# Maximum Likelihood and Kernel Density Estimation

**Prereq (To understand this lecture, you must have already mastered):**

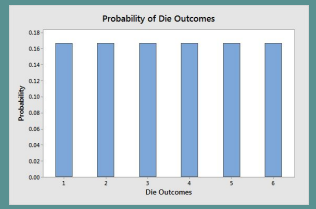
1. Probability basics
2. Expectation and Variance
3. Recognizing the most common distributions

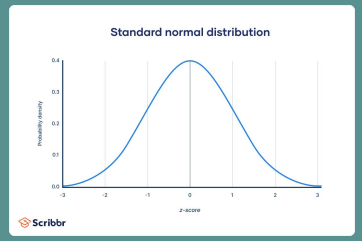
**Topics:**

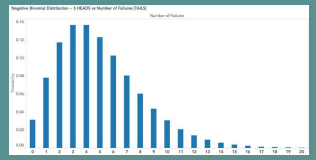
1. Understanding independence
2. IID Joint distributions
3. Maximum Likelihood
4. Kernel Density Estimation

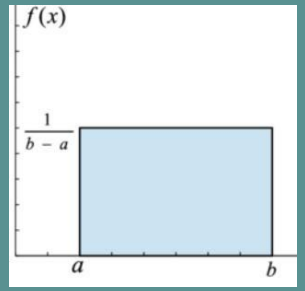
Lecture Written by : Prof. Wu
@ chiehwu.com

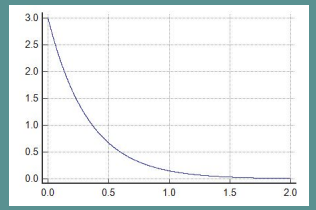
# Previously, we learn how to obtain p(x) for several distributions

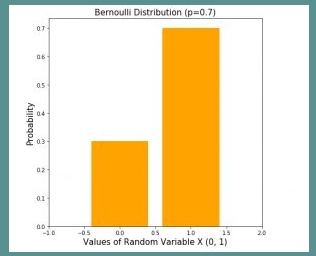


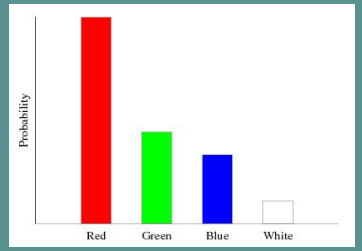






With these distributions, we got lucky because

- Θ is normally just the mean or variance

What what do you do, when you have more very complex distributions where

- The mean or variance is NOT necessarily θ, for example

$$p(x) = ReLU\left[\frac{1}{\sqrt{m}}\sum_{j=1}^{m} a_j[\langle\theta_j, x\rangle]_+\right] \longleftarrow \text{There are } m \text{ number of } \theta\text{s}$$

How are you going to find θ for these cases?

One solution, Maximum Likelihood.

# Independent Variables and how it impacts the joint distribution

Sometimes one variable can influence the outcome of another variable

X : How much you study
Y : Probability of higher exam grade

Sometimes two variable may have nothing to do with each other.

X : how much you workout
Y : will it rain in Alaska today.

We previously learned that the conditional probability is defined as $p(x|y) = \frac{p(x,y)}{p(y)}$, this implies

$$p(x|y) = \frac{p(x,y)}{p(y)} \implies p(x,y) = p(x|y)p(y)$$

However, if knowing the probability of y doesn't influence x at all, then

$$p(x|y) = p(x) \implies p(x,y) = p(x)p(y)$$

When $p(x,y) = p(x)p(y)$, this is called **Independence**.

Under this condition, we would say that x is independent of y

# The common assumption of IID (Independent, Identically distributed.

Often, when we collect data samples, we assume that each sample is independent of the other

For example:

You are collecting blood pressure from 100 people.

You assume that the blood pressure from any person is **independent** from everybody else

This is a reasonable assumption (normally)

- It is possible the 2 people live together and have similar life style
- But most of the time, we can assume they have nothing to do with each other

The assumption of data independence is the basis of most statistical machine learning algorithms, it is call

**Independent, identically distribution ( IID )**

**This implies that each sample in the data came from the same distribution, but are independent from each other.**

# Joint Gaussian Distribution for 1000 samples.

1. Let's say that we are using the Gaussian distribution to model 1000 samples we collected. The probability distribution of a Gaussian follows the equation

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}.$$

2. If there are 1000 samples, which we denote them as $x_1, x_2, ..., x_{1000}$, then the probability of each distinct event would be

$$p(x_1) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_1-\mu)^2}{2\sigma^2}} \quad \text{and} \quad p(x_2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_2-\mu)^2}{2\sigma^2}} \quad \text{and} \quad ... \quad \text{and} \quad p(x_{1000}) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_{1000}-\mu)^2}{2\sigma^2}}$$

   Notice that we can get the probability of each event by plugging $x_1, x_2$, etc individually into the probability distribution.

3. I previously mentioned that we can assume that each event is independent of the other event, which we called **IID**.

4. If all 1000 events are independent of each other, then the joint distribution would be

$$p(x_1, x_2, ..., x_{1000}) = p(x_1)p(x_2)...p(x_{1000}) = \prod_{i=1}^{n} p(x_i).$$

5. Since we don't want to write $p(x_1, x_2, ..., x_{1000})$, every single time, let's set $x = [x_1, x_2, ..., x_{1000}]$, so when we talk about $p(x)$, we know that $x$ is a vector of multiple variables. So for 1000 samples, their joint Gaussian distribution would be

$$p(x) = \prod_{i}^{1000} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}} \tag{1}$$

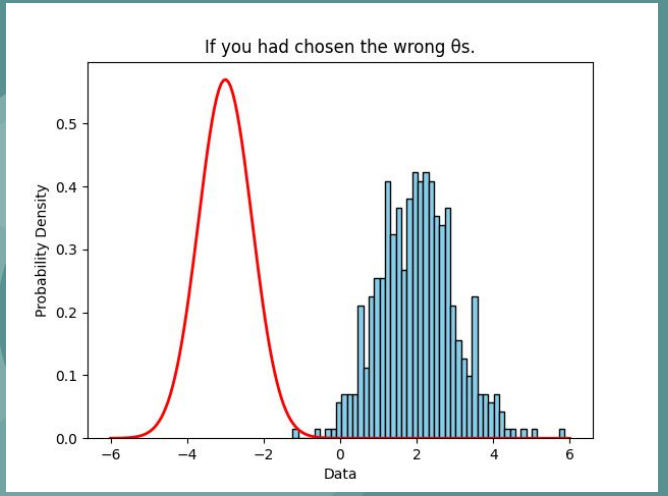
# Result of picking an inaccurate $\theta$.

