CSE 312

Foundations of Computing II

Lecture 11: Bloom Filters continued, Zoo of Discrete RVs, part I

Review Variance – Properties

Definition. The **variance** of a (discrete) RV X is

$$\mathrm{Var}(X) = \mathbb{E}[(X - \mathbb{E}[X])^2] = \sum_x p_X(x) \cdot (x - \mathbb{E}[X])^2$$

Theorem. For any $a, b \in \mathbb{R}$, $\mathrm{Var}(a \cdot X + b) = a^2 \cdot \mathrm{Var}(X)$

Theorem. $\mathrm{Var}(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2$

Review Important Facts about Independent Random Variables

Theorem. If X, Y independent, $\mathbb{E}[X \cdot Y] = \mathbb{E}[X] \cdot \mathbb{E}[Y]$

Theorem. If X, Y independent, $\mathrm{Var}(X + Y) = \mathrm{Var}(X) + \mathrm{Var}(Y)$

Corollary. If $X_1, X_2, \dots, X_n$ mutually independent,

$$\mathrm{Var}\left(\sum_{i=1}^{n} X_i\right) = \sum_{i}^{n} \mathrm{Var}(X_i)$$

Agenda

- Bloom Filters Example & Analysis ◀
- Zoo of Discrete RVs
  - – Uniform Random Variables
  - – Bernoulli Random Variables
  - – Binomial Random Variables
  - – Applications

Basic Problem

Problem: Store a subset S of a large set U.

Example. $U =$ set of 128 bit strings $|U| \approx 2^{128}$
$S =$ subset of strings of interest $|S| \approx 1000$

Two goals:

1. **Very fast** (ideally constant time) answers to queries "Is $x \in S$?" for any $x \in U$.
2. **Minimal storage** requirements.

Bloom Filters

to the rescue

(Named after Burton Howard Bloom)

Bloom Filters

- Stores information about a set of elements $S \subseteq U$.
- Supports two operations:
  1. `add`(x) - adds $x \in U$ to the set S
  2. `contains`(x) – ideally: true if $x \in S$, false otherwise

Possible *false positives*

Combine with fallback mechanism – can distinguish false positives from true positives with extra cost

Bloom Filters – Ingredients

t_1	1	0	1	0	0
t_2	0	1	0	0	1
t_3	1	0	0	1	0

Basic data structure is a $k \times m$ binary array "the Bloom filter"

- k rows $t_1, \dots, t_k$, each of size m
- Think of each row as an m-bit vector

k different hash functions $\mathbf{h}_1, \dots, \mathbf{h}_k: U \rightarrow [m]$

Bloom Filters – Three operations

- Set up Bloom filter for $S = \emptyset$

```
function INITIALIZE($k, m$)
    for $i = 1, \ldots, k$: do
        $t_i$ = new bit vector of $m$ 0s
```

- Update Bloom filter for $S \leftarrow S \cup \{x\}$

```
function ADD($x$)
    for $i = 1, \ldots, k$: do
        $t_i[h_i(x)] = 1$
```

- Check if $x \in S$

```
function CONTAINS($x$)
    return $t_1[h_1(x)] == 1 \wedge t_2[h_2(x)] == 1 \wedge \cdots \wedge t_k[h_k(x)] == 1$
```

Bloom Filters - Initialization

Number of hash functions

Size of array associated to each hash function.

```
function INITIALIZE($k, m$)
    for $i = 1, \dots, k$: do
        $t_i$ = new bit vector of $m$ 0s
```

for each hash function, initialize an empty bit vector of size m

Bloom Filters: Example

Bloom filter $\boldsymbol{t}$ of length $\boldsymbol{m}$ = 5 that uses $\boldsymbol{k}$ = 3 hash functions

```
function INITIALIZE($k, m$)
    for $i = 1, \ldots, k$: do
        $t_i$ = new bit vector of $m$ 0s
```

Index →	0	1	2	3	4
t_1	0	0	0	0	0
t_2	0	0	0	0	0
t_3	0	0	0	0	0

Bloom Filters: Add

```
function ADD($x$)
    for $i = 1, \dots, k$: do
        $t_i[h_i(x)] = 1$
```

for each hash function $\mathbf{h}_i$

Index into i-th bit-vector, at index produced by hash function and set to 1

$\mathbf{h}_i(x) \rightarrow$ result of hash function $\mathbf{h}_i$ on x

Bloom Filters: Example

Bloom filter t of length $\boldsymbol{m}$ = 5 that uses $\boldsymbol{k}$ = 3 hash functions

```
function ADD($x$)
    for $i = 1, \ldots, k$: do
        $t_i[h_i(x)] = 1$
```

add("thisisavirus.com")

