

# ① Shuffling (= uniformly random permutation)

Given a list of  $n$  elements, randomly permute  
Specifically, want each possible permutation of  
 $[n] = (1, 2, \dots, n)$  to have an equal chance, i.e.,  
 $\frac{1}{n!}$  chance.

Naively: draw  $n$  iid samples from any absolutely  
cts. probability distribution (eg. Gaussian)  
then sort these.

Downside: a sort costs  $O(n \cdot \log n)$  flops.

Can we get a linear time algo.?

Yes! "Knuth-shuffle" aka "Fisher Yates shuffle"

Input  $X = (x_1, x_2, \dots, x_n)$

For  $i = 1, 2, \dots, n-1$

$j \in \{i, i+1, \dots, n\}$  chosen uniformly at random

exchange  $x_i$  and  $x_j$

time to compute this  
is independent of  $i$

Then: this is a uniform random permutation,

i.e.,  $P(\text{any permutation on } [n]) = \frac{1}{n!}$

Proof via induction, aka "loop invariant" in CS terms

def a " $k$ -permutation" on  $[n]$  is a list of  $k$  elements,  
each element from  $[n]$  and no repetitions.

i.e., the 1<sup>st</sup>  $k$  elements of any permutation

There are  $\frac{n!}{(n-k)!}$  of these (  $n!$  total permutations,  
only count those whose  
1<sup>st</sup>  $k$  elements differ )

So... we want a  $n$ -permutation.

claim: after iteration  $K$ , all  $K$ -permutations have an equal chance of equalling  $\underbrace{X(1:K)}_{\text{Matlab notation}}$ , i.e.,  $\frac{(n-K)!}{n!}$  chance, since there are  $n!/(n-K)!$   $K$ -perms.

proof-of-claim:

iter 1: there are  $n$  1-permutations. By construction, all are equally likely.

iter  $K+1$ : (induction step)

Let  $(X_{\sigma_1}, X_{\sigma_2}, \dots, X_{\sigma_K}, X_{\sigma_{K+1}})$  be any  $K+1$  permutation.

$$P(X(1:K+1) = (X_{\sigma_1}, \dots, X_{\sigma_K}, X_{\sigma_{K+1}}))$$

$$= P(X(K+1) = X_{\sigma_{K+1}} \mid X(1:K) = (X_{\sigma_1}, \dots, X_{\sigma_K})) \cdot$$

$$P(X(1:K) = (X_{\sigma_1}, \dots, X_{\sigma_K})) \rightarrow \underbrace{= \frac{(n-K)!}{n!}}_{\text{via induct.}}$$

$X_{\sigma_{K+1}}$  is some entry not already selected

In algo,  $j$  is chosen uniformly at random among  $\{K+1, \dots, n\}$   $n-K$  entries that have not already been selected.

So, this is  $\frac{1}{n-K}$

$$\dots = \frac{1}{n-K} \cdot \frac{(n-K)!}{n!} = \frac{(n-(K+1))!}{n!}$$

==

So, after  $K=n$  iterations,  $\frac{0!}{n!} = \frac{1}{n!}$  chance of any given permutation,

i.e., this is a shuffle.  $\square$

## ② Reservoir Sampling to form a Simple Random Sample (SRS) "randSample(n, k)" in Matlab

i.e. combination  
not  
permutation  
not ordered  
list

Def A "SRS" of  $k$  elements (from  $n$  possible) is  
a subset of size  $k$  from  $[n]$  such that all such  
subsets have equal chance, namely  $1/\binom{n}{k} = \frac{k!(n-k)!}{n!}$

Example: • shuffle the data and take 1st  $k$  entries  
(inefficient)

"overkill"  
since these  
also  
shuffle

- do the 1st  $k$  steps of Knuth / Fisher-Yates
- a SRS followed by a  $k$ -element shuffle is a  $k$ -perm.

Vitter '85  
but  
also '60's

Reservoir Sampling is for the situation where

1) we want only 1-pass over the data,  
eg a data stream, aka streaming.

- 2a)  $n$  is unknown! ← classical
- or 2b) other things unknown! ← applicable to our class

Start here:

2a) First, observe Fisher-Yates looped  $i=1, 2, \dots, n-1$   
 $j \in \{i, \dots, n\}$

but we can do an "inside-out" version

- Input  $x \in \mathbb{R}^n$ , output  $y \in \mathbb{R}^n$  (shuffled), no longer in-place
- For  $i=1, 2, \dots, n-1$ ,  $n \leftarrow$  different!

$j \in \{1, 2, \dots, i\}$  uniformly at random  
(not  $\{i, i+1, \dots, n\}$  anymore)

if  $j \neq i$ ,

$y_i \leftarrow y_j$

$y_j \leftarrow x_i$

} if we initialized  $y = x$ ,  
then this is like  
swap  $(y_i, y_j)$

... and running outer loop  
backward!

(we won't show it here, but similar inductive proof shows this is correct) i.e., reverse input, then flip "for" loop.

Observe that since  $j \in \{1, \dots, i\}$  not  $\{1, \dots, n\}$ , we don't need to know  $n$ .

This is essentially the basic reservoir sampling algo, "Algo R" (Alan Waterman)

Input:  $x \in \mathbb{R}^n$ ,  $n$  "unknown" or "we won't know it 'til we see it"

Output:  $y \in \mathbb{R}^k$ ,  $y$  is the "reservoir"

initialize  $y = x(1:k)$

For  $i = k+1, \dots, n$  ← only place  $n$  appears

$j \in \{1, 2, \dots, i\}$  uniformly at random

if  $j \leq k$

$y_j = x_i$

} if  $j > k$ , we'd be updating  $y$  past its last entry. we don't care, so skip it.

