MINIQUIZ

Für beliebige Zufallsvariablen $X_1, X_2, \dots X_n$ und Konstanten $a_1, \dots a_n$ gilt $\mathbb{E}[a_1 X_1 + a_2 X_2 + \dots a_n X_n] = a_1 \mathbb{E}[X_1] + \dots + a_n \mathbb{E}[X_n]$

- [x] Wahr
- [ ] Falsch

Für beliebige Zufallsvariablen X_1, X_2 gilt $Var(X_1 + X_2) = Var(X_1) + Var(X_2)$.

- [ ] Wahr
- [x] Falsch

X_1 und X_2 nicht zwingend unabhängig

Ist die erwartete Anzahl von Vergleichen für den QuickSort-Algorithmus $O(n \log n)$?

- [x] Wahr
- [ ] Falsch

Sind X und Y unabhängige Zufallsvariablen, dann ist $\mathbb{E}[XY] = \mathbb{E}[X]\mathbb{E}[Y]$.

- [x] Wahr
- [ ] Falsch

Sind $X_1, X_2, \dots, X_n$ unabhängige Zufallsvariablen mit gleicher Verteilung, sodass $Var(X_i) = 1$ für alle i, dann ist $Var(X_1 + \dots + X_n) = n$.

- [x] Wahr
- [ ] Falsch

Für eine Zufallsvariable $Var(X) = \sigma^2$ und $\mathbb{E}[X] = 0$ gilt:

$\Pr(X > \lambda\sigma) \leq 1/\lambda^2$.

$\Pr[X > \lambda\sigma] \leq \Pr[X \geq \lambda\sigma] \leq \Pr[|X - \underbrace{\mathbb{E}[X]}_{=0}| \geq \lambda\sigma] \leq \frac{\sigma^2}{(\lambda\sigma)^2} = \frac{1}{\lambda^2}$

- [x] Wahr
- [ ] Falsch

Wenn $A(x)$ ein randomisierter Algorithmus ist, der mit einer Wahrscheinlichkeit von $3/4$ die richtige Ja/Nein-Antwort liefert, können wir diesen Algorithmus $O(\log(1/\delta))$ mal mit unabhängigen Zufallsbits wiederholen, und die häufigste Antwort wird mit einer Wahrscheinlichkeit von $1 - \delta$ richtig sein.

- [x] Wahr
- [ ] Falsch

Wenn X eine nicht-negative Zufallsvariable ist, mit $\mathbb{E}[X] > 100$, dann ist $\mathbf{Pr}[X > 10] \geq 1/2$.

$\Omega = \{0, 1, 1000\}$ $X(\omega) = \omega$ $E[X] = \frac{1}{3} + \frac{1000}{3}$

- ○ Wahr
- ○ Falsch

Für eine Zufallsvariable X und eine Konstante a gilt $Var(X + a) = Var(X) + a$.

$Var[X + a] = Var[X]$

- ○ Wahr
- ✗ Falsch

Wenn wir unabhängig voneinander zufällig n Bälle in n Behälter werfen (wobei jeder Ball mit gleicher Wahrscheinlichkeit in jedem Behälter landet) und X_1 die Anzahl der Bälle im ersten Behälter am Ende des Prozesses bezeichnet, dann ist $\mathbb{E}[X_1] = 1$.

- ✗ Wahr
- ○ Falsch

Peergrading 3

Exercise 1 – *Balls and bins*

Consider a process where n balls are thrown indpendently, each to a uniformly random bin among $\{1, \dots n\}$. Let X_i denote the number of balls in the bin i at the end of this process. Show that for some large constant C, we have

$$\Pr(\max_{i \in \{1,\dots n\}} X_i > \underbrace{C\frac{\log n}{\log\log n}}_{k}) < 1/4.$$

Hint: You can use without proof that $\binom{n}{k} \le \left(\frac{en}{k}\right)^k$.