h_1("thisisavirus.com") → 2

Index →	0	1	2	3	4
t_1	0	0	0	0	0
t_2	0	0	0	0	0
t_3	0	0	0	0	0

Bloom Filters: Example

Bloom filter t of length $\boldsymbol{m} = 5$ that uses $\boldsymbol{k} = 3$ hash functions

```
function ADD($x$)
    for $i = 1, \dots, k$: do
        $t_i[h_i(x)] = 1$
```

add("thisisavirus.com")

h_1("thisisavirus.com") → 2

h_2("thisisavirus.com") → 1

Index →	0	1	2	3	4
t_1	0	0	1	0	0
t_2	0	0	0	0	0
t_3	0	0	0	0	0

Bloom Filters: Example

Bloom filter t of length $\boldsymbol{m} = 5$ that uses $\boldsymbol{k} = 3$ hash functions

```
function ADD($x$)
    for $i = 1, \ldots, k$: do
        $t_i[h_i(x)] = 1$
```

add("thisisavirus.com")

h_1("thisisavirus.com") → 2

h_2("thisisavirus.com") → 1

h_3("thisisavirus.com") → 4

Index →	0	1	2	3	4
t_1	0	0	1	0	0
t_2	0	1	0	0	0
t_3	0	0	0	0	0

Bloom Filters: Example

Bloom filter t of length $\boldsymbol{m} = 5$ that uses $\boldsymbol{k} = 3$ hash functions

```
function ADD($x$)
    for $i = 1, \ldots, k$: do
        $t_i[h_i(x)] = 1$
```

add(“thisisavirus.com”)

h_1(“thisisavirus.com”) → 2

h_2(“thisisavirus.com”) → 1

h_3(“thisisavirus.com”) → 4

Index →	0	1	2	3	4
t_1	0	0	1	0	0
t_2	0	1	0	0	0
t_3	0	0	0	0	1

Bloom Filters: Contains

```
**function** CONTAINS($(x)$)
    **return** $t_1[h_1(x)] == 1 \wedge t_2[h_2(x)] == 1 \wedge \cdots \wedge t_k[h_k(x)] == 1$
```

Returns True if the bit vector t_i for each hash function has bit 1 at index determined by $h_i(x)$,

Returns False otherwise

Bloom Filters: Example

Bloom filter t of length $\boldsymbol{m} = 5$ that uses $\boldsymbol{k} = 3$ hash functions

```
**function** CONTAINS($x$)
    **return** $t_1[h_1(x)] == 1 \land t_2[h_2(x)] == 1 \land \cdots \land t_k[h_k(x)] == 1$
```

contains("thisisavirus.com")

Index →	0	1	2	3	4
t_1	0	0	1	0	0
t_2	0	1	0	0	0
t_3	0	0	0	0	1

Bloom Filters: Example

Bloom filter t of length $\boldsymbol{m}$ = 5 that uses $\boldsymbol{k}$ = 3 hash functions

```
function CONTAINS($x$)
    return $t_1[h_1(x)] == 1 \land t_2[h_2(x)] == 1 \land \cdots \land t_k[h_k(x)] == 1$
```

True

contains("thisisavirus.com")

h_1("thisisavirus.com") → 2

Index →	0	1	2	3	4
t_1	0	0	1	0	0
t_2	0	1	0	0	0
t_3	0	0	0	0	1

Bloom Filters: Example

Bloom filter t of length $\boldsymbol{m}$ = 5 that uses $\boldsymbol{k}$ = 3 hash functions

```
function CONTAINS($x$)
    return $t_1[h_1(x)] == 1 \wedge t_2[h_2(x)] == 1 \wedge \cdots \wedge t_k[h_k(x)] == 1$
```

True True

contains("thisisavirus.com")

h_1("thisisavirus.com") → 2

h_2("thisisavirus.com") → 1

Index →	0	1	2	3	4
t_1	0	0	1	0	0
t_2	0	1	0	0	0
t_3	0	0	0	0	1

Bloom Filters: Example

Bloom filter t of length $\boldsymbol{m} = 5$ that uses $\boldsymbol{k} = 3$ hash functions

```
function CONTAINS($x$)
    return $t_1[h_1(x)] == 1 \wedge t_2[h_2(x)] == 1 \wedge \cdots \wedge t_k[h_k(x)] == 1$
```

True True True

contains("thisisavirus.com")

h_1("thisisavirus.com") → 2

h_2("thisisavirus.com") → 1

h_3("thisisavirus.com") → 4

Index →	0	1	2	3	4
t_1	0	0	1	0	0
t_2	0	1	0	0	0
t_3	0	0	0	0	1

Bloom Filters: Example

Bloom filter t of length $\boldsymbol{m} = 5$ that uses $\boldsymbol{k} = 3$ hash functions

```
**function** CONTAINS($x$)
    **return** $t_1[h_1(x)] == 1 \land t_2[h_2(x)] == 1 \land \cdots \land t_k[h_k(x)] == 1$
```