The joint pdf as defined by the previous slide is

$$p(x) = \prod_{i}^{1000} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}} \tag{1}$$

Now, let's pretend that we picked a really bad choice of $\theta$, so our distribution is way off. What would be the probability of $p(x_1)p(x_2)...p(x_{1000})$?
According to our poorly proposed distribution

- $p(x_1)$ would be very small
- $p(x_2)$ would be very small
- In fact every $p(x_i)$ would be very small
- resulting in an even lower $p(x) = p(x_1)p(x_2)...p(x_{1000})$.
- From this, we see that a bad choice of $\theta$ leads to a very small $p(x)$.



# Result of picking better $\theta$s.

Now, let's pretend that we picked a much better choice of $\theta$, our estimated distribution is now much closer to the truth. What would be the probability of $p(x_1)p(x_2)...p(x_{1000})$?
According to our poorly proposed distribution

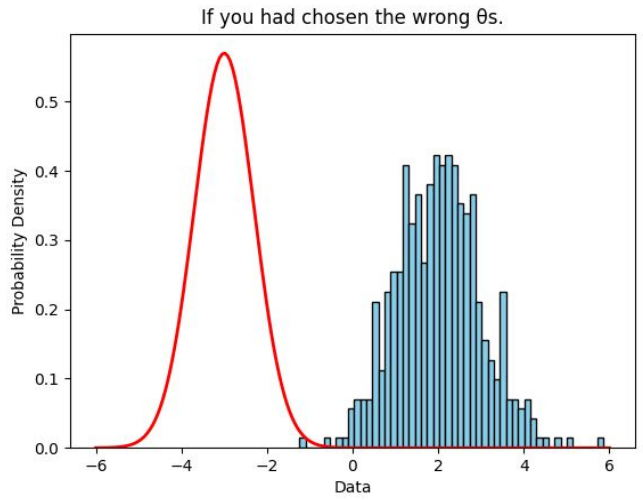
- $p(x_1)$ would be larger
- $p(x_2)$ would be larger
- In fact every $p(x_i)$ would be larger
- resulting in an even larger $p(x) = p(x_1)p(x_2)...p(x_{1000})$.
- From this, we see that a better choice of $\theta$ leads to larger and larger values of $p(x)$.

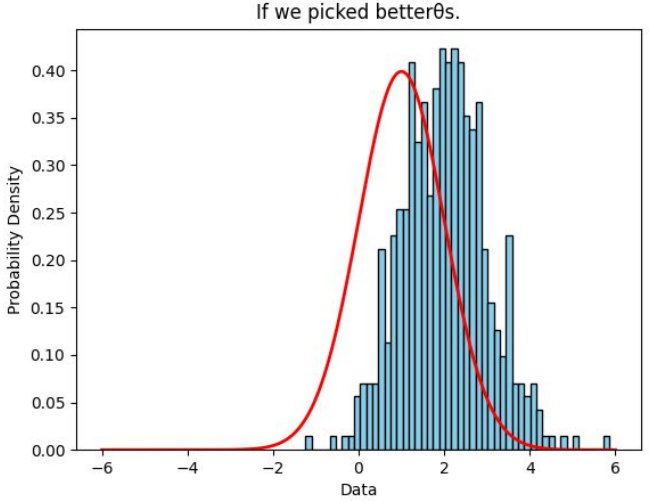
**Maximum Likelihood came from this observation.**

- What if we just pick the $\theta$ that gives us the maximum $p(x)$?
- That if, what if we solve the problem

$$\max_{\theta_1,\theta_2} \quad p_\theta(x) \implies \max_{\theta_1,\theta_2} \prod_i^{1000} \quad \frac{1}{\sqrt{2\pi\theta_2^2}} e^{-\frac{(x_i-\theta_1)^2}{2\theta_2^2}}$$

- Indeed, this is how the problem of maximum likelihood is posed.



# Using Gradient Descent to solve Maximum Likelihood (MLE)

I wrote the code that solves this maximization via gradient descent.

Here is a video showing as gradient descent improve the maximization, we get a better and better estimation of the “true” p(x).

https://www.youtube.com/watch?v=G6MUrbDCaHY

In this case, we just used autograd to calculate the gradient.

# Here is the code I wrote that generates the maximum likelihood example

```python
#!/usr/bin/env python
#
import autograd.numpy as np
from autograd.numpy import sqrt
from autograd import grad
import matplotlib.pyplot as plt
#
# Generate a random sample from a Gaussian distribution
np.random.seed(42)
n = 500
μᵀ = 2.0
σᵀ = 1
2π = 2*np.pi
sample = np.random.normal(μᵀ, σᵀ, n)
#
# Define the log-likelihood function
def gaussian_log_likelihood(params):
    μ, σ = params
    log_likelihood = -n/2 * np.log(2π * σ**2) - (1 / (2 * σ**2)) * np.sum((sample - μ)**2)
    return -log_likelihood  # Negative log-likelihood for minimization
#
# Define the gradient of the log-likelihood fun
gradient = grad(gaussian_log_likelihood)
#
η = 0.001
num_iterations = 50
θ₀ = np.array([0.0, 2.0])  # Initial values for mu and σ
#
θ = θ₀
for i in range(num_iterations):
    ∇f = gradient(θ)
    θ = θ - η*∇f              # Perform gradient descent
    ## Plot the histogram of the data
    [μ, σ] = θ
    plt.hist(sample, bins=50, density=True, color='skyblue', edgecolor='black')

    # Plot the probability density function (PDF) of the normal distribution
    x = np.linspace(-4, 5, 100)
    y = 1 / (σ * sqrt(2π)) * np.exp(-(x - μ)**2 / (2 * σ**2))
    plt.plot(x, y, color='red', linewidth=2)

    # Set plot title and labels
    plt.title('Normal Distribution')
    plt.xlabel('Data')
    plt.ylabel('Probability Density')
    plt.savefig('./video_frames/img_' + str(i).zfill(3) )
    plt.clf()
```

