CIS192 Python Programming

Probability and Monte Carlo Methods

Robert Rand

University of Pennsylvania

October 15, 2015

Outline

The Two Child Problem

Suppose I have two children.

I announce that I have a son.

What is the probability that both children are boys?

Considering the Possibilities

First child	Second child
Boy	Boy
Boy	Girl
Girl	Boy
~~Girl~~	~~Girl~~

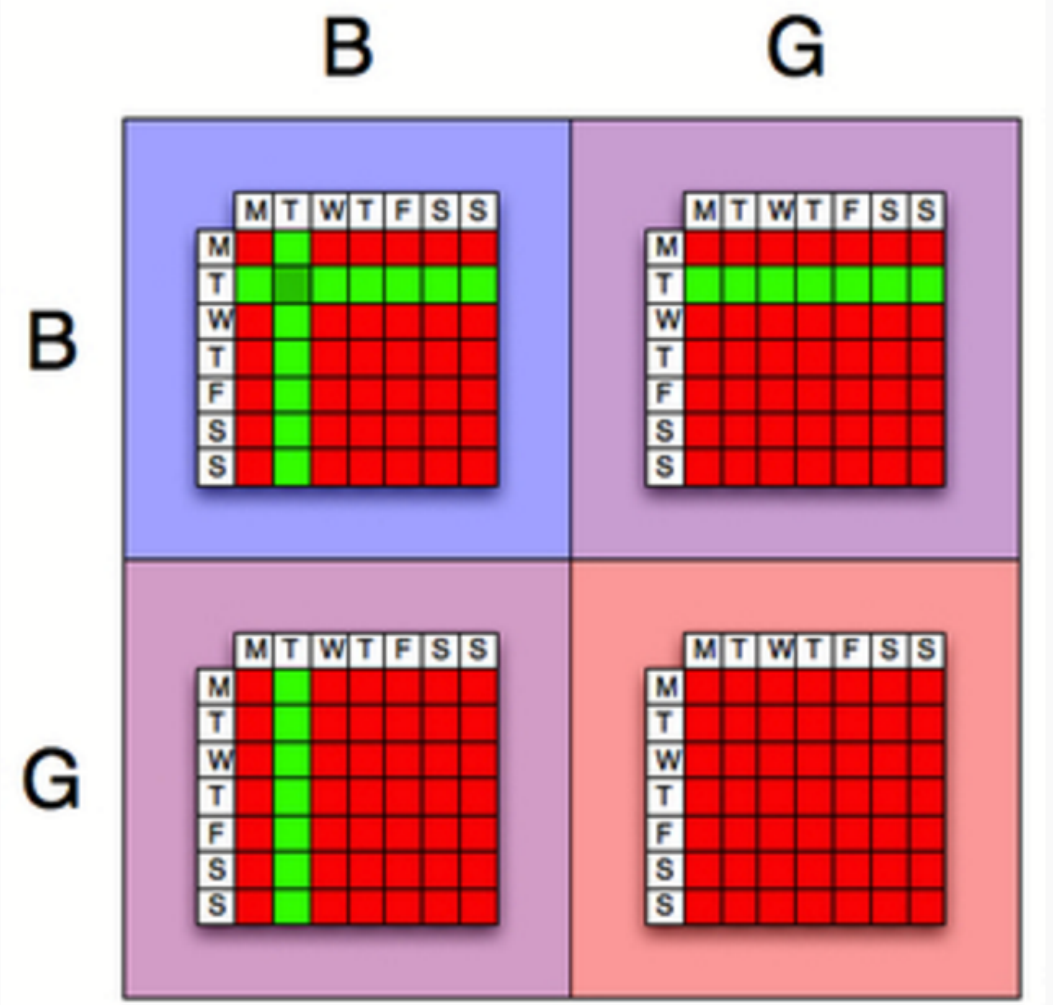
Tuesday's Child

I announce that

"I have a son who was born on a Tuesday."

Now what is the probability that both children are boys?

A Handy Visualization



The **Tuesday Child Problem**, visually demonstrated

Random Variables

- Definition: A *Random Variable* is a function from a set of possible outcomes to associated (generally numeric) values.
  - For a coin toss we've been mapping [lands heads] to 1 and [lands tails] to 0.
  - The values associated with a die roll are the numbers 1 through 6.
- Commonly represented by capital X.
- We will model random variables with (random) Python functions.