True True True

contains("thisisavirus.com")

h_1("thisisavirus.com") → 2

h_2("thisisavirus.com") → 1

h_3("thisisavirus.com") → 4

Index →	0	1	2	3	4
t_1	0	0	1	0	0
t_2	0	1	0	0	0
t_3	0	0	0	0	1

Since all conditions satisfied, returns True (correctly)

Bloom Filters: False Positives

Bloom filter t of length $\boldsymbol{m}$ = 5 that uses $\boldsymbol{k}$ = 3 hash functions

add(“totallynotsuspicious.com”)

```
function ADD($x$)
    for $i = 1, \ldots, k$: do
        $t_i[h_i(x)] = 1$
```

Index →	0	1	2	3	4
t_1	0	0	1	0	0
t_2	0	1	0	0	0
t_3	0	0	0	0	1

Bloom Filters: False Positives

Bloom filter t of length $\boldsymbol{m}$ = 5 that uses $\boldsymbol{k}$ = 3 hash functions

```
function ADD($x$)
    for $i = 1, \dots, k$: do
        $t_i[h_i(x)] = 1$
```

add("totallynotsuspicious.com")

h_1("totallynotsuspicious.com") → 1

Index →	0	1	2	3	4
t_1	0	0	1	0	0
t_2	0	1	0	0	0
t_3	0	0	0	0	1

Bloom Filters: False Positives

Bloom filter t of length $\boldsymbol{m}$ = 5 that uses $\boldsymbol{k}$ = 3 hash functions

```
function ADD($x$)
    for $i = 1, \ldots, k$: do
        $t_i[h_i(x)] = 1$
```

add("totallynotsuspicious.com")

h_1("totallynotsuspicious.com") → 1

h_2("totallynotsuspicious.com") → 0

Index →	0	1	2	3	4
t_1	0	1	1	0	0
t_2	0	1	0	0	0
t_3	0	0	0	0	1

Bloom Filters: False Positives

Bloom filter t of length $\boldsymbol{m} = 5$ that uses $\boldsymbol{k} = 3$ hash functions

```
function ADD($x$)
    for $i = 1, \ldots, k$: do
        $t_i[h_i(x)] = 1$
```

add(“totallynotsuspicious.com”)

h_1(“totallynotsuspicious.com”) → 1

h_2(“totallynotsuspicious.com”) → 0

h_3(“totallynotsuspicious.com”) → 4

Index →	0	1	2	3	4
t_1	0	1	1	0	0
t_2	1	1	0	0	0
t_3	0	0	0	0	1

Bloom Filters: False Positives

Bloom filter t of length $\boldsymbol{m}$ = 5 that uses $\boldsymbol{k}$ = 3 hash functions

```
function ADD($x$)
    for $i = 1, \ldots, k$: do
        $t_i[h_i(x)] = 1$
```

add("totallynotsuspicious.com")

h_1("totallynotsuspicious.com") → 1

h_2("totallynotsuspicious.com") → 0

h_3("totallynotsuspicious.com") → 4

Index →	0	1	2	3	4
t_1	0	1	1	0	0
t_2	1	1	0	0	0
t_3	0	0	0	0	1

Bloom Filters: False Positives

Bloom filter t of length $\boldsymbol{m} = 5$ that uses $\boldsymbol{k} = 3$ hash functions

```
function CONTAINS($x$)
    return $t_1[h_1(x)] == 1 \wedge t_2[h_2(x)] == 1 \wedge \cdots \wedge t_k[h_k(x)] == 1$
```

contains("verynormalsite.com")

Index →	0	1	2	3	4
t_1	0	1	1	0	0
t_2	1	1	0	0	0
t_3	0	0	0	0	1

Bloom Filters: False Positives

Bloom filter t of length $\boldsymbol{m}$ = 5 that uses $\boldsymbol{k}$ = 3 hash functions

```
function CONTAINS($x$)
    return $t_1[h_1(x)] == 1 \wedge t_2[h_2(x)] == 1 \wedge \cdots \wedge t_k[h_k(x)] == 1$
```

True

contains(“verynormalsite.com”)

h_1(“verynormalsite.com”) → 2

Index →	0	1	2	3	4
t_1	0	1	1	0	0
t_2	1	1	0	0	0
t_3	0	0	0	0	1

Bloom Filters: False Positives

Bloom filter t of length $\boldsymbol{m}$ = 5 that uses $\boldsymbol{k}$ = 3 hash functions

```
function CONTAINS($x$)
    return $t_1[h_1(x)] == 1 \wedge t_2[h_2(x)] == 1 \wedge \cdots \wedge t_k[h_k(x)] == 1$
```

