Random Variable – Variable whose value is determined by a random experiment.

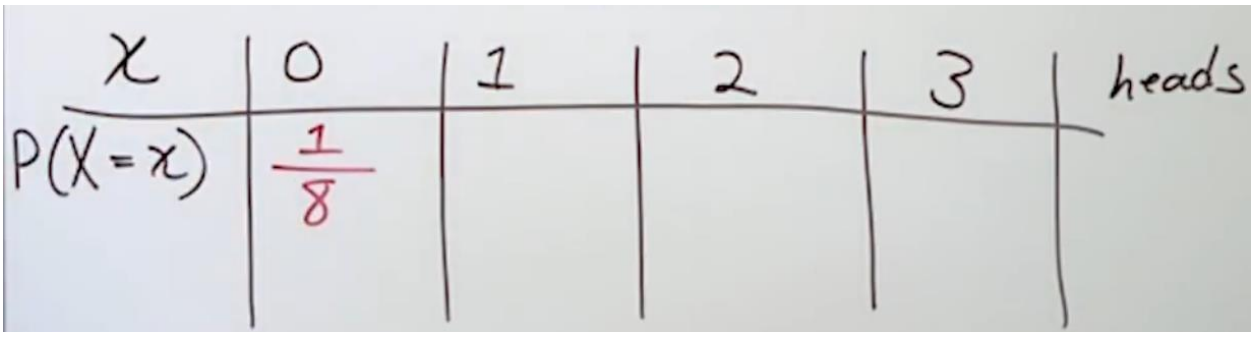
Discrete Prob Dist – Table or Formula that lists the probabilities for each outcome of the random variable, X.

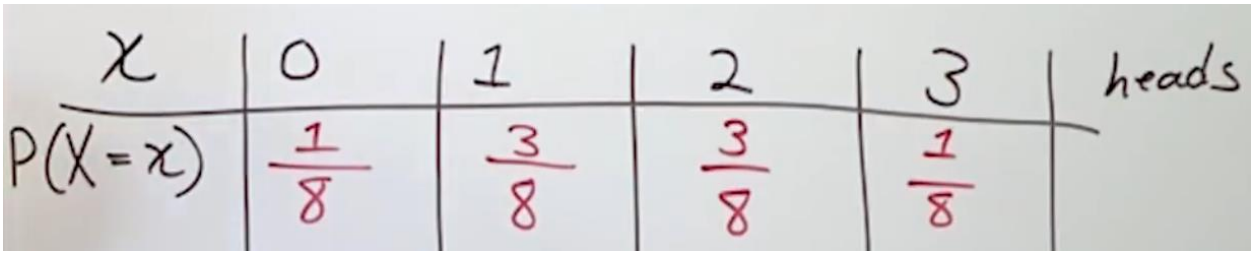
Ex) Flip 3 coins at same time.
Let random variable X be # heads showing.

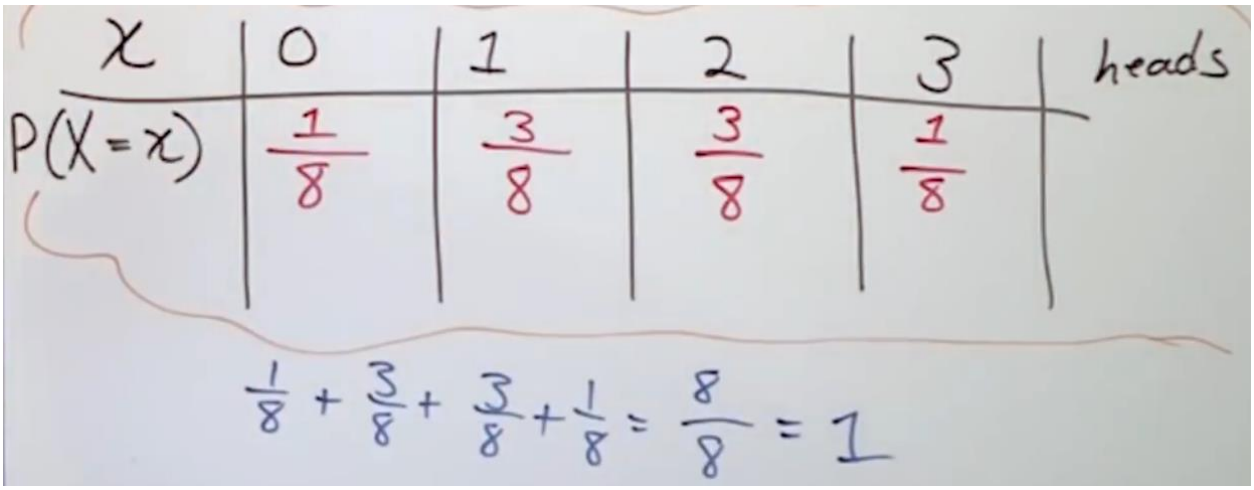
Discrete Prob Dist

HHH	→	3 heads
HHT	→	2 heads
HTH	→	2 heads
HTT	→	1 head
THH	→	2 heads
THT	→	1 head
TTH	→	1 head
TTT	→	0 heads

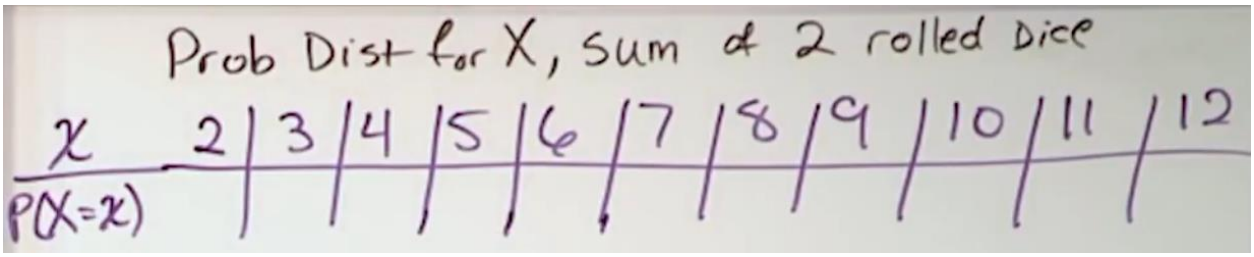
X	0	1	2	3	heads
P(X=x)					