Notice that the estimated and the true parameters are very close

$$p(x_1, x_2, ..., x_n) = p(x_1)p(x_2)...p(x_n) = \prod_{i=1}^{n} p(x_i)$$

Automatically calculating the gradient

$$\max_{\theta} \prod_{i=1}^{n} p(x_i|\theta)$$

Extract the estimated parameters

```python
μ_estimated, σ_estimated = θ
#
# Print the estimated parameters
print("Estimated μ:", μ_estimated)
print("Estimated σ:", σ_estimated)
#
```

```
Estimated μ: 2.006837994588541
Estimated σ: 0.9802715029966268
```

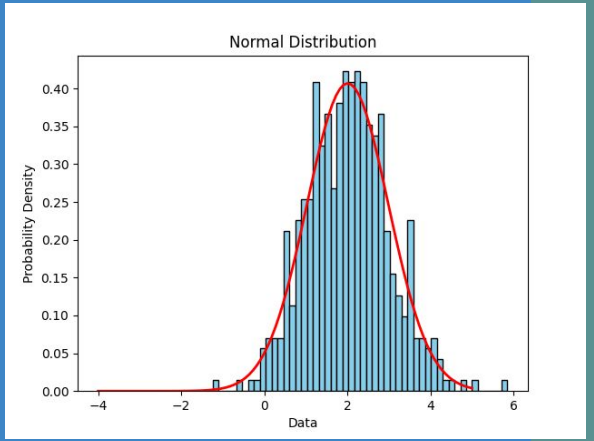
$$\underbrace{\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\theta_1)^2}{2\theta_2^2}}}_{\text{Gaussian Distribution}}$$

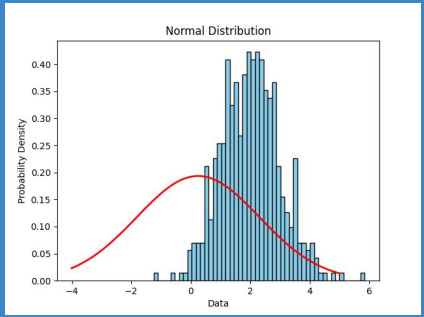
# In summary

1. You have some data with a certain distribution p(x)

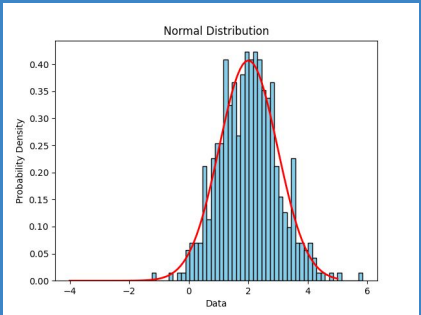
2. You don't know p(x), but you can
   a. Use the mean and variance method if you recognize it as a simple and well known distribution.
   b. Use MLE and find the θ that makes the data maximally likely.

3. If you use Maximum likelihood, you can solve it 2 ways
   a. If the function is simple,
      you can calculate the gradient and set it to 0 that gives you the best θ. (Closed-form solution)
   b. If the function is complicated
      You can use gradient descent to approximate the value of θ
      You don't even need to know how to calculate the derivative (numpy does it for you)




Gradient Descent




We showed a 1D example, but MLE works on higher dimensions too !!

# Functions of machine learning algorithms.

Given $z \in \mathbb{R}, x, y, w \in \mathbb{R}^d, A \in \mathbb{R}^{d \times d}$

1. Linear: $f(x) = y^\top x$
2. Quadratic:
   $f(x) = x^\top A x$
3. Trace of Quadratic:
   $f(x) = \text{Tr}(x^\top A x)$
4. Activation Function:
   $f(w) = \text{ReLU}(w^\top x)$
5. Linear Regression:
   $f(w) = \frac{1}{2} \sum_{i=1}^{n} (w^\top x_i - y_i)^2$
6. Multivariate Gaussian Tr:
   $f(x) = e^{-\text{Tr}(x^\top A x)}$
7. Multivariate Gaussian:
   $f(x) = e^{-x^\top A x}$
8. Sigmoid Function:
   $f(z) = \frac{1}{1+e^{-z}}$
9. Logistic Regression Objective:
   $f(w) = \frac{1}{1+e^{-w^\top x}}$
10. L1 Norm:
    $f(x) = ||x||_1$.
11. L2 Norm:
    $f(x) = ||x||_2$.
12. SVM objective:
    $f(w) = \sum_{i=1}^{n} \text{ReLU}(1 - y_i \langle x_i, w \rangle)$
13. PCA objective:
    $f(x) = x^\top A x - \lambda(x^\top x - 1)$