$\Pr[\max_{i\in\{1,\dots,n\}} X_i > k] \le \sum_{i=1}^{n} \Pr[X_i > k] \overset{!}{<} \frac{1}{4}$

$\Pr[X_1 > k] < \frac{1}{4n}$, $Y_j :=$ Eimer vom i-ten Ball

$X_1 > k \Leftrightarrow \exists S \subset \{1,\dots,n\} : |S| > k \wedge Y_i = 1 \quad i \in S$

$\Pr[X_1 > k] \le \sum_{S \in \binom{[n]}{k+1}} \Pr[\forall j \in S : Y_j = 1] = \sum_{S \in \binom{[n]}{k+1}} \prod_{j \in S} \underbrace{\Pr[Y_j = 1]}_{\frac{1}{n}} = \sum_{S \in \binom{[n]}{k+1}} \left(\frac{1}{n}\right)^{k+1} = \binom{n}{k+1} \cdot \left(\frac{1}{n}\right)^{k+1} \le \left(\frac{en}{k+1}\right)^{k+1} \cdot \left(\frac{1}{n}\right)^{k+1}$

$= \left(\frac{e}{k+1}\right)^{k+1} \overset{!}{<} \frac{1}{4n}$

$[n] = \{1,\dots,n\}$

$k+1 > \frac{10\log n}{\log\log n} > \frac{\sqrt{\log n}}{e}$

$\left(\frac{e}{k+1}\right)^{k+1} < \left(\frac{1}{\frac{\sqrt{\log n}}{e}}\right)^{k+1} = \left(\frac{e}{\sqrt{\log n}}\right)^{k+1} = \log(n)^{-\frac{1}{2}(k+1)} = e^{-\frac{1}{2}\log(\log n)(k+1)} < e^{-5\log n} = e^{\log(\frac{1}{n^5})} = \frac{1}{n^5} < \frac{1}{4n}$

$\log(n) = e^{\log\log n}$

Schranken (Part 2)

Satz 2.67. *(Ungleichung von Markov)* Sei X eine Zufallsvariable, die nur nicht-negative Werte annimmt. Dann gilt für alle $t \in \mathbb{R}$ mit $t > 0$, dass

$$\Pr[X \geq t] \leq \frac{\mathbb{E}[X]}{t}.$$

Oder äquivalent dazu $\Pr[X \geq t \cdot \mathbb{E}[X]] \leq 1/t$.

Satz 2.68. *(Ungleichung von Chebyshev)* Sei X eine Zufallsvariable und $t \in \mathbb{R}$ mit $t > 0$. Dann gilt

$$\Pr[|X - \mathbb{E}[X]| \geq t] \leq \frac{\mathrm{Var}[X]}{t^2}$$

oder äquivalent dazu $\Pr[|X - \mathbb{E}[X]| \geq t\sqrt{\mathrm{Var}[X]}] \leq 1/t^2$.

Satz 2.70 (Chernoff-Schranken). Seien $X_1, \ldots, X_n$ unabhängige Bernoulli-verteilte Zufallsvariablen mit $\Pr[X_i = 1] = p_i$ and $\Pr[X_i = 0] = 1 - p_i$. Dann gilt für $X := \sum_{i=1}^{n} X_i$:

(i) $\Pr[X \geq (1+\delta)\mathbb{E}[X]] \leq e^{-\frac{1}{3}\delta^2 \mathbb{E}[X]}$ für alle $0 < \delta \leq 1$,

(ii) $\Pr[X \leq (1-\delta)\mathbb{E}[X]] \leq e^{-\frac{1}{2}\delta^2 \mathbb{E}[X]}$ für alle $0 < \delta \leq 1$,

(iii) $\Pr[X \geq t] \leq 2^{-t}$ für $t \geq 2e\mathbb{E}[X]$.

Prüfungsaufgabe [FS20]

Exercise 3 – *Obere Schranken*

Wir werfen eine faire Münze $n \geq 1$ mal. Sei X die Anzahl der Würfe bei denen die Münze 'Kopf' zeigt. Geben Sie möglichst gute obere Schranken für $\Pr[X \geq 0.75\,n]$ an.

(a) Mithilfe der Ungleichung von Markov. *(1 points)*

$$E[X] = \frac{n}{2}$$

$$\Pr[X \geq 0.75n] \leq \frac{\frac{n}{2}}{0.75n} = \frac{2}{3}$$

(b) Mithilfe der Ungleichung von Chebychev. *(2 points)*

$$\Pr[|X - E[X]| > t]$$

$$E[X] = \frac{n}{2}$$

$$\mathrm{Var}[X] = np(1-p) = \frac{n}{4}$$

$$\Pr[X \geq \tfrac{3}{4}n] = \Pr[X - E[X] \geq \tfrac{1}{4}n]$$

$$\leq \Pr[|X - E[X]| \geq \tfrac{1}{4}n] \leq \frac{\mathrm{Var}[X]}{\frac{n^2}{16}} = \frac{\frac{n}{4}}{\frac{n^2}{16}} = \frac{4}{n}$$