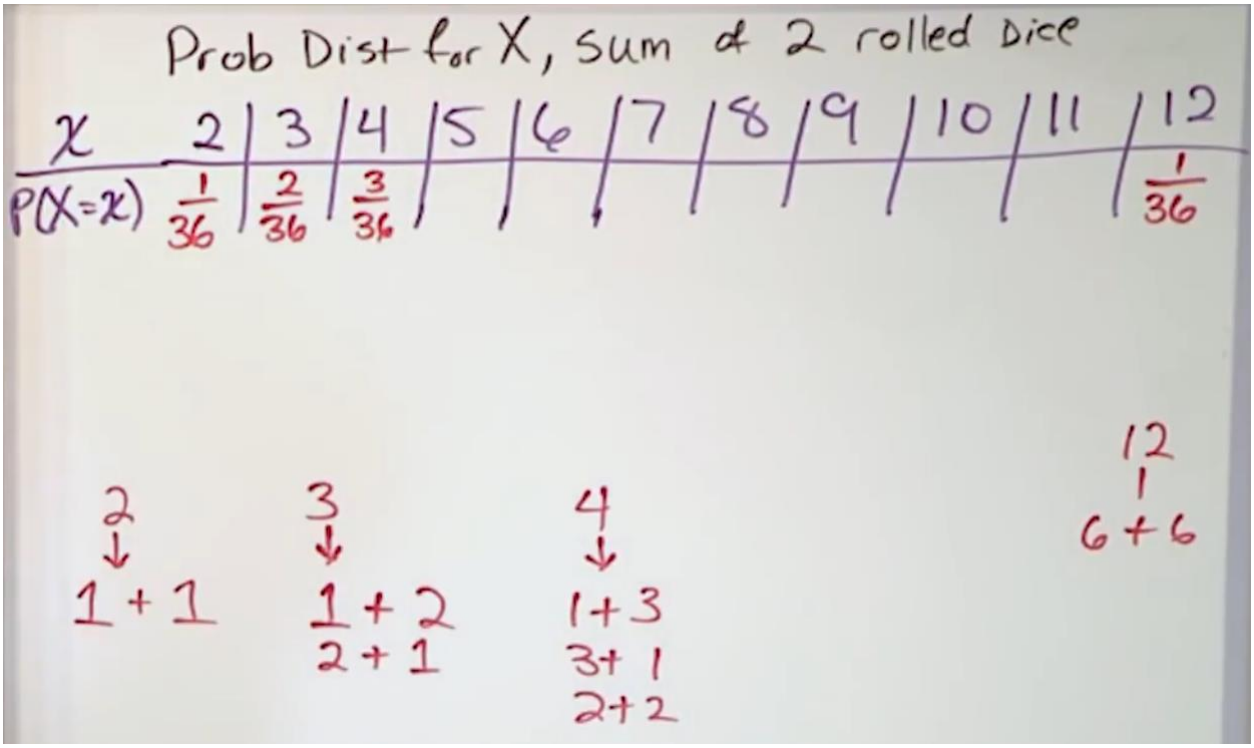

x
0
1
2
3
heads
P(X = x)
1/8

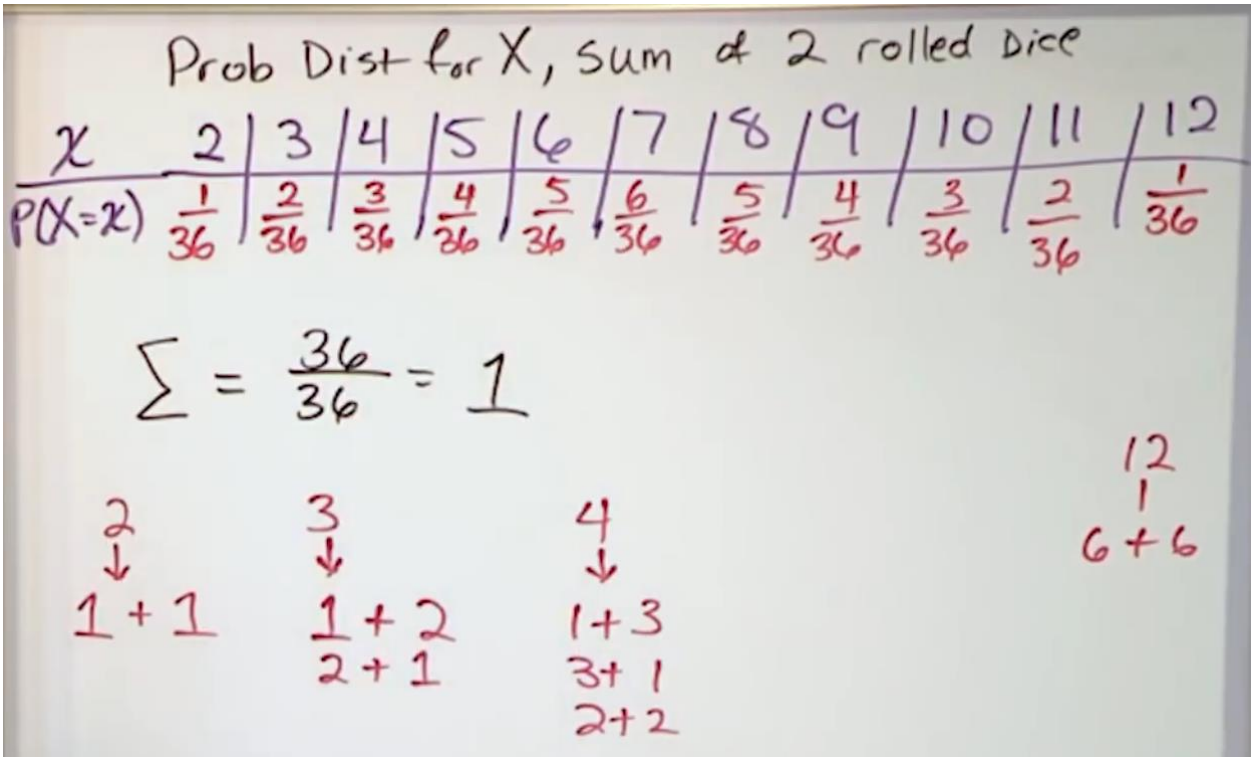

x
0
1
2
3
heads
P(X = x)
1/8
3/8
3/8
1/8

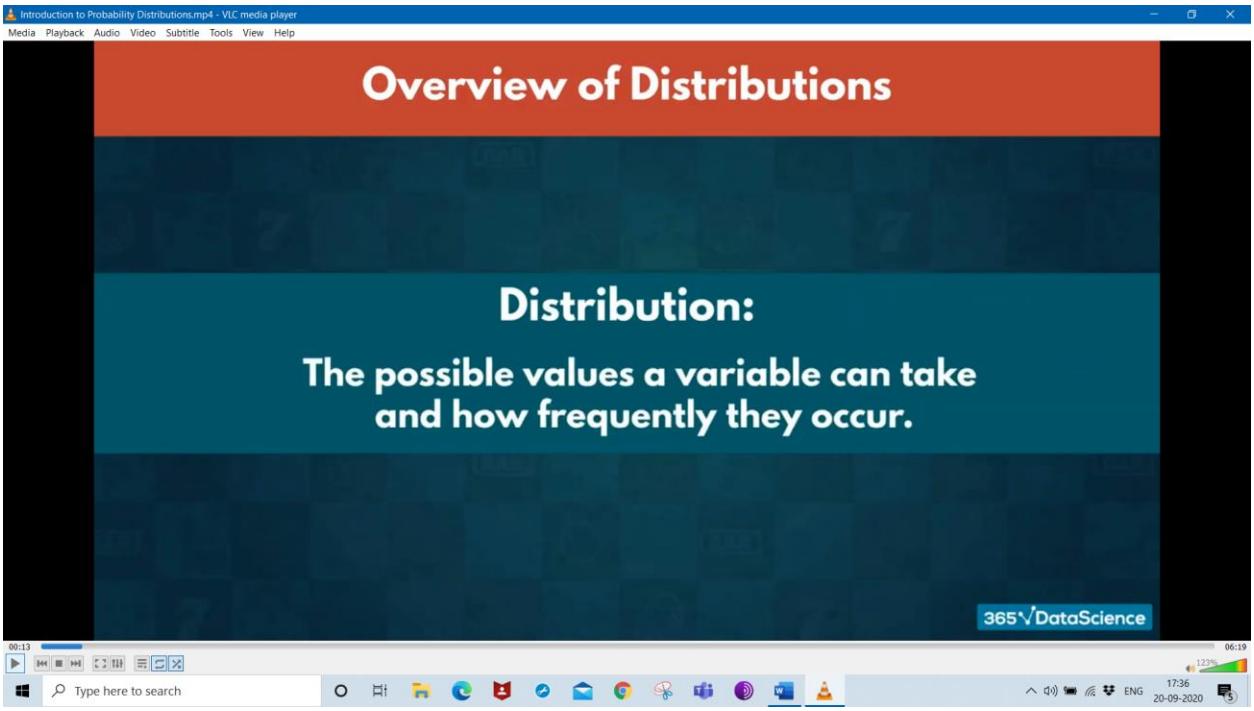

x
0
1
2
3
heads
P(X = x)
1/8
3/8
3/8
1/8
1/8 + 3/8 + 3/8 + 1/8 = 8/8 = 1

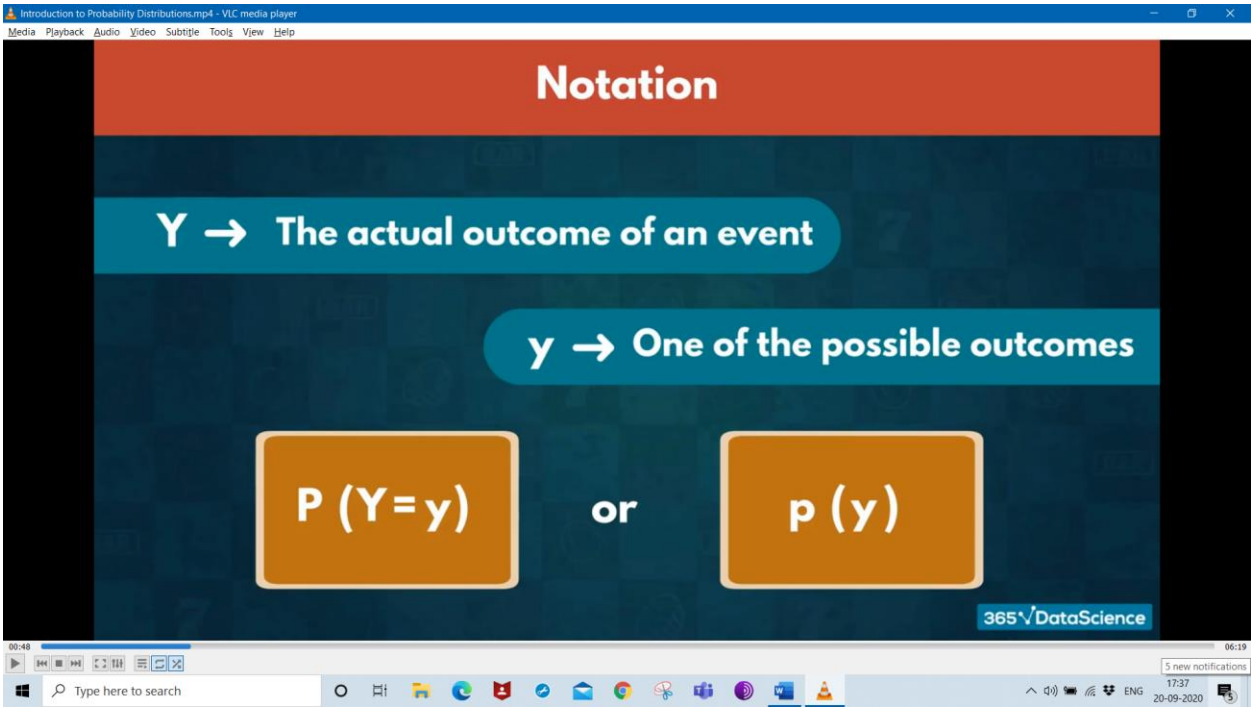
Prob Dist for X, sum of 2 rolled Dice

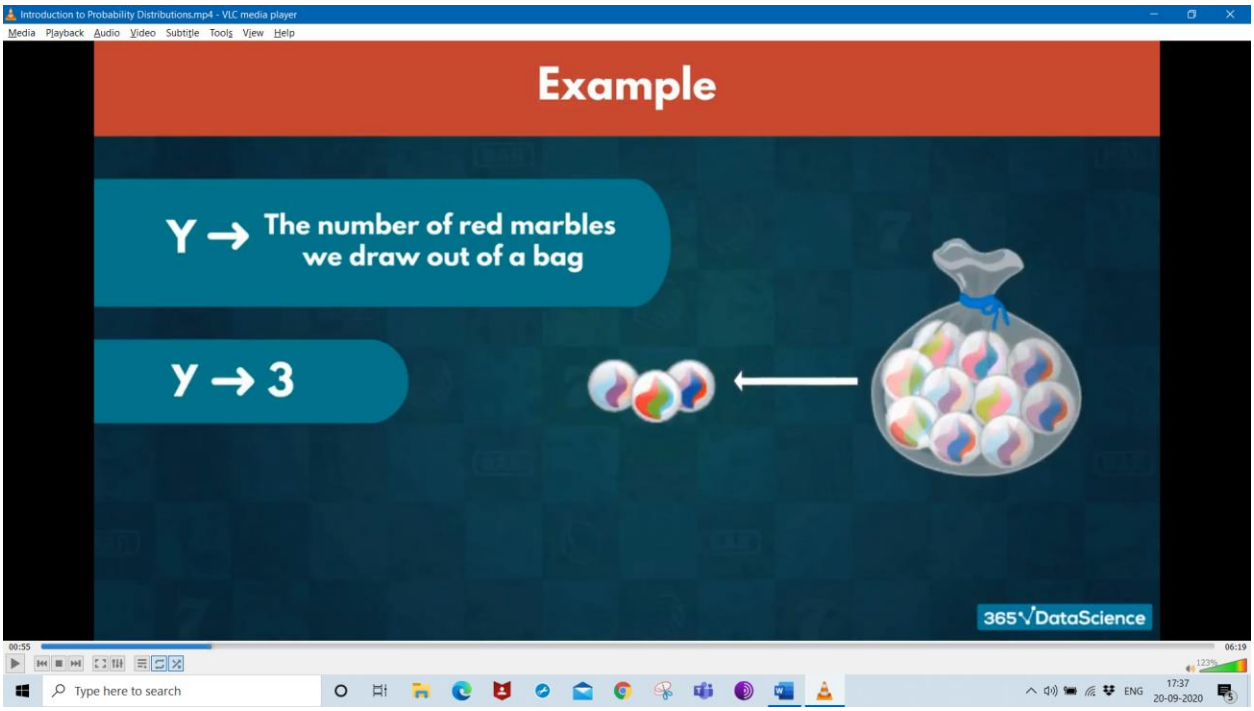


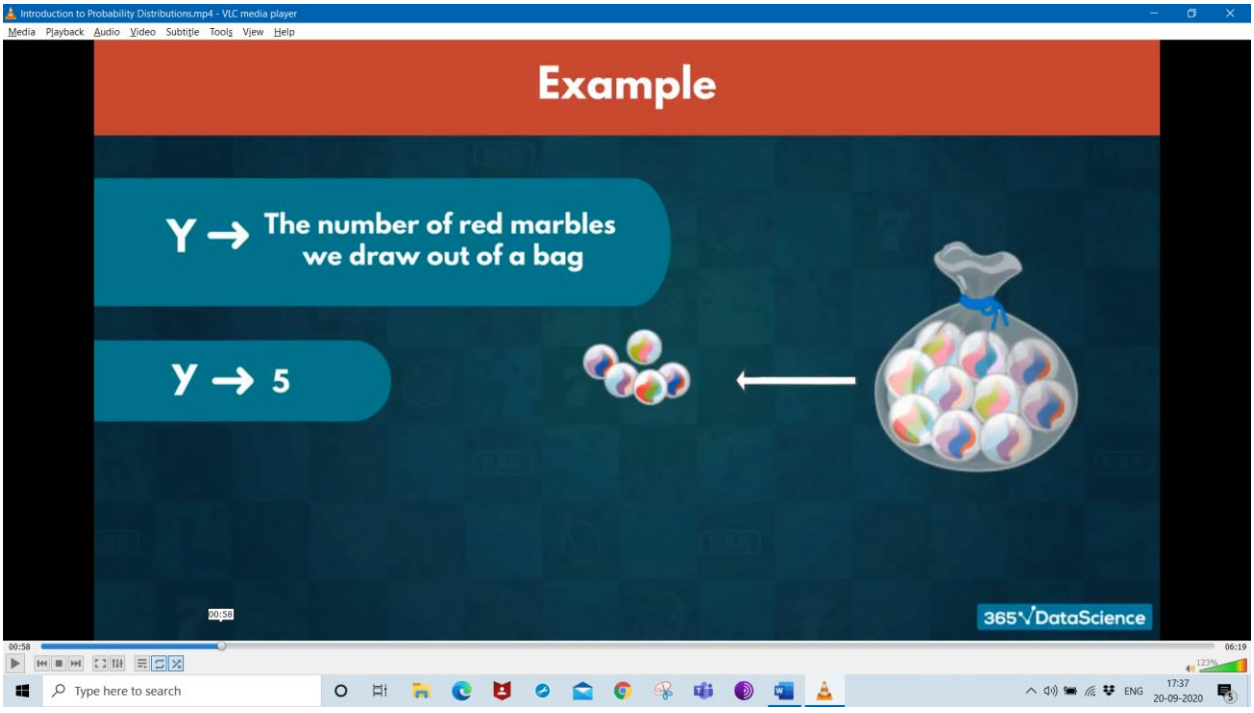


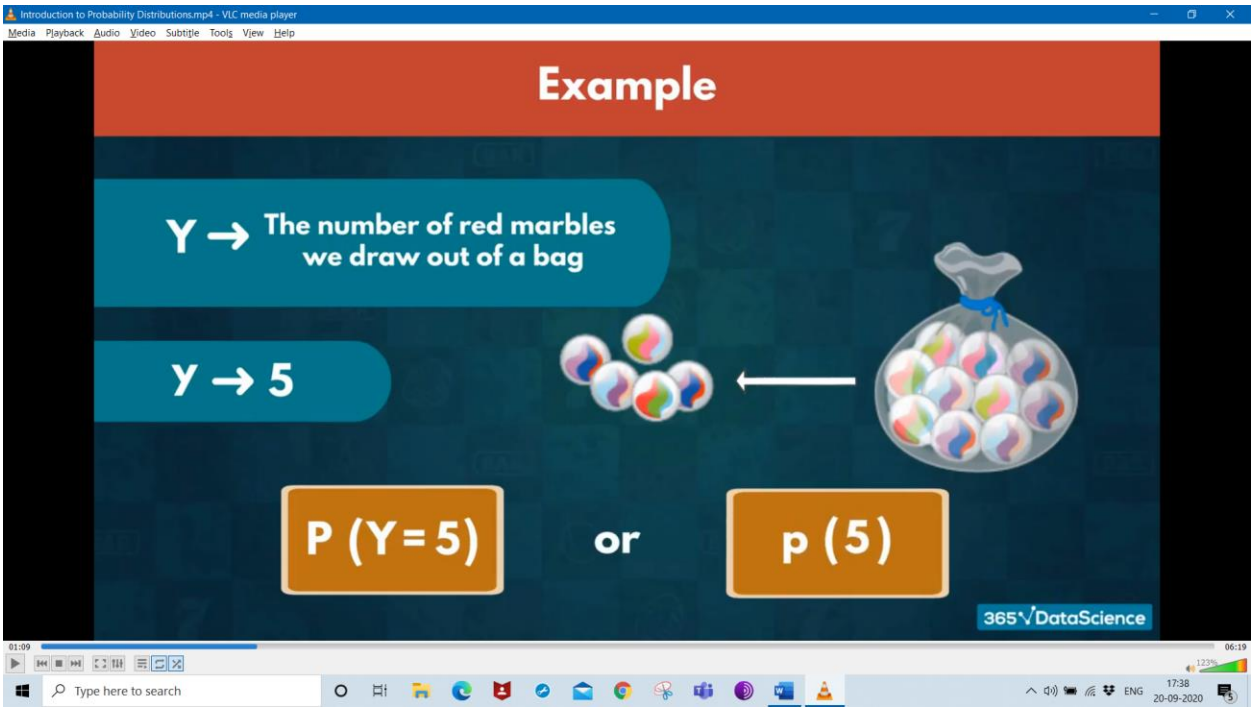

Prob Dist for X, sum of 2 rolled dice
X 2 3 4 5 6 7 8 9 10 11 12
P(X=x) 1/36 2/36 3/36 4/36 5/36 6/36 5/36 4/36 3/36 2/36 1/36
∑ = 36/36 = 1
2
1 + 1
3
1 + 2
2 + 1
4
1+3
3+1
2+2
12
6+6

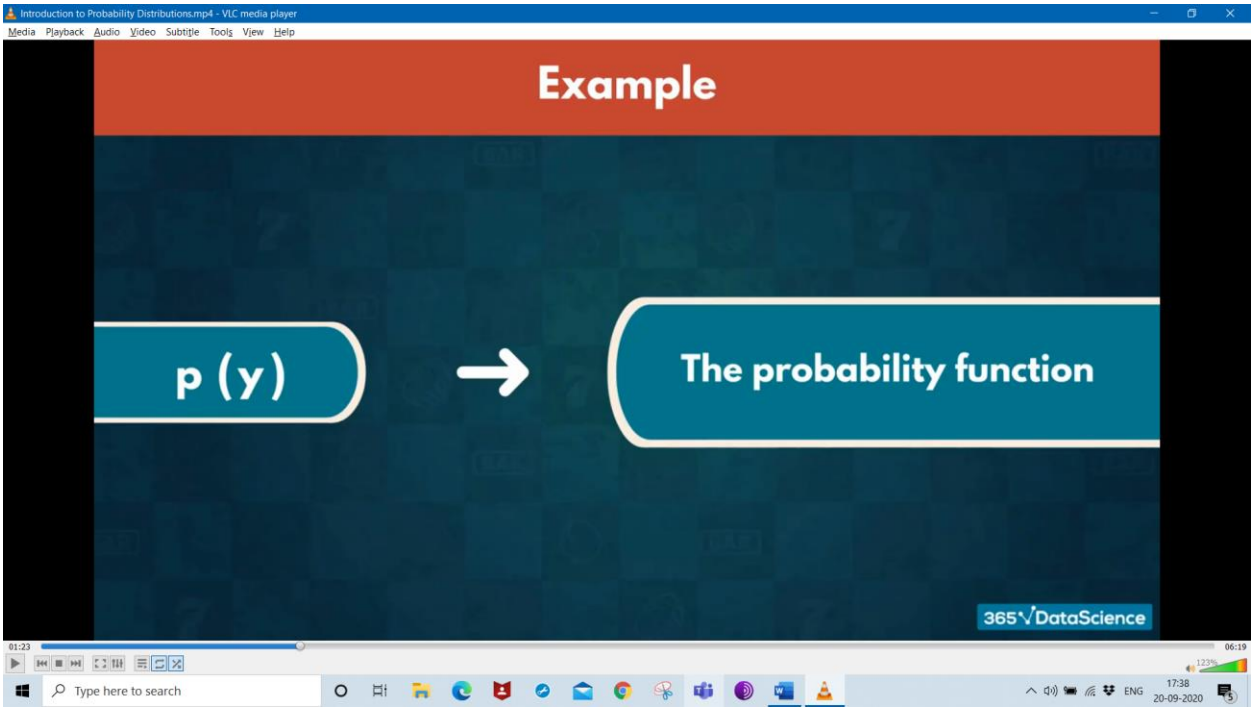

Introduction to Probability Distributions.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Overview of Distributions
Distribution:
The possible values a variable can take
and how frequently they occur.
365√DataScience
00:13
06:19
Type here to search
ENG
17:36
20-09-2020

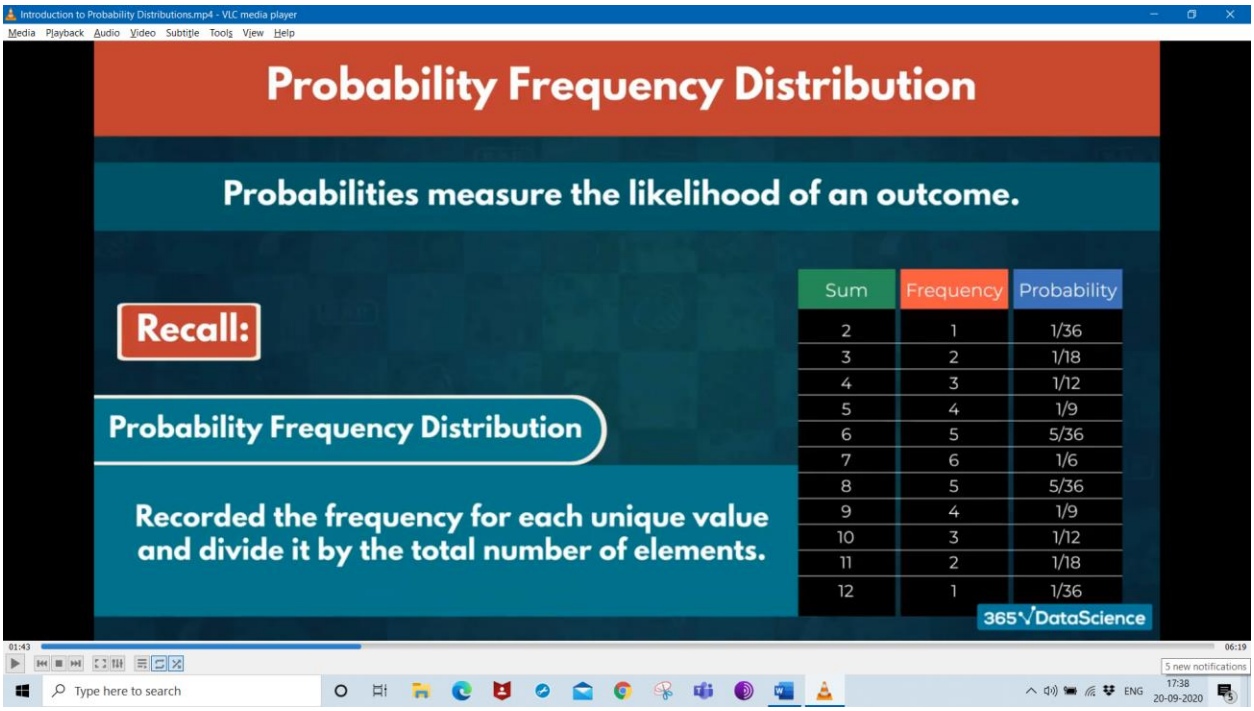

Introduction to Probability Distributions.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Notation
Y → The actual outcome of an event
y → One of the possible outcomes
P (Y = y)
or
p (y)
365√DataScience
00:48
06:19
5 new notifications
Type here to search
ENG
17:37
20-09-2020

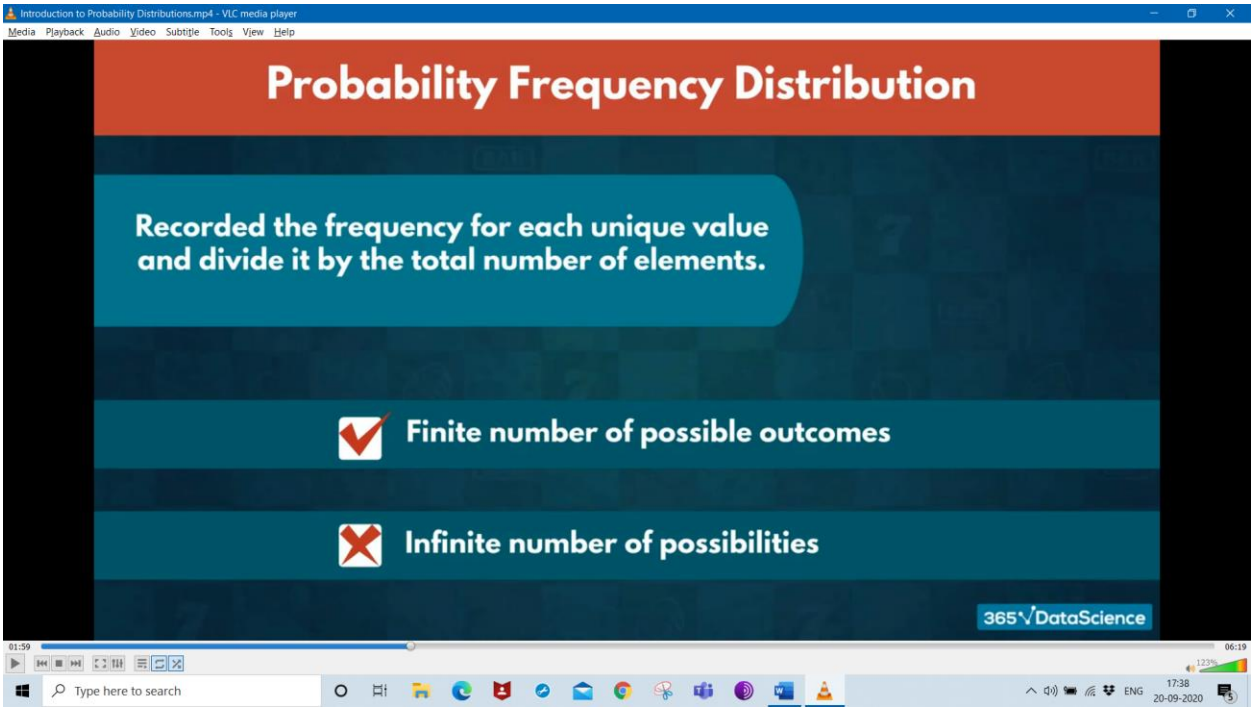

Introduction to Probability Distributions.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Example
Y → The number of red marbles we draw out of a bag
y → 3
365√DataScience
00:55
06:19
Type here to search
ENG
17:37
20-09-2020

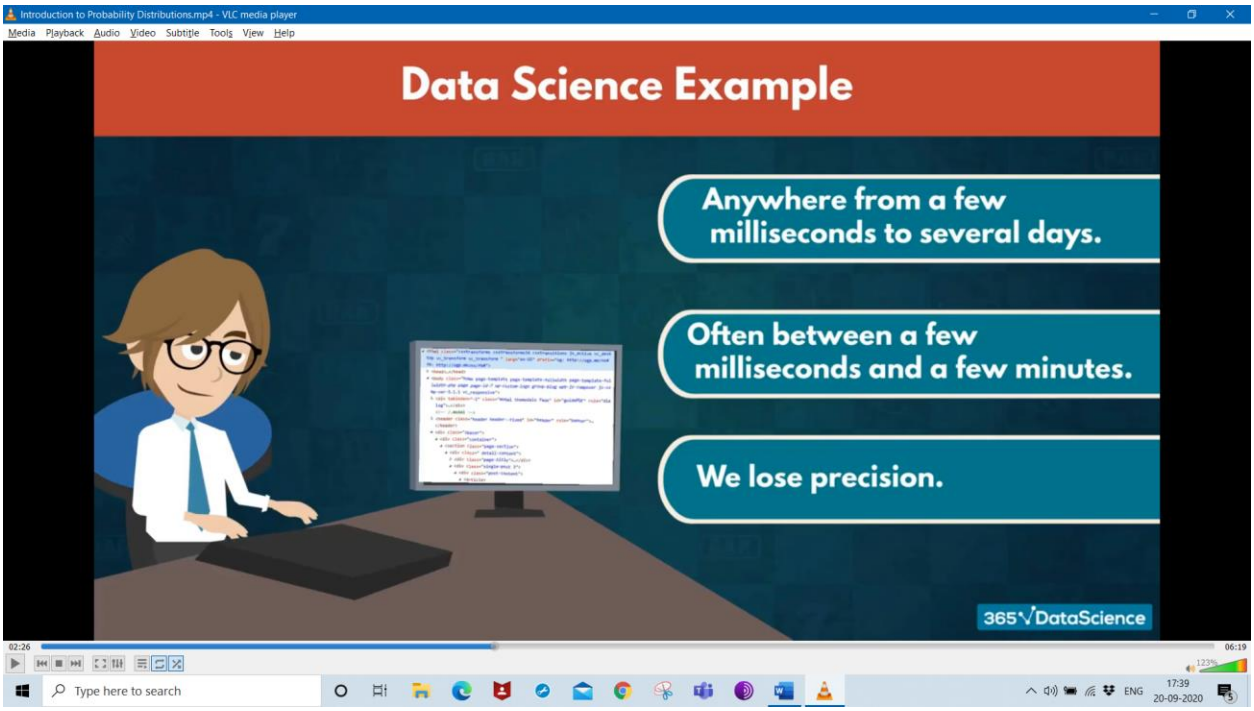

Introduction to Probability Distributions.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Example
Y → The number of red marbles we draw out of a bag
y → 5
365√DataScience
00:58
06:19
Type here to search
ENG
17:37
20-09-2020

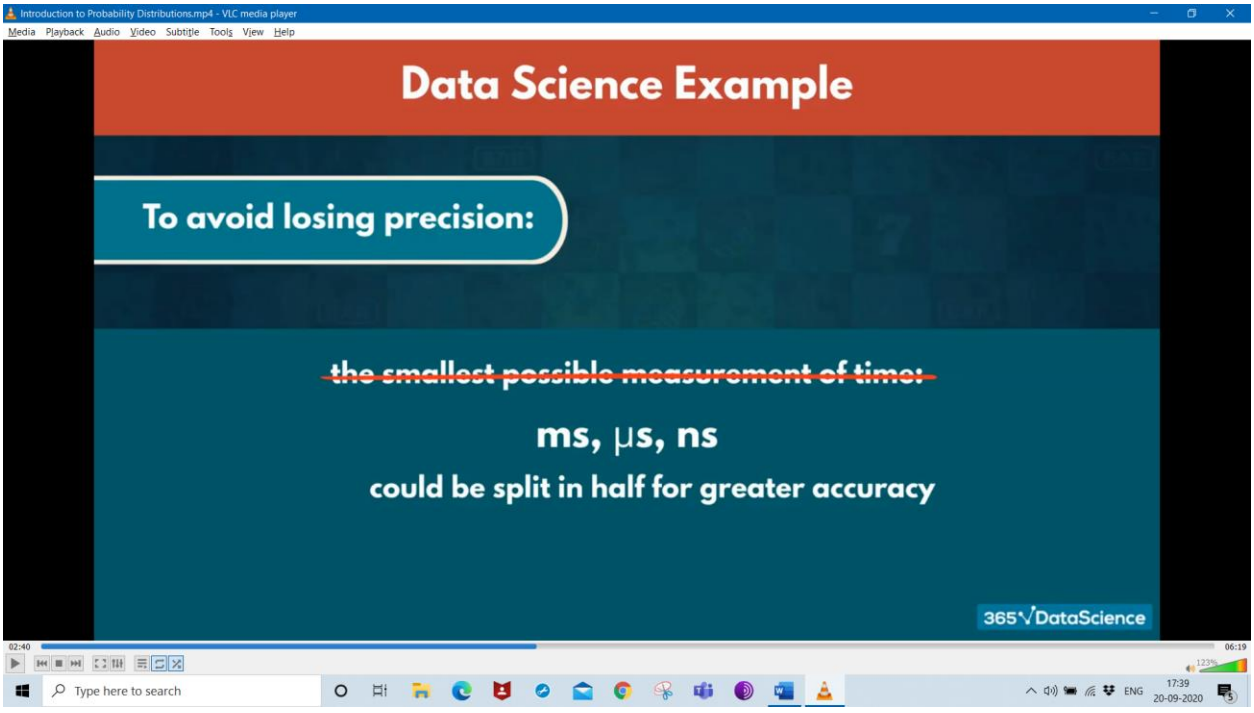

Introduction to Probability Distributions.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Example
Y → The number of red marbles we draw out of a bag
y → 5
P (Y = 5)
or
p (5)
365√DataScience
01:09
06:19
Type here to search
ENG
17:38
20-09-2020

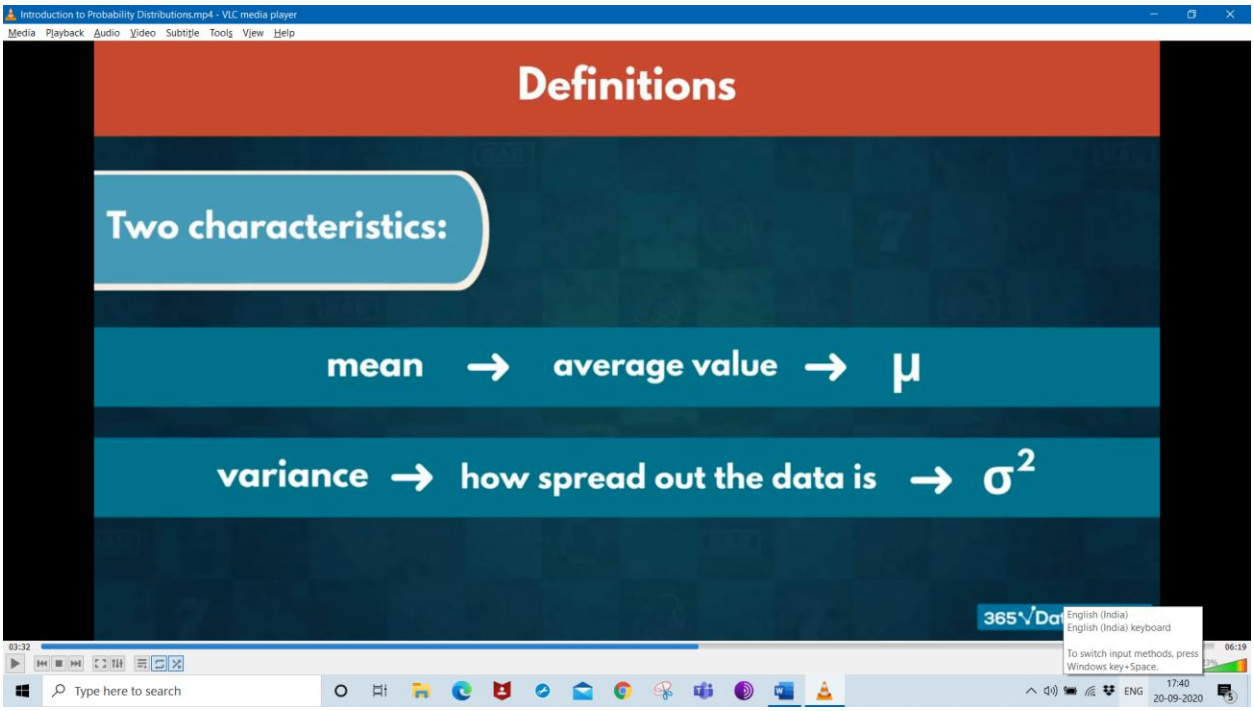

Introduction to Probability Distributions.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Example
p (y)
The probability function
365√DataScience
01:23
06:19
Type here to search
ENG
17:38
20-09-2020