Maximum Likelihood Objectives

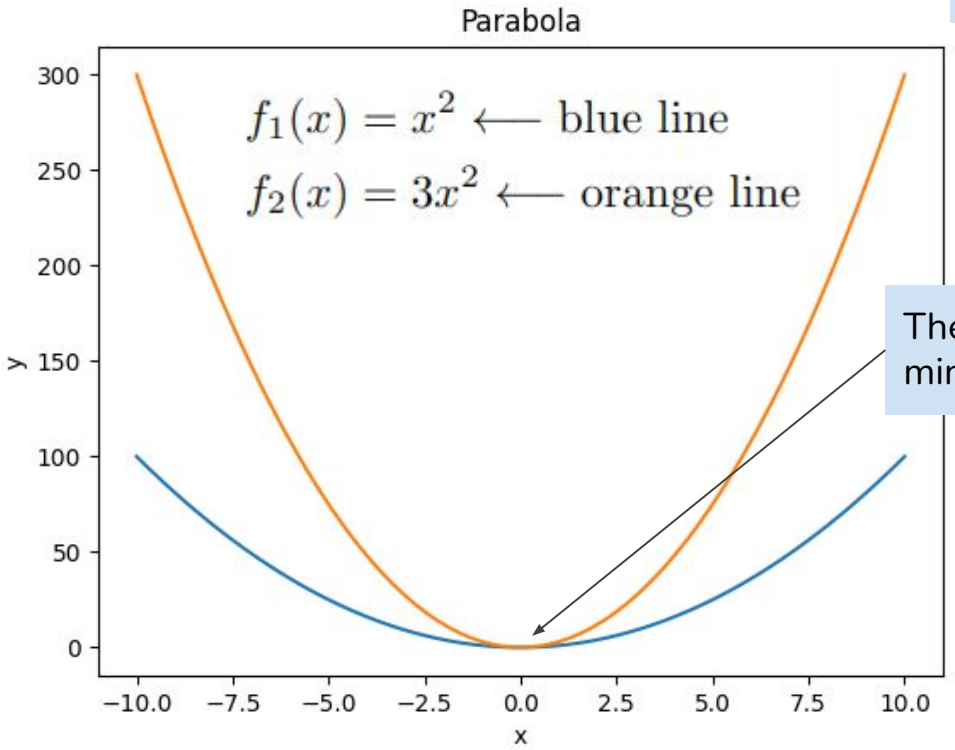
14. Gaussian Maximum Likelihood:
    $\mu \in \mathbb{R}$
    $f(\mu) = \log\left(\prod_i^n \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}\right)$
15. Exponential Maximum Likelihood:
    $\lambda \in \mathbb{R}$
    $f(\lambda) = \log\left(\prod_i^n \lambda e^{-\lambda x_i}\right)$
16. Bernoulli Maximum Likelihood:
    $\alpha$ can only be 1 or 0, $\alpha \in \{0, 1\}$
    $p \in \mathbb{R}$
    $f(p) = \log\left(\prod_i^n p^{\alpha_i}(1-p)^{1-\alpha_i}\right)$
17. Uniform Maximum Likelihood:
    $a, b \in \mathbb{R} : a \leq b$
    $f(a, b) = \log\left(\prod_i^n \frac{1}{b-a}\right)$

- You don't need to know the usage of these functions for now.
- You just need to know how to take their derivative.
- The purpose of this class is to understand the purpose of these functions.
- We will go over some of them together.

# Part 2:

# Tricks of optimization

# Review of minimization rule min f(x) = min α f(x)



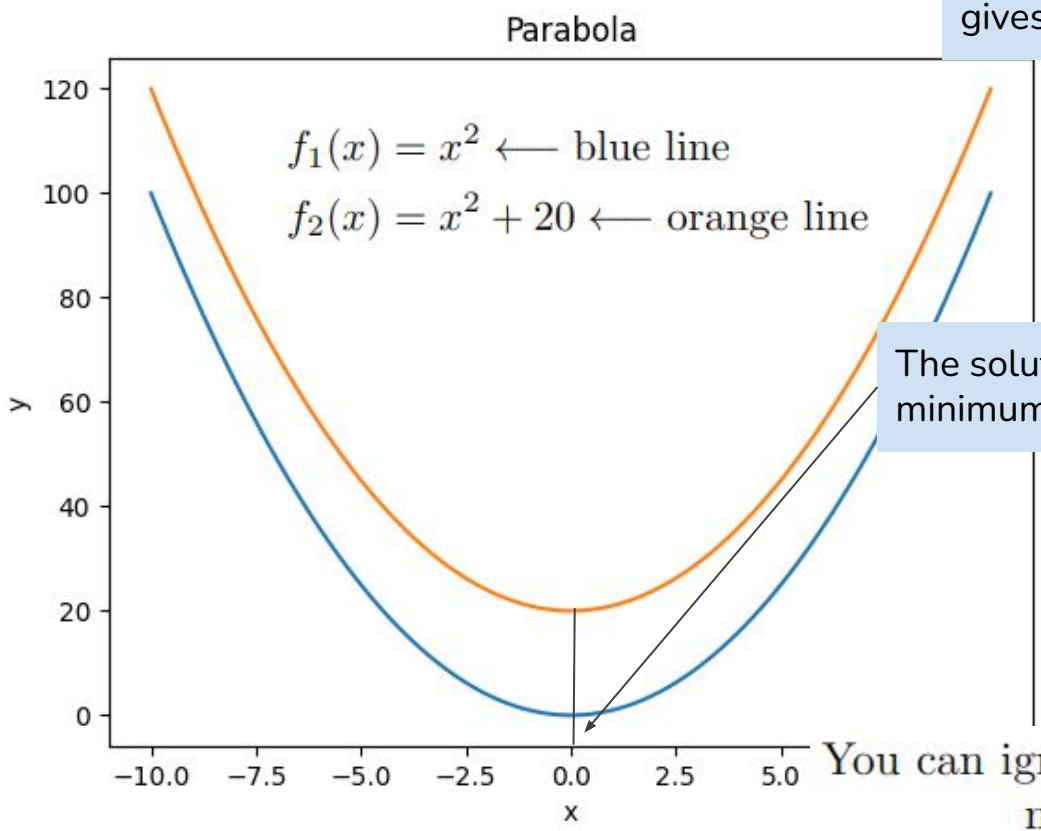

Multiplying a constant to a optimizing function still gives the same optimal solution x

