Lecture 1

Review & Intro to non-linear regression

About Me

- PhD in Mechanical Engineering (2019)
- Lead Data Scientist & Innovation Projects Manager at Fujitsu
- Adjunct Professor for the MDS program

Jeffrey.English@ubc.ca

Course Outline

- Topics
  - Fitting probability distributions
  - Fitting *conditional* probability distributions (GLMs and GAMs)
  - Fitting polynomials & splines
  - Non-parametric estimation (kernel density)
  - Parametric estimation with neural networks
  - Time series data
- Grades
  - Labs – 4 starting this week (10% each)
  - Project (60%)

Project

- In teams of 2-3, you will conduct an analysis of a dataset using methods covered in this course (and previous courses)

- There are three parts of the project
  - Data proposal – due February 26
  - Exploratory analysis – due March 7
  - Report – due March 21

- More information is available on the Github (and hopefully soon, Canvas)

First example

0.767359
0.894135
0.835443
0.97312
0.631978
0.967231
0.99357
0.736777
0.944434
0.920174
0.746323
0.673962
0.812484
0.702883
0.753294
0.718707
0.724407
0.630302
0.67623
0.792855
0.671233
0.53399
…
0.57358
0.586896
0.62967

- Consider a series of values I’ve measured
- I want to understand the *process that generated these numbers,* ***statistically***
- This will help me answer all sorts of questions:
    - What is the likelihood of a particular value, or range of values, occurring?
    - Does the process change if I compare different samples?

The Likelihood Function

$$L(\Theta) = f(y; \Theta)$$

The likelihood depends on some value or values Θ (the parameters of the distribution)

$$L(\Theta) = \prod_{i=1}^{n} f(y_i; \Theta)$$

Likelihood can be expressed as the product of the probabilities of N different observations

The Likelihood Function

For a normal distribution:

$$f(y;\Theta) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{\left(x_j-\mu_0\right)^2}{2\sigma_0^2}\right)$$

Consider a set of observations:

Value	Probability
2.1	0.266
3.6	0.333
3.2	0.391
2.9	0.397
2.5	0.352
Likelihood	**0.00485**

Log-Likelihood

Taking the log of the likelihood gives the log-likelihood function (sound familiar?)

Taking the log transformation also converts the product into a sum – much easier to calculate!

Log-Likelihood

$$L(\Theta) = \prod_{i=1}^{n} f(y_i; \Theta)$$

$$\log\big(L(\Theta)\big) = \log\left(\prod_{i=1}^{n} f(y_i; \Theta)\right)$$

$$\log\big(L(\Theta)\big) = \log\big(f(y_0; \Theta)\big) + \log\big(f(y_1; \Theta)\big) + \ \ldots$$

$$\boxed{\log\big(L(\Theta)\big) = \sum_{i=1}^{n} f(y_i; \Theta)}$$

Finding the best log likelihood

Principal of calculus/optimization – the minimum or maximum of a function is where the derivative is zero

For the normal distribution, these parameters are well known:

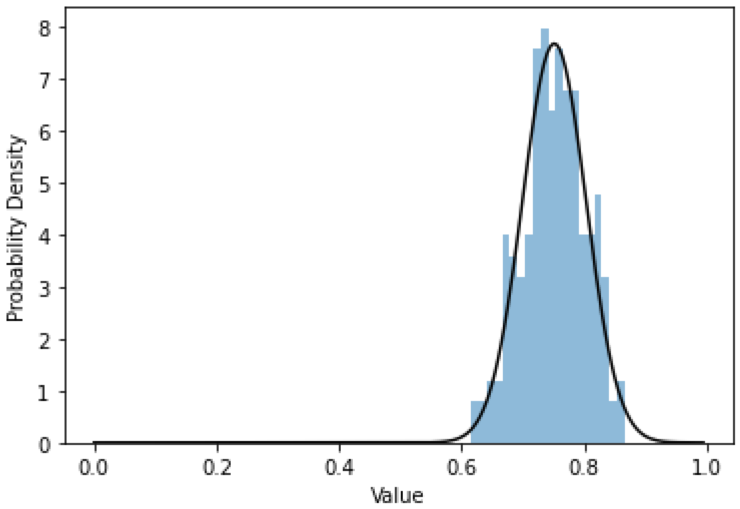
$$\hat{\mu} = \frac{1}{n}\sum_{j=1}^{n} x_j$$

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{j=1}^{n}\left(x_j - \hat{\mu}\right)^2$$

Completing the example

```
# Values is a numpy array of observations
n = len(values)
mean = 1/n * sum(values)
variance = 1/n * sum((values-mean)**2)
print(mean, variance)
>>> 0.7559705424951697 0.02141794314880534
```

