge about the algorithm)

    Here randomization provably more powerful than any thinkable deterministic algorithm!

    ↖ constant factor for #cmps

- What can we gain for (NP-)hard problems?
- But first, let's define things properly.

7.3 Recap of Probability Theory

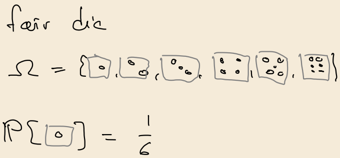
Probability Theory

- We will quickly revisit some key terms from probability theory
  - Single place to look up notation etc.
- Much will focus on discrete probability, but some continuous tools useful, too

Probability Spaces

Discrete probability space $(\Omega, \mathbb{P})$:

- $\Omega = \{\omega_1, \omega_2, \ldots\}$ a (finite or) *countable* set
- $\mathbb{P} : 2^{\Omega} \longrightarrow [0, 1]$ a discrete probability measure, i. e.,
  - $\mathbb{P}[\Omega] = 1$
  - $\mathbb{P}[A] = \sum_{\omega \in A} \mathbb{P}[\omega] \quad \rightsquigarrow \quad \mathbb{P}$ determined by $w_i = \mathbb{P}[\omega_i]$.



Probability Spaces

Discrete probability space $(\Omega, \mathbb{P})$:

- $\Omega = \{\omega_1, \omega_2, \ldots\}$ a (finite or) *countable* set
- $\mathbb{P} : 2^\Omega \to [0,1]$ a discrete probability measure, i. e.,
  - $\mathbb{P}[\Omega] = 1$
  - $\mathbb{P}[A] = \sum_{\omega \in A} \mathbb{P}[\omega] \quad \rightsquigarrow \quad \mathbb{P}$ determined by $w_i = \mathbb{P}[\omega_i]$.

General probability space $(\Omega, \mathcal{F}, \mathbb{P})$:

- Ω is a set of points (the universe)
- $\mathcal{F} \subseteq 2^\Omega$ is a σ-algebra, i. e., (discrete case: $\mathcal{F} = 2^\Omega$; $\Omega = \mathbb{R}$: Borel σ-algebra $\mathcal{B}$ generated by (a, b))
  - $\emptyset \in \mathcal{F}$
  - closed under complementation: $A \in \mathcal{F} \implies \overline{A} = \Omega \setminus A \in \mathcal{F}$
  - closed under *countable* union: $A_1, A_2, \ldots \in \mathcal{F} \implies \bigcup_{i=1}^{\infty} A_i \in \mathcal{F}$
- $\mathbb{P} : \mathcal{F} \to [0,1]$ is a probability measure, i. e., ($\Omega = \mathbb{R} \rightsquigarrow$ Lebesgue measure $\lambda((a,b)) = b - a$)
  - $\mathbb{P}[\Omega] = 1$
  - If $A_1, A_2, \ldots \in \mathcal{F}$ are pairwise *disjoint* then $\mathbb{P}\left[\bigcup_{i=1}^{\infty} A_i\right] = \sum_{i=1}^{\infty} \mathbb{P}[A_i]$

Events

something we can assign a probability to ↙

$A \in \mathcal{F}$ is called an *event* of $(\Omega, \mathcal{F}, \mathbb{P})$; also a *measurable set*.

Events

something we can assign a probability to

$A \in \mathcal{F}$ is called an *event* of $(\Omega, \mathcal{F}, \mathbb{P})$; also a *measurable set*.

Basic properties

- $\mathbb{P}\left[\,\overline{A}\,\right] = 1 - \mathbb{P}[A]$ counter-probability $\quad (\overline{A} = \Omega \setminus A)$
- $\mathbb{P}\left[\bigcup A_i\right] \le \sum_i \mathbb{P}[A]$ the *union bound* (a.k.a. Boole's inequality a.k.a. σ-subadditivity)

Events

something we can assign a probability to

$A \in \mathcal{F}$ is called an *event* of $(\Omega, \mathcal{F}, \mathbb{P})$; also a *measurable set*.