The solution for the x that gives minimum value is exactly the same

You can ignore the constant during optimization

$$\min_x \ \alpha f(x) \implies \min_x \ f(x)$$

# Review of minimization rule min f(x) + c = min f(x)



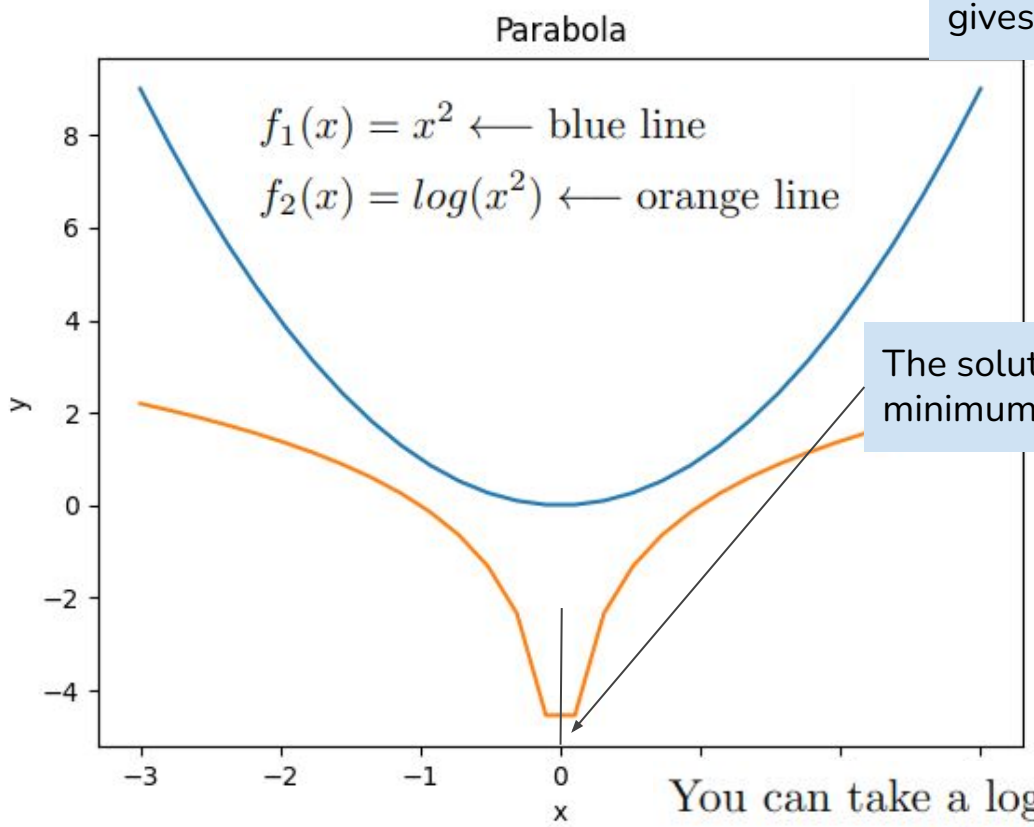

Add a constant to a optimizing function still gives the same optimal solution x

The solution for the x that gives minimum value is exactly the same

You can ignore the constant during optimization

$$\min_x f(x) + c \implies \min_x f(x)$$

# Review of minimization rule min f(x) = min log(f(x))




Taking a log of the optimizing function still gives the same optimal solution x

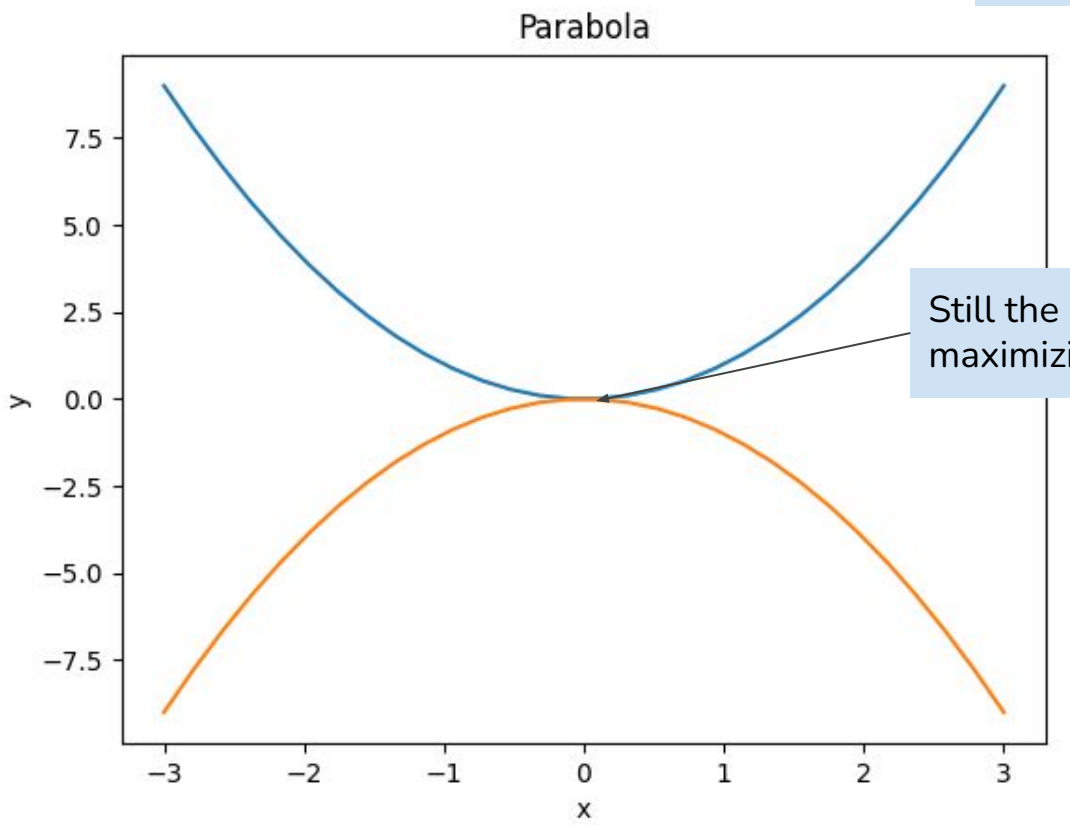
The solution for the x that gives minimum value is exactly the same

You can take a log of the function during optimization

$$\min_x f(x) \implies \min_x log(f(x))$$

# Review of minimization rule max f(x) = min - f(x)

Minimizing the negative of a function is the same as optimizing the positive of a function




Still the same solution, just now a maximizing problem

Most software assumes minimization. So if you have a maximization problem, you can convert it to minimization.

