CSCI 2011: Discrete Probability

Chris Kauffman

Last Updated:
Thu Jul 19 14:15:06 CDT 2018

Logistics

Reading: Rosen

- Now: 6.1 - 6.5, 8.5
- Now: 7.1 - 7.4
- Next: 8.1 - 8.3

Assignments

- A06: due today
- A07: up Thu

Quiz Thursday

- Strong/Structural Induction Proofs
- Basic Counting
- Permutations/Combinations

Basic Probability with Sets

- There are S possible outcomes in an experiment
- Interested in $E \subset S$ of these, called an **Event**
- Probability of the even occurring: $P(E) = |E|/|S|$

Examples

- Roll a 6-sided die, want a 3 or more
  - $S = \{1, 2, 3, 4, 5, 6\}$
  - $E = \{3, 4, 5, 6\}$
  - $P(E) = |E|/|S| = 4/6 = 0.666\overline{6}$
- Flip coin 3 times, interested in odd number of Heads
  - $S = \{HHH, HHT, HTH, HTT, THH, THT, TTH, TTT\}$
  - $E = \{HHH, HTT, THT, TTH\}$
  - $P(E) = |E|/|S| = 4/8 = 0.5$

Probability with Combinations and Permutations

- Combinations/Permutations are useful in calculating probabilities
- Must calculate possibilities available when distinct outcomes possible

Examples

- A **lottery** has participants pick 6 numbers from 1 to 40; if all 6 match participant wins big money
- Possibilities: $C(40, 6) = 3,838,380 = |S|$
- 1 winning combination so $E = |1|$
- Chance of me winning big: $P(E) = 1/3,838,380 \approx 2.6 \times 10^{-8}$
- Picking 5 numbers correctly gives small prize, winning choices are then $C(6, 6) + C(6, 5) = 1 + 6 = |E|$
- Not much better odds for $P(E)$

Unions, Intersections, Complements in Probability

- Set operations have play in probability

Complement: $P(\overline{E}) = 1 - P(E)$

- Pick-6 Lottery winning chance is $P(E) = \frac{1}{3{,}838{,}380}$
- Pick-6 Lottery losing chance is

$$P(\overline{E}) = \frac{3,838,399}{3,838,380} = 1 - P(E)$$

Union: $P(A \cup B) = P(A) + P(B) - P(A \cap B)$

- Pick random number between 1 and 1000, $|S| = 1000$
- $E = \{x | x \text{ evenly divisible by 2 or 5}\}$
- $|E| = \lfloor 1000/2 \rfloor + \lfloor 1000/5 \rfloor - \lfloor 1000/(2 \cdot 5) \rfloor =$ $500 + 200 - 100 = 600$
- $P(E) = 600/1000 = 0.6$

Discrete Probability Distributions

- A **discrete probability distribution** assigns a probability to each of a finite set of objects
- May not be uniform, often specified as an array

x_i	Rain	Sun	Snow	Hail
$P(x_i)$	0.25	0.50	0.10	0.15

- Individual elements are **not events** but comprise individual possibilities
- Sum of outcome probabilities must be 1 for a proper distribution

$$\sum_{i=1}^{n} P(x_i) = 1$$

- **Continuous probabilities distributions** differ
  - Probability of a continuous objects often on the real line
  - Integrals not summations, requires calculus

Exercise: Generalized Distribution Properties

x_i	Rain	Sun	Snow	Hail
$P(x_i)$	0.25	0.50	0.10	0.15

- An event is some combination of these such as
  - $E_1 = \{Snow, Hail\}$
  - $E_2 = \{Rain, Snow, Hail\}$
  - $E_3 = \{Rain, Sun\}$
- For events, have following general identities
  - $P(E) = \sum_{x_i \in E} P(x_i)$
  - $P(A \cup B) = P(A) + P(B) - P(A \cap B)$
  - $P(\overline{E}) = \sum_{x_i \notin E} P(x_i) = 1 - P(E)$

Calculate Probabilities

$$
\begin{array}{ll}
E_1 = \{Snow, Hail\} & P(E_1) = ??? \\
E_2 = \{Rain, Snow, Hail\} & P(E_2) = ??? \\
E_1 \cup E_2 & P(E_1 \cup E_2) = ??? \\
 & \\
\overline{E_1 \cup E_2} & P(\overline{E_1 \cup E_2}) = ???
\end{array}
$$

Answers: Generalized Distribution Properties

x_i	Rain	Sun	Snow	Hail
$P(x_i)$	0.25	0.50	0.10	0.15

- An event is some combination of these such as
  - $E_1 = \{Snow, Hail\}$
  - $E_2 = \{Rain, Snow, Hail\}$
  - $E_3 = \{Rain, Sun\}$
- For events, have following general identities
  - $P(E) = \sum_{x_i \in E} P(x_i)$
  - $P(A \cup B) = P(A) + P(B) - P(A \cap B)$
  - $P(\overline{E}) = \sum_{x_i \notin E} P(x_i) = 1 - P(E)$

