The Principle of Maximum Entropy for Discrete Random Variables

Contents


```
import numpy as np
np.set_printoptions(precision=3)
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set(rc={"figure.dpi":100, 'savefig.dpi':300})
sns.set_context('notebook')
sns.set_style("ticks")
```

Objectives

- Learn how to find the maximum entropy distribution for discrete random variables.

The Brandeis dice problem

This is from the 1962 Brandeis lectures of E. T. Jaynes.

> When a die is tossed, the number of spots up can have any value x in $1, \ldots, 6$. Suppose a die has been tossed N times and we are told only that the average number of spots up was not 3.5 (as we might expect from an "honest" die) but 4.5. Given this information, and nothing else, what probability should we assign to x spots on the next toss?

Let X be a random variable corresponding to the result of tossing the die. The description above imposes the following mean value constraint on the random variable X:

$$\sum_{x=1}^{6} xp(x) = 4.5.$$

As we discussed in the lecture, to come up with a probability mass function for X you have to maximize the entropy subject to the constraints above. We saw that this constrained optimization problem has a unique solution of the form:

$$p(x) = \frac{\exp\{\lambda x\}}{Z(\lambda)},$$

where $Z(\lambda)$ is the *partition function*:

$$Z(\lambda) = \sum_i e^{\lambda i} = e^{\lambda} + e^{2\lambda} + \cdots + e^{6\lambda},$$

https://en.wikipedia.org/wiki/Partition_function_(statistical_mechanics)

and λ is a parameter to be tuned so that the constraint is satisfied. We will identify λ by solving a root finding problem. To this end, let us write the partition function as:

$$Z(\lambda) = \left(e^{\lambda}\right)^1 + \left(e^{\lambda}\right)^2 + \cdots + \left(e^{\lambda}\right)^6.$$

According to the theory, in order to find λ we must solve:

$$\frac{\partial \log Z}{} = 4.5$$

$$\frac{\partial}{\partial \lambda} - 4.5.$$

Or equivalently:

$$\frac{1}{Z(\lambda)} \sum_{i=1}^{6} i e^{-\lambda i} = 4.5.$$

So, to find λ, we need to find the root of this function:

the i's here are the summing variable, not the imaginary identifier

$$f(\lambda) = \frac{1}{Z(\lambda)} \sum_{i=1}^{6} i e^{-\lambda i} - 4.5.$$

Let's code it up:

```
def f(lam : float):
    """The function of which the root we want to find."""
    p_unormalized = np.exp(np.arange(1, 7) * lam)
    p = p_unormalized / np.sum(p_unormalized)
    E_X = np.sum(np.arange(1, 7) * p)
    return E_X - 4.5
```

recall that Z is the normalization constant

To find the root, we will use the Brent's method as implemented in `scipy`:

```
import scipy.optimize

# Left bound for x
a = -2
# Right bound for x
b = 2
res = scipy.optimize.root_scalar(
    f,
    bracket=(a,b),
    method='brentq',
    xtol=1e-20,
    rtol=1e-15
)

print(res)

lam = res.root

print(f'Lambda = {lam:.2f}')

# The maximum entropy probabilities
p = np.exp(lam * np.arange(1, 7))
p = p / np.sum(p)

print(f'p = {p}')
```

```
      converged: True
           flag: 'converged'
 function_calls: 11
     iterations: 10
           root: 0.3710489380810334
Lambda = 0.37
p = [0.054 0.079 0.114 0.165 0.24  0.347]
```

Check that the expectation turns out to be correct:

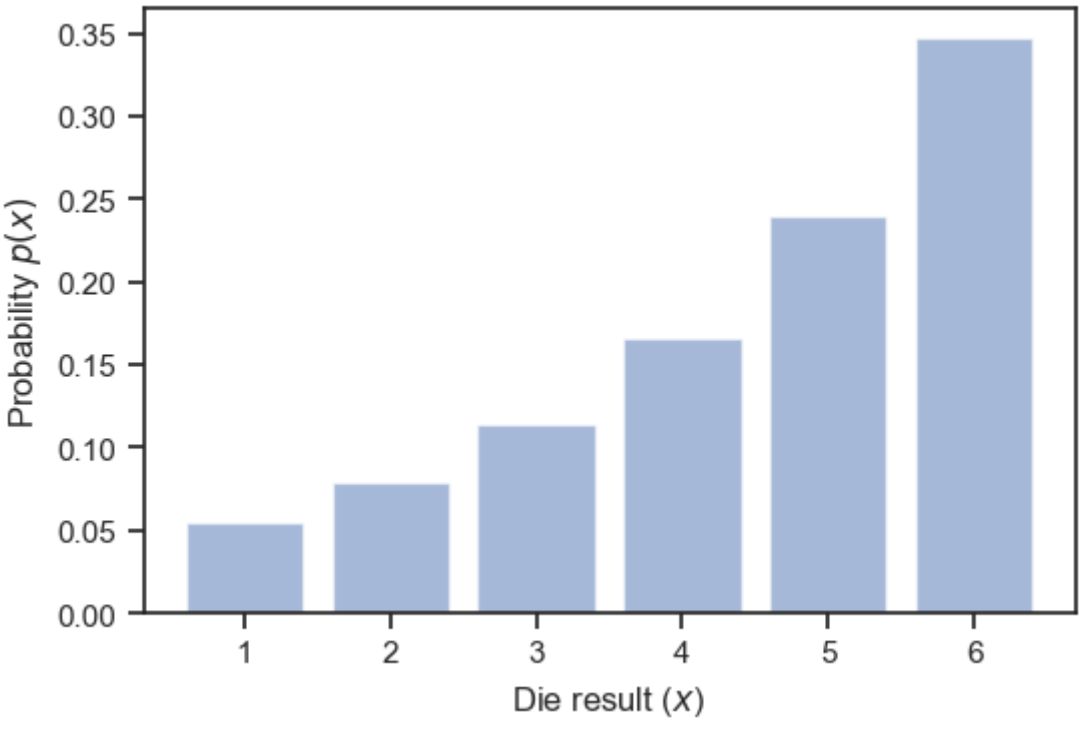
```
(p * np.arange(1, 7)).sum()
```

```
4.5
```

We are good!

Now, let's plot the maximum entropy probabilities:

```
fig, ax = plt.subplots()
plt.bar(np.arange(1, 7), p, alpha=0.5)
ax.set_xlabel('Die result ($x$)')
ax.set_ylabel('Probability $p(x)$');
```



Questions

E[X^2] = V[X] + (E[X])^2 = 0.2 + 4.5^2

- Rerun the code above assuming that the mean is 3.5. What kind of distribution do you find? Why?
- If you have some time to spare, modify the example above to add the constraint that the variance of X should be 0.2. Hint: First, translate the constraint about the variance to a constraint about $\mathbb{E}[X^2]$. Second, you need to introduce one more parameter to optimize for. Call it μ. The distribution would be $p(x) = \frac{\exp\{\lambda x + \mu x^2\}}{Z(\lambda,\mu)}$. Then derive the set of non-linear equations you need solve to find λ and μ by expanding these two equations:

- Z will have a new definition
- set up the two equations
- use root solver to identify the parameters

$$\frac{\partial Z}{\partial \lambda} = \mathbb{E}[X],$$

and

$$\frac{\partial Z}{\partial \mu} = \mathbb{E}[X^2].$$

Finally, use scipy.optimize.root to solve the root-finding problem. Be careful with this because it could take several hours to do right...

By Ilias Bilionis (ibilion[at]purdue.edu)