Basic properties

- $\mathbb{P}\left[\,\overline{A}\,\right] = 1 - \mathbb{P}[A]$ counter-probability $\quad (\overline{A} = \Omega \setminus A)$
- $\mathbb{P}\left[\bigcup A_i\right] \;\le\; \sum_i \mathbb{P}[A]$ the *union bound* (a.k.a. Boole's inequality a.k.a. σ-subadditivity)
- $\{A_1, \ldots, A_k\}$ *(mutually) independent* $\iff$ $\mathbb{P}\left[\bigcap_i A_i\right] \;=\; \prod_i \mathbb{P}[A_i]$

  An infinite set of events is mutually independent if every finite subset is so.

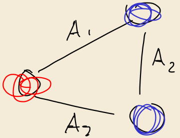
  k-wise independence means that only all size-k subsets are independent.

pairwise indp. $\quad \exists\, i \neq j \in [k] \quad \mathbb{P}[A_i \cap A_j] = \mathbb{P}[A_i] \cdot \mathbb{P}[A_j]$

$\not\Rightarrow$ mutual independence

$\{A_1, A_2, A_3\}$ 2-wise indp.

but not mut. indp.



A_i = edge has 2 colors at endpoints

Events

something we can assign a probability to

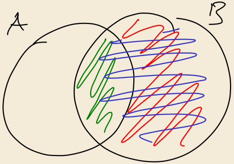
$A \in \mathcal{F}$ is called an *event* of $(\Omega, \mathcal{F}, \mathbb{P})$; also a *measurable set*.

Basic properties

- $\mathbb{P}[\overline{A}] = 1 - \mathbb{P}[A]$ counter-probability $\quad (\overline{A} = \Omega \setminus A)$
- $\mathbb{P}\left[\bigcup A_i\right] \le \sum_i \mathbb{P}[A]$ the *union bound* (a.k.a. Boole's inequality a.k.a. σ-subadditivity)
- $\{A_1, \ldots, A_k\}$ *(mutually) independent* $\iff \mathbb{P}\left[\bigcap_i A_i\right] = \prod_i \mathbb{P}[A_i]$

  An infinite set of events is mutually independent if every finite subset is so.

  k-wise independence means that only all size-k subsets are independent.



- *conditional probability* for A given B: $\mathbb{P}[A \mid B] = \mathbb{P}[A \cap B] / \mathbb{P}[B]$

  generally undefined if $\mathbb{P}[B] = 0$
- *law of total probability*: If $\Omega = B_1 \dot\cup B_2 \dot\cup \cdots$ is a partition of Ω, we have

$$\mathbb{P}[A] = \sum_{\substack{i \\ \mathbb{P}[B_i] \neq 0}} \mathbb{P}[A \mid B_i] \cdot \mathbb{P}[B_i].$$

Random Variables

Random variables (r.v.) $X : \Omega \to \mathcal{X}$; often $\mathcal{X} = \mathbb{R}$ (in general spaces: only *measurable* functions)

fair die

$\Omega = \{⚀, ⚁, ⚂, ⚃, ⚄, ⚅\}$

$\mathbb{P}[⚀] = \frac{1}{6}$

$X : \Omega \to \{1, \dots, 6\}$

$Y : \Omega \to \{\text{even}, \text{odd}\}$

Random Variables

Random variables (r.v.) $X : \Omega \to \mathcal{X}$; often $\mathcal{X} = \mathbb{R}$ (in general spaces: only *measurable* functions)

Basic properties and conventions:

- event $\{X = x\}$ is defined as $\{\omega \in \Omega : X(\omega) = x\}$.
- For event A define the indicator r.v. $\mathbb{1}_A$ via $\mathbb{1}_A(\omega) = [\omega \in A]$ $= \begin{cases} 1 & \omega \in A \\ 0 & \text{else} \end{cases}$