Calculate Probabilities

$$
\begin{array}{ll}
E_1 = \{Snow, Hail\} & P(E_1) = 0.25 \\
E_2 = \{Rain, Snow, Hail\} & P(E_2) = 0.50 \\
E_1 \cup E_2 & P(E_1 \cup E_2) = 0.50 \\
 & \\
\overline{E_1 \cup E_2} & P(\overline{E_1 \cup E_2}) = 0.50
\end{array}
$$

Bernoulli Trials and the Binomial Distribution

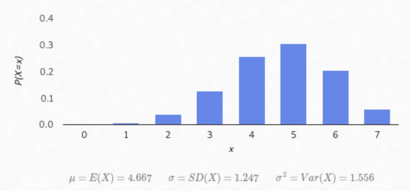
- Experiments in which only two outcomes are possible are called **Bernoulli Trials**
  - Coin flips, random bits, pick a ball when all are red or black
- Repeatedly running the experiment results in "trials" e.g. flip a coin 7 times
- Follows a regular pattern related to Binomial Coefficients
  - p: probability of outcome A
  - q: probability of outcome B $= 1 - p$
  - n: number of trials
  - k: number of times A occurs in n trials
  - E: event that A occurs k times in n trials
  - $P(E) = C(n, k)p^k q^{n-k} = \binom{n}{k} p^k q^{n-k}$
- Referred to as the **Binomial Distribution**

Example of the Binomial distribution

A 6-sided die has 4 Red sides and 2 Black sides

- $P(Red) = 2/3 = p$
- $P(Black) = 1/3 = q.$

Roll 7 times, probability of rolling



Red	Black	$P(E)$	
0	7	$C(7,0) \cdot (\frac{2}{3})^0 \cdot (\frac{1}{3})^7$	$= 0.00046$
1	6	$C(7,1) \cdot (\frac{2}{3})^1 \cdot (\frac{1}{3})^6$	$= 0.00640$
2	5	$C(7,2) \cdot (\frac{2}{3})^2 \cdot (\frac{1}{3})^5$	$= 0.03841$
3	4	$C(7,3) \cdot (\frac{2}{3})^3 \cdot (\frac{1}{3})^4$	$= 0.12803$
4	3	$C(7,4) \cdot (\frac{2}{3})^4 \cdot (\frac{1}{3})^3$	$= 0.25606$
5	2	$C(7,5) \cdot (\frac{2}{3})^5 \cdot (\frac{1}{3})^2$	$= 0.30727$
6	1	$C(7,6) \cdot (\frac{2}{3})^6 \cdot (\frac{1}{3})^1$	$= 0.20485$
7	0	$C(7,7) \cdot (\frac{2}{3})^7 \cdot (\frac{1}{3})^0$	$= 0.05853$

Chip Testing

- Real world problems in manufacturing: defects in products
- CPU makers like Intel produce "chips" in batches but chemistry may go wrong producing a "bad batch"
  - 10,000 chips in a batch
  - Good batch is ALL good
  - Bad batch has 10% of chips bad
- Determine if a batch is Good or Bad
- Could check all chips **but** too time/$$$ intensive: $O(N)$ tests to do
- Suggest an alternative algorithm that is more efficient

Monte Carlo Solution

- **Sample** the chips until the chance of is very low that that the batch is bad
- In a Bad batch testing a 1 good chip is $P(Good = 1) = 0.90$
- Since the batches are large, chance of testing n good chips is about $P(Good = n) \approx 0.90^n$
- Choose $n = 66$ gives $P(Good = 66) = 0.90^{66} \approx 0.001$
- During testing, if any chip is bad, know that the batch is bad
- If all 66 are test Good, there only a 1% chance that the batch is bad

Gambling with Algorithms

- City in Manaco most famous for its casinos/gambling
- Algorithms that employ Sampling approaches can be more efficient at the cost of potential errors
- Many numerical approximation algorithms employ this technique such as approximating π

"Random Variables"

- Often experiments have numerical outcomes
  - 6-sided die with 1-6 on it, outcome is the value of die
  - Flip an unfair coin 10 times, outcome is number of heads
- Value of the outcome is referred to as a **Random Variable**
- Often use the notation
  - X for random variable (RV)
  - x for a specific value the RV may take on
  - $P(X = x)$ for probability that X takes on value x