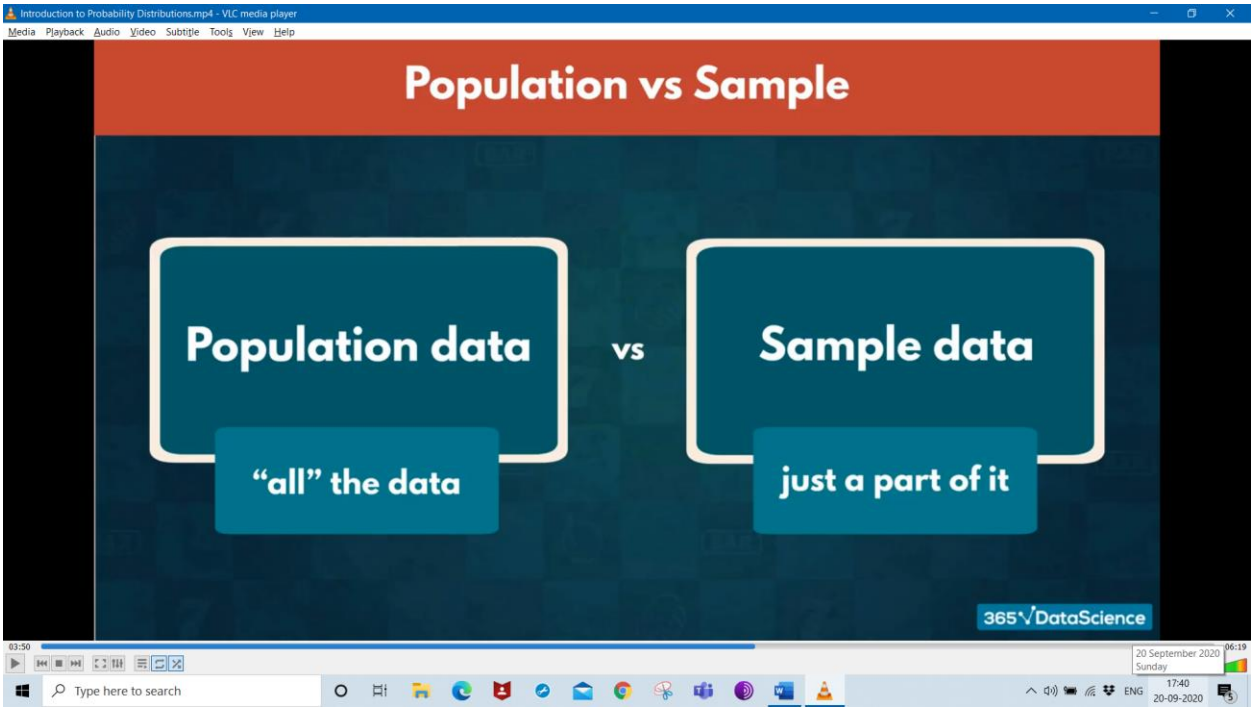

Introduction to Probability Distributions.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Probability Frequency Distribution
Probabilities measure the likelihood of an outcome.
Recall:
Probability Frequency Distribution
Recorded the frequency for each unique value and divide it by the total number of elements.
Sum | Frequency | Probability
2 | 1 | 1/36
3 | 2 | 1/18
4 | 3 | 1/12
5 | 4 | 1/9
6 | 5 | 5/36
7 | 6 | 1/6
8 | 5 | 5/36
9 | 4 | 1/9
10 | 3 | 1/12
11 | 2 | 1/18
12 | 1 | 1/36
365√DataScience
01:43
06:19
5 new notifications
Type here to search
ENG
17:38
20-09-2020

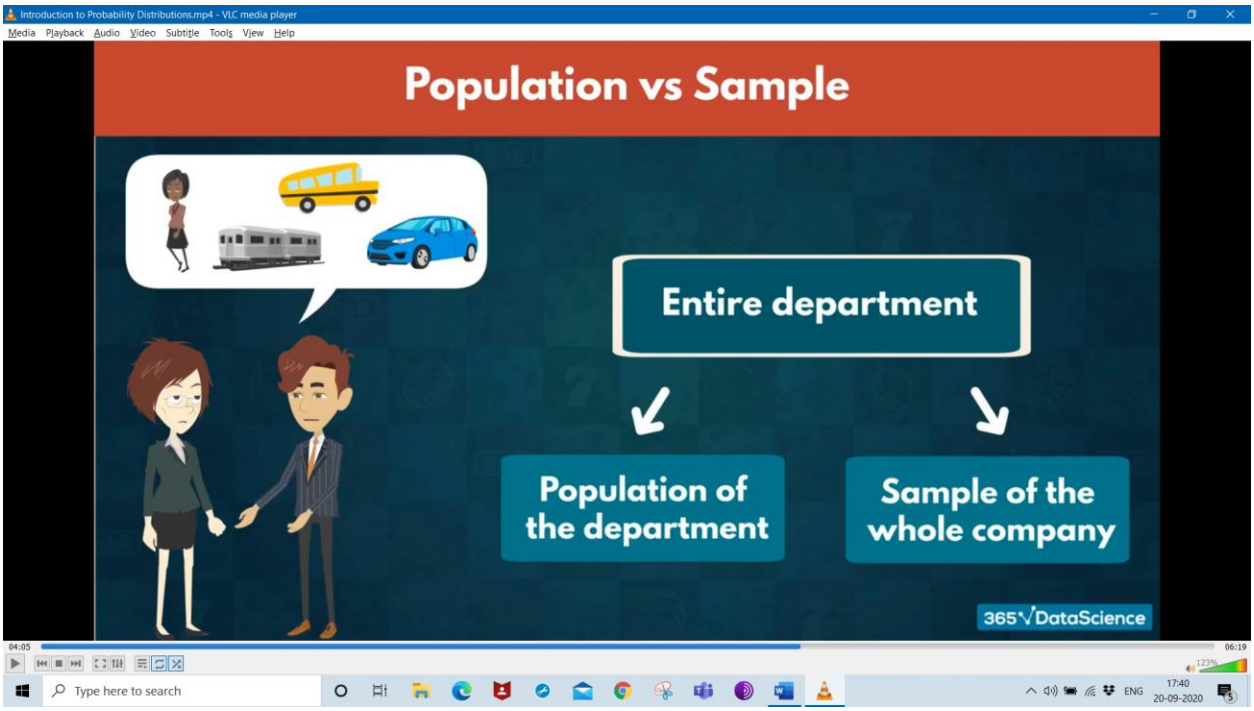

Introduction to Probability Distributions.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Probability Frequency Distribution
Recorded the frequency for each unique value and divide it by the total number of elements.
Finite number of possible outcomes
Infinite number of possibilities
365√DataScience
01:59
06:19
Type here to search
ENG
17:38
20-09-2020

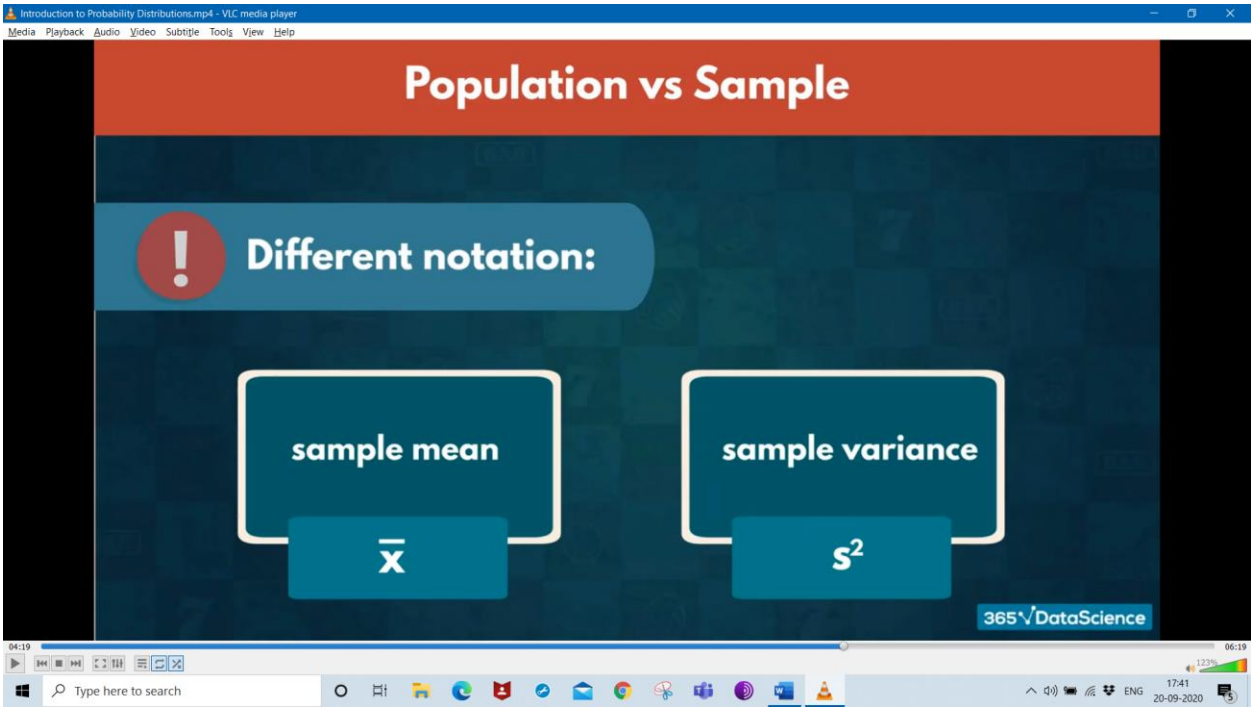

Introduction to Probability Distributions.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Data Science Example
Anywhere from a few milliseconds to several days.
Often between a few milliseconds and a few minutes.
We lose precision.
365√DataScience
02:26
06:19
Type here to search
ENG
17:39
20-09-2020

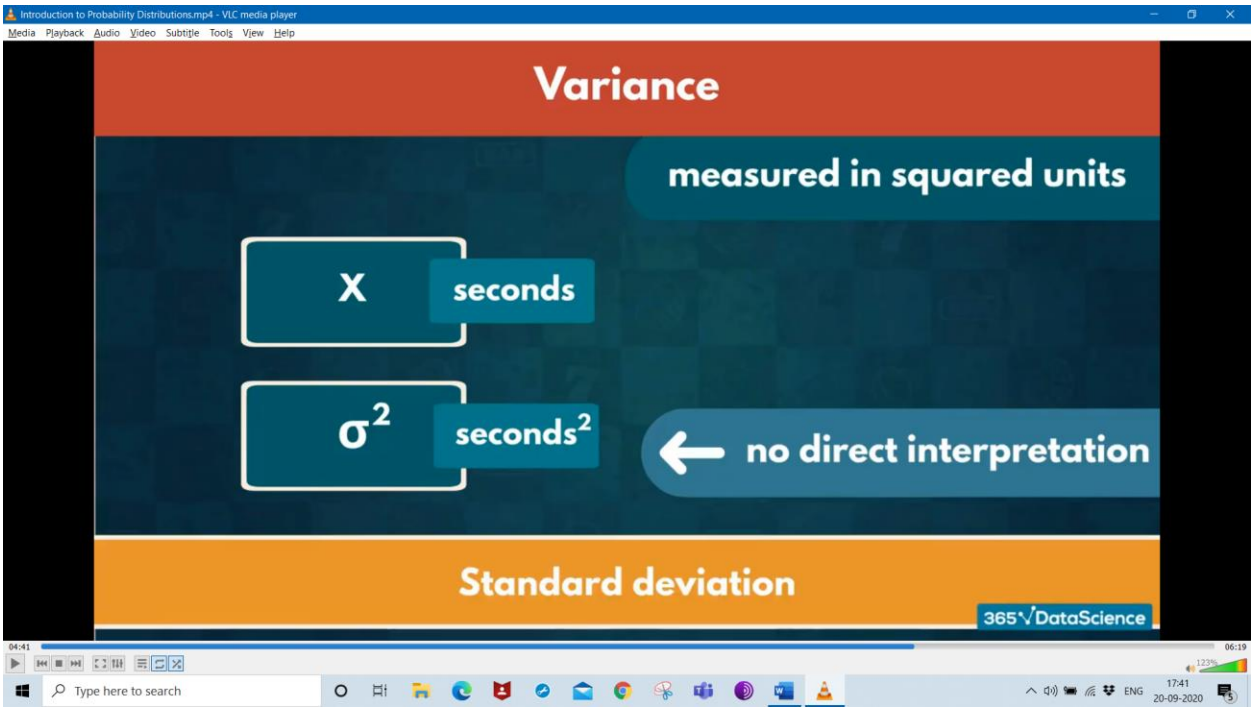

Introduction to Probability Distributions.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Data Science Example
To avoid losing precision:
the smallest possible measurement of time:
ms, μs, ns
could be split in half for greater accuracy
365√DataScience
02:40
06:19
Type here to search
ENG
17:39
20-09-2020

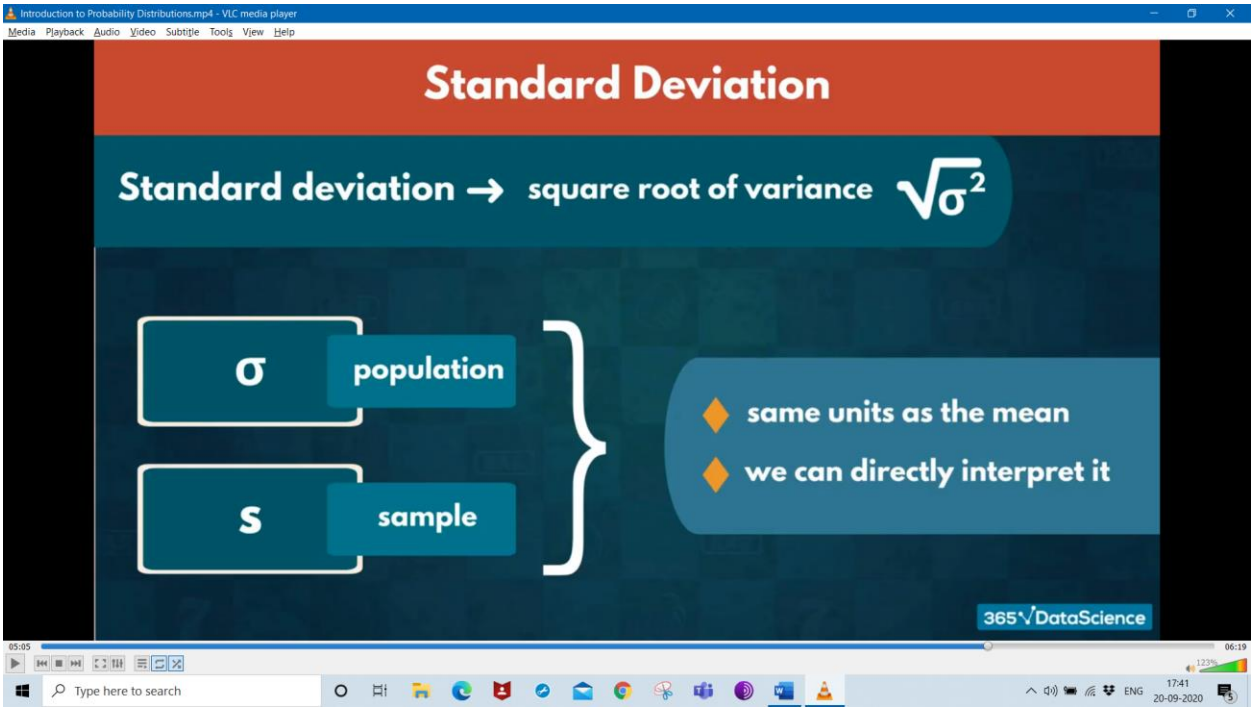

Introduction to Probability Distributions.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Definitions
Two characteristics:
mean → average value → μ
variance → how spread out the data is → σ²
365√Da
English (India)
English (India) keyboard
To switch input methods, press Windows key+Space.
03:32
06:19
Type here to search
ENG
17:40
20-09-2020

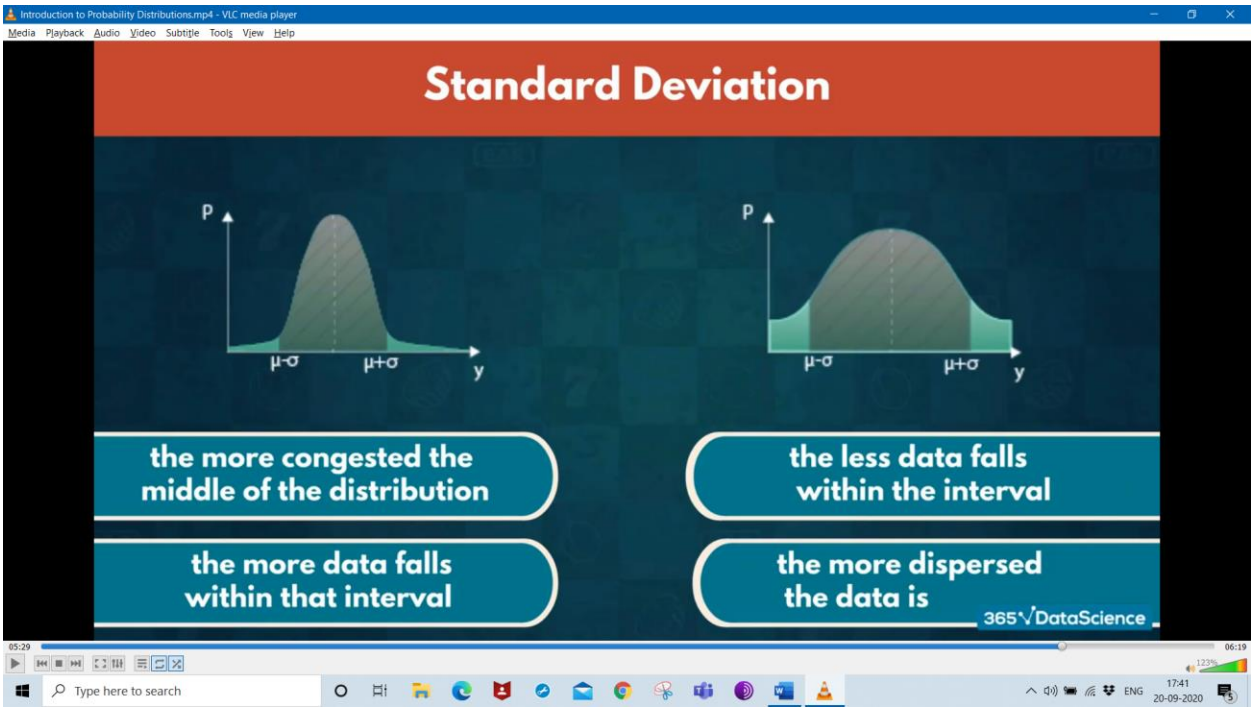

Introduction to Probability Distributions.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Population vs Sample
Population data
vs
Sample data
"all" the data
just a part of it
365√DataScience
03:50
06:19
20 September 2020
Sunday
Type here to search
ENG
17:40
20-09-2020

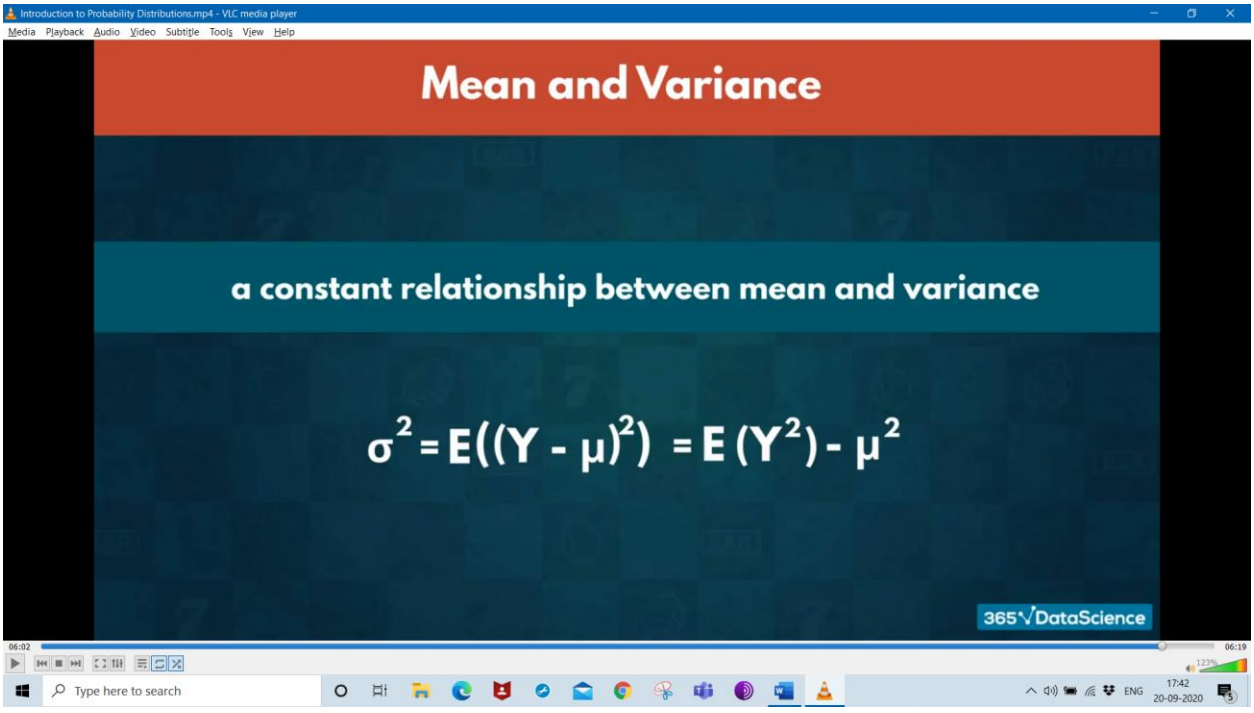

Introduction to Probability Distributions.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Population vs Sample
Entire department
Population of the department
Sample of the whole company
365√DataScience
04:05
06:19
Type here to search
ENG
17:40
20-09-2020


Introduction to Probability Distributions.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Population vs Sample
Different notation:
sample mean
$\bar{x}$
sample variance
s^2
365√DataScience
04:19
06:19
Type here to search
ENG
17:41
20-09-2020


Introduction to Probability Distributions.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Variance
measured in squared units
X
seconds
σ²
seconds²
no direct interpretation
Standard deviation
365√DataScience
04:41
06:19
Type here to search
ENG
17:41
20-09-2020


Introduction to Probability Distributions.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Standard Deviation
Standard deviation → square root of variance √σ²
σ
population
s
sample
same units as the mean
we can directly interpret it
365√DataScience
05:05
06:19
Type here to search
ENG
17:41
20-09-2020


Introduction to Probability Distributions.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Standard Deviation
P
μ-σ
μ+σ
y
P
μ-σ
μ+σ
y
the more congested the middle of the distribution
the less data falls within the interval
the more data falls within that interval
the more dispersed the data is
365√DataScience
05:29
06:19
Type here to search
ENG
17:41
20-09-2020


Introduction to Probability Distributions.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Mean and Variance
a constant relationship between mean and variance
$\sigma^2 = E((Y - \mu)^2) = E(Y^2) - \mu^2$
365√DataScience
06:02
06:19
Type here to search
ENG
17:42
20-09-2020

In probability theory, a probability density function (PDF), or density of a continuous random variable, is a function that describes the relative likelihood for this random variable to take on a given value.

