CSC 535 HW1

Simon Swenson

8/30/2018

Introduction

I completed all required homework problems. For Q2, I went with method A (assuming no duplicate results) rather than method B.

1 Q1

As the number of dice throws (of 2d6) approach infinity, the experimental results should converge to the actual probability distribution. In this case, since we are checking two independent events (each die in the dice roll), we *could* create a chart for the possible outcomes. However, in this case, that is not really necessary. We know that a roll of 12 can only happen when each die comes up a 6. Therefore, we know the probability is $\frac{1}{n}$, where n is the number of possible outcomes (when both dice are considered). This is 36, since each die has six possible outcomes. Therefore, we expect the experimental results to converge to the value $\frac{1}{36} \approx 0.028 = 2.8\%$. However, even after 1000 dice throws, the experimental results are off by a non-negligible amount. The first result of the random number generator was 3.3%, and only one of the ten experiments (each of 1000 throws) resulted in 3.3% as well. However, reseeding the random number generator immediately yielded the same result, as expected:

1. 3.3%
2. 2.1%
3. 2.9%
4. 2.2%
5. 2.6%
6. 2.8%
7. 2.5%
8. 2.1%
9. 3.3%
10. 3.1%
11. 3.3% (after seed reset)

In any large system that uses random number generation, the random seed is important for reproduceability and debugging. In the best-cased scenario, the initial state of the random number generator shouldn't matter, however, that's not always the case, especially when the system is still being built and debugged (from my experience with deep learning).

2 Q2

We can see here that our world is very well-structured. The first picture is easily recognizeable as a tiger, whereas the following two could be described as ”snow” or random noise. We cannot make sense of them; we are used to interacting with a structured world.

My hypothesis is that, though generating a picture of a tiger with a random number generator may be possible, given infinite time and computing power, that outcome is *extremely* unlikely. I'm reminded of the old idiom of a thousand monkeys on a typewriter eventually producing the works of Shakespeare, or a more down-to-earth example, the origin and evolution of life on Earth. However, in my finite frame of reference, I do not expect any random number generator that *I* encounter to produce such an image. Rather, the vast majority of images will simply be resolved to ”nondescript noise” by our visual processes. Essentially, though they all differ in which pixels are turned on and which are off, those pictures will all resolve to the same meaninglessness in our brains.

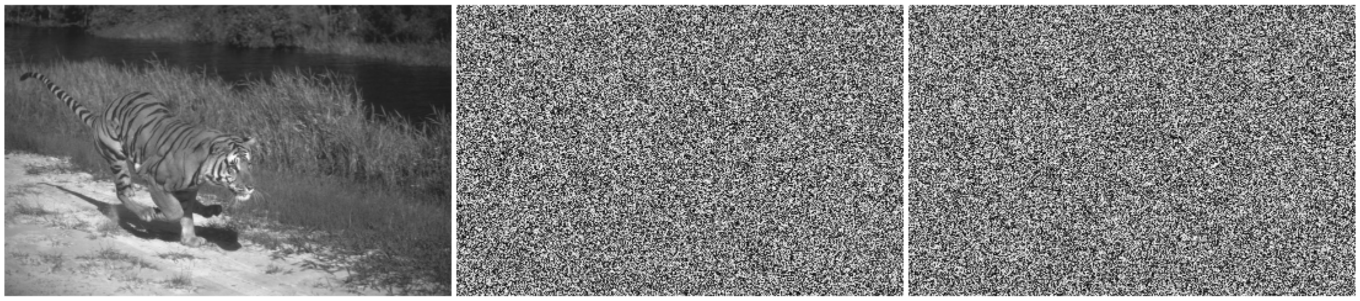

Figure 1: So-called "Random noise," alongside two more pictures of random noise

2.1 A

It is an interesting question, then, to muse on the probability of such an outcome: a random number generator producing a 236x364 matrix of values in the range [0, 256). (Note, however, that similar images would also resolve to a tiger in our brain. We can tweak the pixel values slightly to create a completely different image that also resolve to a tiger.[1]) For that specific tiger image, the probability is like rolling 236×364 256-sided dice. The actual probability is then $\frac{1}{236\times364\times256} \approx 4.547e-8$. Put another way, you would have to generate $236 \times 364 \times 256 = 21991424$ images to get one with a tiger.