Random Variables

Random variables (r.v.) $X : \Omega \to \mathcal{X}$; often $\mathcal{X} = \mathbb{R}$ (in general spaces: only *measurable* functions)

Basic properties and conventions:

- event $\{X = x\}$ is defined as $\{\omega \in \Omega : X(\omega) = x\}$.
- For event A define the indicator r.v. $\mathbb{1}_A$ via $\mathbb{1}_A(\omega) = [\omega \in A]$
- $F_X(x) = \mathbb{P}[X \le x]$ is the *cumulative distribution function (CDF)*.
- X is *discrete* if $X(\Omega) = \{X(\omega) : \omega \in \Omega\}$ is countable. $\mathcal{X} = \mathbb{N}$
- for discrete r.v. X define $f_X(n) = \mathbb{P}[X = n]$ the *probability mass function (PMF)*.
- If F_X is everywhere differentiable, X is *continuous*.
  Then $f_X = F'_X$ is its *probability density function*.

Random Variables

Random variables (r.v.) $X : \Omega \to \mathcal{X}$; often $\mathcal{X} = \mathbb{R}$ (in general spaces: only *measurable* functions)

Basic properties and conventions:

- event $\{X = x\}$ is defined as $\{\omega \in \Omega : X(\omega) = x\}$.
- For event A define the indicator r.v. $\mathbb{1}_A$ via $\mathbb{1}_A(\omega) = [\omega \in A]$
- $F_X(x) = \mathbb{P}[X \le x]$ is the *cumulative distribution function (CDF)*.
- X is *discrete* if $X(\Omega) = \{X(\omega) : \omega \in \Omega\}$ is countable.
- for discrete r.v. X define $f_X(n) = \mathbb{P}[X = n]$ the *probability mass function (PMF)*.
- If F_X is everywhere differentiable, X is *continuous*.
  Then $f_X = F'_X$ is its *probability density function*.

$X \overset{\mathcal{D}}{=} Y + Z$

Equality in distribution:

- We write $X \overset{\mathcal{D}}{=} Y$ if $F_X = F_Y$

Independent Random Variables

Independence:

- Consider *vector* $\boldsymbol{X} = (X_1, \ldots, X_k)$ as single function from Ω to $\mathbb{R}^k$.
  CDF/PMF/PDF of $\boldsymbol{X}$ is called *joint CDF/PMF/PDF* of $X_1, \ldots, X_k$.
- r.v.s *independent* $\iff$ joint PMF/PDF *factors*:
  X and Y independent $\iff$ $\mathbb{P}[X = x \wedge Y = y] = \mathbb{P}[X = x] \cdot \mathbb{P}[Y = y]$ for all x, y.

  (Naturally follows from independent events)

Independent Random Variables

Independence:

- Consider *vector* $\boldsymbol{X} = (X_1, \ldots, X_k)$ as single function from Ω to $\mathbb{R}^k$.
  CDF/PMF/PDF of $\boldsymbol{X}$ is called *joint CDF/PMF/PDF* of $X_1, \ldots, X_k$.
- r.v.s *independent* $\iff$ joint PMF/PDF *factors*:
  X and Y independent $\iff \mathbb{P}[X = x \wedge Y = y] = \mathbb{P}[X = x] \cdot \mathbb{P}[Y = y]$ for all x, y.

  (Naturally follows from independent events)

i.i.d. sequences

- We often talk about sequences of random variables $X_1, X_2, \ldots$
- a sequence of *i.i.d.* r.v. $X_1, X_2, \ldots$ mutually (*independent and identically distributed*)
  has $X_i \overset{\mathcal{D}}{=} X_1$ and $\{X_i\}_{i \geq 1}$ are mutually independent
  - typical example: sequence of coin tosses (with same coin)

Expected Values

Expectation of an $\mathcal{X}$-valued r.v. X, written $\mathbb{E}[X]$, is given by

- $\mathbb{E}[X] = \sum_{x \in \mathcal{X}} x \cdot f_X(x)$ for *discrete* X with PMF f_X,
- $\mathbb{E}[X] = \int_{x \in \mathcal{X}} x \cdot f_X(x) \, dx$ for *continuous* X with PDF f_X.
- undefined if sum does not converge / integral does not exist.

