Stochastic Processes

Kevin Chang

January 20, 2022

1 1.2

- (a)
  - $Range(F(X)) := [0, 1]$
  - For $x \in [0, 1]$
    - $P[F(X) \leq x] = P[X \leq F^{-1}(x)] = F(F^{-1}(x)) = x$
  - For $x > 1$
    - $P[F(X) \leq x] = P[F(X) \leq 1] = 1$
  - For $x < 0$
    - $P[F(X) \leq x] = P[F(X) < 0] = 0$
- (b)
  - $Range(F(X)) := [0, 1]$
  - $P[F^{-1}(U) \leq x] = P[U \leq F(x)] = F(x)$

2 1.3

- $P[X_n = i] = \binom{n}{i} p_n^k (1 - p_n)^{n-k}$
- $P[X_n = i] = \lim_{n\to\infty} \binom{n}{i} (p_n)^i (1 - p_n)^{n-i}$
  $= \lim_{n\to\infty} \binom{n}{i} (\frac{\lambda}{n})^i (\frac{n-\lambda}{n})^{n-i}$
  $= \frac{\lambda^i}{i!} \lim_{n\to\infty} \frac{n! \times n^i}{(n-i)! n^i (n-\lambda)^i} (\frac{n-\lambda}{n})^n$
  $= \frac{\lambda^i}{i!} (1 - \frac{\lambda}{n})^n = \frac{\lambda^i}{i!} \exp(-\lambda)$

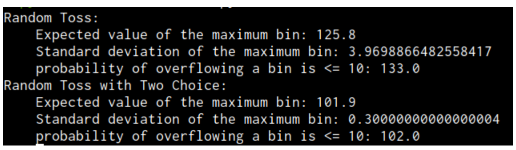
3 1.35

- (a)
  - $\mathbb{E}[h(X)] = \int_{-\infty}^{\infty} h(x) f(x) dx = \int_{-\infty}^{\infty} h(x) e^{-tx} M(t) f_t(x) dx = \mathbb{E}[e^{-tX_t} h(X_t)] M(t)$
- (b)
  - $P[X > a] = \int_a^{\infty} f(x) dx = \int_a^{\infty} e^{-tx} M(t) f_t(x) dx \leq \int_a^{\infty} e^{-ta} M(t) f_t(x) dx$
    $= M(t) e^{-ta} P[X_t > a]$
- (c)
  -

4 1.36

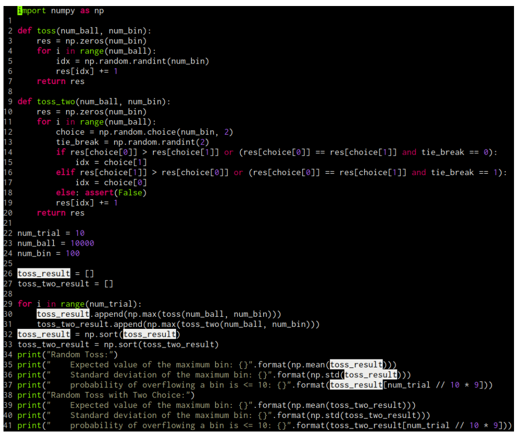
- Jensen's Inequality
  - $f(\sum_i w_i x_i) \leq \sum_i w_i f(x_i)$ with non-negative w_i summing to 1
- choose $f(x) = -\log x$ and $w_i = \frac{1}{n}$
  $\rightarrow -\log(\frac{1}{n}\sum_i x_i) \leq -\frac{1}{n}\sum_i \log x_i$
- inverse the equation and apply exponential on both side
  $\rightarrow \frac{1}{n}\sum_i x_i \geq (\prod_i x_i)^{\frac{1}{n}}$

5 Computer Problem



```
Random Toss:
    Expected value of the maximum bin: 125.8
    Standard deviation of the maximum bin: 3.9698866482558417
    probability of overflowing a bin is <= 10: 133.0
Random Toss with Two Choice:
    Expected value of the maximum bin: 101.9
    Standard deviation of the maximum bin: 0.30000000000000004
    probability of overflowing a bin is <= 10: 102.0
```

(a) Result of the simulation



```python
import numpy as np

def toss(num_ball, num_bin):
    res = np.zeros(num_bin)
    for i in range(num_ball):
        idx = np.random.randint(num_bin)
        res[idx] += 1
    return res

def toss_two(num_ball, num_bin):
    res = np.zeros(num_bin)
    for i in range(num_ball):
        choice = np.random.choice(num_bin, 2)
        tie_break = np.random.randint(2)
        if res[choice[0]] > res[choice[1]] or (res[choice[0]] == res[choice[1]] and tie_break == 0):
            idx = choice[1]
        elif res[choice[1]] > res[choice[0]] or (res[choice[0]] == res[choice[1]] and tie_break == 1):
            idx = choice[0]
        else: assert(False)
        res[idx] += 1
    return res

num_trial = 10
num_ball = 10000
num_bin = 100

toss_result = []
toss_two_result = []

for i in range(num_trial):
    toss_result.append(np.max(toss(num_ball, num_bin)))
    toss_two_result.append(np.max(toss_two(num_ball, num_bin)))
toss_result = np.sort(toss_result)
toss_two_result = np.sort(toss_two_result)
print("Random Toss:")
print("    Expected value of the maximum bin: {}".format(np.mean(toss_result)))
print("    Standard deviation of the maximum bin: {}".format(np.std(toss_result)))
print("    probability of overflowing a bin is <= 10: {}".format(toss_result[num_trial // 10 * 9]))
print("Random Toss with Two Choice:")
print("    Expected value of the maximum bin: {}".format(np.mean(toss_two_result)))
print("    Standard deviation of the maximum bin: {}".format(np.std(toss_two_result)))
print("    probability of overflowing a bin is <= 10: {}".format(toss_two_result[num_trial // 10 * 9]))
```

(b) Code