Probability density function is defined by following formula:

$$P(a \leq X \leq b) = \int_a^b f(x)d_x$$

Where –

- $[a, b]$ = Interval in which x lies.
- $P(a \leq X \leq b)$ = probability that some value x lies within this interval.
- d_x = b-a

Probability Density Function (PDF) is used to define the probability of the random variable coming within a distinct range of values, as objected to taking on anyone value.

The Probability Density Function(PDF) is the probability function which is represented for the density of a continuous random variable lying between a certain range of values. It is also called a probability distribution function or just a probability function. However, in many other sources, this function is stated as the function over a general set of values or sometimes it is referred to as cumulative distribution function or sometimes as probability mass function(PMF). But the actual truth is PDF is defined for continuous random variables whereas PMF is defined for discrete random variables.

The probability density function is defined in the form of an integral of the density of the variable density over a given range. It is denoted by f (x). This function is positive or non-negative at any point of the graph and the integral of PDF over the entire space is always equal to one.

Probability Density Function Formula

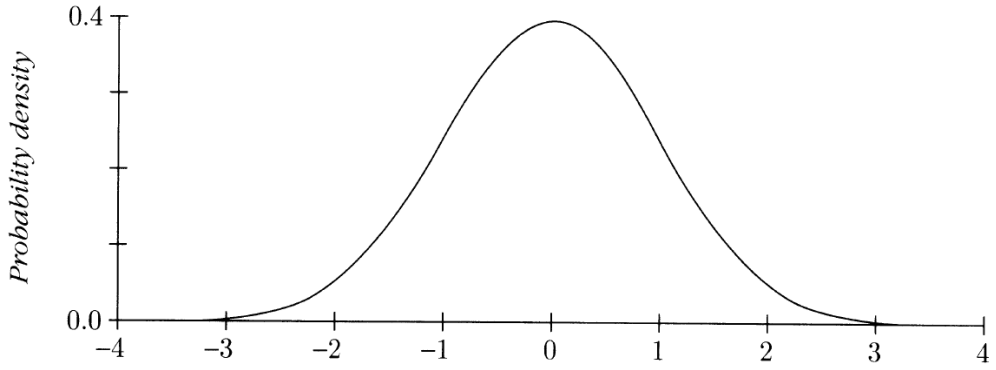
In case of a continuous random variable, the probability taken by X on some given value x is always 0. In this case, if we find P(X = x), it does not work. Instead of this, we require to calculate the probability of X lying in an interval (a, b). Now, we have to calculate it for P(a< X< b). This can be done by using a PDF. The Probability distribution function formula is defined as,

$$P(a < X < b) = \int_a^b f(x)$$

The Normal Distribution

A random variable Z has *standard normal distribution* if Z has as its probability density the *standard normal density*

$$\phi(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2} \qquad (-\infty < z < \infty)$$



The constant $1/\sqrt{2\pi}$ is put in the definition of the standard normal density so the total area under the standard normal curve $y = \phi(z)$ is 1. This is the first integral in the following box:

Standard Normal Integrals

$$\int_{-\infty}^{\infty} \phi(z)\, dz = 1; \qquad \int_{-\infty}^{\infty} z\phi(z)\, dz = 0; \qquad \int_{-\infty}^{\infty} z^2\phi(z)\, dz = 1.$$

Normal (μ, σ^2) Distribution

If Z has standard normal distribution and μ and σ are constants with $\sigma \geq 0$, then

$$X = \mu + \sigma Z$$

has mean μ, standard deviation σ, and variance σ^2. The distribution of X is called the *normal distribution with mean μ and variance σ^2*, abbreviated normal (μ, σ^2). So X has normal (μ, σ^2) distribution if and only if the standardized variable

$$Z = (X - \mu)/\sigma$$

has normal $(0, 1)$ or *standard normal* distribution. To find $P(c < X < d)$, change to standard units and use the standard normal table

$$P(c < X < d) = P(a < Z < b) = \Phi(b) - \Phi(a)$$

where $\quad a = (c - \mu)/\sigma \qquad Z = (X - \mu)/\sigma \qquad b = (d - \mu)/\sigma$

Formula for the normal (μ, σ^2) density. For $\sigma > 0$, the formula is

$$\frac{1}{\sigma}\phi((x - \mu)/\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}(x-\mu)^2/\sigma^2} \qquad (-\infty < x < \infty).$$

Joint Distributions

Given two random variables X and Y defined in the same setting, we can consider their *combined* or *joint* outcome (X, Y) as a random pair of values. By definition, (X, Y) has value (x, y) if X has value x and Y has value y. Thus the event that $((X, Y) = (x, y))$ is the intersection of the events $(X = x)$ and $(Y = y)$, and is usually denoted $(X = x, Y = y)$. So commas mean intersections in statements about random variables.

The range of the joint outcome (X, Y) is the set of all ordered pairs (x, y) with x in the range of X, y in the range of Y, and $P(X = x, Y = y) > 0$. If the range of X is represented by points on a horizontal line, and the range of Y by points on a vertical line, then the range of (X, Y) is represented by a set of points in the plane. Alternatively, the range of (X, Y) may be represented by a set of paths through a tree diagram, as in Chapter 1.

The distribution of (X, Y) is called the *joint distribution* of X and Y. This distribution is determined by the probabilities

$$P(x, y) = P(X = x, Y = y)$$

which must satisfy

$$P(x, y) \geq 0 \qquad \text{and} \qquad \sum_{\text{all } (x,y)} P(x, y) = 1$$

Random Variables with the Same Distribution

Random variables X and Y have the *same* or *identical distribution* if X and Y have the same range, and for every value v in this range,

$$P(X = v) = P(Y = v).$$

Computing probabilities from a joint distribution. Once the joint distribution of X and Y has been calculated, the probability of any event defined in terms of X and Y can be found. Simply sum the probabilities $P(x, y)$ over the relevant set of pairs (x, y):

Independent Random Variables

Random variables X and Y are *independent* if

$$P(X = x, Y = y) = P(X = x)P(Y = y) \quad \text{for all } x \text{ and } y$$

If X and Y are independent random variables, then every event determined by X is independent of every event determined by Y:

$$P(X \in A, Y \in B) = P(X \in A)P(Y \in B)$$

Conceptually, independence means that conditioning on a given value of X does not affect the distribution of Y, and vice-versa. Thus the above definition of independence can be re-expressed as follows in terms of conditional distributions:

Conditional Distributions and Independence

The following three conditions are equivalent:

- X and Y are independent;
- the conditional distribution of Y given $X = x$ does not depend on x;
- the conditional distribution of X given $Y = y$ does not depend on y.

Standard Deviation and Normal Approximation

If you try to predict the value of a random variable X by its mean $E(X) = \mu$, you will be off by the random amount $X - \mu$. It is often important to have an idea of how large this deviation is likely to be. Because

$$E(X - \mu) = E(X) - \mu = 0$$

it is necessary to consider either the absolute value or the square of $X - \mu$ to get an idea of the size of the deviation without regard to sign. Because the algebra is easier with squares than with absolute values, it is natural to first consider $E[(X - \mu)^2]$, then take a square root to get back to the same scale of units as X.

Definition of Variance and Standard Deviation

The *variance* of X, denoted $Var(X)$, is the mean squared deviation of X from its expected value $\mu = E(X)$:

$$Var(X) = E[(X - \mu)^2]$$

The *standard deviation* of X, denoted $SD(X)$, is the square root of the variance of X:

$$SD(X) = \sqrt{Var(X)}$$

Intuitively, $SD(X)$ should be understood as a measure of how spread out the distribution of X is around its mean μ. Because $Var(X)$ is a central value in the distribution of $(X - \mu)^2$, its square root $SD(X)$ gives a rough idea of the typical size of the absolute deviation $|X - \mu|$. Variance always appears as an intermediate step in the calculation of standard deviation. Variance is harder to interpret than SD, but has simpler algebraic properties. Notice that $E(X)$, $Var(X)$, and $SD(X)$ are all determined by the distribution of X. That is to say, if two random variables have the same distribution, then they have the same mean, variance, and SD. So we may speak of the mean, variance, and SD of a distribution rather than a random variable.

Parameters of a normal curve. If a histogram displaying the distribution of X follows an approximately normal curve, the curve will be centered near the mean $E(X)$, and $SD(X)$ will be approximately the distance between the center of the curve and its shoulders, where the curve switches from being concave to convex.

The **Normal Distribution** was used as an approximation to the distribution of a sum or average of a large number of independent random variables.

The idea there was to approximate a discrete distribution of many small individual probabilities by scaling the histogram to make it follow a continuous curve.

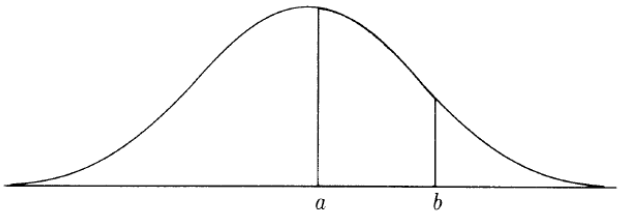
The function defining such a curve is called a ***probability density,*** **denoted *f(x)*** here.

This function ***determines probabilities over an infinite continuous range of possible values***.

The basic idea is that probabilities are defined by areas under the graph of $f(x)$. That is, a random variable X has density $f(x)$ if for all $a \leq b$

$$P(a \leq X \leq b) = \int_a^b f(x)dx,$$

which is the area shaded in the following diagram:



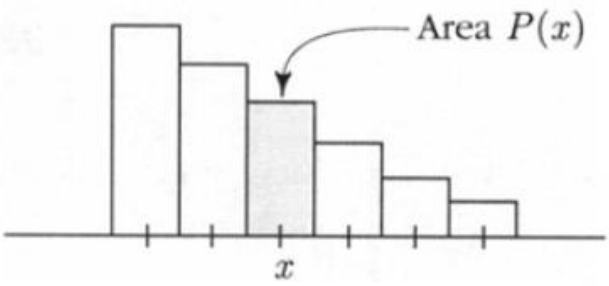
The PDF is used to specify the probability of the random variable falling *within a particular range of values*, as opposed to taking on any one value.

This probability is given by the integral of this variable's PDF over that range—that is, it is given by the area under the density function, but above the horizontal axis and between the lowest and greatest values of the range.

The probability density function is nonnegative everywhere, and its integral over the entire space is equal to 1.

Discrete Distributions

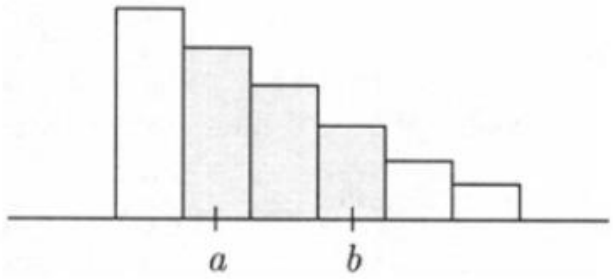
Point Probability:



$$P(X = x) = P(x)$$

So $P(x)$ is the probability that X has integer value x.

Interval Probability:



$$P(a \leq X \leq b) = \sum_{a \leq x \leq b} P(x)$$

the relative area under a histogram between $a - 1/2$ and $b + 1/2$.

Maximum likelihood estimation

$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i \qquad \varepsilon_i \overset{iid}{\sim} N(0, \sigma^2)$

$$f(\varepsilon_i) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2} \frac{(\varepsilon_i - 0)^2}{\sigma^2} \right]$$

Likelihood function : $f(\varepsilon_1, \varepsilon_2 \cdots \varepsilon_n) = \prod_{i=1}^{n} f(\varepsilon_i)$

$$\mathcal{L} = \prod_{i=1}^{n} \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{1}{2} \frac{\varepsilon_i^2}{\sigma^2} \right]$$

$$= \left(\frac{1}{\sigma\sqrt{2\pi}} \right)^n \exp\left[-\frac{1}{2} \sum_{i=1}^{n} \frac{\varepsilon_i^2}{\sigma^2} \right]$$

$$= \left(\frac{1}{\sigma\sqrt{2\pi}} \right)^n \exp\left[-\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2 \right]$$

$$\mathcal{L}^* = \ln \mathcal{L} = -\frac{n}{2} \ln 2\pi - \frac{n}{2} \ln \sigma^2 - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2$$

$\mathcal{L}^* =$ function of β_0, β_1 & σ^2.

Normal equations

i) $$\frac{\partial \mathcal{L}^*}{\partial \beta_0} = -\frac{1}{\sigma^2} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i) = 0$$

ii) $$\frac{\partial \mathcal{L}^*}{\partial \beta_1} = -\frac{1}{\sigma^2} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i) x_i = 0$$

iii) $$\frac{\partial \mathcal{L}^*}{\partial \sigma^2} = -\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4} \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2 = 0$$

(i) $\sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i) = 0$ or $n\bar{y} - n\beta_0 - \beta_1 n\bar{x} = 0$

$\Rightarrow \tilde{\beta}_0 = \bar{y} - \tilde{\beta}_1 \bar{x}$ * Same as

(ii) $$\tilde{\beta}_1 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$ * OLSE

$\bar{x} = \frac{1}{n} \sum x_i$

$\bar{y} = \frac{1}{n} \sum y_i$

(iii) $$\tilde{\sigma}^2 = \frac{1}{n} \sum_{i=1}^{n} (y_i - \tilde{\beta}_0 - \tilde{\beta}_1 x_i)^2$$ → different from l.s.e.

Hessian matrix of 2nd order partial derivatives of L^* with respect to β_0, β_1 and σ^2 at $\beta_0 = \tilde{\beta}_0$, $\beta_1 = \tilde{\beta}_1$, $\sigma^2 = \tilde{\sigma}^2$ turns out to be negative definite

$\left.\begin{matrix}\tilde{\beta}_0 \\ \tilde{\beta}_1 \\ \tilde{\sigma}^2\end{matrix}\right\}$ m.l.e.

Testing of hypothesis and confidence interval estimation

$y_i = \beta_0 + \beta_1 x_i + \varepsilon_i \quad i = 1, \dots, n$

$\varepsilon_i \overset{iid}{\sim} N(0, \sigma^2)$

Two cases $<$ σ^2 is known / σ^2 is unknown

3 setups — β_0

— β_1

— σ^2

Slope parameter

Null hypothesis $H_0 : \beta_1 = \beta_{10}$ $\quad \beta_{10}$: given constant : known

$\hat{\beta}_1 = \frac{S_{xy}}{S_{xx}}$ (OLS, m.l.e)

When σ^2 is known

$E(\varepsilon_i) = 0 \quad V(\varepsilon_i) = \sigma^2$

$E(\hat{\beta}_1) = \beta_1 \quad V(\hat{\beta}_1) = \frac{\sigma^2}{S_{xx}}$

$y_i \sim N(\beta_0 + \beta_1 x_i, \sigma^2)$

$\hat{\beta}_1 \sim N(\beta_1, \frac{\sigma^2}{S_{xx}})$

$$\frac{\hat{\beta}_1 - \beta_{10}}{\sqrt{\frac{\sigma^2}{S_{xx}}}} \sim N(0,1)$$

Slope parameter

Null hypothesis $H_0 : \beta_1 = \beta_{10}$ $\quad \beta_{10}$: given constant : known

$\hat{\beta}_1 = \frac{S_{xy}}{S_{xx}}$ (OLS, m.l.e)

When σ^2 is known

$E(\varepsilon_i) = 0 \quad V(\varepsilon_i) = \sigma^2$

$E(\hat{\beta}_1) = \beta_1 \quad V(\hat{\beta}_1) = \frac{\sigma^2}{S_{xx}}$

$y_i \sim N(\beta_0 + \beta_1 x_i, \sigma^2)$

$\hat{\beta}_1 \sim N(\beta_1, \frac{\sigma^2}{S_{xx}})$

$$Z_1 = \frac{\hat{\beta}_1 - \beta_{10}}{\sqrt{\frac{\sigma^2}{S_{xx}}}} \sim N(0,1)$$ when H_0 is true.

→ test statistic can be used to test the significance of β_1 when σ^2 is known

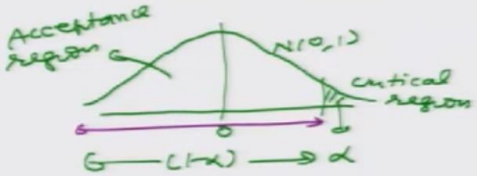
Decision rules

$H_0: \beta_1 > \beta_{10}$

Reject H_0 if $z_1 > z_\alpha$

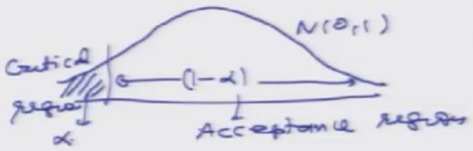
α : level of significance

z_α : upper α % points on the $N(0,1)$ distribution

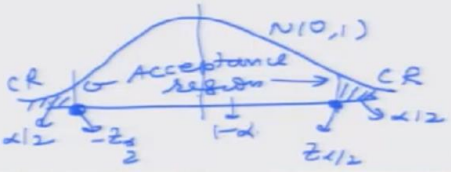


$H_1: \beta_1 < \beta_{10}$

Reject H_0 if $z_1 < z_\alpha$



$H_1: \beta_1 \neq \beta_{10}$



Decision rules

$H_0: \beta_1 > \beta_{10}$

Reject H_0 if $z_1 > z_\alpha$

α : level of significance

z_α : upper α % points on the $N(0,1)$ distribution

$H_1: \beta_1 < \beta_{10}$

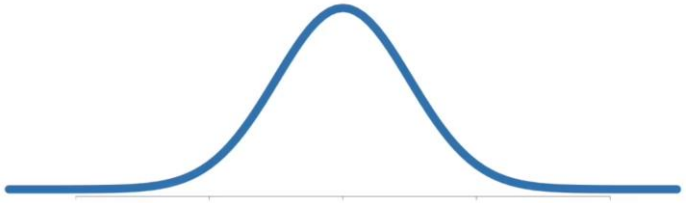
Reject H_0 if $z_1 < z_\alpha$

$H_1: \beta_1 \neq \beta_{10}$

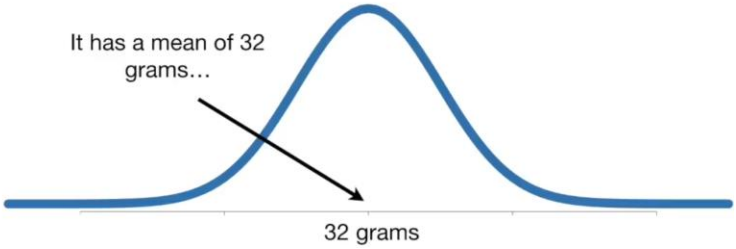
Reject H_0 if $|z_1| > z_{\frac{\alpha}{2}}$

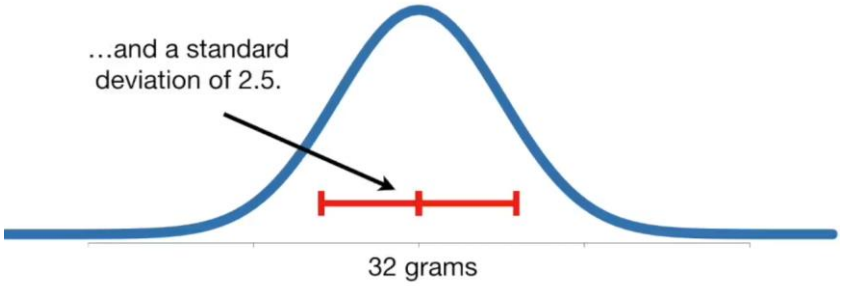
Maximum Likelihood Estimation

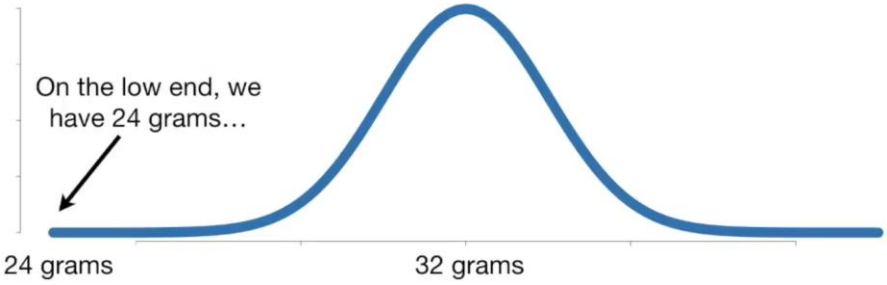
So let's start by looking at probability with respect to a normal distribution (keeping in mind that this concept applies to all continuous distributions).

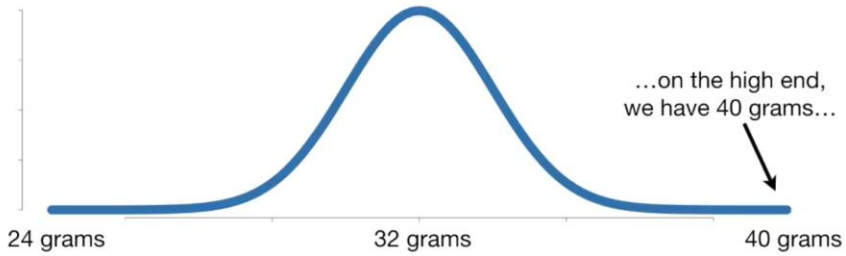


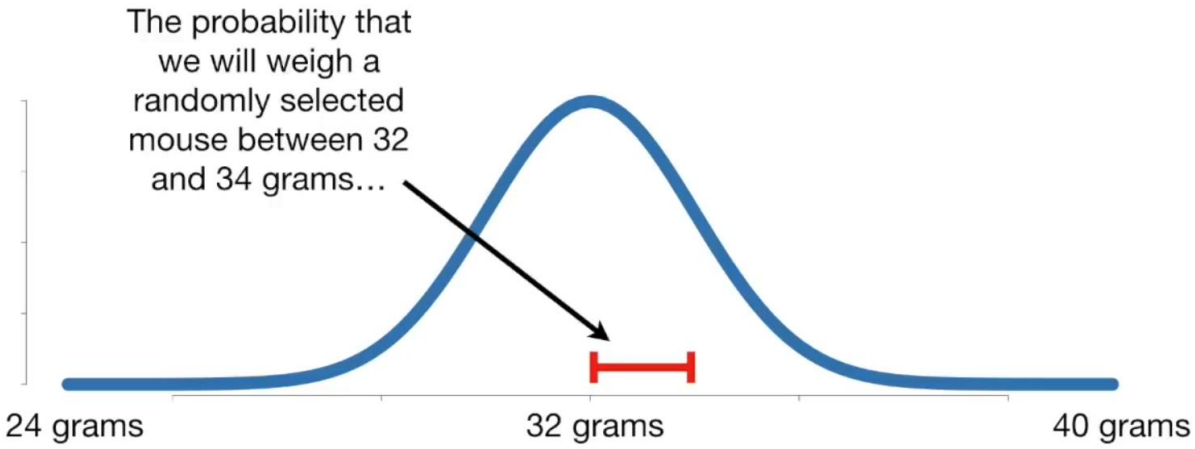
In this case, let's imagine that this is a distribution of mouse weights.

