7. Recitation. Markov Chains: Detailed Balance. Mixing time.

B. Lecture #5. From Bernoulli Processes to Poisson Processes [discrete space, discrete & continuous time].

The two processes discussed here are the simplest dynamic random processes. Simplicity here is related to the fact that the processes are defined with the least number of characteristics. We will also focus on important properties of the processes, e.g. memorylessness, and also on working out interesting (and rather general) questions one may ask (and answer).

1. Bernoulli Process: Definition

Defined as a sequence of independent Bernoulli trials. At each trial

- P(success)=P(x=1)=p
- P(failure)=P(x=0)=1-p

Can be represented as a simple MC (two nodes + two self-loops, please draw one). The sequence looks like $00101010001 = **S*S*S***S$. S here stands for "success".

Examples:

- Sequence of discrete updates – ups and downs (stock market).
- Sequence of lottery wins.
- Arrivals of busses at a station, checked every 1/5/? minutes.

2. Bernoulli: Number of Successes

Number of k successes in n steps follows the binomial distribution

$$\forall k = 0, \cdots, n: \quad P(S=k) = \left(\begin{array}{c} n \\ k \end{array} \right) p^k (1-p)^{n-k} \tag{II.19}$$

$$\text{mean}: \quad \mathbb{E}[S] = np \tag{II.20}$$

$$\text{variance}: \quad var(S) = \mathbb{E}[(S - \mathbb{E}[S])^2] = np(1-p) \tag{II.21}$$

3. Bernoulli: Distribution of Arrivals

Call T_1 the number of trials till the first success (including the success event too). The Probability Mass Function (PMF) for the time of the first success is

$$t = 1, 2, \cdots: \quad P(T_1 = t) = p(1-p)^{t-1} [\text{Geometric PMF}] \tag{II.22}$$

The answer is the product of the probabilities of $(t-1)$ failures and one success (thus memoryless). It is called geometric because checking that the probability distribution is normalized involves summing up the geometric sequence (progression). Naturally, $\sum_{t=1}^{\infty} (1-p)^{t-1} = 1/p$. Mean and variance of the geometric distribution are

$$\text{mean}: \quad \mathbb{E}[T_1] = \frac{1}{p} \tag{II.23}$$

$$\text{variance}: \quad var(T_1) = \mathbb{E}[(T_1 - \mathbb{E}[T_1])^2] = \frac{1-p}{p^2} \tag{II.24}$$

More on the memoryless property. Given n, the future sequence $x_{n+1}, x_{n+2}, \cdots$ is also a Bernoulli process and it is independent of the past. Moreover, suppose we have observed the process for n times and no success has occurred. Then the PMF for the remaining arrival times is also geometric

$$P(T - n = k | T > n) = p(1-p)^{k-1} \tag{II.25}$$

And how about the k^{th} arrival? Let y_k be the number of trials until k^{th} success (inclusive), then we write

$$t = k, k+1, \cdots : \quad P(y_k = t) = \begin{pmatrix} t-1 \\ k-1 \end{pmatrix} p^k (1-p)^{t-k} \text{[Pascal PMF]} \tag{II.26}$$

$$\text{mean :} \quad \mathbb{E}[y_k] = \frac{k(1-p)}{p^2} \tag{II.27}$$

$$\text{variance :} \quad var(y_k) = \mathbb{E}[(y_k - \mathbb{E}[y_k])^2] = \frac{k(1-p)}{p^2} \tag{II.28}$$

The combinatorial factor accounts for the number of configurations of the "k arrivals in y_k trials" type.

Exercise: Define $T_k = y_k - y_{k-1}, \quad k = 2, 3, \cdots$, where thus T_k is the inter-arrival time between $k-1$-th and k-th arrivals. Write down the mass probability distribution function for the k-th inter-arrival time, T_k.

4. *Poisson Process: Definition*

Examples:

- All examples from the Bernoulli case considered in continuous time.
- E-mail arrivals with infrequent check.
- High-energy beams collide at a high frequency (10 MHz) with a small chance of a good event (actual collision).
- Radioactive decay of a nucleus with the trial being to observe a decay within a small time interval.
- Spin flip in a magnetic field.