- one-pass ✓
- independent of  $n$  ✓
- Running time:  $O(n)$  OK, though "Algo L" is  $O(k + k \log(n/k))$  w/ trees.
- $k$ -perm? No, not shuffled (easy to see if  $n=k$ )
- SRS? yes.

2b) Let's keep  $k$  of  $n$  items (order still unimportant) such that  $x_i$  is kept with probability proportional to its weight,  $w_i = w(x_i) \geq 0$ ,  
 i.e.,  $w_i / W = \frac{w_i}{\sum_{j=1}^n w_j}$ .

New complication: if  $n$  is unknown, so is  $W$ .

Ex: Sample a row of a matrix  $X(i, :)$  proportional to its  $l_2$ -norm squared. Then  $W = \|X\|_F^2$ .  
 If we stream rows,  $W$  keeps increasing.

Note: not all weights realizable,

ex.  $k=2, n=3$ , wts =  $[1, 0, 0]$

or  $k=n$ , must have

wts = uniform

incompatible w,  $k=2$

→ link on wikipedia

Algo A-Chao (cf. Wikipedia, or M. Chao '82)  
 or P. Efraimidis '15

Input:  $x \in \mathbb{R}^n$

Output:  $y \in \mathbb{R}^k$

initialize  $y = x(1:k)$

$$W = \sum_{i=1}^k w(y_i)$$

For  $i = k+1, k+2, \dots, n$

$$W \leftarrow W + w(x_i)$$

$$p = \frac{k \cdot w(x_i)}{W}$$

with prob.  $p$ , keep this sample  $x_i$  by assigning it to

$y_j$  with  $j \in [k]$  chosen uniformly at random

else, do nothing.

counter-intuitive.

Not obvious!

### (3) Different types of sampling (about) $K$ elements from $[n]$

① SRS (w/o replacement): choose subset  $\Omega \subseteq [n]$  of size  $K$  such that all subsets equally likely: "uniform"

let  $\Omega_K$  be such a set

pros: nice conceptually, get right size, no duplicates

cons: no longer independent (but exchangeable)

uniform in diff. ways

② SRS w/ replacement:

choose a list  $y \in \mathbb{R}^K$  s.t. each  $y_i \sim \text{Unif}([n])$  i.i.d.

pros: independent, get right size

cons: may contain duplicates which seems like a waste

③ "Bernoulli": keep each  $x_i$  w/ probability  $K/n =: p$

Expect to keep  $K$  entries total

pros: independent, no duplicates

cons: the size of our sample is only  $K$  on average, not deterministically

#### Relations

We'll be looking at  $P(\text{Failure}(\Omega))$  of some randomized algo. that uses samples  $\Omega$

Formalize:

Prop (3.1 in Recht "A Simpler Approach...")

$P(\text{failure via } ①) \leq P(\text{failure via } ③)$  if  $P(\text{failure}(\Omega_K)) \geq$

Proof Let  $\Omega'$  be sampled via ② w/  $K$  entries.

$P(\text{failure}(\Omega'_K))$

i.e.,  $\Omega' = \text{unique}(y)$

whenever  $K \leq K'$   
(more is better)

↑  
set of length  $\leq K$   
K list of length  $K$

$$\begin{aligned}
 \text{then } P(\text{failure}(\Omega')) &= \sum_{i=0}^K \underbrace{P(\text{fail}(\Omega') \mid |\Omega'|=i)}_{\substack{\text{is main} \\ \text{observation}}} \cdot P(|\Omega'|=i) \\
 &\geq P(\text{fail}(\Omega_K)) \text{ since } i \leq K \\
 &\geq P(\text{fail}(\Omega_K)) \cdot \underbrace{\sum_{i=0}^K P(|\Omega'|=i)}_{=1} \\
 &\geq P(\text{fail}(\Omega_K)). \quad \square
 \end{aligned}$$

Prop (p.15 "Robust Uncertainty Principles..." Candès et al '05)

Let  $\Omega_K$  be sampled via ① and let

$\Omega'$  be sampled via ③ w/ parameter  $p = K/n$  so  $\mathbb{E}|\Omega'| = K$

then if  $P(\text{failure}(\Omega_K))$  is non-increasing in  $K$  (same as in prev. prop.),

$$P(\text{failure}(\Omega')) \leq 2 \cdot P(\text{failure}(\Omega_K)).$$

[see Appendix of "Robust Principal Component Analysis" Candès et al. '09 for more refined 2-way result]

proof

$$P(\text{failure}(\Omega')) = \sum_{i=0}^{\overset{\text{new!}}{n}} P(\text{fail}(\Omega') \mid |\Omega'|=i) \cdot P(|\Omega'|=i)$$

(same as before, but sum goes to  $n$ , not  $K$ )

$$\geq \sum_{i=0}^K P(\dots \mid \dots) \cdot P(\dots)$$

since non-negative

$$\geq \underbrace{P(\text{fail}(\Omega_K))}_{\text{as before}} \cdot \underbrace{\sum_{i=0}^K P(|\Omega'|=i)}_{\leq 1 \text{ now.}}$$

$p \cdot n = K \in \mathbb{Z}$

is the median

of  $|\Omega'|$  so  $P(|\Omega'| \leq K-1) < \frac{1}{2} < P(|\Omega'| \leq K)$  cf. Japles + Samuels '68

$$\text{so } \sum_{i=0}^K P(|\Omega'|=i) = P(|\Omega'| \leq K) > \frac{1}{2}$$

$$\text{so } \dots > \frac{1}{2} P(\text{fail}(\Omega_K)). \quad \square$$