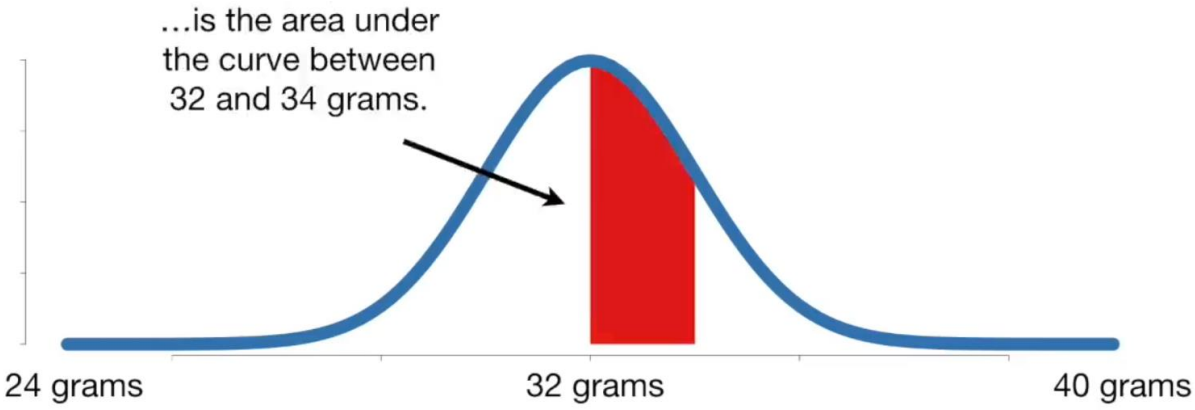


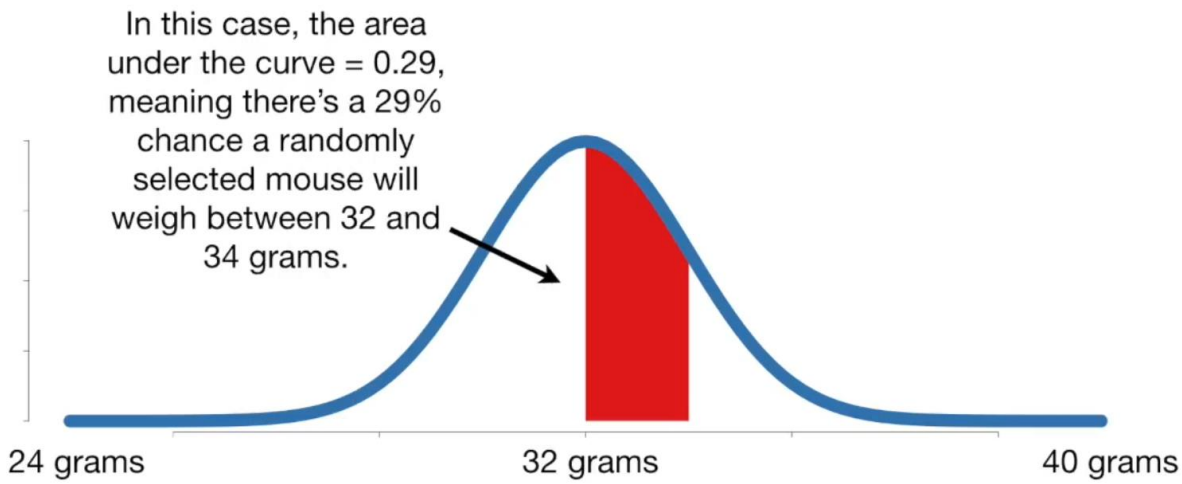





…on the high end,
we have 40 grams…
24 grams
32 grams
40 grams

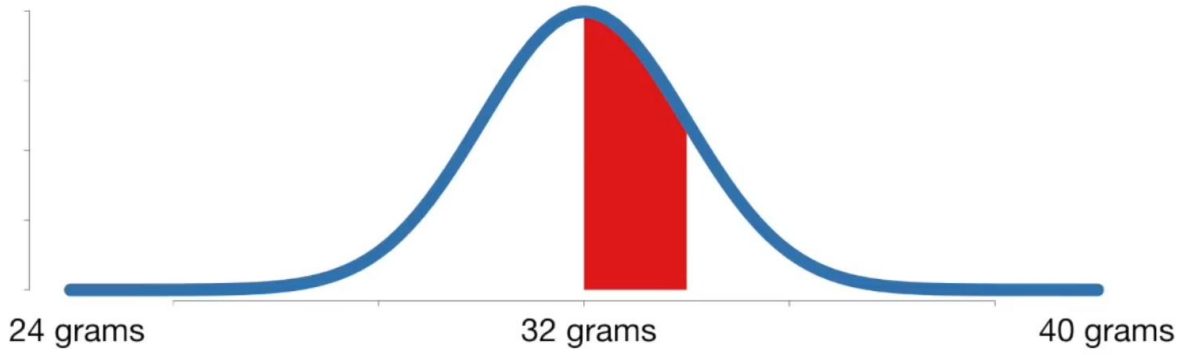

The probability that
we will weigh a
randomly selected
mouse between 32
and 34 grams…
24 grams
32 grams
40 grams


…is the area under
the curve between
32 and 34 grams.
24 grams
32 grams
40 grams

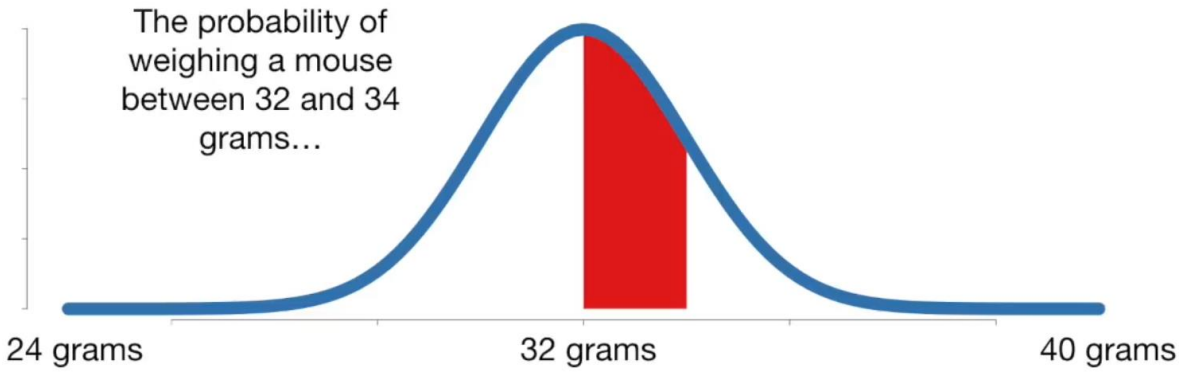


Mathematically, we say this with the following notation….

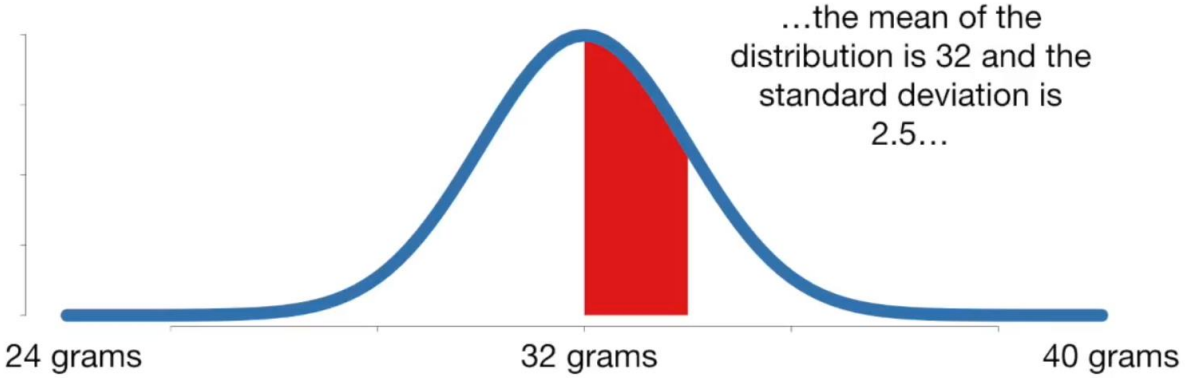
pr(weight between 32 and 34 grams | mean = 32 and standard deviation = 2.5)



pr(weight between 32 and 34 grams | mean = 32 and standard deviation = 2.5)

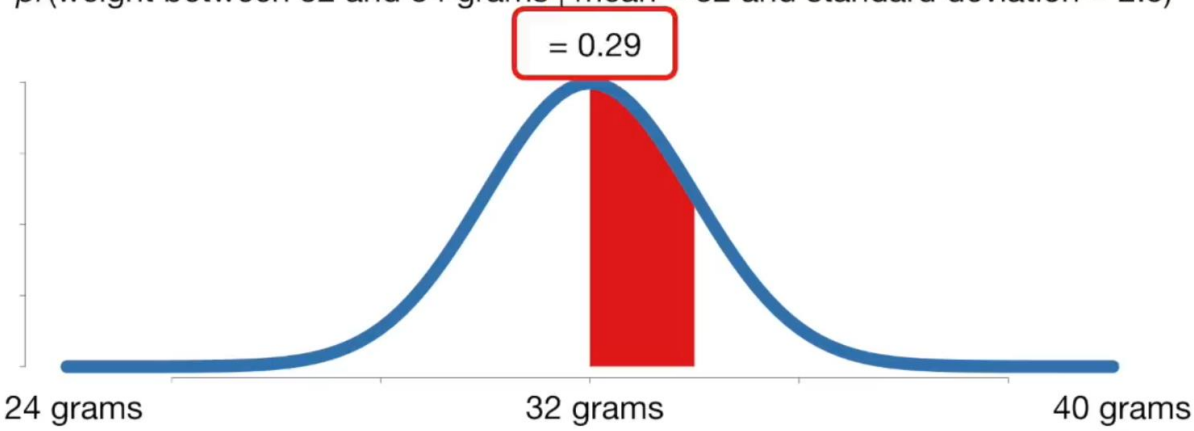


pr(weight between 32 and 34 grams | mean = 32 and standard deviation = 2.5)

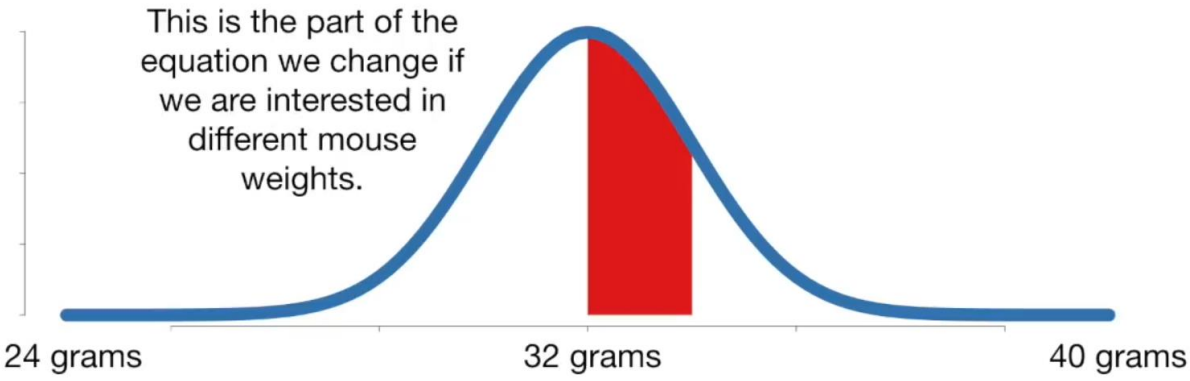


pr(weight between 32 and 34 grams | mean = 32 and standard deviation = 2.5)

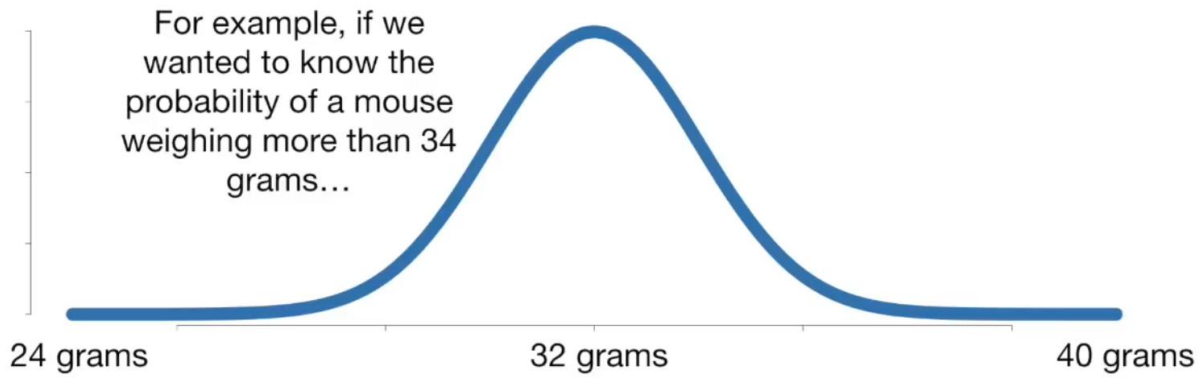
= 0.29



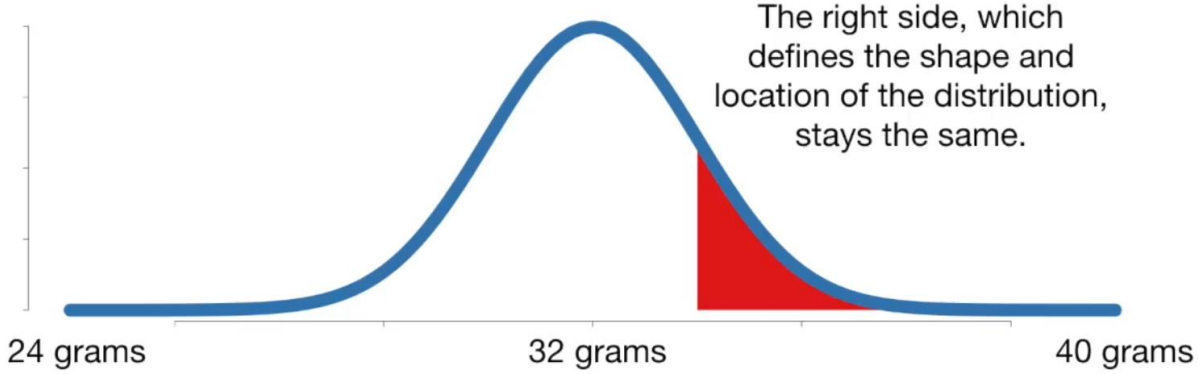
pr(weight between 32 and 34 grams | mean = 32 and standard deviation = 2.5)



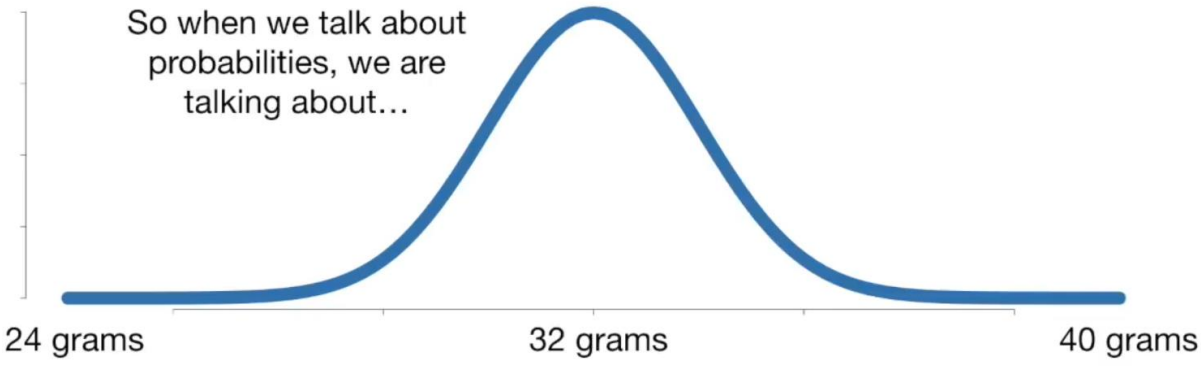
pr(mouse weighs > 34 grams | mean = 32 and standard deviation = 2.5)



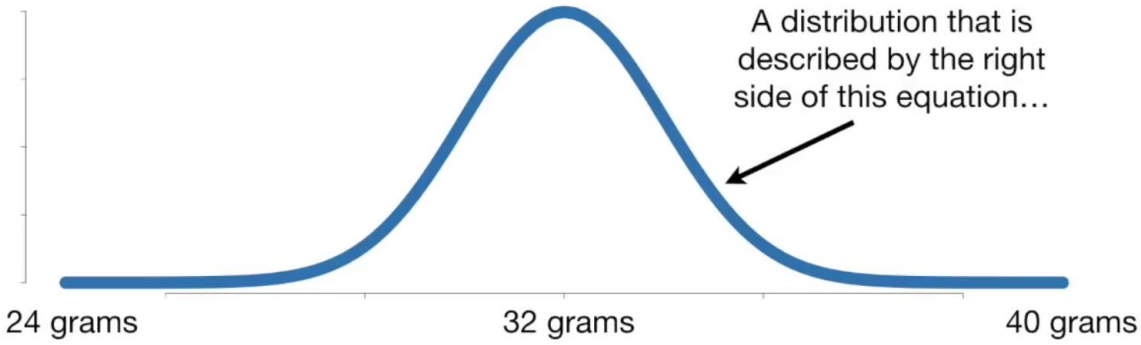
pr(mouse weighs > 34 grams | mean = 32 and standard deviation = 2.5)



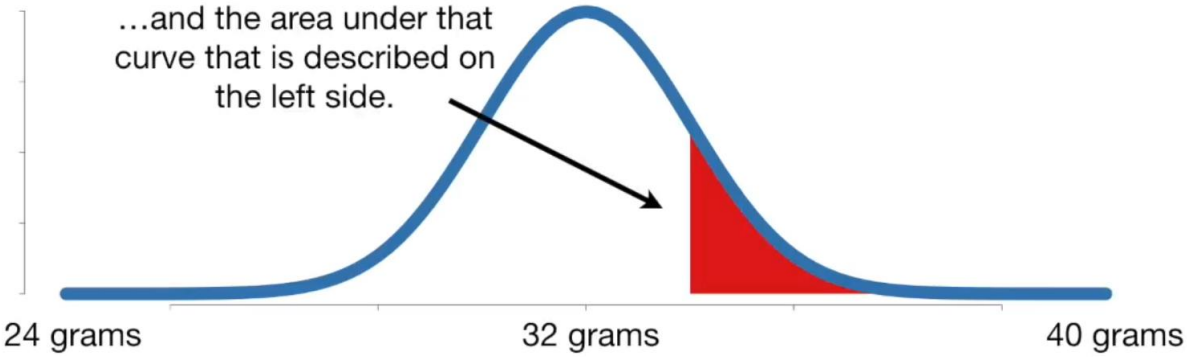
pr(mouse weighs > 34 grams | mean = 32 and standard deviation = 2.5)



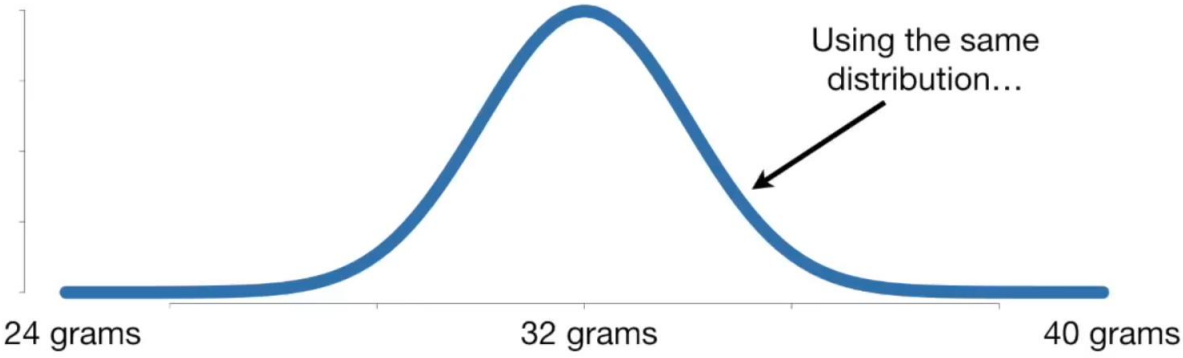
pr(mouse weighs > 34 grams | mean = 32 and standard deviation = 2.5)



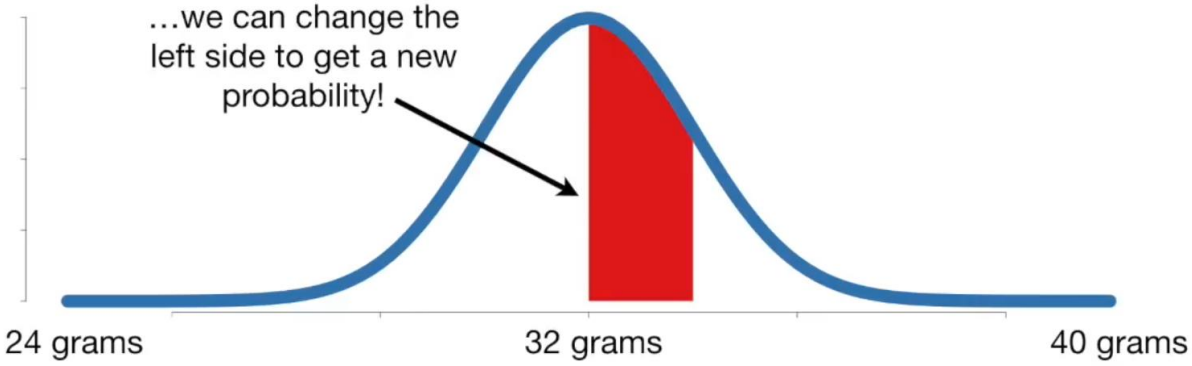
pr(mouse weighs > 34 grams | mean = 32 and standard deviation = 2.5)