Exercise: Suggest an example of a Poisson process event inspired by your work or daily life.

There are two ways of thinking about the Poisson process. We can consider it just as a continuous-time version of the (discrete-time) Bernoulli process. The second option is to approach it through random time intervals between successes.

Start discussing the continuous time limit/version of the Bernoulli process. Intervals become infinitesimally small and we replace the probabilities (of successes) by respective probability densities (per unit time). Let $P(k, \tau)$ be the probability of k arrivals in an interval of duration τ. We assume that

- Number of arrivals in disjoint time intervals are independent.
- When the regularization parameter δ is sufficiently small

$$P(k, \delta) \approx \begin{cases} 1 - \lambda\delta & k = 0 \\ \lambda\delta & k = 1 \\ 0 & k > 0 \end{cases} . \tag{II.29}$$

- λ is the arrival rate of the process.

Assume that $n = t/\delta$ and $p = \lambda\delta$, then relations between Bernoulli and Poisson processes are summarized in the following table

Bernoulli	Possion
arrival probability in each time slot $= p$	arrival probability in each δ-interval $= \lambda\delta$
number of arrivals in t intervals	number of successes in n time slots

5. *Poisson: Arrival Time*

Probability density of the first arrival, y_1:

$$p_{Y_1}(y) = \lambda \exp(-\lambda y), \quad y \geq 0 \quad \text{[exponential]}$$

Then

$$P(Y_1 \leq y) = 1 - P(0, y) = 1 - \int_0^y dy' p_{Y_1}(y') = 1 - \exp(-\lambda y)$$

Like Bernoulli, the Poisson keeps the two key properties

- **Fresh Start Property**: the time of the next arrival is independent of the past
- **Memoryless property:** suppose we observe the process for t seconds and no success occurred. Then the density of the remaining time of arrival is exponential.

By extension (taking limit), for the probability density of time of the k^{th} arrival one derives

$$p_{Y_k}(y) = \frac{\lambda^k y^{k-1} \exp(-\lambda y)}{(k-1)!}, \quad y > 0 \quad \text{(Erlang "of order" k)} \tag{II.30}$$

Summary of the relations between Bernoulli process and Poisson process is summarized as follows

	Bernoulli	Poisson
Times of Arrival	Discrete	Continuous
Arrival Rate	p/per trail	λ/unit time
PMF of Number of arrivals	Binomial	Poisson
PMF of Interarrival Time	Geometric	Exponential
PMF of k^{th} Arrival Time	Pascal	Erlang

6. *From Bernoulli to Poisson: # of arrivals in t-intervals as $n \to \infty$*

Let us now check the second way of relating Bernoulli process to the Poisson process – through analysis of the $n \to \infty$ limit for PMF of k arrivals in t intervals:

$$\binom{n}{k} p^k (1-p)^{n-k} = \binom{n}{k} \left(\frac{\lambda t}{n}\right)^k (1 - \frac{\lambda t}{n})^{n-k} \quad \text{[Binomial]} \tag{II.31}$$

$$= \underbrace{\frac{n!}{(n-k)!n^k}}_{\to 1} \frac{(\lambda t)^k}{k!} \underbrace{\left(1 - \frac{\lambda t}{n}\right)^n}_{\to \exp(-\lambda t)} \underbrace{\left(1 - \frac{\lambda t}{n}\right)^{-k}}_{\to 1} \quad \text{(reoder terms)} \tag{II.32}$$

$$\to|_{n\to\infty} P(k=\tau) = \frac{(\lambda t)^k}{k!} e^{-\lambda t} \quad \text{[Poisson}(\lambda\tau)\text{]} \tag{II.33}$$

$$\text{mean :} \quad \mathbb{E}[N] = \lambda t \tag{II.34}$$

$$\text{variance :} \quad var(N) = \mathbb{E}[(N - \mathbb{E}[N])^2] = \lambda\tau \tag{II.35}$$

7. *Merging and Splitting Processes*

Most important feature shared by Bernoulli and Poisson processes is their invariance with respect to mixing and splitting. We will show it on an example of Poisson process but the same applies to Bernoulli process.