True True

contains("verynormalsite.com")

h_1("verynormalsite.com") → 2

h_2("verynormalsite.com") → 0

Index →	0	1	2	3	4
t_1	0	1	1	0	0
t_2	1	1	0	0	0
t_3	0	0	0	0	1

Bloom Filters: False Positives

Bloom filter t of length $\boldsymbol{m}$ = 5 that uses $\boldsymbol{k}$ = 3 hash functions

```
function CONTAINS($x$)
    return $t_1[h_1(x)] == 1 \wedge t_2[h_2(x)] == 1 \wedge \cdots \wedge t_k[h_k(x)] == 1$
```

True True True

contains("verynormalsite.com")

h_1("verynormalsite.com") → 2

h_2("verynormalsite.com") → 0

h_3("verynormalsite.com") → 4

Index →	0	1	2	3	4
t_1	0	1	1	0	0
t_2	1	1	0	0	0
t_3	0	0	0	0	1

Bloom Filters: False Positives

Bloom filter t of length $\boldsymbol{m}$ = 5 that uses $\boldsymbol{k}$ = 3 hash functions

```
**function** CONTAINS($x$)
    **return** $t_1[h_1(x)] == 1 \land t_2[h_2(x)] == 1 \land \cdots \land t_k[h_k(x)] == 1$
```

True True True

contains("verynormalsite.com")

h_1("verynormalsite.com") → 2

h_2("verynormalsite.com") → 0

h_3("verynormalsite.com") → 4

Index →	0	1	2	3	4
t_1	0	1	1	0	0
t_2	1	1	0	0	0
t_3	0	0	0	0	1

Since all conditions satisfied, returns True (incorrectly)

Analysis: False positive probability

Question: For an element $x \in U$, what is the probability that `contains`(x) returns true if `add`(x) was never executed before?

Probability over what?! Over the choice of the $\boldsymbol{h}_1, \ldots, \boldsymbol{h}_k$

Assumptions for the analysis (somewhat stronger than for ordinary hashing):

- Each $\mathbf{h}_i(x)$ is uniformly distributed in $[m]$ for all x and i
- Hash function outputs for each $\mathbf{h}_i$ are mutually independent (not just in pairs)
- Different hash functions are independent of each other

False positive probability – Events

Assume we perform **`add`**$(x_1), \ldots,$**`add`**(x_n)
+ **`contains`**(x) for $x \notin \{x_1, \ldots, x_n\}$

Event E_i holds iff $\mathbf{h}_i(x) \in \{\mathbf{h}_i(x_1), \ldots, \mathbf{h}_i(x_n)\}$

$$P(\text{false positive}) = P(E_1 \cap E_2 \cap \cdots \cap E_k) = \prod_{i=1}^{k} P(E_i)$$

$\mathbf{h}_1, \ldots, \mathbf{h}_k$ independent

False positive probability – Events

Event E_i holds iff $\mathbf{h}_i(x) \in \{\mathbf{h}_i(x_1), \dots, \mathbf{h}_i(x_n)\}$

Event E_i^c holds iff $\mathbf{h}_i(x) \neq \mathbf{h}_i(x_1)$ and … and $\mathbf{h}_i(x) \neq \mathbf{h}_i(x_n)$

$$P(E_i^c) = \sum_{z=1}^{m} P(\mathbf{h}_i(x) = z) \cdot P(E_i^c \mid \mathbf{h}_i(x) = \mathrm{z})$$

LTP

False positive probability – Events

Event E_i^c holds iff $\mathbf{h}_i(x) \neq \mathbf{h}_i(x_1)$ and … and $\mathbf{h}_i(x) \neq \mathbf{h}_i(x_n)$

$$P(E_i^c \mid \mathbf{h}_i(x) = z) = \ P(\mathbf{h}_i(x_1) \neq z, \ldots, \mathbf{h}_i(x_n) \neq z \mid \mathbf{h}_i(x) = z)$$

Independence of values of $\boldsymbol{h}_i$ on different inputs

$$= \ P(\mathbf{h}_i(x_1) \neq z, \ldots, \mathbf{h}_i(x_n) \neq z\ )$$

$$= \prod_{j=1}^{n} P(\mathbf{h}_i(x_j) \neq z)$$

Outputs of $\boldsymbol{h}_i$ uniformly spread

$$= \prod_{j=1}^{n} \left(1 - \frac{1}{m}\right) = \left(1 - \frac{1}{m}\right)^n$$

$$\Rightarrow P(E_i^c) = \sum_{z=1}^{m} P(\mathbf{h}_i(x) = z) \cdot P(E_i^c \mid \mathbf{h}_i(x) = \mathrm{z}) = \left(1 - \frac{1}{m}\right)^n$$

False positive probability – Events

Event E_i holds iff $\mathbf{h}_i(x) \in \{\mathbf{h}_i(x_1), \dots, \mathbf{h}_i(x_n)\}$