Solving the example

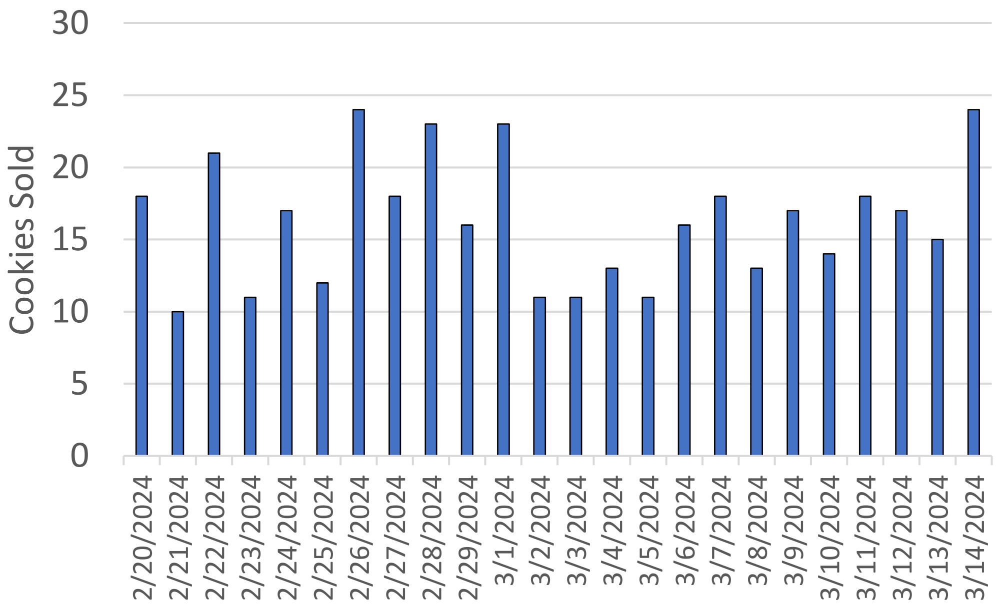


Our model fits (at least visually)

MLE for Other Distributions

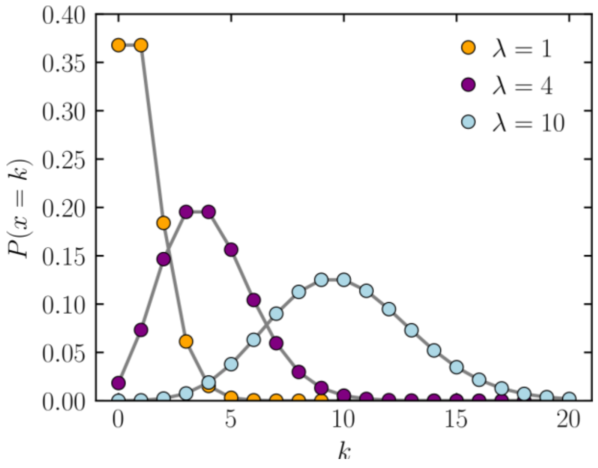
Let's consider another scenario:

I run a business that sells cookies to hungry students. I want to understand how many cookies I sell per day so that I can plan how much flour and chocolate to buy.



MLE for the Poisson Distribution

$$P(k, \lambda) = \frac{\lambda^k e^{-\lambda}}{k!}$$



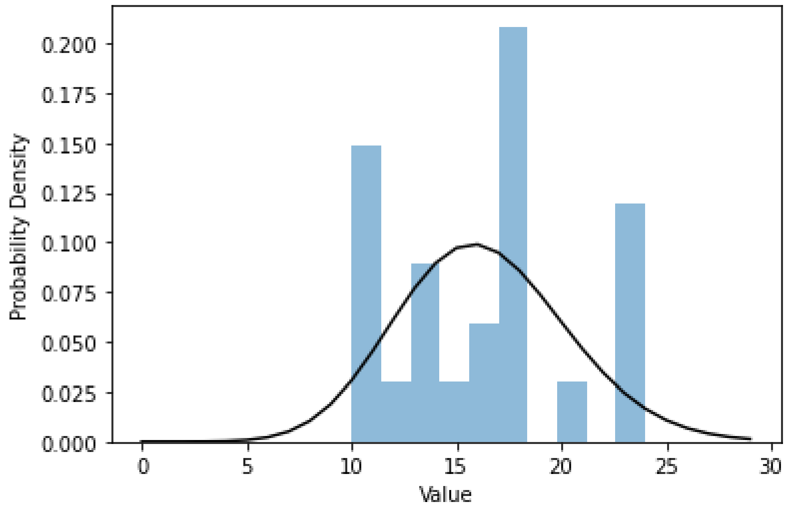
$$L(\lambda) = \prod_{i=1}^{n} \frac{\lambda^{k_i} e^{-\lambda}}{k_i!}$$

$$\log\big(L(\lambda)\big) = \sum_{i=1}^{n} -\lambda + k_i * \log(\lambda) - \log(k_i!)$$

$$\frac{\partial L}{\partial \lambda} = -n + \frac{1}{\lambda} \sum_{i=1}^{n} k_i$$

$$\lambda = \frac{1}{n} \sum_{i=1}^{n} k_i$$

Did it work?