Expected Values

Expectation of an $\mathcal{X}$-valued r.v. X, written $\mathbb{E}[X]$, is given by

- $\mathbb{E}[X] = \sum_{x \in \mathcal{X}} x \cdot f_X(x)$ for *discrete* X with PMF f_X,
- $\mathbb{E}[X] = \int_{x \in \mathcal{X}} x \cdot f_X(x)\, dx$ for *continuous* X with PDF f_X.
- undefined if sum does not converge / integral does not exist.

Properties:

- *linearity*: $\mathbb{E}[aX + bY] = a\mathbb{E}[X] + b\mathbb{E}[Y]$ (X, Y r.v. and a, b constants)
  even if X and Y are not independent
  only for *finite* sums / linear combinations!
- X and Y *independent* $\implies \mathbb{E}[X \cdot Y] = \mathbb{E}[X] \cdot \mathbb{E}[Y]$.

Conditional Expectation

Similar to conditional *probability*, we can define conditional *expectations*.

- *conditional expectation* on event $\mathbb{E}[X \mid A] = \sum_x \mathbb{P}[X = x \mid A]$ for *discrete* X.
  for general A, continuous definition problematic

- *conditional expectation* on $\{Y = y\}$, written $\mathbb{E}[X \mid Y = y]$.
  - for *discrete* X and Y

$$\mathbb{E}[X \mid Y = y] = \sum_{x \in \mathcal{X}} x \cdot \mathbb{P}\big[X = x \,\big|\, \{Y = y\}\big]$$

Conditional Expectation

Similar to conditional *probability*, we can define conditional *expectations*.

- *conditional expectation* on event $\mathbb{E}[X \mid A] = \sum_x \mathbb{P}[X = x \mid A]$ for *discrete* X.
  for general A, continuous definition problematic
- *conditional expectation* on $\{Y = y\}$, written $\mathbb{E}[X \mid Y = y]$.
  - for *discrete* X and Y
    $$\mathbb{E}[X \mid Y = y] = \sum_{x \in \mathcal{X}} x \cdot \mathbb{P}\big[X = x \,\big|\, \{Y = y\}\big]$$
  - for *continuous* X and Y, use the joint density $f_{(X,Y)}$ and define the *marginal density* of Y as $f_Y(y) = \int_{x \in \mathcal{X}} f(x, y)\, dx$. Then
    $$\mathbb{E}[X \mid Y = y] = \int_{\mathcal{X}} x \cdot f_{X|Y}(x, y)\, dx \qquad \text{with} \qquad f_{X|Y}(x, y) = \frac{f_{(X,Y)}(x, y)}{f_Y(y)}$$

Conditional Expectation

Similar to conditional *probability*, we can define conditional *expectations*.

- *conditional expectation* on event $\mathbb{E}[X \mid A] = \sum_x \mathbb{P}[X = x \mid A]$ for *discrete* X.
  for general A, continuous definition problematic
- *conditional expectation* on $\{Y = y\}$, written $\mathbb{E}[X \mid Y = y]$.
  - for *discrete* X and Y
  $$\mathbb{E}[X \mid Y = y] = \sum_{x \in \mathcal{X}} x \cdot \mathbb{P}\big[X = x \,\big|\, \{Y = y\}\big]$$
  - for *continuous* X and Y, use the joint density $f_{(X,Y)}$ and define the *marginal density* of Y as $f_Y(y) = \int_{x \in \mathcal{X}} f(x, y)\, dx$. Then
  $$\mathbb{E}[X \mid Y = y] = \int_{\mathcal{X}} x \cdot f_{X|Y}(x, y)\, dx \qquad \text{with} \qquad f_{X|Y}(x, y) = \frac{f_{(X,Y)}(x, y)}{f_Y(y)}$$
- With $g(y) := \mathbb{E}[X \mid Y = y]$ we obtain a *new r.v.* $\mathbb{E}[X \mid Y] = g(Y)$.
- *law of total expectation*: $\mathbb{E}[X] = \mathbb{E}_Y\big[\mathbb{E}_X[X \mid Y]\big]$.