Event E_i^c holds iff $\mathbf{h}_i(x) \neq \mathbf{h}_i(x_1)$ and … and $\mathbf{h}_i(x) \neq \mathbf{h}_i(x_n)$

$$P(E_i^c) = \left(1 - \frac{1}{m}\right)^n$$

$$\Rightarrow \qquad \mathrm{FPR} = \prod_{i=1}^{k} \left(1 - P(E_i^c)\right) = \left(1 - \left(1 - \frac{1}{m}\right)^n\right)^k$$

False Positivity Rate – Example

$$\mathrm{FPR} = \left(1 - \left(1 - \frac{1}{m}\right)^{n}\right)^{k}$$

e.g., $n = 5{,}000{,}000$

$k = 30$

$m = 2{,}500{,}000$

→ FPR = 1.28%

Comparison with Hash Tables - Space

- Google storing 5 million URLs, each URL 40 bytes.
- Bloom filter with $k = 30$ and $m = 2{,}500{,}000$

Hash Table

(optimistic)

$5{,}000{,}000 \times 40B = 200\text{MB}$

Bloom Filter

$2{,}500{,}000 \times 30 = 75{,}000{,}000$ bits

< 10 MB

Time

- Say avg user visits 102,000 URLs in a year, of which 2,000 are malicious.
- 0.5 seconds to do lookup in the database, 1ms for lookup in Bloom filter.
- Suppose the false positive rate is 3%

0.5 seconds DB lookup

false positives

$$1\text{ms} + \frac{100000 \times 0.03 \times 500\text{ms} + 2000 \times 500\text{ ms}}{102000} \approx 25.51\text{ms}$$

total URLs

malicious URLs

Bloom filter lookup

Bloom Filters typical of… .

… randomized algorithms and randomized data structures.

- **Simple**
- **Fast**
- **Efficient**
- **Elegant**
- **Useful!**

Brain Break

Motivation for “Named” Random Variables

Random Variables that show up all over the place.

- Easily solve a problem by recognizing it’s a special case of one of these random variables.

Each RV introduced today will show:

- A general situation it models
- Its name and parameters
- Its PMF, Expectation, and Variance
- Example scenarios you can use it

Welcome to the Zoo! (Preview) 🦍 🐘🦁🐯🦓 🐪🦒

$X \sim \text{Unif}(a, b)$

$$P(X = k) = \frac{1}{b - a + 1}$$

$$\mathbb{E}[X] = \frac{a + b}{2}$$

$$\text{Var}(X) = \frac{(b - a)(b - a + 2)}{12}$$

$X \sim \text{Ber}(p)$

$$P(X = 1) = p, P(X = 0) = 1 - p$$

$$\mathbb{E}[X] = p$$

$$\text{Var}(X) = p(1 - p)$$

$X \sim \text{Bin}(n, p)$

$$P(X = k) = \binom{n}{k} p^k (1 - p)^{n-k}$$

$$\mathbb{E}[X] = np$$

$$\text{Var}(X) = np(1 - p)$$

$X \sim \text{Geo}(p)$

$$P(X = k) = (1 - p)^{k-1} p$$

$$\mathbb{E}[X] = \frac{1}{p}$$

$$\text{Var}(X) = \frac{1 - p}{p^2}$$

$X \sim \text{NegBin}(r, p)$

$$P(X = k) = \binom{k - 1}{r - 1} p^r (1 - p)^{k-r}$$

$$\mathbb{E}[X] = \frac{r}{p}$$

$$\text{Var}(X) = \frac{r(1 - p)}{p^2}$$

$X \sim \text{HypGeo}(N, K, n)$

$$P(X = k) = \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}}$$

$$\mathbb{E}[X] = n\frac{K}{N}$$

$$\text{Var}(X) = n\frac{K(N - K)(N - n)}{N^2(N - 1)}$$

Agenda

- Bloom Filters Example & Analysis
- Zoo of Discrete RVs, Part I
  - Uniform Random Variables ◀
  - Bernoulli Random Variables
  - Binomial Random Variables
  - Applications

Discrete Uniform Random Variables

A discrete random variable X **equally likely** to take any (integer) value between integers a and b (inclusive), is uniform.

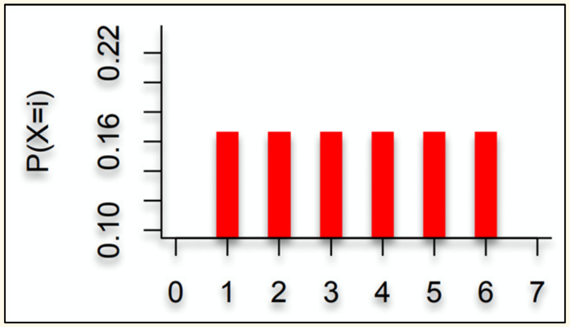
Notation:

PMF:

Expectation:

Variance:

Example: value shown on one roll of a fair die



Discrete Uniform Random Variables

A discrete random variable X **equally likely** to take any (integer) value between integers a and b (inclusive), is uniform.