$$\max_x f(x) \implies \min_x -f(x)$$

Review of log rules

# Log Rules and Tricks

Log of exponential is just 1

$$log(e) = 1 \tag{1}$$

Multiplication of variables in log is equal to addition

$$log(xy) = log(x) + log(y) \tag{2}$$
$$log(xyz) = log(x) + log(y) + log(z) \tag{3}$$

Division of variables in log is equal to subtraction

$$log(\frac{x}{y}) = log(x) - log(y) \tag{4}$$
$$log(\frac{x}{yz}) = log(x) - log(y) - log(z) \tag{5}$$

Power is moved to the front

$$log(x^n) = n\, log(x) \implies log(e^x) = x \tag{6}$$

# Practice Solving Maximum Likelihood Problem

Let's say we want to perform maximum likelihood given that each sample came from an exponential distribution.

$$p(x) = \theta e^{-\theta x}.$$

1. What is then the IID joint distribution we want to maximize for Maximum Likelihood?
2. Now maximize this problem using the closed-form solution.
    - This is where you take the derivative and set $\theta = 0$.
    - Remember to use all the log properties you have learned.
3. From the last class, do you remember what $\theta$ was supposed to be?
4. How does your answer differ from the what $\theta$ was supposed to be?

# MLE for exponential distribution

The exponential distribution for a single sample is

$$p(x|\theta) = \theta e^{-\theta x}$$

Given $n$ samples, the IID joint distribution becomes

$$p(x_1, x_2, ..., x_n) = p(x_1)p(x_2)...p(x_n) = \prod_{i=1}^{n} p(x_i) = \prod_{i=1}^{n} \theta e^{-\theta x_i}$$

In maximum likelihood, we are going to maximize the joint distribution with $\theta$ as the variable

$$\begin{aligned}
\max_{\theta} \prod_{i=1}^{n} \theta e^{-\theta x_i} &= \max_{\theta} \left[\theta e^{-\theta x_1}\right]\left[\theta e^{-\theta x_2}\right] ... \left[\theta e^{-\theta x_n}\right] \\
&= \max_{\theta} \theta^n e^{-\sum_{i=1}^{n} \theta x_n} = \max_{\theta} log\{\theta^n e^{-\sum_{i=1}^{n} \theta x_n}\} \\
&= \max_{\theta} log(\theta^n) + log\left(e^{-\sum_{i=1}^{n} \theta x_n}\right) \\
&= \max_{\theta} n\, log(\theta) - \sum_{i=1}^{n} \theta x_n
\end{aligned}$$

Now we can take the derivative and set it to 0

$$\frac{d}{d\theta}\{n\, log(\theta) - \sum_{i=1}^{n} \theta x_n\} = \frac{n}{\theta} - \sum_{i=1}^{n} x_n = 0$$

this results in

$$\begin{aligned}
\frac{n}{\theta} - \sum_{i=1}^{n} x_n &= 0 \\
\frac{n}{\theta} &= \sum_{i=1}^{n} x_n \\
\frac{1}{\theta} &= \frac{1}{n} \sum_{i=1}^{n} x_n = \mathbb{E}[X]
\end{aligned}$$

$$\mathbb{E}[X] = \int_0^{\infty} x\, \theta e^{-\theta x}\, dx = \frac{1}{\theta}$$

In the previous lecture, we came to the exact same conclusion by finding the expectation of the distribution.

This is a very good practice of all the optimization rules, we just covered.

## Log Rules and Tricks

Log of exponential is just 1

$$log(e) = 1 \tag{1}$$

Multiplication of variables in log is equal to addition

$$log(xy) = log(x) + log(y) \tag{2}$$
$$log(xyz) = log(x) + log(y) + log(z) \tag{3}$$

Division of variables in log is equal to subtraction

$$log(\frac{x}{y}) = log(x) - log(y) \tag{4}$$
$$log(\frac{x}{yz}) = log(x) - log(y) - log(z) \tag{5}$$

Power is moved to the front

$$log(x^n) = n\, log(x) \implies log(e^x) = x \tag{6}$$

You can follow these same steps for most of the simple distributions, i.e., Gaussian, poisson, binomial, etc.

**Question 4.** (100 points) From Class Data in Canvas, find and load the file

`time_until_phone_drop_exam_4k.csv`

This file consists of 1000 individuals recording the time they first dropped their phone(resulting in some damage). The time is in terms of years, so the number 1.6 is equivalent to 1.6 years.

1) Look at the histogram and identify the distribution as $p_1(x)$ by matching it to a known distribution.
2) Identify $p_2(x)$ by maximizing the log-likelihood of its joint distribution **using gradient descent**.
3) Plot out $p_1(x)$ and $p_2(x)$ side by side along with the histogram.
4) The phone company you work for wants to know when is a good time to release the next phone. The manager has identified that the best release time is when 90% of the customers have dropped and damaged their phones. Based on your manager's assessment, when would you recommend releasing the next phone using $p_1(x)$?

```
# Note: use this command to help you
from scipy.stats import expon
expon.ppf(area, loc=0, scale= mean)
```

Really try to think through it first.
Only look at my code if you are stuck (this is only a couple of line): link

# Part 3:

# Kernel Density Estimation

# Kernel Density Estimation

Basic and commonly used Methods:

- Histograms
- Via the mean and variance
- Maximum Likelihood
- Kernel Density Estimation
- Method of moments
- MAP
- Bayesian Estimation
- And a lot more .....

We previously saw some of the basic ways to estimate p(x).
So far, we have covered 3.

**We are now ready to cover the 4th method.**

Kernel Density Estimation or KDE

# Kernel Density Estimation

Basic and commonly used Methods:

- Histograms
- Via the mean and variance
- Maximum Likelihood
- Kernel Density Estimation
- Method of moments
- MAP
- Bayesian Estimation
- And a lot more .....