Famous Distributions

discrete

- *Bernoulli r.v.* $X \overset{\mathcal{D}}{=} \mathrm{B}(p) \quad \rightsquigarrow \quad \mathbb{P}[X=1]=p,\ \mathbb{P}[X=0]=1-p$
- *Binomial r.v.* $Y \overset{\mathcal{D}}{=} \mathrm{Bin}(n,p) \quad \rightsquigarrow \quad Y = X_1 + \cdots + X_n$ for $X_1,\ldots,X_n$ i.i.d. $X_i \overset{\mathcal{D}}{=} \mathrm{B}(p)$
- discrete *uniform r.v.* $X \overset{\mathcal{D}}{=} \mathcal{U}([0..n)) \quad \rightsquigarrow \quad \mathbb{P}[X=i]=\frac{1}{n}$ for $i \in [0..n)$ (else 0) die $\overset{\mathcal{D}}{=} \mathcal{U}([1..6])$
- *Geometric r.v.* $X \overset{\mathcal{D}}{=} \mathrm{Geo}(p) \quad \rightsquigarrow \quad \mathbb{P}[X=k]=(1-p)^{k-1}p$ for $k \in \mathbb{N}_{\geq 1}$

continuous

- continuous uniform $X \overset{\mathcal{D}}{=} \mathcal{U}([0,1)) \quad \rightsquigarrow \quad f_X(x)=1$ for $x \in [0,1)$ (else 0)

(of course there are many more)

7.4 Computing with Randomness

Model of Computation

Definition 7.3 (Probabilistic Turing Machine)

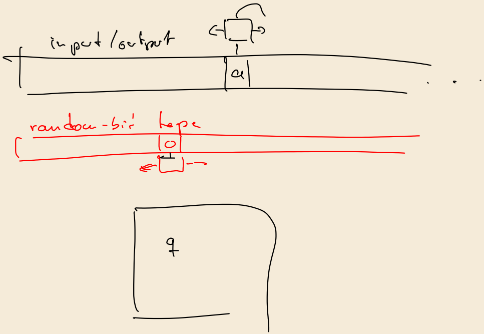
A *probabilistic Turing Machine* (PTM) $M = (Q, \Sigma, \Gamma, \delta, q_0, \Box, q_{\text{halt}})$ is a deterministic TM with an additional read-only tape, filled with random bits.

The *transition function* δ takes as input

- the current state q
- the current tape symbol a
- the current *random-tape symbol* $r \in \{0, 1\}$

and outputs

- the next state q'
- the new tape symbol b
- the tape-head movement $d \in \{L, R, N\}$
- the *random-tape head movement* $d_r \in \{L, R, N\}$



Model of Computation

Definition 7.3 (Probabilistic Turing Machine)

A *probabilistic Turing Machine* (PTM) $M = (Q, \Sigma, \Gamma, \delta, q_0, \square, q_{\text{halt}})$ is a deterministic TM with an additional read-only tape, filled with random bits.
The *transition function* δ takes as input

- the current state q
- the current tape symbol a
- the current *random-tape symbol* $r \in \{0, 1\}$

and outputs

- the next state q'
- the new tape symbol b
- the tape-head movement $d \in \{L, R, N\}$
- the *random-tape head movement* $d_r \in \{L, R, N\}$ ◀

Intended semantics: random tape filled with i.i.d. $\mathrm{B}(\frac{1}{2})$ r.v.