3 Q3

3.1 A

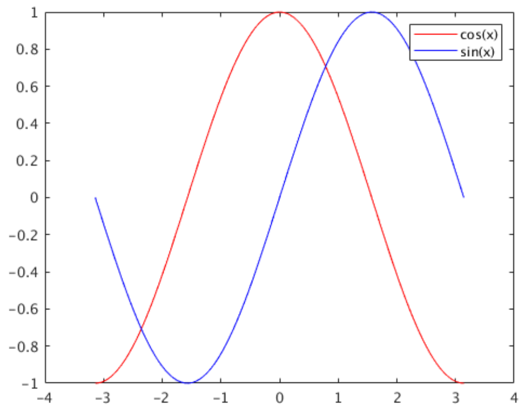

Figure 2: Graphs of both sine and cosine

3.2 B

4 Q4

We can also compute the sum of the bars in the nearby figure. Doing so yieds the result of 0.9414.

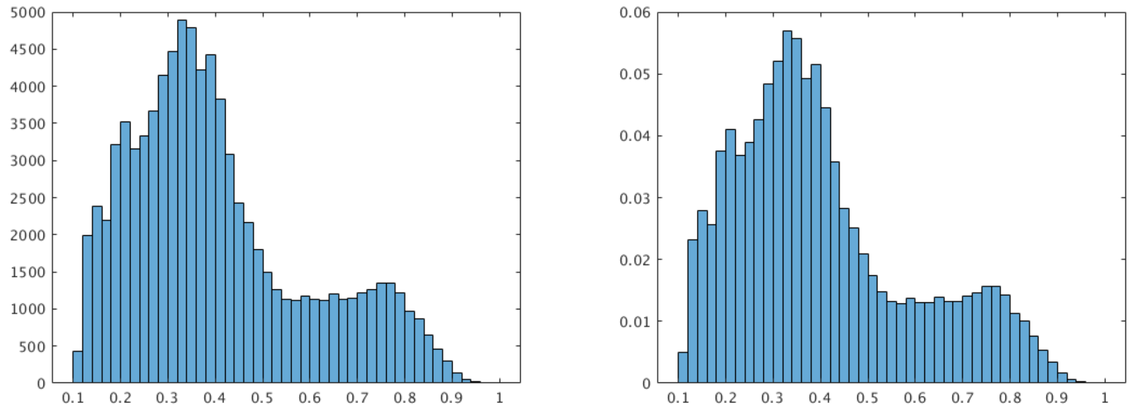

Figure 3: Histogram of the tiger image, normed on right (number of bins is MATLAB default)

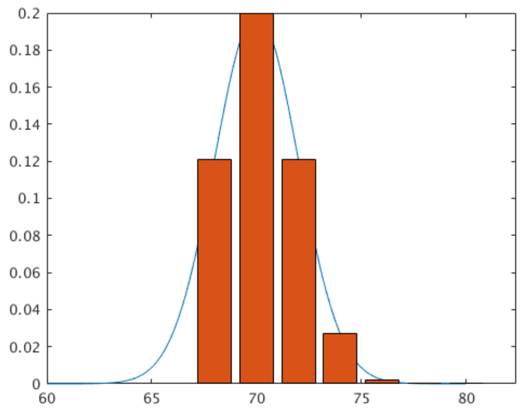

Figure 4: The normal distribution for heights adult men with bars showing an estimation method with $\delta = 2$. Note that some bars are not visible on the graph because the probabilities for them are so low.

If we extend the bars in the nearby figure over a larger distance and vary the bar deltas, we can empirically measure how accurate such a method is, comparing it to one, the area under any probability density function.

As expected, lowering the bar deltas increases the accuracy substantially. Also expected, diminishing returns is at play here, leading to a logarithmic shape.

