Expectation and Hashing

COMPSCI 230 — Discrete Math

April 14, 2015
Outline

1. Expectation
2. Repeated Bernoulli Trials
3. Associative Storage and Retrieval
Coin Flip

• Bet on $H$ outcome, $p = P(H) = 0.6$, win $w = $10 if head, $l = -$10 if tail
• Experiment outcome $C$ with sample space $S = \{H, T\}$
• Random variable

$$W : \{H, T\} \rightarrow \mathbb{R}$$

defined as

$$W(C) = \begin{cases} w & \text{for } C = H \\ l & \text{for } C = T \end{cases}$$

with probability distribution

$$p = P(W = w) = 0.6$$
$$q = P(W = l) = 0.4$$

where $p + q = 1$
Expectation of a Random Variable

• What amount \( a \) do I win on average if \( p = P(H) = 0.6 \)?
• Approximate frequentist interpretation:
  Play the game 1000 times, win close to \( $1000a \)
  [However, we play once]
• The **expected win** is defined as

\[
a = \mathbb{E}[W] = \mathbb{E}[W(C)] = pW(H) + qW(T)
= pw + (1 - p) \ell = 0.6 \cdot 10 + 0.4 \cdot (-10) = $2
\]

• Play the game 1000 times, win close to \( 1000 \mathbb{E}[W] = $2000 \)
• More generally the **expected value** (or **expectation**) of random variable \( X = X(O) \) is

\[
E[X] = E[X(O)] = \sum_{O \in \mathcal{S}} P(O)X(O)
\]
Expectation is Linear

- \[ \mathbb{E}[a U + b V] = a \mathbb{E}[U] + b \mathbb{E}[V] \] because

\[
\mathbb{E}[a U + b V] = \sum_{O \in S} P(O) [a U(O) + b V(O)] \\
= a \sum_{O \in S} P(O) U(O) + b \sum_{O \in S} P(O) V(O) \\
= a \mathbb{E}[U] + b \mathbb{E}[V]
\]

- Example: Upton wins or loses 8 pounds sterling. Valerie wins or loses 10 dollars
- \[ \mathbb{E}[U] = 0.6 \cdot 8 - 0.4 \cdot 8 = 1.6 \text{ pounds} \]
- \[ \mathbb{E}[V] = 0.6 \cdot 10 - 0.4 \cdot 10 = 2 \text{ dollars} \]
- Exchange rate: 1.49 dollars per pound sterling
- Combined win: \[ 1.49 \mathbb{E}[U] + \mathbb{E}[V] = 1.49 \cdot 1.6 + 2 = \$4.384 \]
Sequences of Bernoulli Trials

• Coin flip is an example of a Bernoulli trial: two outcomes, fixed probability $p$ of “success,” repeated trials are independent

• Independence is a property of the unbounded repeated experiment

$$C = (C_1, C_2, \ldots)$$ with sample space $\mathbb{S}^\infty = \mathbb{S} \times \mathbb{S} \times \ldots$

• The infinite sequence $C$ is one outcome of the repeated experiment

• So we can define random variables

$$X : \mathbb{S}^\infty \to \mathbb{R}$$

• Example:

$$N(C) = n \text{ iff the first } H \text{ is in trial } n \in \mathbb{N}$$
Expected Number of Trials to First Success

- $N(C) = n$ iff the first $H$ is in trial $n \in \mathbb{N}$
- $N = n$ iff $n - 1$ $T$s are followed by one $H$
- Example: $n = 4$. $(T, T, T, H, ...)$
- Because of independence, $P(N = n) = q^{n-1}p$
- $\mathbb{E}[N]$: expected number of trials until the first success
- How many times do you need to flip a coin on average until it comes up $H$? [2 clickers]
A Summation Lemma

• For $x \neq 1$

\[
\sum_{n=0}^{\infty} nx^n = \frac{x}{(1 - x)^2}
\]

• Proof: We know that for $x \neq 1$

\[
\sum_{n=0}^{\infty} x^n = \frac{1}{1 - x}
\]

\[
\sum_{n=0}^{\infty} nx^n = x \sum_{n=0}^{\infty} nx^{n-1} = x \sum_{n=0}^{\infty} \frac{d}{dx} x^n = x \frac{d}{dx} \sum_{n=0}^{\infty} x^n
\]

\[
= x \frac{d}{dx} \frac{1}{1 - x} = x \frac{1}{(1 - x)^2} = \frac{x}{(1 - x)^2}
\]

\[\square\]
Expected Number of Trials to First Success

- The expected number of trials to first success in a Bernoulli experiment with success probability $p$ is

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

- Proof:

$$\mathbb{E}[N] = \sum_{C \in S^\infty} N(C) P(C) = \sum_{n=1}^{\infty} n P(N = n)$$

$$= \sum_{n=1}^{\infty} n q^{n-1} p = \sum_{n=0}^{\infty} n q^{n-1} p = \frac{p}{q} \sum_{n=0}^{\infty} n q^n$$

(from lemma)$$= \frac{p}{q} \frac{q}{(1 - q)^2} = \frac{p}{q} \frac{q}{p^2} = \frac{1}{p} \square$$
Associative Storage and Retrieval

• Store and lookup data items by key. Examples:
  • Purchase orders by customer
  • Student record by last name
  • Catalog by model number

• Associate a **key** to each **value**. In Racket:

```racket
> (define grade (make-hash))
> (hash-set! grade "Smith" '(87 80 93)) ;! signals a **side effect**
> (hash-ref grade "Adams")
'(82 80 78)
> (hash-ref grade "Jones")
**hash-ref**: no value found for key
  **key**: "Jones"
```
Implementation 1

- **Insert:** \((\text{key . value})\) pairs cons-ed to a list as they come in.
- **Lookup:** Search the list sequentially until the key is found (return value) or the end of the list is reached (raise an error).

```scheme
(define (make-hash-1) null)

(define-syntax-rule (hash-set-1! table key value)
  (set! table (cons (cons key value) table)))

(define (hash-ref-1 table key)
  (cond ((empty? table)
         (error 'hash-ref-1
               "no value found for key\n\ntkey: ~a" key))
        ((equal? key (car (first table)))
         (cdr (first table)))
        (else (hash-ref-1 (rest table) key))))
```
Implementation 1 $\rightarrow$ Implementation 2

- Functionality issue: if you insert two values with the same key, you only get the last one back
- Desired behavior? Overwrite? Raise error? Ignore second value? Store and return both?
- Efficiency:
  - Insertion is straightforward and efficient
  - Lookup takes on average $n/2$ steps if there are $n$ items in the table
  - Lookup bogs down when $n$ is large
- Improvement 1: If key already exists, overwrite
- Improvement 2: Insert $(\text{key . value})$ pairs so that keys are sorted