Randomized Computation

- *Configuration* of PTM: $(\alpha q\beta, \rho q\sigma)$
  $\alpha q\beta$ normal TM config
  $\rho\sigma$ content of random tape, with head on first bit of σ

- *computation relation* $\vdash$ similar to TM
  content of random tape unchanged, heads can move independently

- *function computed* by PTM M:
  for input x and **fixed random bits** ρ, computation is deterministic:
  $M(x, \rho) = y$ if $(q_0 x, q_0 \rho) \vdash^\star (q_{\text{halt}} y, \rho' q_{\text{halt}} \rho'')$

Randomized Computation

- *Configuration* of PTM: $(\alpha q\beta, \rho q\sigma)$
  $\alpha q\beta$ normal TM config
  $\rho\sigma$ content of random tape, with head on first bit of σ
- *computation relation* $\vdash$ similar to TM
  content of random tape unchanged, heads can move independently
- *function computed* by PTM M:
  for input x and **fixed random bits** ρ, computation is deterministic:
  $M(x,\rho) \;=\; y \;\text{ if }\; (q_0x, q_0\rho) \;\vdash^\star\; (q_{\text{halt}}y, \rho' q_{\text{halt}}\rho'')$

⇝ ***Randomized computation of PTM:***
random variable $M(x, B_0B_1B_2\ldots)$ where $B_0, B_1, B_2, \ldots$ are i.i.d. $\mathrm{B}(\frac{1}{2})$ distributed

⇝ Write $\mathbb{P}[M(x) = y] \;=\; \sum_b \mathbb{P}[B_0B_1\ldots = b] \cdot [M(x,b) = y]$

- Hope: PTM M so that correct output computed with high probability

Warmup: Rejection Sampling

We assume only random *bits*. How to simulate, say, a fair (6-sided) die?

Warmup: Rejection Sampling

We assume only random *bits*. How to simulate, say, a fair (6-sided) die?

```
procedure rollDie():
    do
        Draw 3 random bits b_2, b_1, b_0
        // Interpret as binary representation of a number in [0..7]
        n = Σ_{i=0}^{2} 2^i b_i
    while (n = 0 ∨ n = 7)
    return n
```

Warmup: Rejection Sampling

We assume only random *bits*. How to simulate, say, a fair (6-sided) die?

---

procedure rollDie():
    do
        Draw 3 random bits b_2, b_1, b_0
        // Interpret as binary representation of a number in $[0..7]$
        $n = \sum_{i=0}^{2} 2^i b_i$
    while ($n = 0 \vee n = 7$)
    return n

Correctness: Every output $1, \ldots, 6$ equally likely by construction.

Warmup: Rejection Sampling

We assume only random *bits*. How to simulate, say, a fair (6-sided) die?

```
procedure rollDie():
    do
        Draw 3 random bits b_2, b_1, b_0
        // Interpret as binary representation of a number in [0..7]
        n = ∑_{i=0}^{2} 2^i b_i
    while (n = 0 ∨ n = 7)
    return n
```

Correctness: Every output $1, \ldots, 6$ equally likely by construction.

Termination: *Infinite* runs possible!

***Expected* Running Time:** Leave loop with probability $\frac{6}{8} = \frac{3}{4}$ in each iteration

$\leadsto$ in expectation, only $\frac{4}{3} = \sum_{i \ge 1} i \cdot \left(\frac{1}{4}\right)^{i-1} \frac{3}{4}$ repetitions.

iterations $\overset{D}{=} \mathrm{Geo}(\frac{3}{4})$

/ $\mathbb{E}[\mathrm{Geo}(p)] = \frac{1}{p}$

rollDie is a correct and practically efficient algorithm.

What can go wrong?

What can go wrong in a randomized computation?

- Computation could run into a deterministic infinite loop (as for deterministic TM)
  - ϟ don't ever terminate, no output
  - ⇝ Clearly don't want that (just as before)

    (annoyingly undecidable to check . . . also just as before)

What can go wrong?

What can go wrong in a randomized computation?