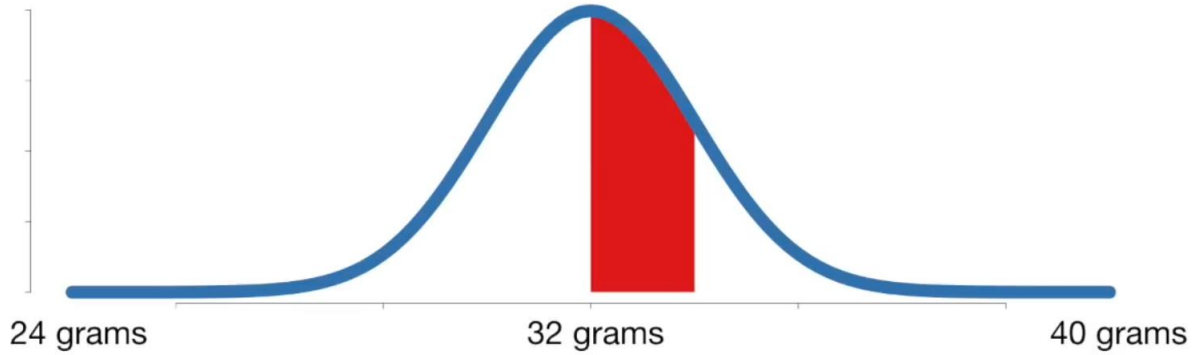
pr(mouse weighs > 34 grams | mean = 32 and standard deviation = 2.5)



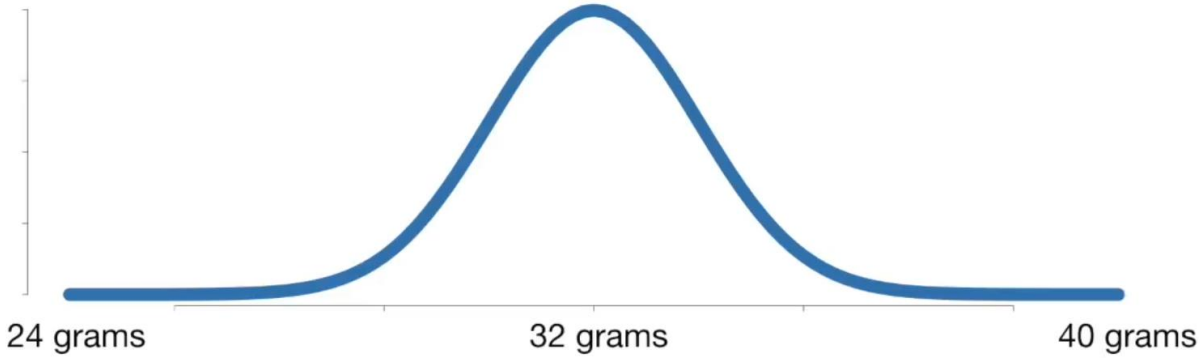
pr(weight between 32 and 34 grams | mean = 32 and standard deviation = 2.5)

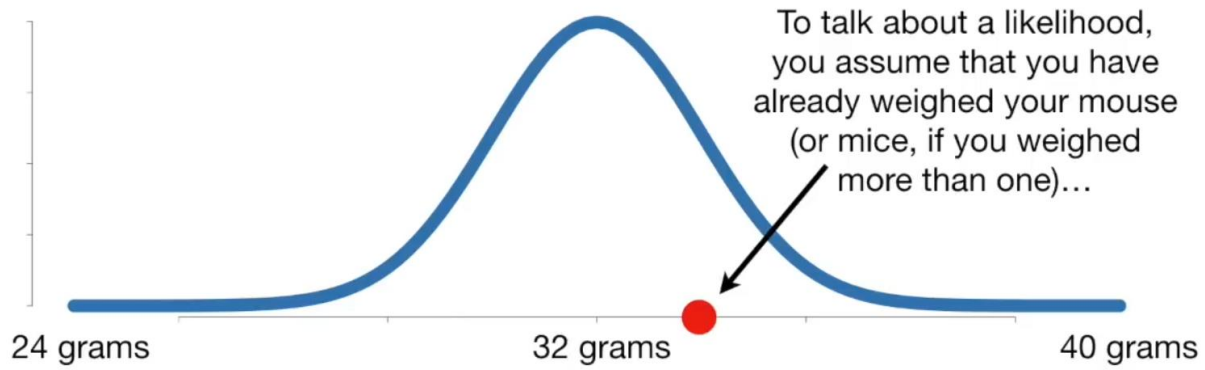


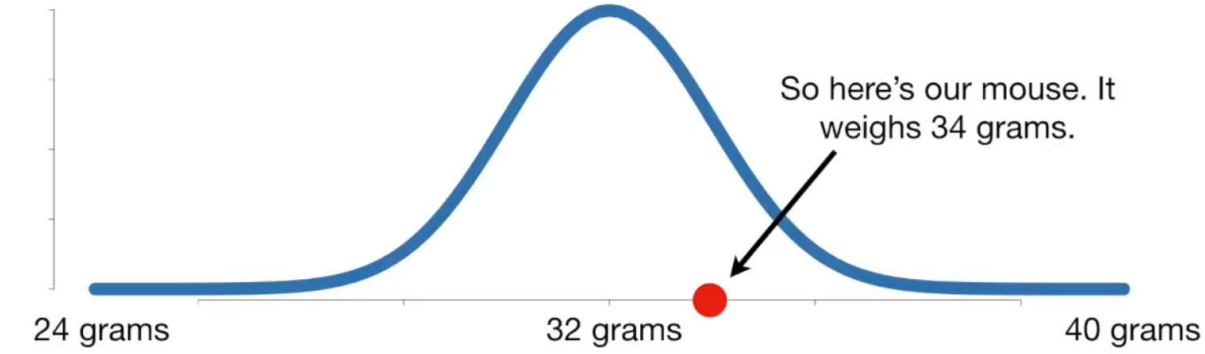
pr(weight between 32 and 34 grams | mean = 32 and standard deviation = 2.5)

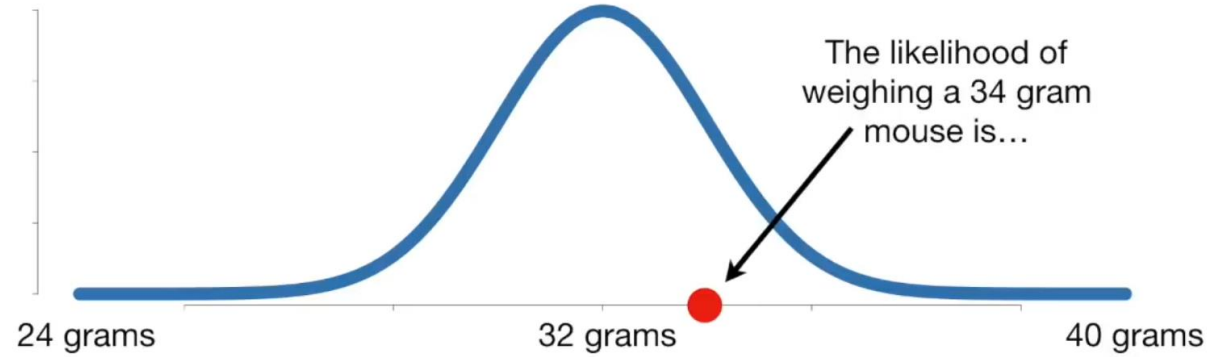


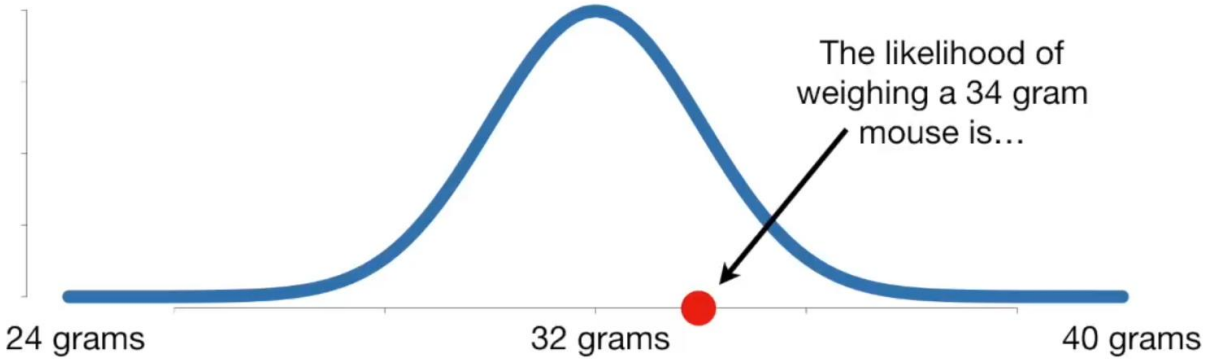
Now that we have probability worked out, let's talk about *likelihood*…

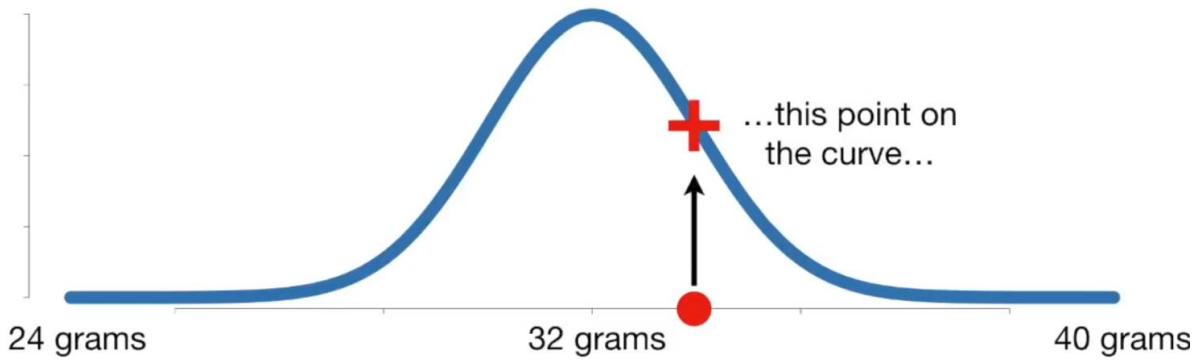


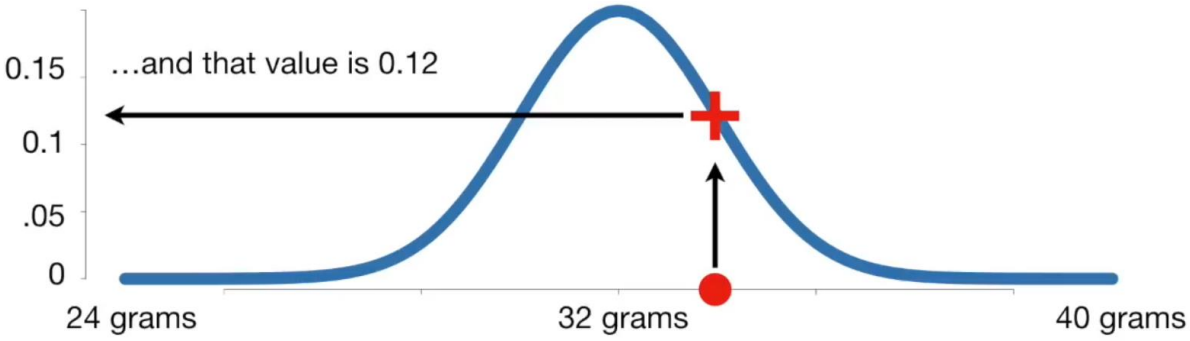

To talk about a likelihood, you assume that you have already weighed your mouse (or mice, if you weighed more than one)…
24 grams
32 grams
40 grams


So here's our mouse. It weighs 34 grams.
24 grams
32 grams
40 grams


The likelihood of weighing a 34 gram mouse is…
24 grams
32 grams
40 grams

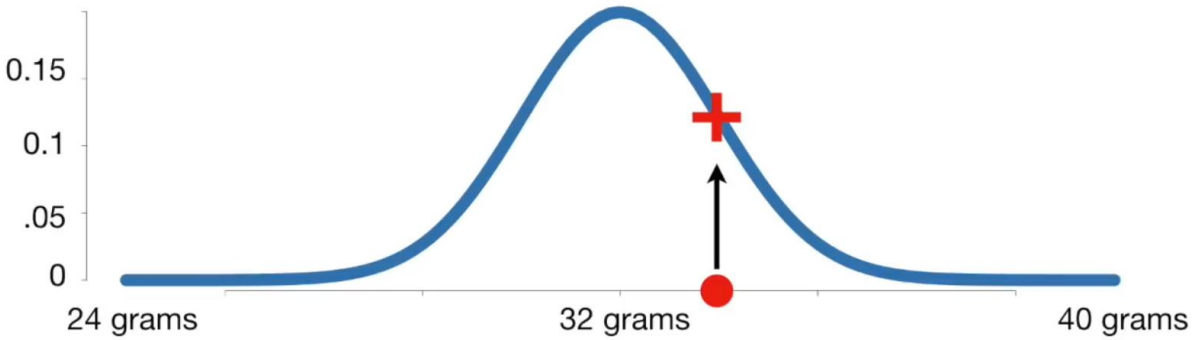

The likelihood of weighing a 34 gram mouse is…
24 grams
32 grams
40 grams

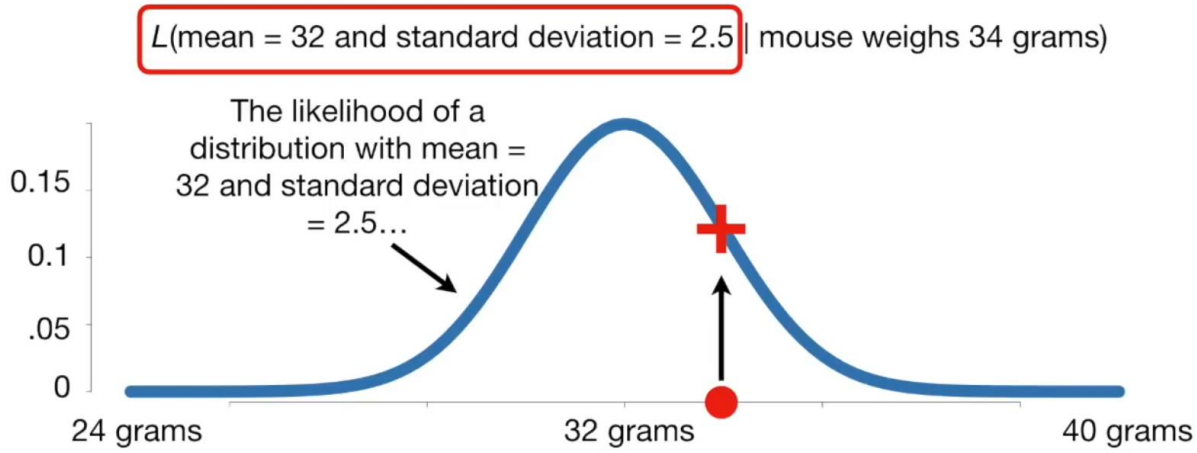


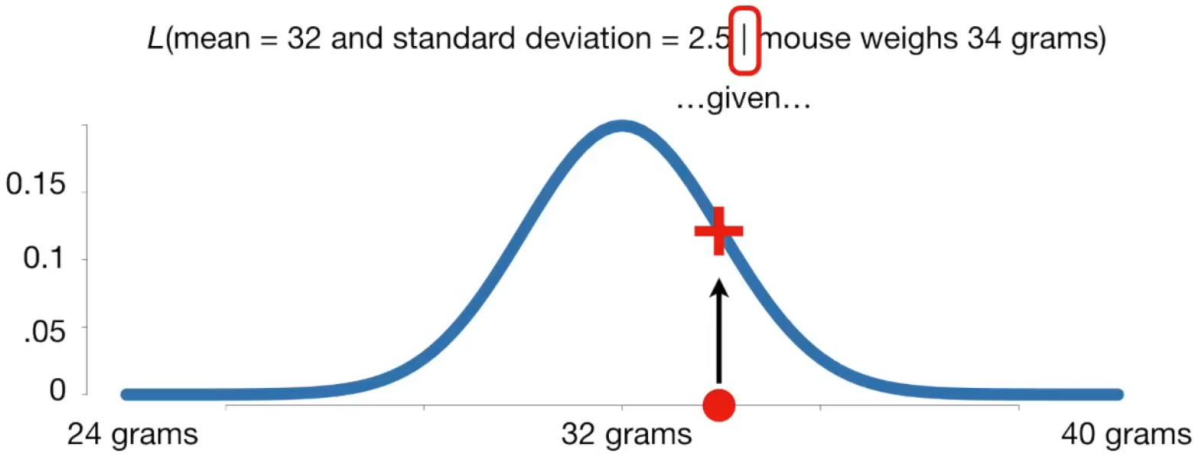


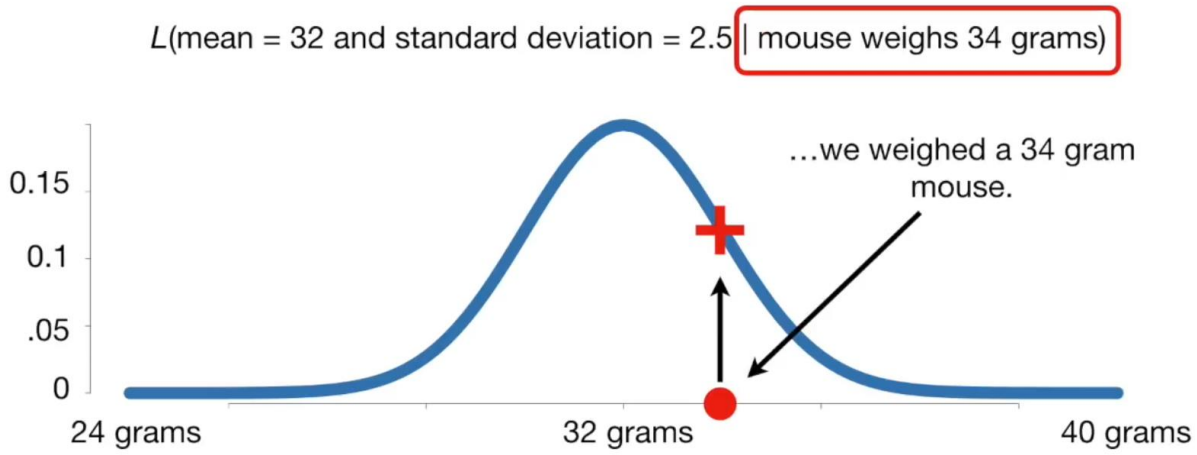
Mathematically, we say this with the following notation….

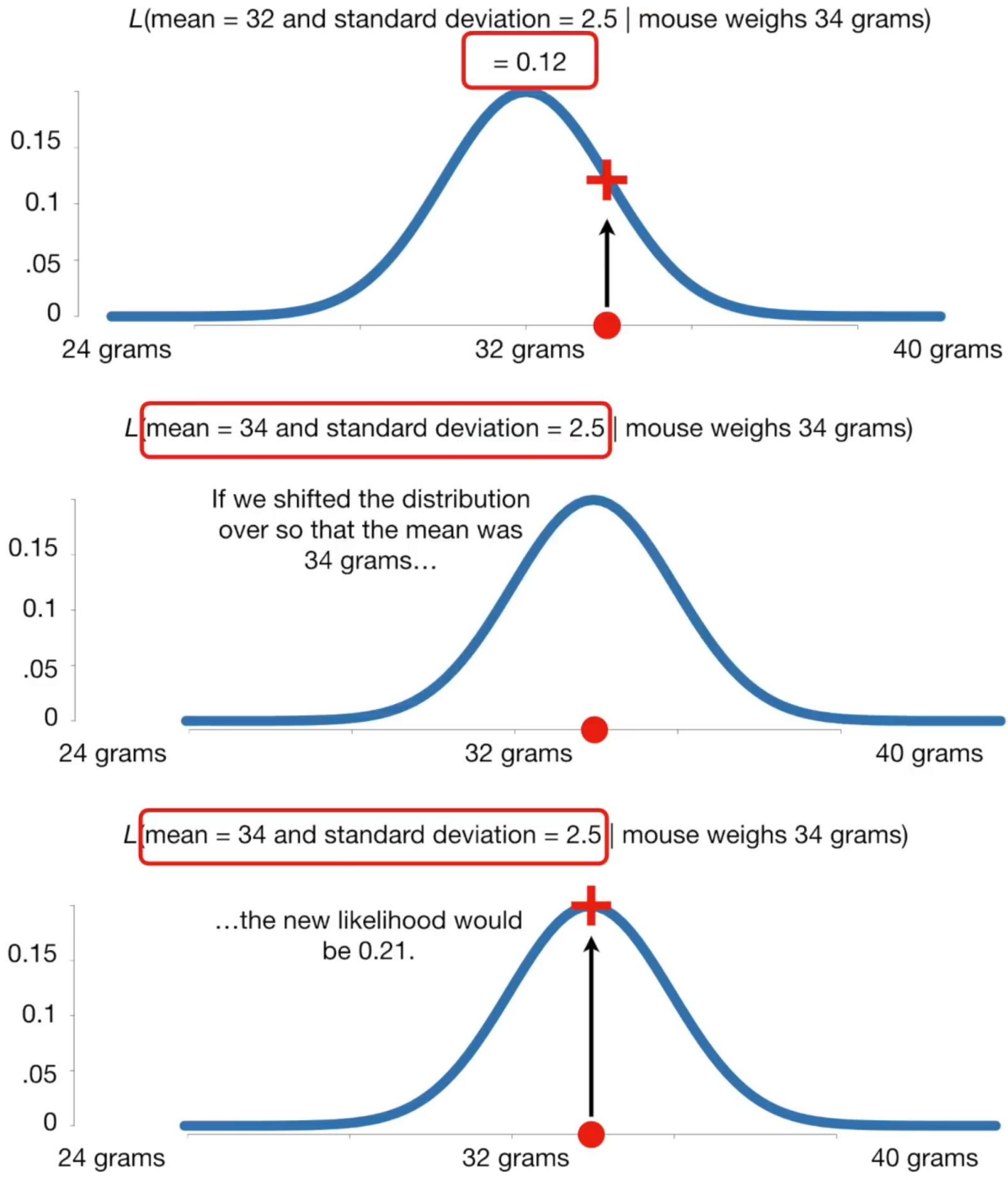
L(mean = 32 and standard deviation = 2.5 | mouse weighs 34 grams)

