Foundations of data science, summer 2020

JONATHAN LENNARTZ, MICHAEL NÜSKEN, ANNIKA TARNOWSKI

4. Exercise sheet
Hand in solutions until Thursday, 14 May 2020, 12:00

Exercise 4.1 (Random point in a ball). (8+3 points)

Write a Python routine (eg. in a jupyter notebook) to choose a uniformly random point in a unit ball of dimension d. $\boxed{5}$

Use your routine with $d = 2$ to pick $1\,000$ points and plot them, so that we can visually check the correctness of the algorithm. $\boxed{3}$

Can you also plot a sample with $d = 3$? $\boxed{+3}$

Solution. The idea here was to stay close to the algorithm discussed in class. You can use one of the possible numpy commands that generate points with respect to a Gaussian and then normalise. This yields points uniformly random on the surface of a ball.

To get a uniform distribution in the interior of the ball, each vector has to be scaled with a scalar chosen with a certain distribution, as discussed in class. Put $R \leftarrow U^{\frac{1}{d}}$. ⌈Then for $r \in [0, 1]$:

$$\operatorname{prob}(R < r) = \operatorname{prob}\left(U < r^d\right) = r^d.$$

This is the cumulative distribution function. The density is its derivative:

$$\frac{\mathrm{d}}{\mathrm{d}r} \operatorname{prob}(R < r) = dr^{d-1}.$$

And that is exactly what we found in the course. Thus putting $R \leftarrow U^{\frac{1}{d}}$ is exactly the result of inverse transform sampling.⌋

Other methods were also possible. You can check the correctness of your own algorithm easily by considering the plot. ○

Exercise 4.2 (Random exit). (10 points)

Consider an experiment for which some general indicator variable X has

$$\operatorname{prob}(X = 1) = p \neq 0.$$

We create new random variables as following: Repeat the experiment to obtain X_i for all $i \in \mathbb{N}'$. That is, ... (fill in!)

Then we set the exit time

$$T = \min \{i \mid X_i = 1\}.$$

We simply may assume that this minimum always exists.

2 (i) Prove that

$$T = i \iff X_i = 1 \wedge X_{i-1} = 0 \wedge \cdots \wedge X_1 = 0.$$

Solution. By definition it holds that $T = i$ if and only if $i \in \{j \mid X_j = 1\}$ and $\forall \ell < i : \ell \notin \{j \mid X_j = 1\}$. By the nature of our experiment, $i \in \{j \mid X_j = 1\}$ means that $X_i = 1$ and $\ell \notin \{j \mid X_j = 1\}$ means that $X_\ell = 0$ as there are only two possible outcomes.

So

$$T = i \iff X_i = 1 \wedge \forall \ell < i : X_\ell = 0,$$

hence

$$T = i \iff X_i = 1 \wedge X_{i-1} = 0 \wedge \cdots \wedge X_1 = 0.$$

$\bigcirc$

1 (ii) Compute(!) prob $(T = 1)$, prob $(T = 2)$, prob $(T = 3)$.

Solution. We have prob $(T = 1) = p$, prob $(T = 2) = (1-p)p$ and prob $(T = 3) = (1-p)^2 p$. For an explanation, see the following item.

$\bigcirc$

2 (iii) Prove a formula for prob $(T = i)$.

Solution. We have already shown that

$$T = i \iff X_i = 1 \wedge X_{i-1} = 0 \wedge \cdots \wedge X_1 = 0.$$

As the variables are *independent* (never forget to write down keywords) we have

$$\begin{aligned}
&\text{prob}\,(X_i = 1 \wedge X_{i-1} = 0 \wedge \cdots \wedge X_1 = 0) \\
&= \text{prob}\,(X_i = 1) \cdot \text{prob}\,(X_{i-1} = 0) \cdot \cdots \cdot \text{prob}\,(X_1 = 0) \\
&= p \cdot (1-p) \cdot \cdots \cdot (1-p) \\
&= p \cdot (1-p)^{n-1}.
\end{aligned}$$

$\bigcirc$

1 (iv) Express $\mathrm{E}(T)$ as an infinite sum.

Solution. We have

$$\mathrm{E}(T) = \sum_{i=1}^{\infty} i \cdot \operatorname{prob}(T = i) = \sum_{i=1}^{\infty} i \cdot (p \cdot (1-p)^{i-1}).$$

(v) Use the geometric series to derive a formula for $\sum_{i=1}^{\infty} ix^{i-1}$. 1

Solution. We have for $x < 1$ that

$$\sum_{i=1}^{\infty} ix^{i-1} = \sum_{i=0}^{\infty} ix^{i-1} = \frac{\partial}{\partial x}\left(\sum_{i=0}^{\infty} x^i\right) = \frac{\partial}{\partial x}\left(\frac{1}{1-x}\right) = \frac{1}{(1-x)^2}.$$

(vi) Compute the expected exit time $\mathrm{E}(T)$. 1

Solution.

$$\mathrm{E}(T) = \sum_{i=1}^{\infty} i \cdot (p \cdot (1-p)^{i-1}) = p \sum_{i=1}^{\infty} i(1-p)^{i-1} = p \cdot \frac{1}{p^2} = \frac{1}{p}.$$

(vii) Consider the method 1 algorithm from class. What is its expected runtime? 2

Solution. The method 1 algorithm runs a loop to uniformly at random generate a point x in a cube and exits if and only if $\|x\| \leq 1$. By the nature of uniform distribution

$$\operatorname{prob}\left(x \in B^d\right) = \frac{\mathrm{vol}(B^d)}{\mathrm{vol}([-1,1]^d)} = \frac{\mathrm{vol}(B^d)}{2^d}.$$

Hence the expected runtime of a single loop is $\frac{2^d}{\mathrm{vol}(B^d)}$ by our previous results about random exit experiments. For $d \to \infty$ not only $2^d \to \infty$, also $\mathrm{vol}(B^d) \to 0$.

So the runtime for high dimensions grows even strictly faster than 2^d. With a little more effort, we can even show that grows even faster than $d^{\frac{d}{2}}$. This is a terrible runtime for just obtaining a single point in the ball.