The mean and variance happens to be θ

MLE allows you to find θ, if the function is complex and mean and variance is not necessarily θ

$$p(x) = ReLU\left[\frac{1}{\sqrt{m}}\sum_{j=1}^{m} a_j[\langle \theta_j, x\rangle]_+\right] \longleftarrow \text{There are } m \text{ number of } \theta\text{s}$$

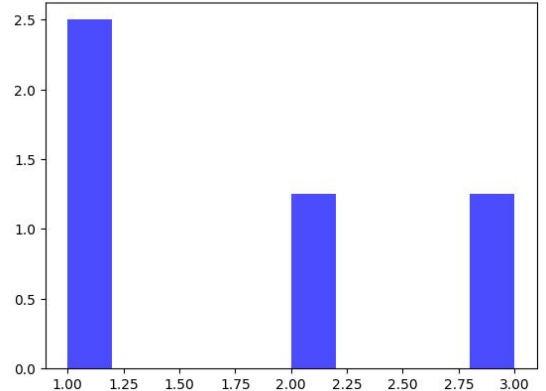
But what if you don't even know what the function look like. And you need to come up with a uncommon p(x)?

This is when KDE is useful

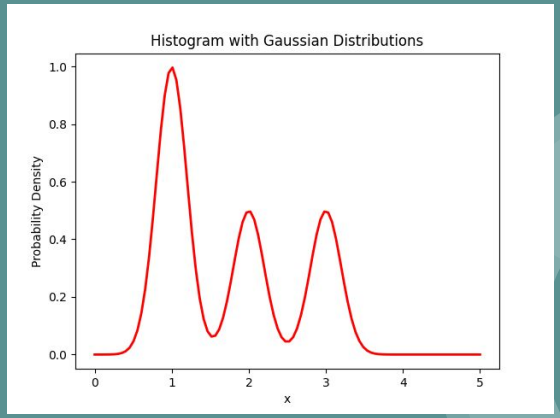
Plus KDE is super simple, it is just histogram with Gaussians

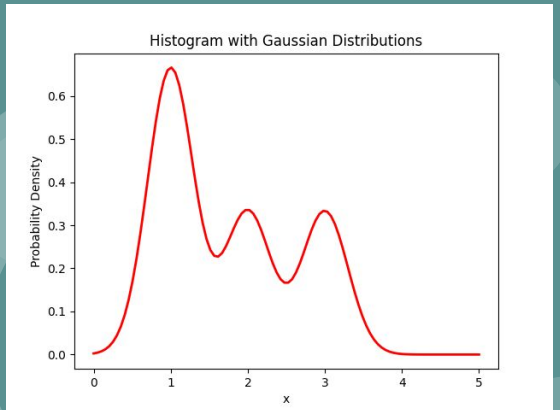
# Kernel Density Estimation

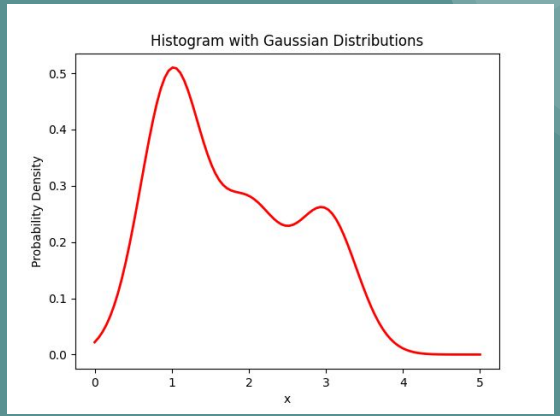
```
#!/usr/bin/env python
#
import numpy as np
import matplotlib.pyplot as plt
#
x = np.array([1,1,2,3])
n, bins, patches = plt.hist(x, 10, facecolor='blue', alpha=0.7, density=True)
plt.show()
```



| x |
|---|
| 1 |
| 1 |
| 2 |
| 3 |



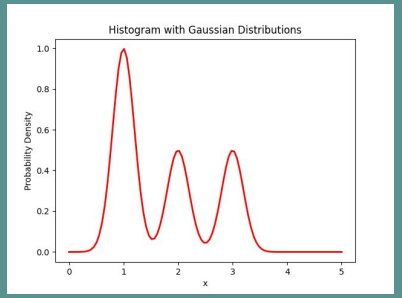

Basically, instead of adding a block to a bar, you would add a Gaussian distribution.

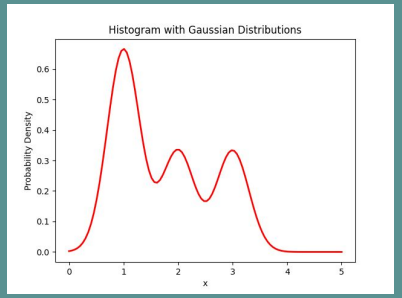
$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\theta_1)^2}{2\theta_2^2}}$$