Notation: $X \sim \mathrm{Unif}(a,b)$

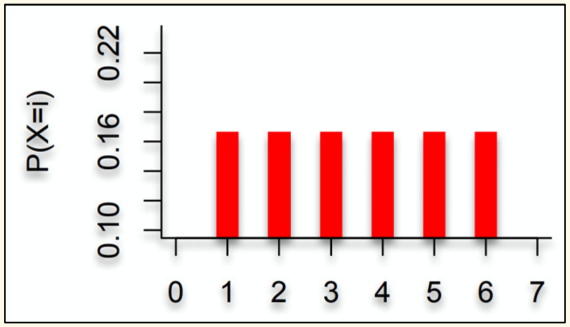
PMF: $\mathrm{P}(X = i) = \frac{1}{b-a+1}$

Expectation: $\mathbb{E}[X] = \frac{a+b}{2}$

Variance: $\mathrm{Var}(X) = \frac{(b-a)(b-a+2)}{12}$

Example: value shown on one roll of a fair die is $\mathrm{Unif}(1,6)$:

- $P(X = i) = 1/6$
- $\mathbb{E}[X] = 7/2$
- $\mathrm{Var}(X) = 35/12$



Agenda

- Bloom Filters Example & Analysis
- Zoo of Discrete RVs, Part I
  - Uniform Random Variables
  - Bernoulli Random Variables ◀
  - Binomial Random Variables
  - Applications

Bernoulli Random Variables

A random variable X that takes value 1 (“Success”) with probability p, and 0 (“Failure”) otherwise. X is called a Bernoulli random variable.

Notation: $X \sim \mathrm{Ber}(p)$

PMF: $P(X = 1) = p,\ P(X = 0) = 1 - p$

Expectation:

Variance:

Poll:

pollev.com/paulbeame028

	Mean	Variance
A.	p	p
B.	p	$1 - p$
C.	p	$p(1 - p)$
D.	p	p^2

Bernoulli Random Variables

A random variable X that takes value 1 ("Success") with probability p, and 0 ("Failure") otherwise. X is called a Bernoulli random variable.

Notation: $X \sim \mathrm{Ber}(p)$

PMF: $P(X = 1) = p, \; P(X = 0) = 1 - p$

Expectation: $\mathbb{E}[X] = p$ Note: $\mathbb{E}[X^2] = p$

Variance: $\mathrm{Var}(X) = \mathbb{E}[X^2] - \mathbb{E}[X]^2 = p - p^2 = p(1-p)$

Examples:
- Coin flip
- Randomly guessing on a MC test question
- A server in a cluster fails
- Any indicator RV

Agenda

- Bloom Filters Example & Analysis
- Zoo of Discrete RVs, Part I
  - Uniform Random Variables
  - Bernoulli Random Variables
  - Binomial Random Variables ◀
  - Applications

Binomial Random Variables

A discrete random variable X that is the number of successes in n independent random variables $Y_i \sim \mathrm{Ber}(p)$.
X is a Binomial random variable where $X = \sum_{i=1}^{n} Y_i$

Examples:

- # of heads in n coin flips
- # of 1s in a randomly generated n bit string
- # of servers that fail in a cluster of n computers
- # of bit errors in file written to disk
- # of elements in a bucket of a large hash table

Poll:
pollev.com/paulbeame028
$P(X = k)$
A. $p^k(1-p)^{n-k}$
B. np
C. $\binom{n}{k} p^k (1-p)^{n-k}$
D. $\binom{n}{n-k} p^k (1-p)^{n-k}$

Binomial Random Variables

A discrete random variable X that is the number of successes in n independent random variables $Y_i \sim \mathrm{Ber}(p)$.
X is a Binomial random variable where $X = \sum_{i=1}^{n} Y_i$

Notation: $X \sim \mathrm{Bin}(n, p)$

PMF: $P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$

Expectation:

Variance:

Poll:
pollev.com/paulbeame028

	Mean	Variance
A.	p	p
B.	np	$np(1-p)$
C.	np	np^2
D.	np	n^2p

Binomial Random Variables

A discrete random variable X that is the number of successes in n independent random variables $Y_i \sim \mathrm{Ber}(p)$.
X is a Binomial random variable where $X = \sum_{i=1}^{n} Y_i$

Notation: $X \sim \mathrm{Bin}(n, p)$

PMF: $P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$

Expectation: $\mathbb{E}[X] = np$

Variance: $\mathrm{Var}(X) = np(1-p)$

Mean, Variance of the Binomial

"i.i.d." is a commonly used phrase.
It means "independent & identically distributed"

If $Y_1, Y_2, \dots, Y_n \sim \mathrm{Ber}(p)$ and independent (i.i.d.), then
$X = \sum_{i=1}^{n} Y_i, \quad X \sim \mathrm{Bin}(n, p)$