Merging: Let $N_1(t)$ and $N_2(t)$ be two independent Poisson processes with rates λ_1 and λ_2 respectively. Let us define $N(t) = N_1(t) + N_2(t)$. This rando process is derived combining the arrivals as shown in Fig. (6). The claim is that $N(t)$ is the Poisson process with the rate $\lambda_1 + \lambda_2$. To see it we first note that $N(0) = N_1(0) + N_2(0) = 0$. Next, since $N_1(t)$ and $N_2(t)$ are independent and have independent increments their sum also have an independent increment. Finally, consider an interval of length τ, $(t, t+\tau]$. Then the number of arrivals in the interval are Poisson$(\lambda_1\tau)$ and Poisson$(\lambda_2\tau)$ and the two numbers are independent. Therefore the number of arrivals in the interval associated with $N(t)$ is Poisson$((\lambda_1 + \lambda_2)\tau)$ - as sum of two independent Poisson random variables. We can obviously generalize the statement to a sum of many Poisson processes. Note that in the case of Bernoulli process the story is identical provided that collision is counted as one arrival.

Splitting: Let $N(t)$ be a Poisson process with rate λ. Here, we split $N(t)$ into $N_1(t)$ and $N_2(t)$ where the splitting is decided by coin tossing (Bernoulli process) - when an arrival occur we toss a coin and with probability p and $1-p$ add arrival to N_1 and N_2 respectively. The coin tosses are independent of each other and are independent of $N(t)$. Then, the following statements can be made

- N_1 is a Poisson process with rate λp.
- N_2 is a Poisson process with rate $\lambda(1-p)$.
- N_1 and N_2 are independent, thus Poisson.

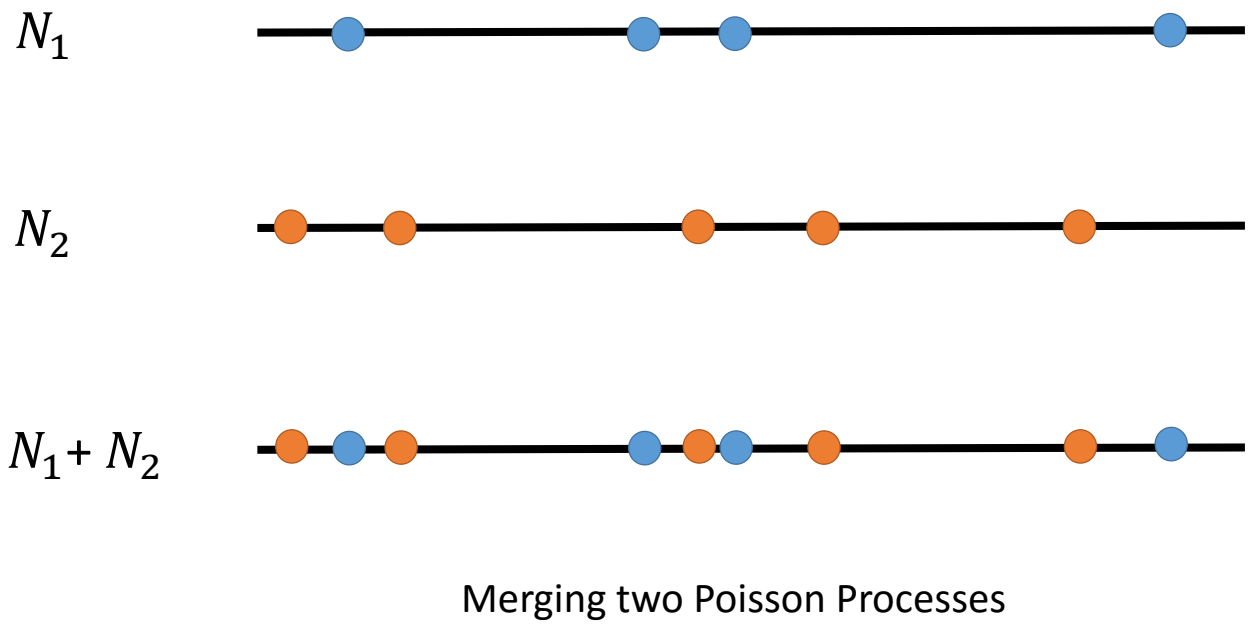

FIG. 6: Merging two Poisson processes.