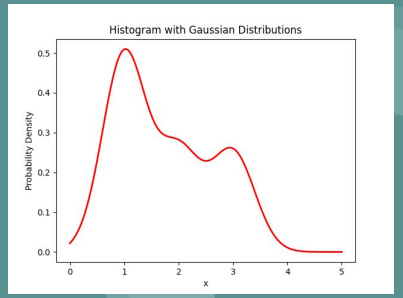
θ1 is the location, θ2 is the width

# Kernel Density Estimation

| x |
|---|
| 1 |
| 1 |
| 2 |
| 3 |




Regular Gaussian of 1 sample

$$p(x) = \frac{1}{\theta_2\sqrt{2\pi}} e^{-\frac{(x-\theta_1)^2}{2\theta_2^2}}$$

KDE from 4 samples

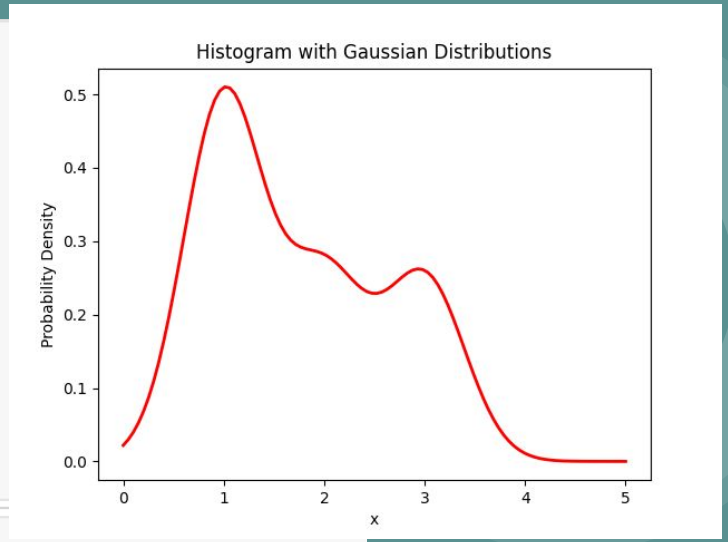
$$p(x) = \frac{1}{4}\left[\frac{1}{\theta_2\sqrt{2\pi}} e^{-\frac{(x-1)^2}{2\theta_2^2}} + \frac{1}{\theta_2\sqrt{2\pi}} e^{-\frac{(x-1)^2}{2\theta_2^2}} + \frac{1}{\theta_2\sqrt{2\pi}} e^{-\frac{(x-2)^2}{2\theta_2^2}} + \frac{1}{\theta_2\sqrt{2\pi}} e^{-\frac{(x-3)^2}{2\theta_2^2}}\right]$$

The larger θ2, the smoother the function. You can also learn θ2 using MLE.

# Kernel Density Estimation

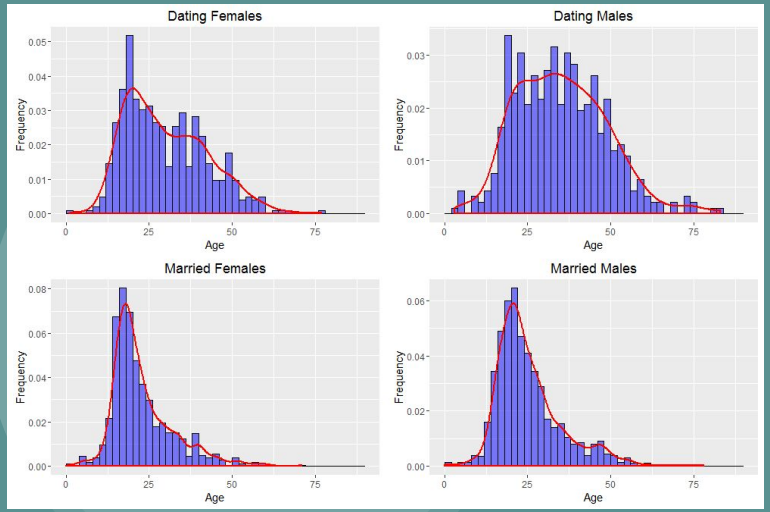
```python
#!/usr/bin/env python
#
import numpy as np
import matplotlib.pyplot as plt
#
# Set the random seed for reproducibility
np.random.seed(0)
2π = 2*np.pi
#
# Parameters for the first Gaussian distribution
μ1 = 1  # Mean
σ1 = 0.4  # Standard deviation
#
μ2 = 2  # Mean
σ2 = 0.4  # Standard deviation
#
μ3 = 3  # Mean
σ3 = 0.4  # Standard deviation
#
```

```python
def gaussian(μ, σ, x):
    return np.exp(-0.5 * ((x - μ) / σ)**2) / (σ * np.sqrt(2π))
#
# Generate x-axis values for the PDF plot
x = np.linspace(0, 5, 100)
#
# Compute the PDF values for the mixture of two Gaussian distributions
pdf = (gaussian(μ1, σ1, x) + gaussian(μ1, σ1, x) + gaussian(μ2, σ2, x) + gaussian(μ3, σ3, x))/4
#
# Plot the PDF of the mixture distribution
plt.plot(x, pdf, 'r-', linewidth=2)
#
# Set plot title and labels
plt.title('Histogram with Gaussian Distributions')
plt.xlabel('x')
plt.ylabel('Probability Density')
#
# Display the plot
plt.show()
#
```



# Histogram and KDE

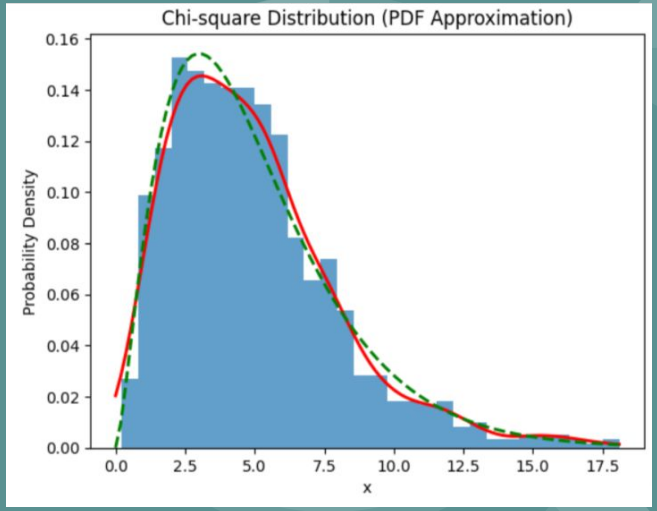
- Once you learned how to plot histograms and perform KDE
- You will be able to generate all sorts of interesting data
- This data is based on a Standard University Research
- It asks 4 populations when they first met their partner.




- This data is based on a research study by Standford University.
- The project is called “How Couples Meet and Stay Together”
- The link can be found at: https://data.stanford.edu/hcmst
- The image was found at: http://ritvikmath.com/Relationship_Math/

In class: write the code that performs the MLE on the following data.

```python
#!/usr/bin/env python
#
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import chi2, gaussian_kde
#
# Set the random seed for reproducibility
np.random.seed(0)
#
# Parameters for the chi-square distribution
df = 5  # Degrees of freedom
#
# Generate random samples from the chi-square distribution
num_samples = 1000
samples = np.random.chisquare(df, num_samples)
```




1. Use the above code to generate ch2 distribution
2. Use Exponential distribution to fit this data. Did it do a good job?
3. Use Gaussian distribution to fit this data. Did it do a good job?
4. Use KDE to approximate it.
5. KDE should end up looking like the red line on the right.

You can find my code here.

# End of the day...

**Remember to submit your In-class work**

Drawn by AI @ https://lexica.art/