- Computation could run into a deterministic infinite loop (as for deterministic TM)
  - ϟ don't ever terminate, no output
  - ⇝ Clearly don't want that (just as before)

    (annoyingly undecidable to check . . . also just as before)

- Computation could repeatedly have branches that keep looping (as for rollDie)
  - ⇝ For every t, there is a probability $p > 0$ to run for more than t time steps
  - This is a new option that deterministic TMs didn't have
    . . . but nondeterministic TMs did, and we just defined running time to be ∞ there!

    So, is that a problem? Or is it not??

Random Termination

Key question: *What is the probability space for the running time of the PTM simulating rollDie?*

- Note: this could indeed be a problem.
  - $\{0,1\}^\star$ (the set of **finite** bitstrings) is countably infinite (=discrete)
  - But the set of ***infinite strings*** (*ω-language*) is not!
    $\{0,1\}^\omega = \{b_0 b_1 \ldots : b_i \in \{0,1\}\} = \{b : b : \mathbb{N}_0 \to \{0,1\}\}$ is in bijection with $[0,1) \subset \mathbb{R}$
    $b \mapsto 0.b_0 b_1 b_2 \ldots$

Random Termination

Key question: *What is the probability space for the running time of the PTM simulating rollDie?*

- Note: this could indeed be a problem.
  - $\{0,1\}^\star$ (the set of **finite** bitstrings) is countably infinite (=discrete)
  - But the set of ***infinite strings*** (ω-*language*) is not!
    $\{0,1\}^\omega = \{b_0 b_1 \ldots : b_i \in \{0,1\}\} = \{b : b : \mathbb{N}_0 \to \{0,1\}\}$ is in bijection with $[0,1) \subset \mathbb{R}$
    $b \mapsto 0.b_0 b_1 b_2 \ldots$
- Config $(\alpha q \beta, \rho q \boldsymbol{\sigma})$ for PTM needs $\sigma \in \{0,1\}^{\boldsymbol{\omega}}$ in general

Random Termination

Key question: *What is the probability space for the running time of the PTM simulating rollDie?*

- Note: this could indeed be a problem.
  - $\{0,1\}^\star$ (the set of **finite** bitstrings) is countably infinite (=discrete)
  - But the set of ***infinite strings*** (ω*-language*) is not!
    $\{0,1\}^\omega = \{b_0 b_1 \ldots : b_i \in \{0,1\}\} = \{b : b : \mathbb{N}_0 \to \{0,1\}\}$ is in bijection with $[0,1) \subset \mathbb{R}$
    $b \mapsto 0.b_0 b_1 b_2 \ldots$
- Config $(\alpha q \beta, \rho q \boldsymbol{\sigma})$ for PTM needs $\sigma \in \{0,1\}^{\boldsymbol{\omega}}$ in general

- Define the random variable $Time_M(x) \in \mathbb{N}_0 \cup \{\infty\}$ on the *Bernoulli probability space*
  - generators: $\{\pi_x : x \in \{0,1\}^\star\}$ where $\pi_x = \{xw : w \in \{0,1\}^\omega\} \subseteq \{0,1\}^\omega$
  - Bernoulli σ-algebra: smallest $\mathcal{F}$ containing all $\{\pi_x\}_x$ that is closed under countable union and complement
  - $\mathbb{P}[\pi_x] = 2^{-|x|}$

rollDie terminates iff $\rho \in T \subseteq \{0,1\}^\omega$ $\mathbb{P}[T]$

Random Termination

Key question: *What is the probability space for the running time of the PTM simulating rollDie?*

- Note: this could indeed be a problem.
  - $\{0,1\}^\star$ (the set of **finite** bitstrings) is countably infinite (=discrete)
  - But the set of ***infinite strings*** (ω-*language*) is not!
    $\{0,1\}^\omega = \{b_0 b_1 \ldots : b_i \in \{0,1\}\} = \{b : b : \mathbb{N}_0 \to \{0,1\}\}$ is in bijection with $[0,1) \subset \mathbb{R}$
    $b \mapsto 0.b_0 b_1 b_2 \ldots$
- Config $(\alpha q \beta, \rho q \boldsymbol{\sigma})$ for PTM needs $\sigma \in \{0,1\}^{\boldsymbol{\omega}}$ in general