8. *Recitation. Examples of Bernoulli & Poisson Processes*

+

C. Lecture #6. Monte-Carlo Algorithms: General Concepts and Direct Sampling.

This lecture should be read in parallel with the respective IJulia notebook file. Monte-Carlo (MC) methods refers to a broad class of algorithms that rely on repeated random sampling to obtain results. Named after Monte Carlo -the city- which once was the capital of gambling, i.e. playing with randomness. The MC algorithms can be used for numerical integration, e.g. computing weighted sum of many contributions, expectations, marginals, etc. MC can also be used in optimization.

Sampling is a selection of a subset of individuals/configurations from within a statistical population to estimate characteristics of the whole population.

There are two basic flavors of sampling. Direct Sampling MC - mainly discussed in this lecture and Markov Chain MC. DS-MC focuses on drawing **independent** samples from a distribution, while MCMC draws correlated (according to the underlying Markov Chain) samples.

Let us illustrate both on the simple example of the 'pebble' game - calculating the value of π by sampling interior of a circle.

1. *Direct-Sampling by Rejection vs MCMC for 'pebble game'*

In this simple example we will construct distribution which is uniform within a circle from another distribution which is uniform within a square containing the circle. We will use direct product of two rand() to generate samples within the square and then simply reject samples which are not in the interior of the circle.

In respective MCMC we build a sample (parameterized by a pair of coordinates) by taking previous sample and adding some random independent shifts to both variables, also making sure that when the sample crosses a side of the square it reappears on the opposite side. The sample "walks" the square, but to compute area of the circle we count only for samples which are within the circle (rejection again).

See IJulia notebook for an illustration.

2. Direct Sampling by Mapping

Direct Sampling by Mapping consists in application of the deterministic function to samples from a distribution you know how to sample from. The method is exact, i.e. it produces independent random samples distributed according to the new distribution. (We will discuss formal criteria for independence in the next lecture.)

For example, suppose we want to generate exponential samples, $y_i \sim \rho(y) = \exp(-y)$ – one dimensional exponential distribution over $[0,\infty]$, provided that one-dimensional uniform oracle, which generates independent samples, x_i from $[0,1]$, is available. Then $y_i = -\log(x_i)$ generates desired (exponentially distributed) samples.

Another example of DS MS by mapping is given by the Box-Miller algorithm which is a smart way to map two-dimensional random variable distributed uniformly within a box to the two-dimensional Gaussian (normal) random variable:

$$\int_{-\infty}^{\infty}\frac{dxdy}{2\pi}e^{-(x^2+y^2)/2}=\int_0^{2\pi}\frac{d\varphi}{2\pi}\int_0^{\infty}rdr\,e^{-r^2/2}=\int_0^{2\pi}\frac{d\varphi}{2\pi}\int_0^{\infty}dz\,e^{-z}=\int_0^1 d\theta\int_0^1 d\psi=1.$$

Thus, the deisred mapping is $(\psi,\theta)\to(x,y)$, where $x=\sqrt{-2\log\psi}\cos(2\pi\Theta)$ and $y=\sqrt{-2\log\psi}\sin(2\pi\theta)$.

See IJulia notebook for numerical illustrations.

3. Direct Sampling by Rejection (another example)

Let us now show how to get positive Gaussian (normal) random variable from an exponential random variable through rejection. We do it in two steps

- First, one samples from the exponential distribution: $x\sim\rho_0(x)=\begin{cases}e^{-x} & x>0,\\ 0 & \text{otherwise}\end{cases}$
- Second, aiming to get a sample from the positive half of Gaussian, $x\sim\rho_0(x)=\begin{cases}\sqrt{2/\pi}\exp(-x^2/2) & x>0,\\ 0 & \text{otherwise}\end{cases}$, one accepts the generated sample with the probability

  $$p(x)=\frac{1}{M}\sqrt{2/\pi}\exp(x-x^2/2)$$

  where M is a constant which should be larger than, $\max(\rho(x)/\rho_0(x))=\sqrt{2/\pi}\,e^{1/2}\approx 1.32$, to guarantee that $p(x)\le 1$ for all $x>0$.