NO!

Visually, we can tell that our model is a bad fit

We might have violated one of the assumptions about the Poisson distribution:

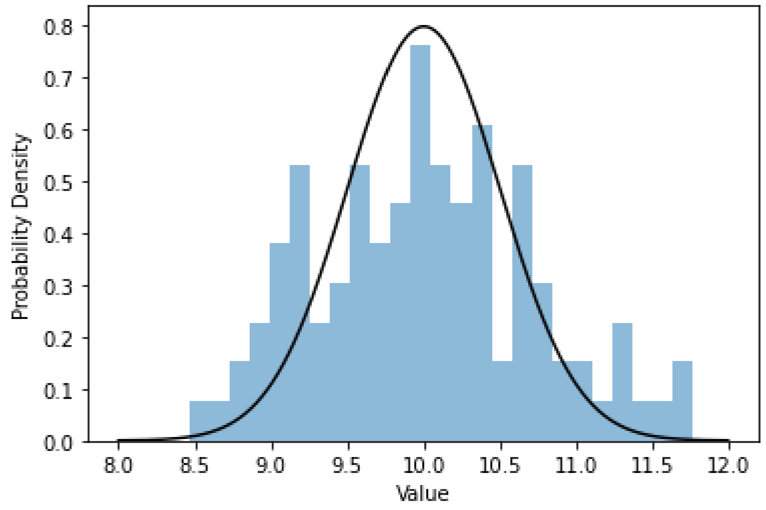
- Our events might not occur independently
- Events might not happen at a constant rate

Hypothesis testing of MLE results

I want to be sure that my cookies are coming out the right size. I would like them to have an average diameter of 10cm and a standard deviation of 0.5cm.

I can use this information to create a normal distribution.

Can we test if this distribution is a good fit?



Hypothesis testing of MLE results

In statistics terms, we want to test a null hypothesis that $\Theta = \Theta_0$

To do this, we calculate the likelihood ratio

$$R_L = \frac{L(\Theta_0)}{L(\Theta_{MLE})}$$

Evidence against the null hypothesis is indicated by higher R_L

Hypothesis testing of MLE results

- The likelihood ratio can be hard to calculate numerically, so we can alternatively use the log-likelihood ratio

$$\log(R_L) = \log\left(\frac{L(\Theta_0)}{L(\Theta_{MLE})}\right)$$

$$= \log\big(L(\Theta_0)\big) - \log\big(L(\Theta_{MLE})\big)$$

Hypothesis testing of MLE results

```
from scipy.stats import norm

# Values of our expected distribution, Theta_0
mean_0 = 10.0
std_0 = 0.5**2

# Values found using MLE, Theta_MLE
n = len(values)
mean_mle = 1/n * sum(values)
std_mle = 1/n * sum((values-mean_mle)**2) ** 0.5

# Likelihood of both distributions
def calculate_loglikelihood(values, mean, std):
    probabilities = [np.log(norm.pdf(x, loc=mean, scale=std)) for x in values]
    return np.sum(probabilities)
loglikelihood_0 = calculate_likelihood(values, mean_0, std_0)
loglikelihood_mle = calculate_likelihood(values, mean_mle, std_mle)

log_RL = loglikelihood_0 / loglikelihood_mle
print(-2*log_RL))
>>> -7.533
```

Assessing Likelihood Ratios

- For independent and normally distributed data, $-2*\log(RL)$ has a $\chi 2$ distribution on 1 degree of freedom for any sample size

- If we have n independent observations from a non-normal population, it can be shown that $-2*\log(RL)$ has a limiting distribution which is $\chi 2$ on 1 degree of freedom as the sample size **n** increases, under the assumption that the null hypothesis is true

- This result can be used to determine approximate p-values

Assessing Likelihood Ratios

```
from scipy.stats import chi2

test_statistic = -2*log_RL
pvalue = 1 - chi2(df=1).cdf(test_statistic)
print(pvalue)
>>> 0.0522
```