- Define the random variable $Time_M(x) \in \mathbb{N}_0 \cup \{\infty\}$ on the *Bernoulli probability space*
  - generators: $\{\pi_x : x \in \{0,1\}^\star\}$ where $\pi_x = \{xw : w \in \{0,1\}^\omega\} \subseteq \{0,1\}^\omega$
  - Bernoulli σ-algebra: smallest $\mathcal{F}$ containing all $\{\pi_x\}_x$ that is closed under countable union and complement
  - $\mathbb{P}[\pi_x] = 2^{-|x|}$

$\rightsquigarrow$ expectations over $\rho \in \{0,1\}^\omega$, the infinite initial random-bit tape input are well-defined

(Expected) Time

Definition 7.4 (PTM running time)

For a PTM M, we define $time_M(x)$ as for nondeterministic TMs as the supremum of time steps over all computations.

Moreover, we define the *expected time* as

$$\mathbb{E}\text{-}time_M(x) \;=\; \mathbb{E}[time_M(x)] \;=\; \mathbb{E}_{\rho}\Big[\inf\{t \in \mathbb{N}_0 : (q_0 x, q_0 \rho) \vdash^{t} (q_{\text{halt}} y, \rho' q_{\text{halt}} \rho'')\}\Big]$$

Similarly

$$\mathbb{E}\text{-}Time_M(n) \;=\; \sup\big\{\mathbb{E}\text{-}time_M(x) : x \in \Sigma^n\big\}$$

(Expected) Time

Definition 7.4 (PTM running time)

For a PTM M, we define $time_M(x)$ as for nondeterministic TMs as the supremum of time steps over all computations.
Moreover, we define the *expected time* as

$$\mathbb{E}\text{-}time_M(x) \;=\; \mathbb{E}[time_M(x)] \;=\; \mathbb{E}_\rho\Big[\inf\{t \in \mathbb{N}_0 : (q_0 x, q_0 \rho) \vdash^t (q_{\text{halt}} y, \rho' q_{\text{halt}} \rho'')\Big]$$

Similarly

$$\mathbb{E}\text{-}Time_M(n) \;=\; \sup\big\{\mathbb{E}\text{-}time_M(x) : x \in \Sigma^n\big\}$$

◀

- We can of course also study full distribution of $time_M(x)$
- Useful property of expected time:
  $\mathbb{E}\text{-}time_M(x) < \infty$ iff $\mathbb{P}[time_M(x) = \infty] = 0$

A New Complexity Measure: Random Bits

Definition 7.5 (Random-bit complexity)

For a PTM M computing with input alphabet Σ, the *random-bit cost* for an input $x \in \Sigma^\star$ is denote by

$$random_M(x) \quad = \quad \sup\left\{|\rho'| : (xq_0, q_0\rho) \vdash^\star (\alpha q \beta, \rho' q \rho'') \vdash^\star (q_{\text{halt}} y, \rho' q_{\text{halt}} \rho'')\right\}$$

and similarly

$$Random_M(n) \quad = \quad \sup\left\{random_M(x) : x \in \Sigma^n\right\}.$$

Further, the *expected random-bit cost* are defined as
$\mathbb{E}\text{-}random_M(x) = \mathbb{E}_\rho[random_M(x)]$ and
$\mathbb{E}\text{-}Random_M(n) = \sup\left\{\mathbb{E}\text{-}random_M(x) : x \in \Sigma^n\right\}$ ◀

Randomization vs. Nondeterminism

- Superficially similar concepts
- Key difference: meaning of number of computations of TM
  - nondeterministic TM: accept if **some (single)** accepting computation is possible
  - randomized TM: accept if **most** possible computations are accepting

$\rightsquigarrow$ nondeterminism = purely theoretical construction (overly powerful yardstick)

- randomization = widely applied efficient design technique