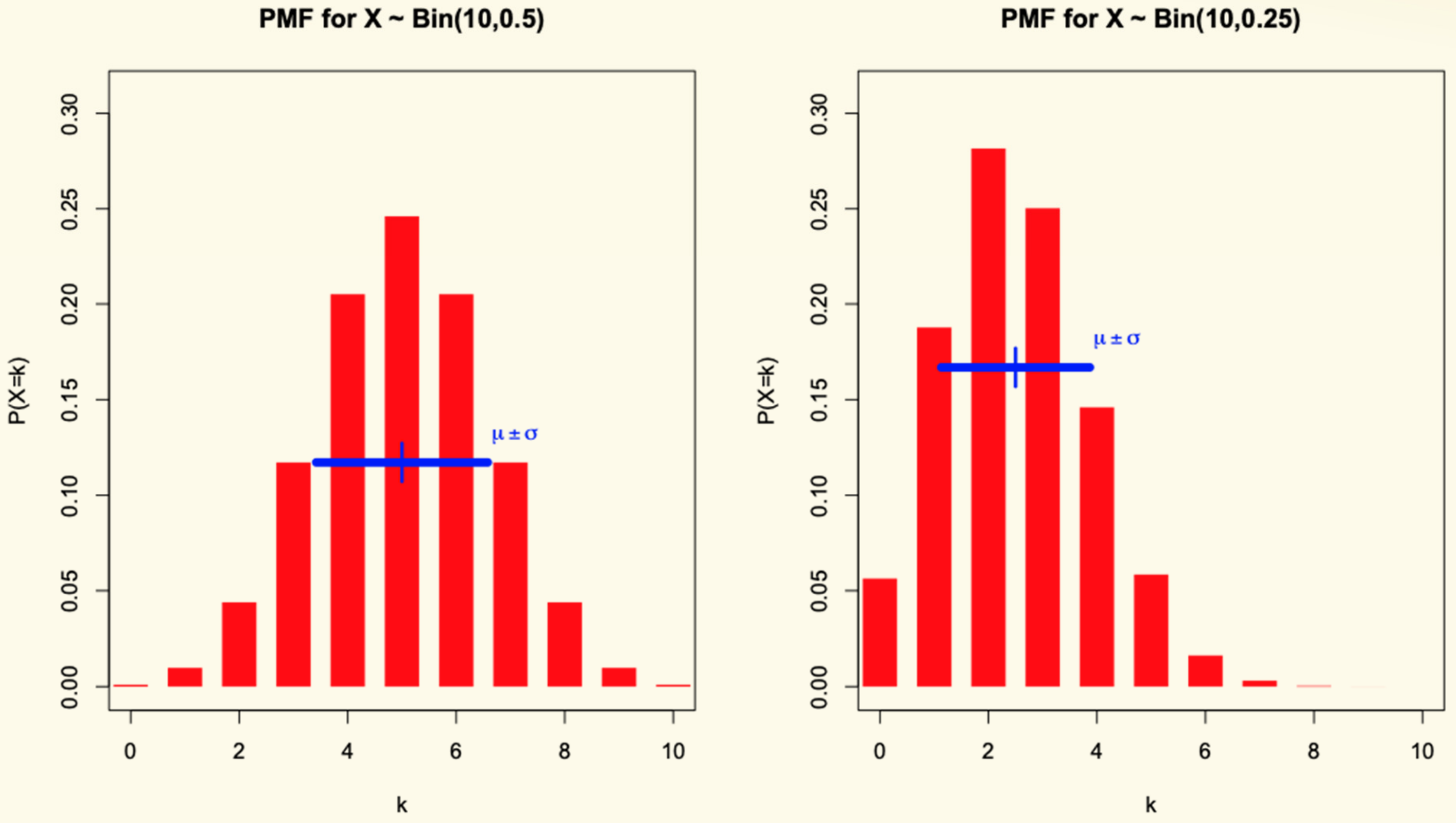
Claim $\mathbb{E}[X] = np$

$$\mathbb{E}[X] = \mathbb{E}\left[\sum_{i=1}^{n} Y_i\right] = \sum_{i=1}^{n} \mathbb{E}[Y_i] = n\mathbb{E}[Y_1] = np$$