5 Q5

My most immediate research interest is to use both point-cloud and color video data to predict landslides. Though it didn't immediately strike me as a task that could use probabilistic graphical models, since I tend to think in terms of algorithms that give an exact answer, there is a lot of missing data and noise in real-world video and point cloud data. If I'm understanding the proposal that Josh linked, correctly, it sounds like most of this video data will be crowdsourced. In other words, that data will be from low-fidelity sources like cell phones. Therefore, they will not represent reality exactly. This is where PGMs can help. Rather than trying to find an exact answer, we can use probability theory to find the most *likely* solution for the surface of the terrain geometry. While we can't reproduce the terrain geometry perfectly, we can certainly come up with a good estimation of it by using probablistic graphical models, each new video source giving a new slice of data, which affects the graph.

In addition to this, the task seems similar to another task that Kobus illustrated in his talk at the graduate meet-and-greet in April this year. The illustration involved generating geometry of a room from a picture of that room. Probablistically, I can conclude that generating geometry from cell phone video and point cloud data may use

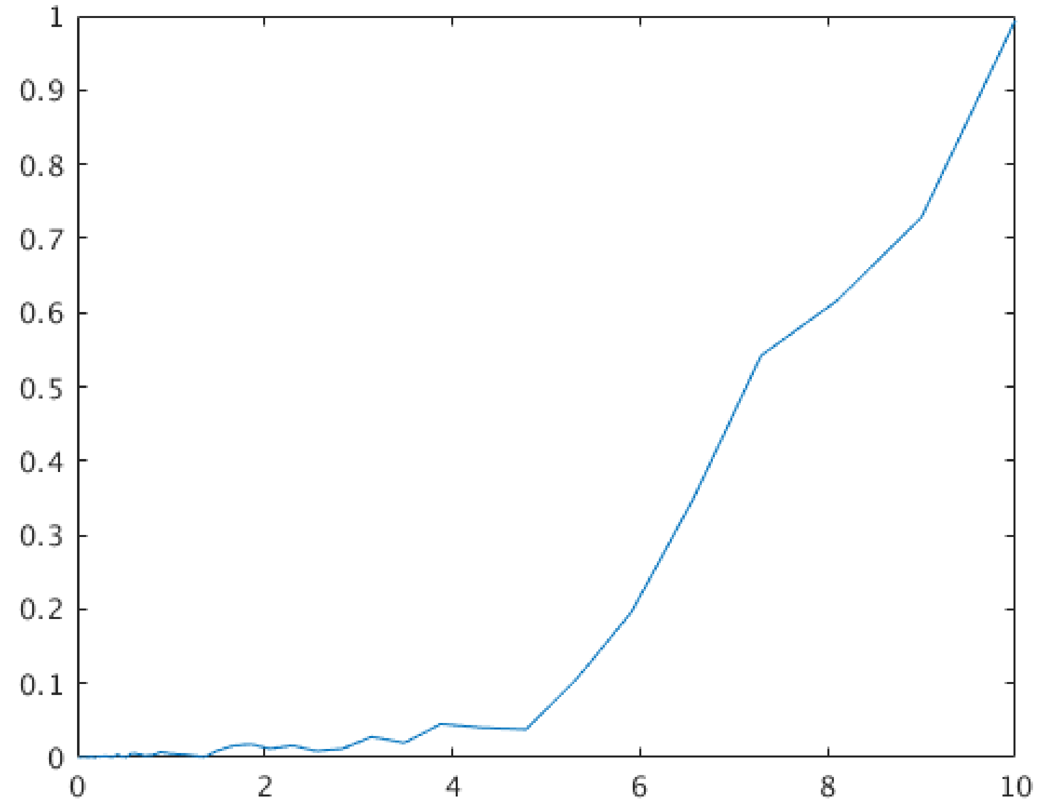

Figure 5: Error as a function of bar deltas

similar techniques.

References

[1] Ian GoodFellow, Nicolas Papernot, Sandy Huang, Yan Duan, Pieter Abbeel, Jack Clark. *Attacking Machine Learning with Adversarial Examples*, OpenAI Blog, February 24, 2017. https://blog.openai.com/adversarial-example-research/