Probability

- Each random variable has an associated *Probability Distribution* that maps its values to *probabilities*.
  - Every probability must be in the interval $[0, 1]$.
  - The sum of the probabilities must equal 1.
- We say $Pr(X = x) = p$ do denote the probability of an event.
  - eg. for a fair die D, $Pr(D = 3) = \frac{1}{6}$

Expectation

$$E(X) = \sum_x Pr(x) * x$$

- Where C represents a coin toss as above, $E(C) = 0.5 * 1 + 0.5 * 0 = 0.5$
- For a die D, $E(D) = \frac{1}{6} * 1 + \cdots + \frac{1}{6} * 6 = 3.5$

The Law of Large Numbers

The *Weak Law of Large Numbers* states that as the number of trials approaches infinity, the frequency of a given outcome approaches its true probability.

- i.e. If we toss a lot of coins, we will get heads close to half the time.
- Which brings us back to Python.

The Monte Hall Problem

You're on a gameshow.

There are three doors in front of you, two have goats behind them and one has a car. After you pick a door, Monty Hall (the host) opens another door, revealing a goat. He then gives you the option of switching.

Should you switch?

Monty Python

```
def monty(switch = False):
  car = randint(1,3)
  pick = randint(1,3)
  open = choice(list({1, 2, 3} - {car, pick}))
  if switch:
     pick = choice(list({1, 2, 3} - {pick, open}))
  if pick == car:
     return "Car"
  else:
     return "Goat"
```

Evil Monty

```
def monty(switch = False, evil = False):
    car = randint(1,3)
    pick = randint(1,3)
    if evil and not pick == car:
        return "Goat*"
    open = choice(list({1, 2, 3} - {car, pick}))
    if switch:
        pick = choice(list({1, 2, 3} - {pick, open}))
    if pick == car:
        return "Car"
    else:
        return "Goat"
```

Joint Probability

Sometimes we're interested in the probability of two events occurring together. For example, we might be interested in the probability that Monty is evil AND we win a car.

We represent this as $Pr(X, Y)$ or $Pr(X \cap Y)$
i.e. $Pr(result = Car, Monty = Evil)$

Conditional Probability

Alternatively we're interested in the probability of some event happening once we know that another event has occured. For example, we might be interested in the probability that we win a car GIVEN the knowledge that Monty is evil.