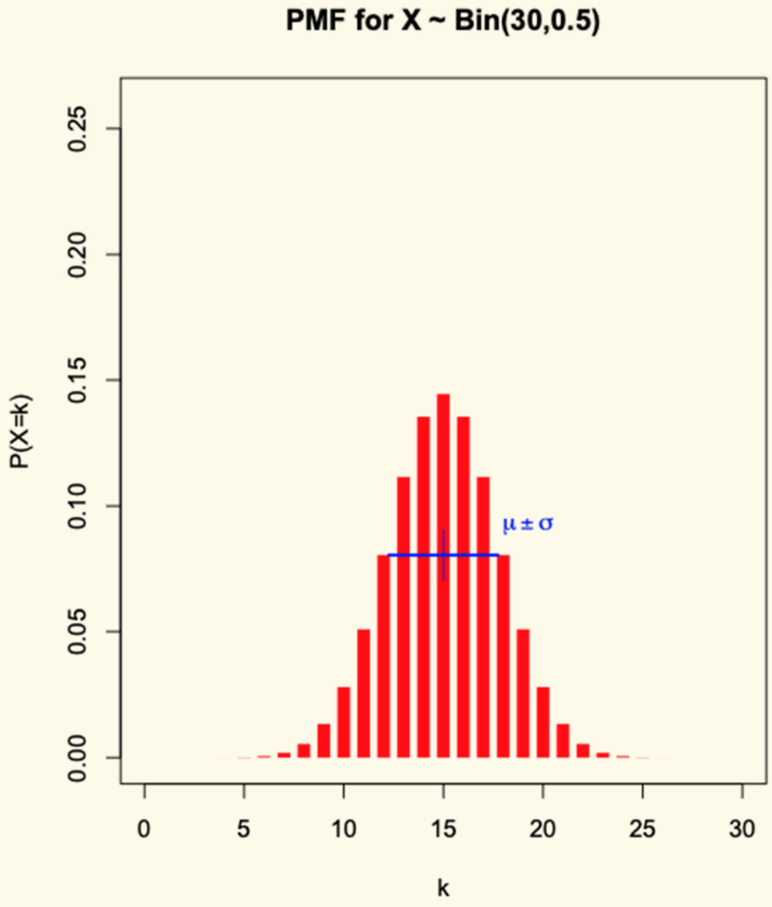
Claim $\mathrm{Var}(X) = np(1-p)$

$$\mathrm{Var}(X) = \mathrm{Var}\left(\sum_{i=1}^{n} Y_i\right) = \sum_{i=1}^{n} \mathrm{Var}(Y_i) = n\mathrm{Var}(Y_1) = np(1-p)$$

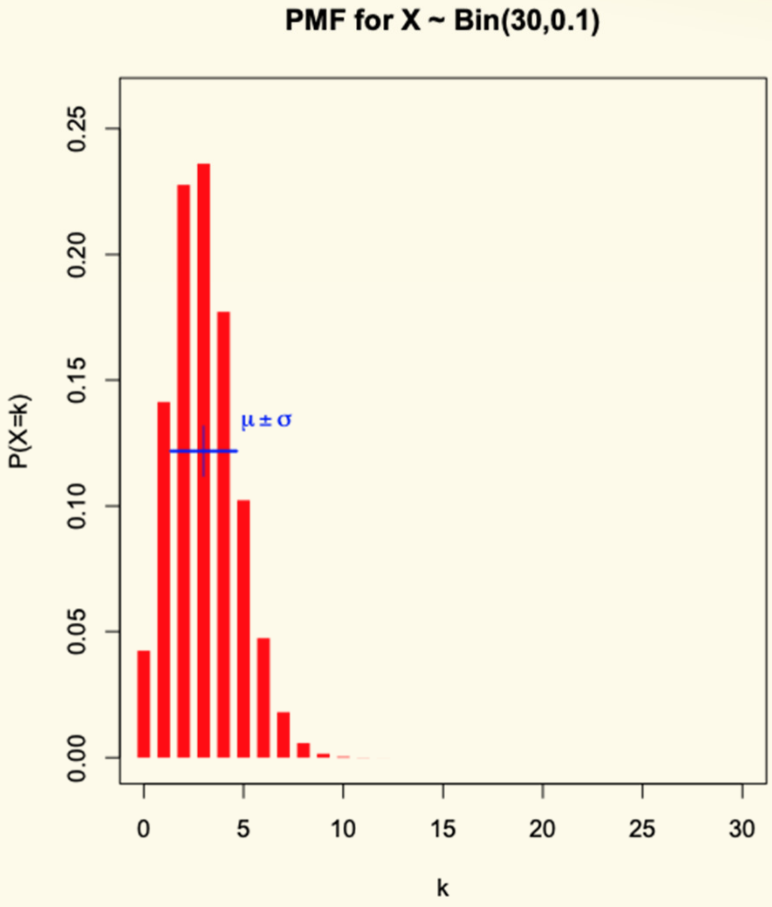
Binomial PMFs



Binomial PMFs



PMF for X ~ Bin(30,0.5)



PMF for X ~ Bin(30,0.1)

Example

Sending a binary message of length 1024 bits over a network with probability 0.999 of correctly sending each bit in the message without corruption (independent of other bits).

Let X be the number of corrupted bits.

What is $\mathbb{E}[X]$?

Poll:

pollev.com/paulbeame028

a. 1022.99
b. 1.024
c. 1.02298
d. 1
e. Not enough information to compute