Note that the rejection algorithm has an advantage of being applicable even when the probability densities are known only up to a multiplicative constant. (We will discuss issues related to this constant, also called in the multivariate case the partition function, extensively.)

4. Importance Sampling

One important application of MC is in computing of sums/integrals/expectations. Suppose we want to compute an expectation of a function, $f(x)$, over the distribution, $\rho(x)$, i.e. $\int dx\rho(x)f(x)$, in the regime where $f(x)$ and $\rho(x)$ are concentrated around very different x. In this case the overlap of $f(x)$ and $\rho(x)$ is small and as a result a lot of MC samples will be 'wasted'.

Importance Sampling is the method which aims to fix the small-overlap problem. The method is based on adjusting the distribution function from $\rho(x)$ to $\rho_a(x)$ and then utilizing the following obvious formula

$$\mathbb{E}_\rho[f(x)]=\int dx\rho(x)f(x)=\int dx\rho_a(x)\frac{f(x)\rho(x)}{\rho_a(x)}=\mathbb{E}_{\rho_a}\left[\frac{f(x)\rho(x)}{\rho_a(x)}\right]$$

See the IJulia notebook contrasting DS example, $\rho(x)=\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right)$ and $f(x)=\exp\left(-\frac{(x-4)^2}{2}\right)$, with IS where the choice of the proposal distribution is, $\rho_a(x)=\frac{1}{\sqrt{\pi}}\exp\left(-(x-2)^2\right)$. This example shows that we are clearly wasting samples with DS.

Note one big problem with IS. In a realistic multi-dimensional case it is not easy to guess the proposal distribution, $\rho_a(x)$, right. One way of fixing this problem is to search for good $\rho_a(x)$ adaptively. A sophisticated adaptive importance sampling package is available at `https://pypi.python.org/pypi/pypmc/1.0`.

See also

- `https://statmechalgcomp.wikispaces.com/`
- `http://www.math.nyu.edu/faculty/goodman/teaching/Monte_Carlo/direct_sampling.ps`
- `http://www.cs.berkeley.edu/~jordan/courses/260-spring10/lectures/lecture17.pdf`

for additional material/discussions of DS-MC.

5. Direct Brut-force Sampling

This algorithm relies on availability of the uniform sampling algorithm from $[0, 1]$, rand(). One splits the $[0, 1]$ interval into pieces according to the weights of all possible states and then use rand() to select the state. The algorithm is impractical as it requires keeping in the memory information about all possible configurations. The use of this construction is in providing the bench-mark case useful for proving independence of samples.

6. Direct Sampling from a multi-variate distribution with a partition function oracle

Suppose we have an oracle capable of computing the partition function (normalization) for a multivariate probability distribution and also for any of the marginal probabilities. (Notice that we are ignoring for now the issue of the oracle complexity.) Does it give us the power to generate independent samples?

We get affirmative answer to this question through the following **decimation** algorithm generating independent sample $x \sim P(x)$, where $x \doteq (x_i | i = 1, \cdots, N)$:

Algorithm 1 Decimation Algorithm

Input: $P(x)$ (expression). Partition function oracle.

$x^{(d)} = \varnothing; \quad I = \varnothing$
while $|I| < N$ **do**
  Pick i at random from $\{1, \cdots, N\} \setminus I$.
  $x^{(I)} = (x_j | j \in I)$
  Compute $P(x_i|x^{(d)}) \doteq \sum_{x \setminus x_i; x^{(I)} = x^{(d)}} P(x)$ with the oracle.
  Generate random $x_i \sim P(x_i|x^{(d)})$.
  $I \cup i \leftarrow I$
  $x^{(d)} \cup x_i \leftarrow x^{(d)}$
end while

Output: $x^{(\mathrm{dec})}$ is an independent sample from $P(x)$.