We represent this as $Pr(X \mid Y)$
i.e. $Pr(result = Car \mid Monty = Evil)$

$$Pr(X \mid Y) = \frac{Pr(X \cap Y)}{Pr(Y)}$$

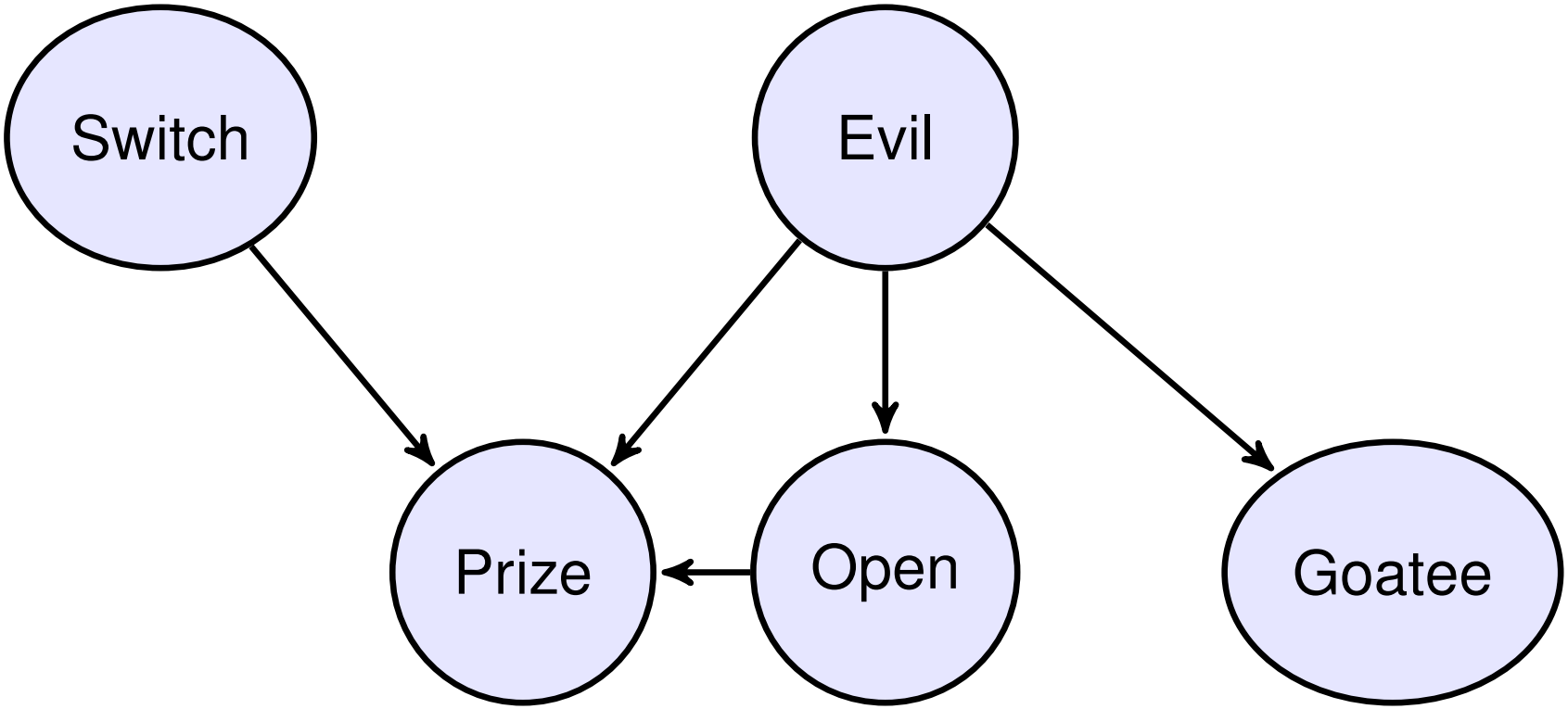
Independence

If the event Y doesn't impact the probability of X, that is:

$$Pr(X \mid Y) = Pr(X)$$

we say that X and Y are *independent*.

Bayes Nets



Bayes' Theorem

$$Pr(X \mid Y) = \frac{Pr(Y \mid X)P(X)}{P(Y)}$$

This follows directly from the definition of conditional probability:

$$Pr(X \mid Y)P(Y) = Pr(X \cap Y) = P(Y \mid X)P(X)$$

PAC Learning

We say that a *concept c* is *Probably Approximately Correct (PAC) Learnable* if we have an time algorithm that:

1. Takes an arbitrary error bound ϵ and failure probability δ,
2. Takes in an arbitrary distribution of labeled items D,
3. Returns an approximation of the concept c_a that (with probability greater than $1 - \delta$) correctly classifies $(1 - \epsilon)$–fraction of the points, and
4. Runs in polynomial time in ϵ and δ.

PAC Example

The concept *c* might be “people who are the right height and weight to become US Air Force pilots”.

Say our distribution is a randomly selected group of adults. We send them all to the airforce academy and record their height, weight and whether they passed.

We will then estimate the minimum weight as the weight of the lightest person who passed the test, and likewise for max weight, and min height and weight.

We won’t go through all the math here, but it should only take $\approx 1/\epsilon$ people to come within ϵ of each parameter, so $\approx 4/\epsilon$ to approximate all four borders. We then scale by a function of δ.

PAC Caveat

We're only guaranteeing high accuracy classification of samples *drawn from our distribution*.

Hence, if we're testing toddlers, we won't have any who pass the test, and our approximate test will always say "No".

Fortunately, that's the correct response to 100% of toddlers who wish to enroll in the Air Force.

Random Functions

- `random()` returns a float between 0 and 1 uniformly at random
- `randint(a,b)` returns a random integer between *a* and *b* inclusive
- `choice(li)` samples a random element from `li`
- `sample(li)` returns a randomly selected subset of `li`
- `shuffle(li)` randomly permutes the list `li`
- `uniform(a, b)` returns a float uniformly distributed between *a* and *b*
- `gauss(mean, stddev)` returns a float from a gaussian (or normal) distribution with the given mean and standard deviation.