Example using Binomial Distribution

A 6-sided die has 4 Red sides and 2 Black sides

- $P(Red) = 2/3$
- X is number of Reds on a 7 rolls
- $P(X = 5) = C(7, 5) \cdot (\frac{2}{3})^5 \cdot (\frac{1}{3})^2 = 0.30727$
- $P(X = 1) = C(7, 1) \cdot (\frac{2}{3})^1 \cdot (\frac{1}{3})^6 = 0.00640$

Expected Value

Often interested in the **Expected Value** of a random variable

- Notated $E(X)$
- Computed by $E(X) = \sum_{s \in S} P(s) \cdot X(s)$ where
  - S is the set of all outcomes
  - s is an individual outcome
  - $P(s)$ is the probability of outcome s
  - $X(s)$ is the value of X when X occurs
- For discrete probability distributions, usually make use of a table of probabilites/outcomes and multiply/sum
- Most common distributions have proven expected values
  - Binomial Distribution: Expected Number of Successes for n trials with p chance of success is $n \cdot p$

Example of Expected Value

Random variable T represents the temperature during various weather conditions.

Outcome	s_i	Rain	Sun	Snow	Hail
Probability	$P(s_i)$	0.25	0.50	0.10	0.15
Temperature	$T(s_i)$	75	85	30	60

$$
\begin{aligned}
E(T) =& P(Rain) \cdot T(Rain) + P(Sun) \cdot T(Sun) \\
& + P(Snow) \cdot T(Snow) + P(Hail) \cdot T(Hail) \\
=& 0.25 \cdot 75 + 0.50 \cdot 85 \\
& + 0.10 \cdot 30 + 0.15 \cdot 60 \\
=& 73.25
\end{aligned}
$$

Expected Value of Runtime of Algorithms

- Discussed runtime of algorithms with Big-O complexity usually as the **Worst Case** runtime: maximum number of operations
- Could also discuss the **Best Case** runtime: minimum number of operations
- Also of interest is the **Average Case**: expected number of operations
- Now in a position to discuss this for some simple algorithms

Exercise: Linear Search Average Case

- Given annotated code for linear search with number of ops per line
- Calculate expected number of ops in the following cases

Case 1

- 100% chance `key` in `a[]`
- Equal chance anywhere in `a[]`

Case 2

- 50% chance `key` in `a[]`, equal chance at any index
- 50% chance NOT in `a[]`

```
boolean linear_search(int a[],
                      int key)
{                           # OPS
  int n = length(a);        # 2
  int i = 0;                # 1
  while(i<n){               # 1
    if(a[i] == key){        # 2
      return true;          # 0
    }
    i++;                    # 1
  }
  return false;             # 0
}
```

Answers: Linear Search Average Case 1

Case 1

- Ops for pos `i` $=$
  $3 + 4 \cdot i + 3 = 4i + 6$
- Equal probability for each spot gives $P(i) = 1/n$

$$
\begin{aligned}
E(Ops) &= \sum_{i=0}^{n-1} \frac{1}{n} 4i + 6 \\
&= \frac{1}{n}(6n + 4\sum_{i=0}^{n-1} i) \\
&= \frac{1}{n}(6n + 4(\frac{1}{2}(n-1)n)) \\
&= 6 + 2(n-1) \\
&= 2n + 4
\end{aligned}
$$

```
boolean linear_search(int a[],
                      int key)
{                       # OPS
  int n = length(a);    # 2
  int i = 0;            # 1
  while(i<n){           # 1
    if(a[i] == key){    # 2
      return true;      # 0
    }
    i++;                # 1
  }
  return false;         # 0
}
```

Answers: Linear Search Average Case 2

Case 2

- If in list, average Ops is $2n + 4$
- Not in list, max Ops is $3 + 4n + 1 = 4n + 4$
- 50% chance in vs out so

$$
\begin{aligned}
E(Ops) &= \frac{1}{2}(2n+4) + \frac{1}{2}(4n+4) \\
&= n + 2 + 2n + 2 \\
&= 3n + 4
\end{aligned}
$$

```
boolean linear_search(int a[],
                      int key)
{                          # OPS
  int n = length(a);       # 2
  int i = 0;               # 1
  while(i<n){              # 1
    if(a[i] == key){       # 2
      return true;         # 0
    }
    i++;                   # 1
  }
  return false;            # 0
}
```

Average Case Analysis

- Much harder to analyze average case than wost/best case but are often of importance
- Requires assumptions about input probabilities
- Related: **Amortized Analysis** which looks at the total number of ops in doing operations repeatedly
- Allows one to show that $O(1)$ array appends are possible if averaging over many appends
- Amortized Analysis studied in Algorithms and Advanced Algorithms Courses

What we Haven't Touched

- Variance of Random Variables
- Independence and Conditional Probability
- Bayes Rule and Spam Filtering

May come up in HW so do some reading.