(c) Mithilfe der Chernoff Schranken. *(2 points)*

(i) $\Pr[X \geq (1+\delta)E[X]] \leq e^{-\frac{1}{3}\delta^2 E[X]}$

$$\Pr[X \geq \tfrac{3}{4}n] = \Pr[X \geq (1+\tfrac{1}{2})\tfrac{n}{2}] \leq e^{-\frac{1}{3}\cdot\frac{1}{4}\cdot\frac{n}{2}} = e^{-\frac{n}{24}}$$

(δ)

Randomisierte Algorithmen

randomisierter Algorithmus: Eingabe $I \rightarrow$ Algorithmus A mit Zufallszahlen $R \rightarrow$ Ausgabe $A(I,R)$

deterministisch: selbe Eingabe, selber Output
nicht-deterministisch: selbe Eingabe, vielleicht unterschiedlicher Output

Monte Carlo	vs	Las Vegas
ZV: Korrektheit		ZV: Laufzeit
immer gleiche Laufzeit		manchmal zu langsam / "???" ausgeben
manchmal falsches Ergebnis		immer korrekte Antwort
Beispiele: Primzahltest, Target-Shooting		Beispiele: Quicksort

Reduktion von Fehlerwahrscheinlichkeiten

① Las Vegas

nie falsche Antwort, aber manchmal "???" als Ausgabe
$Pr[\mathcal{A}(I) \text{ korrekt}] \geq \varepsilon$

Es gilt $\forall S>0$, $\mathcal{A}_S$ ein Algo welcher $\mathcal{A}$ solange aufruft bis richtige Antwort oder $N = \frac{1}{\varepsilon} \ln(\frac{1}{\delta})$ erfolglose Aufrufe, dann

$$Pr[\mathcal{A}_S(I) \text{ korrekt}] \geq 1-S$$

ϵ	δ	N
0.1	0.01	47
0.5	0.01	10
0.5	10^{-80}	369
0.9	10^{-30}	77

Beweis:

$$Pr[\mathcal{A} \text{ gibt } N\text{-mal "???"}] \leq (1-\varepsilon)^N \overset{1-x \leq e^{-x}}{\leq} e^{-\varepsilon N} = e^{-\varepsilon \frac{1}{\varepsilon} \ln(S^{-1})} = e^{\ln(S)} = S$$

$$Pr[\mathcal{A}_S(I) \text{ korrekt}] \geq 1-S$$

② Monte Carlo (einseitiger Fehler)

$\mathcal{A}$ rand. Algo mit Ausgabe $\in \{\text{Ja}, \text{Nein}\}$

$Pr[\mathcal{A}(I) = \text{Ja}] = 1$ falls I Ja-Instanz
$Pr[\mathcal{A}(I) = \text{Nein}] \geq \varepsilon$ falls I Nein-Instanz

$\forall S>0$ $\mathcal{A}_S$ = Algo., welcher $\mathcal{A}$ solange aufruft, bis Nein oder $N = \frac{1}{\varepsilon} \ln(\frac{1}{\delta})$ - mal Ja ausgegeben wird

Dann gilt: $Pr[\mathcal{A}_S(I) \text{ korrekt}] \geq 1-S$

Beweis analog zu ①

③ Monte Carlo (zweiseitiger Fehler)

$\mathcal{A}$ rand. Algo. mit Ausgabe $\in \{Ja, Nein\}$

$\Pr[\underbrace{\mathcal{A}(I) \text{ korrekt}}_{p}] \geq \frac{1}{2} + \varepsilon$

$\forall \delta > 0$ $\mathcal{A}_\delta$:= Algo, welcher $\mathcal{A}$ $N = \frac{4}{\varepsilon^2} \ln(\frac{1}{\delta})$ - mal aufruft und die Mehrheit der erhaltenen Antworten ausgibt

Dann gilt: $\Pr[\mathcal{A}_\delta(I) \text{ korrekt}] \geq 1 - \delta$

Beweis: $X :=$ # richtige Antworten bei N Aufrufen

$\varepsilon \leq \frac{1}{2}$

$\Pr[\mathcal{A}_\delta(I) \text{ korrekt}] \geq \Pr[X > \frac{N}{2}] = 1 - \Pr[X \leq \frac{N}{2}]$