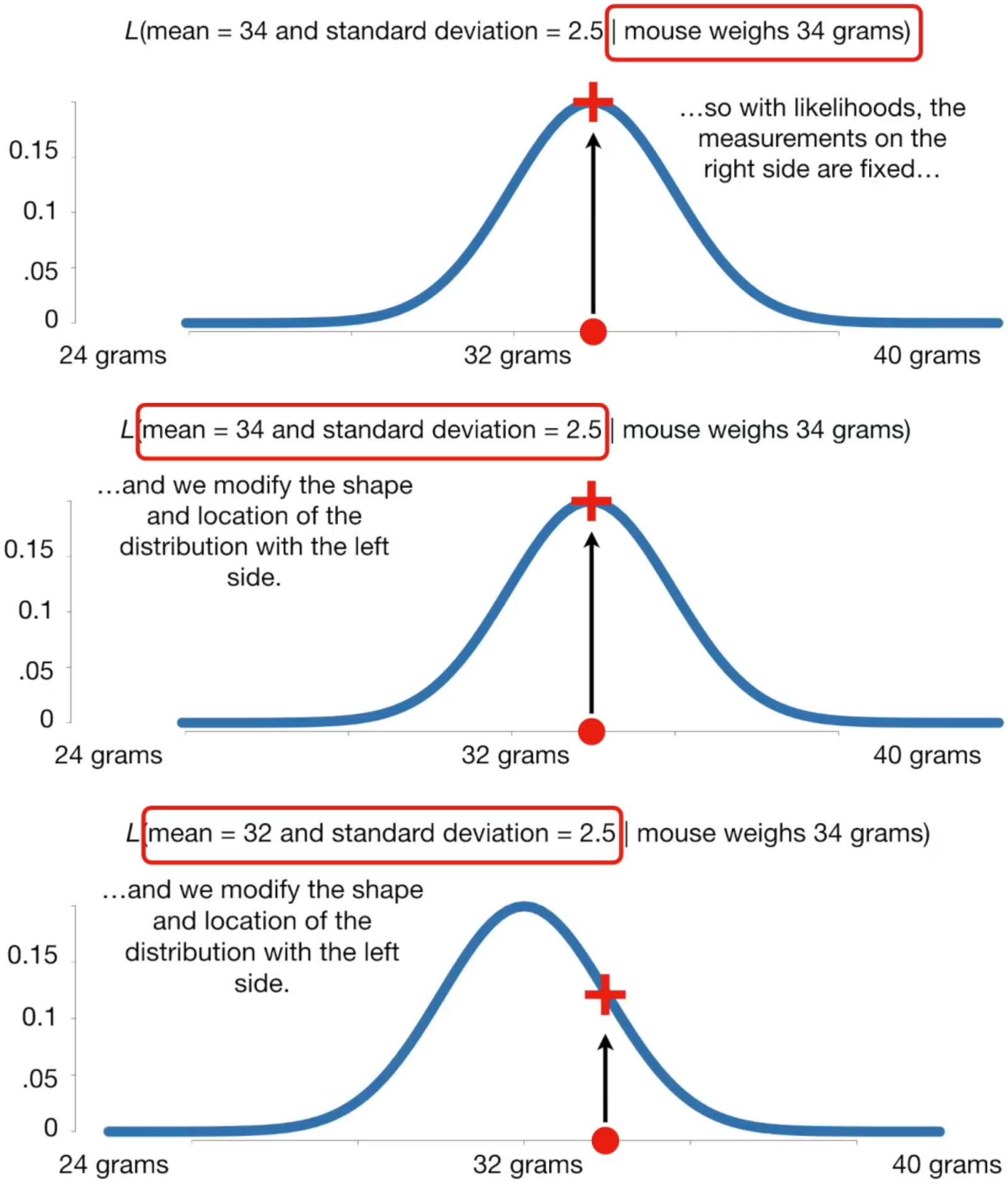


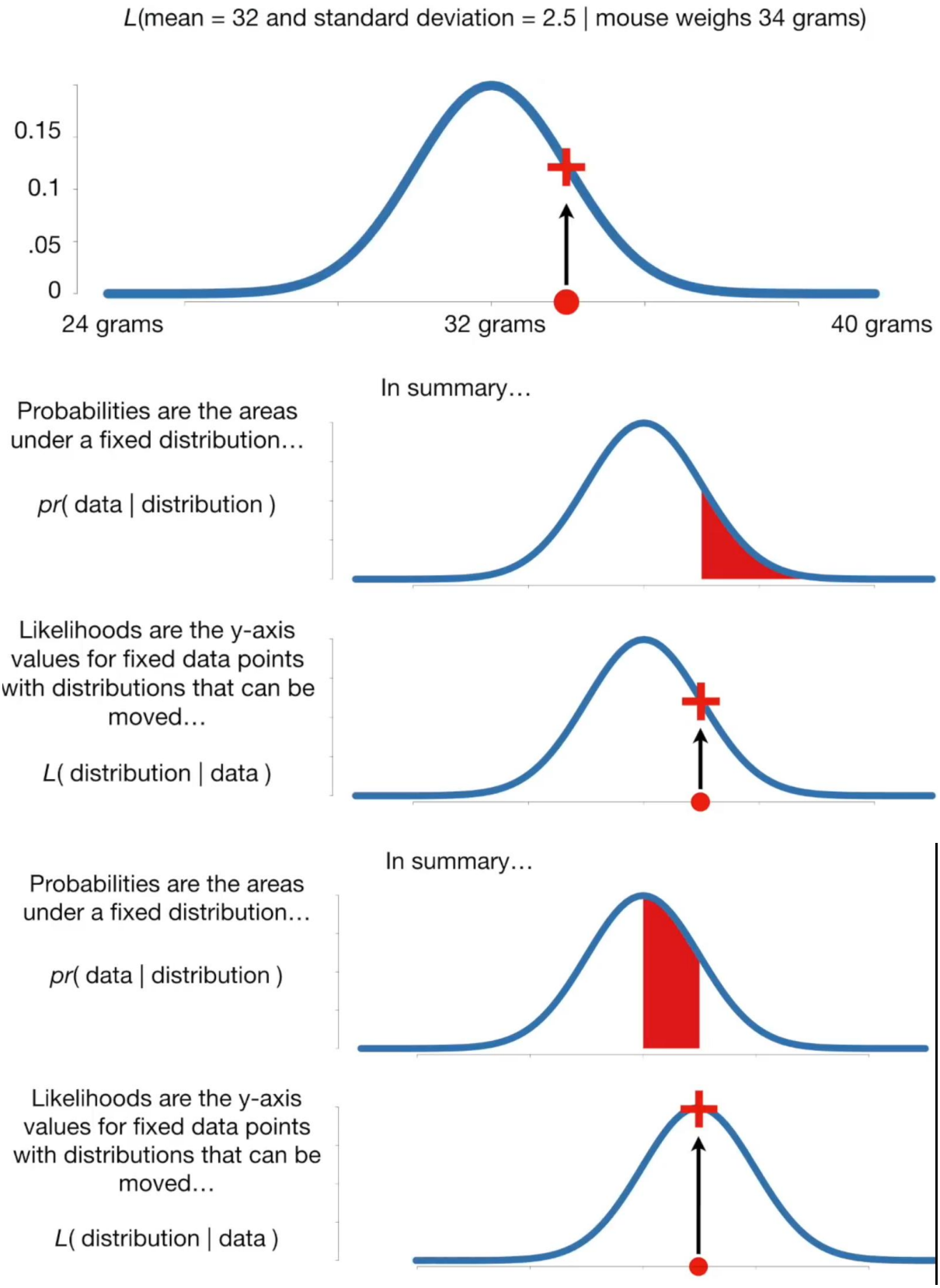

L(mean = 32 and standard deviation = 2.5 | mouse weighs 34 grams)
The likelihood of a distribution with mean = 32 and standard deviation = 2.5…
0.15
0.1
.05
0
24 grams
32 grams
40 grams


L(mean = 32 and standard deviation = 2.5 | mouse weighs 34 grams)
…given…
0.15
0.1
.05
0
24 grams
32 grams
40 grams


L(mean = 32 and standard deviation = 2.5 | mouse weighs 34 grams)
…we weighed a 34 gram mouse.
0.15
0.1
.05
0
24 grams
32 grams
40 grams


L(mean = 32 and standard deviation = 2.5 | mouse weighs 34 grams)
= 0.12
0.15
0.1
.05
0
24 grams
32 grams
40 grams
L(mean = 34 and standard deviation = 2.5 | mouse weighs 34 grams)
If we shifted the distribution over so that the mean was 34 grams…
0.15
0.1
.05
0
24 grams
32 grams
40 grams
L(mean = 34 and standard deviation = 2.5 | mouse weighs 34 grams)
…the new likelihood would be 0.21.
0.15
0.1
.05
0
24 grams
32 grams
40 grams


L(mean = 34 and standard deviation = 2.5 | mouse weighs 34 grams)
…so with likelihoods, the measurements on the right side are fixed…
0.15
0.1
.05
0
24 grams
32 grams
40 grams
L(mean = 34 and standard deviation = 2.5 | mouse weighs 34 grams)
…and we modify the shape and location of the distribution with the left side.
0.15
0.1
.05
0
24 grams
32 grams
40 grams
L(mean = 32 and standard deviation = 2.5 | mouse weighs 34 grams)
…and we modify the shape and location of the distribution with the left side.
0.15
0.1
.05
0
24 grams
32 grams
40 grams


L(mean = 32 and standard deviation = 2.5 | mouse weighs 34 grams)
0.15
0.1
.05
0
24 grams
32 grams
40 grams
In summary…
Probabilities are the areas under a fixed distribution…
pr(data | distribution)
Likelihoods are the y-axis values for fixed data points with distributions that can be moved…
L(distribution | data)
In summary…
Probabilities are the areas under a fixed distribution…
pr(data | distribution)
Likelihoods are the y-axis values for fixed data points with distributions that can be moved…
L(distribution | data)

One last thing…

If you want to see the actual equations, check out the StatQuest that derives the maximum likelihood estimator for the Exponential Distribution.

$$\text{pr}(x_1, x_2, \ldots, x_n \mid \lambda) = \text{pr}(x_1 \mid \lambda)\text{pr}(x_2 \mid \lambda) \ \ldots \ \text{pr}(x_n \mid \lambda)$$

$$= \lambda e^{-\lambda x_1} \lambda e^{-\lambda x_2} \ldots \lambda e^{-\lambda x_n}$$

$$= \lambda^n \left[e^{-\lambda x_1} e^{-\lambda x_2} \ldots e^{-\lambda x_n} \right]$$

$$= \lambda^n \left[e^{-\lambda (x_1 + x_2 + \ldots + xn)} \right]$$

The Gaussian Distribution

The Gaussian, also known as the normal distribution, is a widely used model for the distribution of continuous variables. In the case of a single variable x, the Gaussian distribution can be written in the form

$$\mathcal{N}(x|\mu,\sigma^2) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\} \tag{2.42}$$

where μ is the mean and σ^2 is the variance. For a D-dimensional vector $\mathbf{x}$, the multivariate Gaussian distribution takes the form

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{D/2}}\frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\} \tag{2.43}$$

where $\boldsymbol{\mu}$ is a D-dimensional mean vector, $\boldsymbol{\Sigma}$ is a $D \times D$ covariance matrix, and $|\boldsymbol{\Sigma}|$ denotes the determinant of $\boldsymbol{\Sigma}$.

Another situation in which the Gaussian distribution arises is when we consider the sum of multiple random variables. The *central limit theorem* (due to Laplace) tells us that, subject to certain mild conditions, the sum of a set of random variables, which is of course itself a random variable, has a distribution that becomes increasingly Gaussian as the number of terms in the sum increases (Walker, 1969). We can illustrate this by considering N variables $x_1, \ldots, x_N$ each of which has a uniform distribution over the interval $[0, 1]$ and then considering the distribution of the mean $(x_1 + \cdots + x_N)/N$. For large N, this distribution tends to a Gaussian

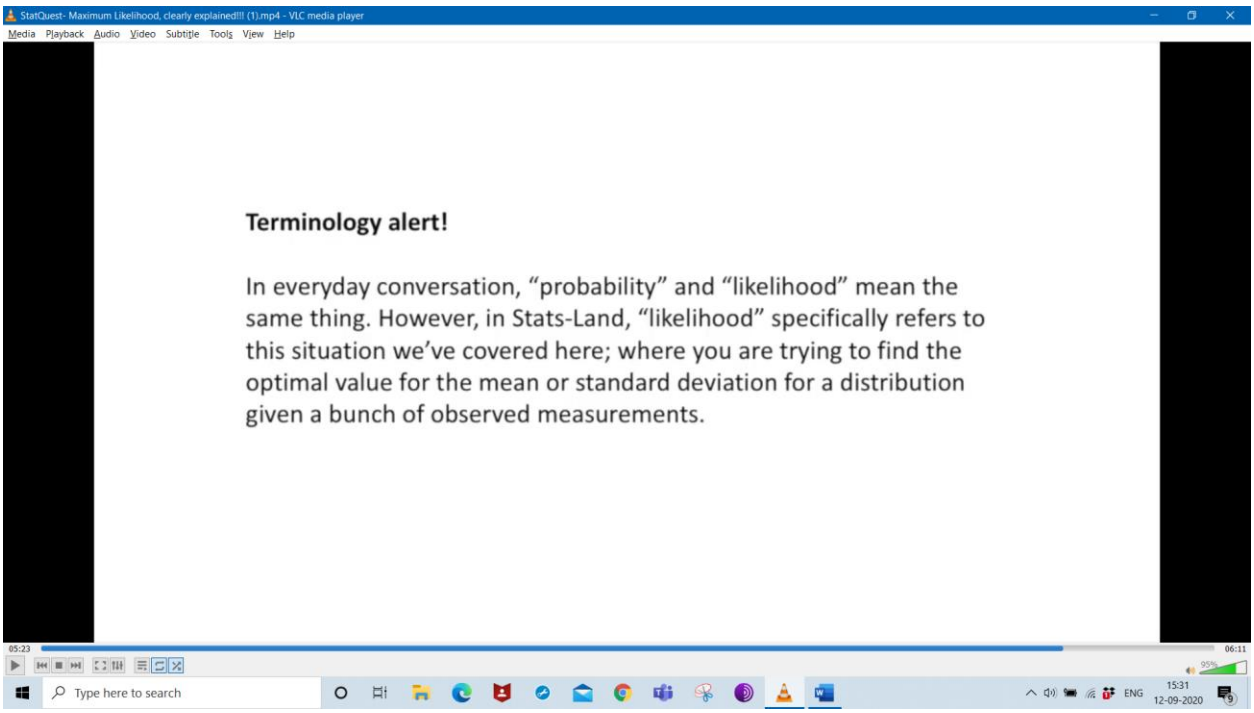
In practice, the convergence to a Gaussian as *N* increases can be very rapid.

One consequence of this result is that the binomial distribution, which is a distribution over *m* defined by the sum of *N* observations of the random binary variable *x*, will tend to a Gaussian as $N \to \infty$.

We begin by considering the geometrical form of the Gaussian distribution. The functional dependence of the Gaussian on $\mathbf{x}$ is through the quadratic form

$$\Delta^2 = (\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}) \tag{2.44}$$

which appears in the exponent. The quantity Δ is called the *Mahalanobis distance* from $\boldsymbol{\mu}$ to $\mathbf{x}$ and reduces to the Euclidean distance when $\boldsymbol{\Sigma}$ is the identity matrix. The Gaussian distribution will be constant on surfaces in $\mathbf{x}$-space for which this quadratic form is constant.



Maximum Likelihood Estimation For Regression

Maximum likelihood estimation or otherwise noted as MLE is a popular mechanism which is used to estimate the model parameters of a regression model. Other than regression, it is very often used in statics to estimate the parameters of various distribution models.

A section wise summary of the artical is as follows. Although post is written with assumption of reader being started from begining, feel free to jump to any section at your desire.

-Normal / Gaussian distribution
-Binomial distribution
-What is MLE
-How to Calculate Likelihood
-How to use MLE for linear regression

Normal / Gaussian distribution

In probability the normal or gaussian distribution is a very famous continous probability distribution. It basically depends on two factors — the mean and the standard deviation. The mean of the distribution determines the location of the center of the graph and the standard deviation determines the height and width of the graph. So notation of normal distribution becomes:

$$N(\mu, \sigma^2) \quad ; \ \mu \text{ - mean} \ , \ \sigma^2 \text{ - variance}$$

PDF of Normal distribution is:

$$f(x \mid \mu,\ \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

Note: from a pdf function, we can get the probability related to a data point x. So by substituting values for x, the relevant probabilities can be obtained. We can think of PDFs as models and that they are defined by their parameters. So parameters of normal distribution are : μ *(mean)* and σ^2 *(variance)*

Binomial distribution:

When a trail of two outcomes (as success and fail) is repeated for n times and when the probabilities of "*number of success event*" is logged, the resultant distribution is called a binomial distribution. For an example lets toss a coin for 10 times ($n = 10$) and the success is getting head. So if we log the probabilities of getting head only one times, two times, three times, … then that distribution of probabilities is in a binomial distribution.

Hence the binomial distribution depends mainly upon "number of trials" and "probability of success in an individual trail". Thus the notation of binomial distribution is:

$b(x; n, P)$;n - number of trials, P - success probability of an individual trial

PDf of Binomial Distribution:

$$b(x; n, P) = {}^{n}C_{x} \cdot P^{x} \cdot (1 - P)^{n-x}$$

So n and P are the parameters of a Binomial distribution.

Maximum Likelihood Estimation

Maximum likelihood estimation (MLE) is a technique used for estimating the parameters of a given distribution, using some observed data. For example, if a population is known to follow a "normal distribution" but the "mean" and "variance" are unknown, MLE can be used to estimate them using a limited sample of the population. MLE does that by finding particular values for the parameters (mean and variance) so that the resultant model with those parameters (mean and variance) would have generated the data.

So generally, likelihood expression is in the form of: ***L(parameters | data).*** Meaning of this is, "*likelihood of having these parameters, once the data are these*".

Likelihood and Probability are two different things although they look and behaves same. We talk about probability when we know the model parameters and when predicting a value from that model. So there we talk about how probable is the resultant value to be come out from that model. So probability is: ***P(data | parameters)***

Now we can see that Likelihood is other side of probability. That is we are going to guess the model parameters from the data. So there we know the results well and we know for sure that they have occured (probability = 1).

A simple example

Suppose we have 3 data points as 2, 2.5 and 3. And let's suppose that these values are from a normal/gaussian distribution. So now we have some data from a model (here model being gaussian), but we don't know it's parameters. Let's see how to estimate model parameters from MLE.

First let's calculate some Likelihoods after assuming some values for parameters. For instance let's think mean and variance as 2 and 1 for one instance and for another instance let's assume they are as 4 and 2.

Calculating Likelihood

It's very important to undestand that likelihood is also calculated from PDF functions but by calculating the joint probabilities of data points from a particular PDF function. That is, we can write likelihood calculation as:

$$L(\text{parameters} \mid \text{data}) = \prod_{i=1}^{n} f(\text{data}_i \mid \text{parameters})$$

So likelihood when model is $N(\mu=2, \sigma^2=1)$:

$$L(\mu=2, \sigma^2=1 \mid x = 2, 2.5, 3) = PDF(x = 2) \times PDF(x = 2.5) \times PDF(x = 3)$$
$$L(2, 1 \mid x = 2, 2.5, 3) = f(x = 2 \mid 2, 1) \cdot f(x = 2.5 \mid 2, 1) \cdot f(x = 3 \mid 2, 1)$$

In above we have calculated the joint probability of 3 points assuming that they are independent. Recall that we calculate the probability from the PDF of that particular distribution.

$$L(2, 1 \mid x = 2, 2.5, 3) = \frac{1}{\sqrt{2\pi 1}} e^{-\frac{(2-2)^2}{2.1}} \cdot \frac{1}{\sqrt{2\pi 1}} e^{-\frac{(2.5-2)^2}{2.1}} \cdot \frac{1}{\sqrt{2\pi 1}} e^{-\frac{(3-2)^2}{2.1}}$$
$$L(2, 1 \mid x = 2, 2.5, 3) = 0.39 * 0.35 * 0.24 = 0.03276$$

Likelihood when model is N(4,2):

$$L(\mu=4, \sigma^2=2 \mid x = 2, 2.5, 3) = PDF(x = 2) \times PDF(x = 2.5) \times PDF(x = 3)$$

$$L(4, 2 \mid x = 2, 2.5, 3) = \frac{1}{\sqrt{2\pi.2}} e^{-\frac{(2-4)^2}{2.2}} \cdot \frac{1}{\sqrt{2\pi.2}} e^{-\frac{(2.5-4)^2}{2.2}} \cdot \frac{1}{\sqrt{2\pi.2}} e^{-\frac{(3-4)^2}{2.2}}$$

$$L(4, 2 \mid x = 2, 2.5, 3) = 0.10 * 0.16 * 0.21 = 0.00336$$

So now we can see that likelihood of parameter values being $\mu=2$, $\sigma^2=1$ is greater than them being $\mu=4$, $\sigma^2=2$. But how can we calculate the precise values for μ and σ^2 ? So in mathematics, to find values regarding optimizations, derivation is used.

So what we do is we write the likelihood calculation function for all the known/selected data points as follows.

$$L(\mu, \sigma^2 \mid x = 2, 2.5, 3) = \prod_{i=1}^{3} f(x_i \mid \mu, \sigma^2)$$

$$L(\mu, \sigma^2 \mid x = 2, 2.5, 3) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(2-\mu)^2}{2\sigma^2}} \cdot \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(2.5-\mu)^2}{2\sigma^2}} \cdot \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(3-\mu)^2}{2\sigma^2}}$$

$$L(\mu, \sigma^2 \mid x = 2, 2.5, 3) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^3 e^{-\frac{(2-\mu)^2}{2\sigma^2}} \cdot e^{-\frac{(2.5-\mu)^2}{2\sigma^2}} \cdot e^{-\frac{(3-\mu)^2}{2\sigma^2}}$$

Let's take the natural logarithms in both sides,

$$\ln(L(\mu, \sigma^2 \mid x = 2, 2.5, 3)) = \ln\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^3 + \ln\left(e^{-\frac{(2-\mu)^2}{2\sigma^2}}\right) + \ln\left(e^{-\frac{(2.5-\mu)^2}{2\sigma^2}}\right) + \ln\left(e^{-\frac{(3-\mu)^2}{2\sigma^2}}\right)$$

$$= 3\ln(1) - 3\ln(\sqrt{2\pi\sigma^2}) - \frac{(2-\mu)^2}{2\sigma^2} - \frac{(2.5-\mu)^2}{2\sigma^2} - \frac{(3-\mu)^2}{2\sigma^2}$$

$$= -3\ln(\sqrt{2\pi\sigma^2}) - \frac{(2-\mu)^2}{2\sigma^2} - \frac{(2.5-\mu)^2}{2\sigma^2} - \frac{(3-\mu)^2}{2\sigma^2}$$

$$= -3\ln(\sqrt{2\pi\sigma^2}) - \frac{1}{2\sigma^2}(4-4\mu+\mu^2+6.25-5\mu+\mu^2+9-6\mu+\mu^2)$$

$$\ln(L(\mu, \sigma^2 \mid x = 2, 2.5, 3)) = -3\ln(\sqrt{2\pi\sigma^2}) - \frac{1}{2\sigma^2}(19.25 - 15\mu + 3\mu^2)$$

Now let's find Maximum likelihood estimators for μ. For that lets partially differentiate the above equation with respect to μ.

$$\frac{\partial(ln(L(\mu,\sigma^2|x))}{\partial\mu} = \frac{1}{2\sigma^2}(0 - 15 + 6\mu)$$

In order to find maximum(optimal) likelihood estimators let's equal above value to 0.