Validity of the algorithm follows from the exact representation for the joint probability distribution function as a product of ordered conditional distribution function (chain rule for distribution):

$$P(x_1, \cdots, x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1, x_2) \cdots P(x_n|x_1, \cdots, x_{n-1}). \tag{II.36}$$

(The chain rule follows directly from the Bias rule/formula. Notice also that ordering of variables within the chain rule is arbitrary.) One way of proving that the algorithm produces an independent sample is to show that the algorithm outcome is equivalent to another algorithm for which the independence is already proven. The benchmark algorithm we can use to state that the Decimation algorithm (1) produces independent samples is the brute-force sampling algorithm described in the beginning of the lecture. The crucial point here is that the decimation algorithm can be interpreted in terms of splitting the $[0, 1]$ interval hierarchically, first according to $P(x_1)$, then subdividing pieces for different x_1 according to $P(x_2, x_1)$, etc. This guidanken experiment will result in the desired proof.

In general efforts of the partition function oracle are exponential. However in some special cases the partition function can be computed efficiently (polynomially in the number of steps). For example this is the case for (glassy) Ising model without magnetic field over planar graph. See `http://surface.syr.edu/cgi/viewcontent.cgi?article=1176&context=phy` and references there in for details.

7. Ising Model

Let us digress and consider the Ising model which is, in fact, an example of a larger class of important/interesting multivariate statistics often referred to (in theoretical engineering) as Graphical Models (GM). We will study GM later in the course. Consider a system of spins or pixels (binary variables) on a graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of nodes/vertices and $\mathcal{E}$ is the set of edges. The graph may be 1d chain, tree, 2d lattice ... or any other graph. (The cases of regular lattices are prevalent in physics, while graphs of a relevance to various engineering disciplines are, generally, richer.) Consider binary variables, residing at every node of the graph, $\forall i \in \mathcal{V}: \ \sigma_i = \pm 1$, we call them "spins". If there are N spins in the system, 2^N is the number of possible configuration of spins — notice exponential scaling with N, meaning, in particular that just counting the number of configurations is "difficult". If we are able to do the counting in an algebraic/polynomial number of steps, we would call it "easy", or rather "theoretically easy", while the practically easy case - which is the goal – would correspond to the case when "complexity" of, say counting, would be $O(N)$ - linear in N at $N \to \infty$. (Btw $o(N)$ is the notation used to state that the behavior is actually slower than $O(N)$, say $\sim \sqrt{N}$ at $N \to \infty$, i.e. asymptotically $o(N) \ll O(N)$.) In magnetism (field of physics where magnetic materials are studied) probability of a spin configuration (vector) is

$$p(\sigma) = \frac{\exp\left(-\beta E(\sigma)\right)}{Z}, \quad E(\sigma) = -\frac{1}{2}\sum_{\{i,j\}\in\mathcal{E}} \sigma_i J_{ij} \sigma_j + \sum_{i\in\mathcal{V}} h_i \sigma_i, \tag{II.37}$$

$$Z = \sum_{\sigma} \exp\left(-\beta E(\sigma)\right). \tag{II.38}$$

$E(\sigma)$ is the energy of a given spin configuration, σ. The first term in $E(\sigma)$ is pair-wise (wrt nodal spins), spin exchange/interaction term. The last term in $E(\sigma)$ stands from (potentially node dependent) contribution of the magnetic field, $h = (h_i | i \in \mathcal{V})$ on individual spins. Z is the partition function, which is the weighted sum of the spin configurations. Formally the partition function is just the normalization condition introduced to enforce, $\sum_\sigma p(\sigma) = 1$. For a general graph with arbitrary values of J and h, Z is the difficult object to compute, i.e. complexity of computing Z is $O(2^N)$. (Notice that for some special cases, such as the case of a tree, or when the graph is planar and $h = 0$, computing the partition function becomes easy.) Moreover, computing other important characteristics, such as the most probable configuration of spins

$$\sigma_{ML} = \arg\max_{\sigma} p(\sigma), \tag{II.39}$$

also called Maximum Likelihood and Ground State in information sciences and physics respectively, or the (so-called marginal) probability of observing a particular node in the state σ_i (can be $+$ or $-$)

$$p_i(\sigma_i) = \sum_{\sigma \setminus \sigma_i} p(\sigma), \tag{II.40}$$

are also difficult problems. (Wrt notations – $\arg\max$ - pronounced argmax - stands for particular σ at which the maximum in Eq. (II.39) is reached. $\sigma \setminus \sigma_i$ in the argument of the sum in Eq. (II.40) means that we sum over all σ consistent with the fixed value of σ_i at the node i.)