$E[X] = Np \geq N(\frac{1}{2} + \varepsilon) = \frac{N}{2} + N\varepsilon$

$\Leftrightarrow \frac{N}{2} \leq Np - \varepsilon N = N(p - \varepsilon) \leq (1-\varepsilon)N \leq (1-\varepsilon)(\frac{N}{2} + \varepsilon N) \leq (1-\varepsilon)E[X]$

$\Pr[X \leq \frac{N}{2}] \leq \Pr[X \leq (1-\varepsilon)E[X]] \leq e^{-\frac{1}{2}\varepsilon^2 \frac{N}{2}} \leq e^{-\frac{1}{2}\varepsilon^2 \cdot \frac{2}{\varepsilon^2} \ln(\frac{1}{\delta})} = \delta$

Aufgabe: Broken Servers

Sie sind mit einem Netzwerk verbunden, das aus n Servern besteht, die von 1 bis n nummeriert sind. Sie können jeden Server i in Zeit $\mathcal{O}(1)$ kontaktieren und erhalten als Antwort entweder eine '0' oder eine '1'. Leider sind einige der Server kaputt uns Sie sollen herausfinden welche Server betroffen sind.

- Falls Server i kaputt ist, sendet er bei jeder Anfrage ein unabhängig gleichverteiltes Bit.
- Falls Server i intakt ist, dann antwortet er auf jede Anfrage mit dem gleichen Bit $a_i \in \{0,1\}$.

  Allerdings ist der Wert a_i unbekannt und kann von Server zu Server variieren.

(a) Seien $\delta > 0$ und $i \in [n]$ gegeben. Beschreiben Sie einen Monte-Carlo Algorithmus, der herausfindet, ob Server i kaputt ist. Berechnen Sie die Fehlerwahrscheinlichkeiten (abhängig davon ob der Server kaputt/intakt ist) Ihres Algorithmus und stellen Sie sicher, dass Ihr Algorithmus Fehlerwahrscheinlichkeit höchstens $\frac{\delta}{n}$ hat.

(b) Sei $\delta > 0$ gegeben. Beschreiben Sie einen Monte-Carlo Algorithmus, der eine Liste aller kaputten Server erstellt und Fehlerwahrscheinlichkeit höchstens δ hat. Hierbei sagen wir, dass der Algorithmus erfolgreich ist, wenn die Liste alle kaputten Server und keinen intakten Server enthält.

a) k Anfragen, wenn alle Antworten gleich $\rightarrow$ Ausgabe: "Server i ist intakt"

X_k = alle k Anfragen bekommen gleiche Antwort

$\Pr[X_k \mid \text{Server } i \text{ ist kaputt}] = \left(\frac{1}{2}\right)^{k-1} \overset{!}{\leq} \frac{\delta}{n}$ $\qquad |\{0,1\}^k| = 2^k$

$\frac{1}{2^k} \cdot 2$

$\Leftrightarrow \left(\frac{1}{2}\right)^k \leq \frac{2\delta}{n}$

$\Leftarrow k \geq -\log_2\left(2\cdot\frac{\delta}{n}\right) = -\log_2\left(\frac{\delta}{n}\right) - 1$ ($-\log_2\left(\frac{\delta}{n}\right) \geq 1$)

b) Nutzen von a) für jeden Server

Y_i: Indikatorvariable für falsches Ergebnis bei Server i

$Y = \sum_{i=1}^{n} Y_i$

$\Pr[\text{Liste inkorrekt}] = \Pr[Y \geq 1] \overset{\text{Markov}}{\leq} \frac{E[Y]}{1} \leq \delta$

$E[Y] = \sum_{i=1}^{n} E[Y_i] \leq \frac{\delta}{n} \cdot n = \delta$

TARGET SHOOTING

Gegeben: endliche Mengen S, U so dass $S \subseteq U$, $I_S : U \to \{0,1\}$ $I_S(u) = 1 \iff u \in S$

Gesucht: $\frac{|S|}{|U|}$

TARGET-SHOOTING

1: Wähle $u_1, \ldots, u_N \in U$ zufällig, gleichverteilt und unabhängig

2: **return** $N^{-1} \cdot \sum_{i=1}^{N} I_S(u_i)$

Sei $\varepsilon > 0$ beliebig klein. Wie groß muss N sein, damit der Algorithmus mit Wahrscheinlichkeit $\geq 1-\delta$ eine Antwort im Intervall $\left[(1-\varepsilon)\frac{|S|}{|U|}, (1+\varepsilon)\frac{|S|}{|U|}\right]$ ausgibt?

Satz 2.79. Seien $\delta, \varepsilon > 0$. Falls $N \geq 3\frac{|U|}{|S|} \cdot \varepsilon^{-2} \cdot \ln(2/\delta)$, so ist die Ausgabe des Algorithmus TARGET-SHOOTING mit Wahrscheinlichkeit mindestens $1 - \delta$ im Intervall $\left[(1-\varepsilon)\frac{|S|}{|U|}, (1+\varepsilon)\frac{|S|}{|U|}\right]$.

Beweis:

PRIMZAHLTEST

übliches Finden von Primzahlen: Aussuchen einer random Zahl n gegebener Länge, dann testen, ob sie eine Primzahl ist

naive Option: alle Teiler bis $\sqrt{n}$ testen → ineffizient

random Zahl $a \in \{2, \dots, \sqrt{n}\}$ auswählen und schauen ob $\gcd(a,n) > 1$

Kleiner fermatscher Satz: Ist $n \in \mathbb{N}$ prim, so gilt für alle Zahlen $0 < a < n$ $a^{n-1} \equiv_n 1$.

Carmichael-Zahl n: für alle teilerfremden Zahlen a gilt $a^{n-1} \equiv_n 1$
kleinste Carmichael-Zahl: $561 = 3 \cdot 11 \cdot 17$

Miller-Rabin-Primzahltest

Idee: wenn n prim, dann ist $(\mathbb{Z}_n, +, *)$ ein Körper
→ $x^2 \equiv_n 1$ hat Lösungen $x = 1$ und $x = -1 \equiv_n n-1$

$n - 1 = d \cdot 2^k$, d ungerade

n prim $\Leftrightarrow \forall a \in \{1, \dots, n-1\}$ gilt $a^{n-1} = (a^d)^{2^k} \equiv_n 1$

MILLER-RABIN-PRIMZAHLTEST(n)

```
1: if n = 2 then
2:     return 'Primzahl'
3: else if n gerade oder n = 1 then
4:     return 'keine Primzahl'
5: Wähle a ∈ {2, 3, ..., n − 1} zufällig und
6: berechne k, d ∈ ℤ mit n − 1 = d2^k und d ungerade.
7: x ← a^d (mod n)
8: if x = 1 or x = n − 1 then
9:     return 'Primzahl'
10: repeat k − 1 mal
11:     x ← x^2 (mod n)
12:     if x = 1 then
13:         return 'keine Primzahl'
14:     if x = n − 1 then
15:         return 'Primzahl'
16: return 'keine Primzahl'
```

Laufzeit $O(\ln(n))$

$\Pr[\text{"nicht prim"} \mid \text{nicht prim}] \geq \frac{3}{4}$

MINIQUIZFRAGEN

1) Wir können jeden Las Vegas Algorithmus mit erwarteter Laufzeit T in einen randomisierten Algorithmus mit Laufzeit höchstens $10T$ und Erfolgsw'keit mind. 0,9 umwandeln.

2) Wir wollen überprüfen, ob ein Graph G auf $n \geq 3$ Knoten die Eigenschaft hat, dass alle Knoten höchstens Grad 10 haben. Wir schlagen folgenden Algorithmus A vor:

Mit Input G, wähle einen Knoten $v \in G$ zufällig gleichverteilt. Falls $d(v) \leq 10$ geben wir "Wahr" aus, ansonsten "Falsch".

(a) Falls G mind. einen Knoten mit $d(v) > 10$ hat, gilt $Pr[A(G) = \text{"Falsch"}] \geq$

(b) Dies ist ein Algorithmus

(c) Falls G keinen Knoten v mit $d(v) > 10$ enthält, dann $Pr[A(G) = \text{"Wahr"}] = 1$

3) Wenn wir wissen, dass 52633 eine Carmichael Zahl ist, folgt daraus, dass $11^{52632} \equiv_{52632} 1$ ist?

4) Betrachte den Fermat-Primzahltest

(a) Die Ausgabe "keine Primzahl" ist immer richtig

(b) Die Ausgabe "Primzahl" ist immer richtig