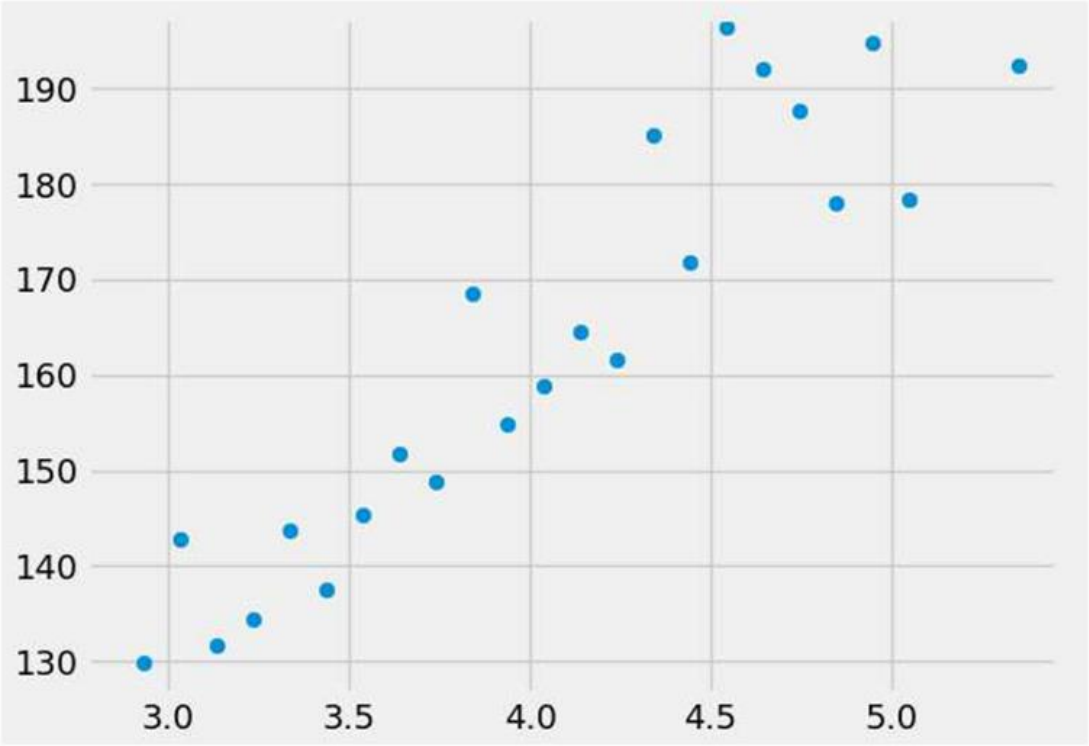
$$\frac{1}{2\sigma^2}(0 - 15 + 6\mu) = 0$$
$$\mu = 15/6 = 2.5$$

So now we know what is the MLE of μ. Like this we can get the MLE of σ^2 also by derivative w.r.t σ^2.

MLE for Linear Regression

As we have used likelihood calculation to find the best parameter values for various distribution models in statistics, MLE method can also be used to find the best model parameters of a linear regression model. But when calculating parameters values for those statistical distribution models, we knew what kind of distributions was it and the relevant PDF function. What kind of distribution are we going to use in linear regression? With that question, we can talk about the main assumption in linear regression:

> "The independent variable (y values) is assumed be in a normal distribution"



An example data set. Blue dots are labels (y values)

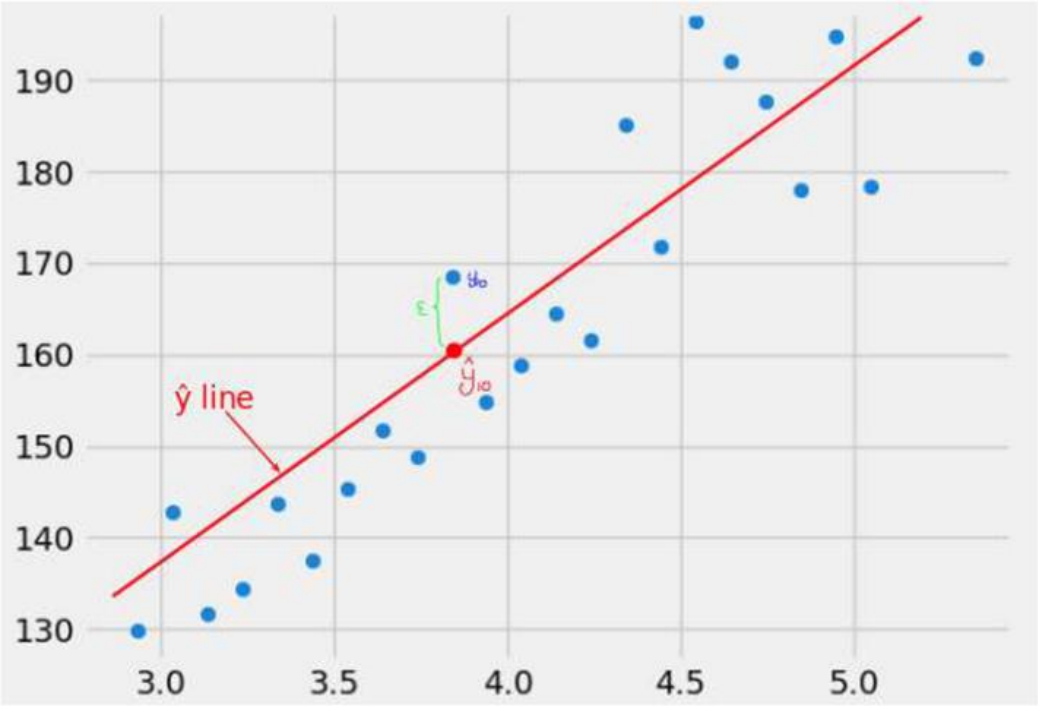
But that's not a full description. When we talk about some values being in a normal distribution, we need to describe more about that normal distribution ; like what kind of normal distribution? More precisely what are the mean and variance of that normal distribution?

In linear regression the trick that we do is, **we take the model that we need to find, as the mean** of the above stated normal distribution. Because we know how to find MLE values of a mean in a normal distribution.

So let's define our linear model that needed to be estimated as $\hat{y}$.

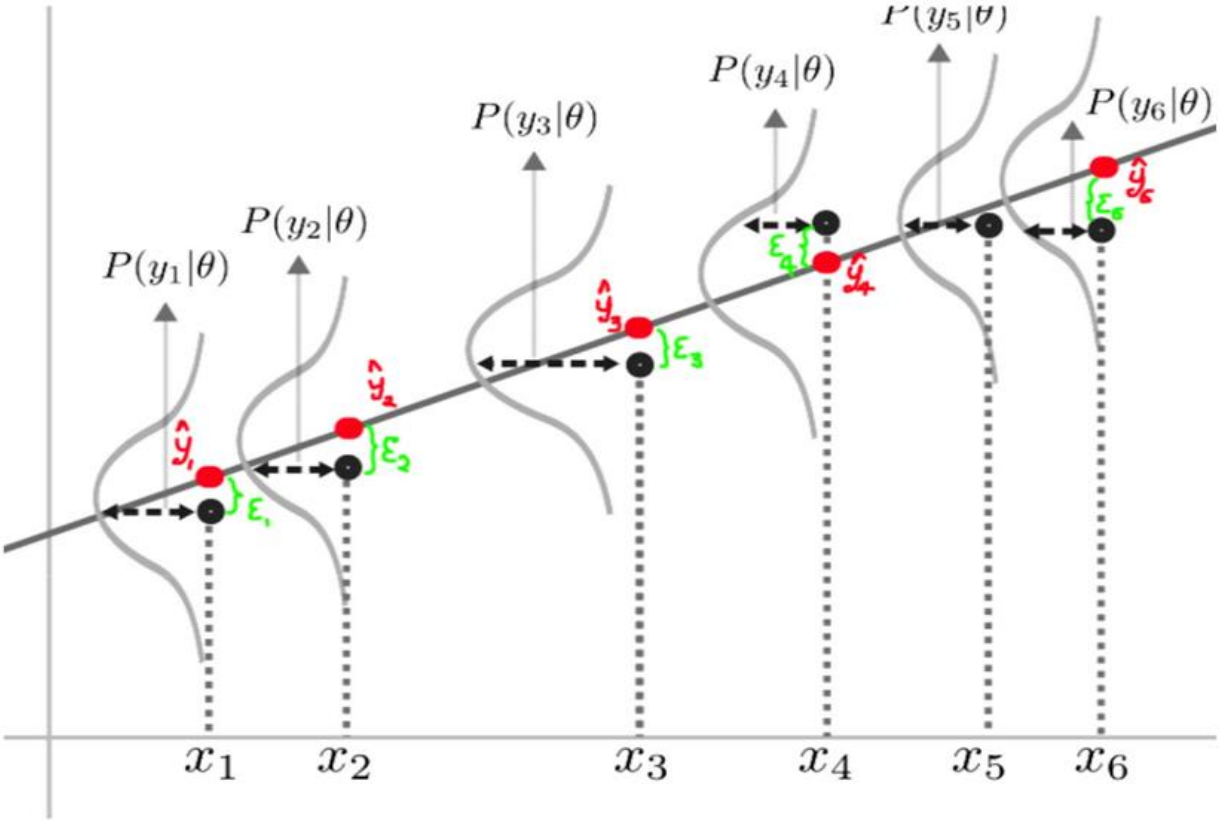
$$\hat{y} = w_0 + w_1x_1 + \ldots + w_dx_d \quad \text{- Model to be predicted}$$

each y value (label) is the value predicted by our model.



As an example, in the above figure, we know that blue dots are labels (y) and they are assumed to be from a normal distribution. And mean for each is not a fixed number, while it's the corresponding value from the ŷ function. So the mean for label y-10 is ŷ-10

and we should understand the graph as follows. That is each labels in the data set have their own mean and variance in a normal distribution.



Note: ŷ is called a linear model as there are only variables with degree one and lesser. If there were variables with degree 2, then it's not a linear model

Note: $\hat{y}$ is called a linear model as there are only variables with degree one and lesser. If there were variables with degree 2, then it's not a linear model

As stated above In linear regression, we treat above line $\hat{y}$ as the "mean" of the normal distribution.

$$\hat{y} = w_0 + w_1 x_1 + \ldots + w_d x_d$$
$$\hat{Y}_{n*1} = X_{n*d}\, W_{d*1} \quad ; \text{in vector form, n - number of records}$$

We can consider this $\hat{y}$ data as also in a normal distribution. But this time, their mean values will be them self since they fall along on top of the $\hat{y}$ line perfectly. And so the variance of these $\hat{y}$ data (predicted labels) will be 0. So, $\hat{y} \sim N(XW, 0)$

We have an error term called ε (residual) which is the distance between predicted value ($\hat{y}$) and actual value(y). And there are some important assumptions that we do in linear

regression regarding these residuals, and they are:

"Residuals are normally distributed"
"Residuals have an equal variance"
"Means of residuals are 0"

So:

$$\varepsilon \sim N(0 , \sigma^2)$$

And we know that $y = \hat{y} + \varepsilon$ and y labels are normally distributed. Our aim is to estimate the best values for mean and variance of normal distribution y. Let's get the mean and variance of y in terms of $\hat{y}$ and ε normal distributions. We know the mean is termed as expectation. So let's get the expectation of $y=\hat{y} + \varepsilon$ equation in order to find the mean (expectation) of y.

$$E(y) = E(\hat{y} + \varepsilon)$$
$$E(y) = E(\hat{y}) + E(\varepsilon)$$
$$E(y) = XW + 0 = XW$$

And,

$$\text{Variance}(y) = \text{Variance}(\hat{y} + \varepsilon)$$
$$\text{Variance}(y) = \text{Variance}(\hat{y}) + \text{Variance}(\varepsilon)$$
$$\text{Variance}(y) = 0 + \sigma^2$$

So we can say that y is a normal distribution with mean XW and variance σ^2.

$$y \sim N(XW , \sigma^2)$$

Now let's calculate the MLEs for XW and σ^2 as we did in previous example. But note that here we hava n data points (in earlier example we had only 3), and each of those data point are of dimension d.

$$L(XW, \sigma^2 | x) = \prod_{i=1}^{n} f_y(y_i, x_i w, \sigma^2)$$

As y is a normal distribution,

$$L(XW,\ \sigma^2|\ x\ ) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y_i - x_i w)^2}{2\sigma^2}}$$

$$L(XW,\ \sigma^2|\ x\ ) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n . e^{-\frac{\sum_{i=1}^{n} (y_i - x_i w)^2}{2\sigma^2}}$$

$$L(XW,\ \sigma^2|\ x\ ) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n . e^{-\frac{(Y - XW)^T (Y - XW)}{2\sigma^2}}$$

Let's get natural logarithms in both sides so that we get the log likelihood,

$$ln(\ L(XW,\ \sigma^2|\ x\ )\ ) = -\frac{n}{2} ln(2\pi) - \frac{n}{2} ln(\sigma^2) - \frac{(Y - XW)^T (Y - XW)}{2\sigma^2}$$

In order to estimate the best set of weights (weight matrix), let's partially differentiate the above equation from w.

$$\frac{\partial ln(\ L(XW, \sigma^2|\ x\ )\ )}{\partial w} = \frac{1}{2\sigma^2} \frac{\partial (Y - XW)^T (Y - XW)}{\partial w}$$

$$\frac{\partial ln(\,L(XW,\sigma^2|\,x\,)\,)}{\partial w} = \frac{1}{2\sigma^2}\,\frac{\partial\,(Y^2 - 2X^T WY + X^T XW^2\,)}{\partial w}$$

$$\frac{\partial ln(\,L(XW,\sigma^2|\,x\,)\,)}{\partial w} = \frac{1}{2\sigma^2}\,(0 - 2X^T Y + 2X^T XW)$$

Optimal values for W is when

$$\frac{\partial ln(\,L(XW,\sigma^2|\,x\,)\,)}{\partial w} = 0$$

Then,

$$\frac{1}{2\sigma^2}\,(0 - 2X^T Y + 2X^T XW) = 0$$

$$W = \frac{X^T Y}{X^T X}$$

$$W = (X^T X\,)^{-1}\,X^T Y$$

So now we have found the optimal values for Ws in our model. And that is the main aim of linear regression since once found the w matrix, we can predict.

Maximum Likelihood Estimation for Linear Regression

Maximum Likelihood Estimation for Linear Regression

The purpose of this article series is to introduce a very familiar technique, Linear Regression, in a more rigourous mathematical setting under a probabilistic, supervised learning interpretation. This will allow us to understand the probability framework that will subsequently be used for more complex supervised learning models, in a more straightforward setting.

We will initially proceed by defining multiple linear regression, placing it in a probabilistic supervised learning framework and deriving an optimal estimate for its parameters via a technique known as maximum likelihood estimation.

In subsequent articles we will discuss mechanisms to reduce or mitigate the dimensionality of certain datasets via the concepts of subset selection and shrinkage. In addition we will utilise the Python Scitkit-Learn library to demonstrate linear regression, subset selection and shrinkage.

Many of these techniques will naturally carry over to more sophisticated models and will aid us significantly in creating effective, robust statistical methods for trading strategy development.

This article is significantly more mathematically rigourous than other articles have been to date. The rationale for this is to introduce you to the more advanced, probabilistic mechanism which pervades machine learning research. Once you have seen a few examples of simpler models in such a framework, it makes it easier to begin looking at the more advanced ML papers for useful trading ideas.

Linear Regression

Linear regression is one of the most familiar and straightforward statistical techniques. It is often taught at highschool, albeit in a simplified manner. It is also usually the first technique considered when studying supervised learning as it brings up important issues that affect many other supervised models.

Linear regression states that the response value y is a linear function of its feature inputs $\mathbf{x}$. That is:

$$y(\mathbf{x}) = \beta^T \mathbf{x} + \epsilon = \sum_{j-0}^{p} \beta_j x_j + \epsilon$$

Where $\beta^T, \mathbf{x} \in \mathbb{R}^{p+1}$ and $\epsilon \sim \mathcal{N}(\mu, \sigma^2)$. That is, β^T and $\mathbf{x}$ are both vectors of dimension $p + 1$ and ϵ, the *error* or *residual* term, is normally distributed with mean μ and variance σ^2. ϵ represents the difference between the predictions made by the linear regression and the true value of the response variable.

Note that β^T, which represents the transpose of the vector β, and $\mathbf{x}$ are both $p + 1$-dimensional, rather than p dimensional, because we need to include an intercept term. $\beta^T = (\beta_0, \beta_1, \ldots, \beta_p)$, while $\mathbf{x} = (1, x_1, \ldots, x_p)$. We must include the '1' in $\mathbf{x}$ as a notational "trick".

Probabilistic Interpretation

An alternative way to look at linear regression is to consider it as a joint probability model[2], [3]. That is, we are interested in the joint probability of how the behaviour of the response y is *conditional* on the values of the feature vector $\mathbf{x}$, as well as any parameters of the model, given by the vector θ. Thus we are interested in a model of the form $p(y \mid \mathbf{x}, \theta)$. This is a **conditional probability density** (CPD) model.

Linear regression can be written as a CPD in the following manner:

$$p(y \mid \mathbf{x}, \theta) = (y \mid \mu(\mathbf{x}), \sigma^2(\mathbf{x}))$$

For linear regression we assume that $\mu(\mathbf{x})$ is linear and so $\mu(\mathbf{x}) = \beta^T \mathbf{x}$. We must also assume that the variance in the model is fixed (i.e. that it doesn't depend on $\mathbf{x}$) and as such $\sigma^2(\mathbf{x}) = \sigma^2$, a constant. This then implies that our parameter vector $\theta = (\beta, \sigma^2)$.

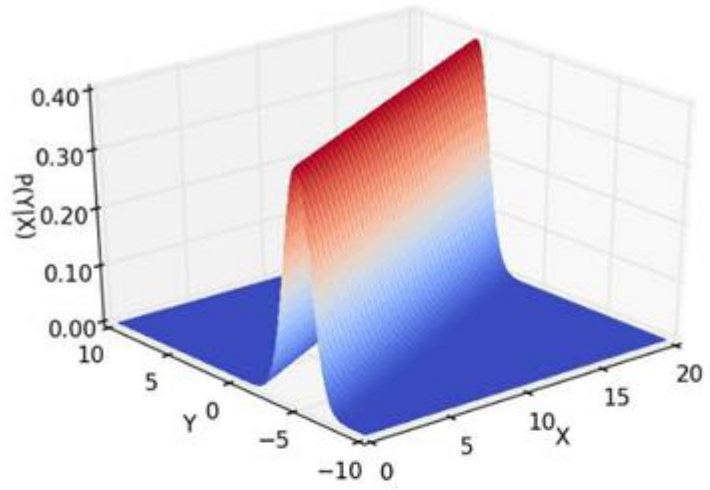
If you recall, we used such a probabilistic interpretation when we considered Bayesian Linear Regression in a previous article.

The benefit of generalising the model interpretation in this manner is that we can easily see how other models, especially those which handle non-linearities, fit into the same probabilistic framework. This allows us to derive results across models using similar techniques.

If we restrict $\mathbf{x} = (1, x)$, we can make a two-dimensional plot $p(y \mid \mathbf{x}, \theta)$ against y and x to see this joint distribution graphically. In order to do so we need to fix the parameters $\beta = (\beta_0, \beta_1)$ and σ^2 (which constitute the θ parameters). Here is a Python script which uses matplotlib to display the distribution:

```
1
2  from matplotlib import cm
3  from matplotlib.ticker import LinearLocator,
4  FormatStrFormatter
5  import matplotlib.pyplot as plt
6  import numpy as np
7  from scipy.stats import norm
8
9
10 if __name__ == "__main__":
11     # Set up the X and Y dimensions
12     fig = plt.figure()
13     ax = fig.gca(projection='3d')
14     X = np.arange(0, 20, 0.25)
15     Y = np.arange(-10, 10, 0.25)
16     X, Y = np.meshgrid(X, Y)
17
18     # Create the univarate normal coefficients
19     # of intercep and slope, as well as the
20     # conditional probability density
21     beta0 = -5.0
22     beta1 = 0.5
23     Z = norm.pdf(Y, beta0 + beta1*X, 1.0)
24
25     # Plot the surface with the "coolwarm" colormap
26     surf = ax.plot_surface(
27         X, Y, Z, rstride=1, cstride=1, cmap=cm.coolwarm,
28         linewidth=0, antialiased=False
29     )
30
31     # Set the limits of the z axis and major line locators
32     ax.set_zlim(0, 0.4)
33     ax.zaxis.set_major_locator(LinearLocator(5))
34
35 ax.zaxis.set_major_formatter(FormatStrFormatter('%.02f'))
```

```
36
37      # Label all of the axes
38      ax.set_xlabel('X')
39      ax.set_ylabel('Y')
40      ax.set_zlabel('P(Y|X)')
41
42      # Adjust the viewing angle and axes direction
43      ax.view_init(elev=30., azim=50.0)
44      ax.invert_xaxis()
45      ax.invert_yaxis()
46
        # Plot the probability density
        plt.show()
```



Plot of $p(y \mid \mathbf{x}, \theta)$ against y and x, influenced from a similar plot in Murphy (2012)[3].

It is clear that the respnse y is linearly dependent upon x. However we are also able to ascertain the probabilistic element of the model via the fact that the probability spreads normally around the linear response.

Basis Function Expansion

One of the benefits of utilising the probabilistic interpretation is that it allows us to easily see how to model non-linear relationships, simply by replacing the feature vector $\mathbf{x}$ with some transformation function $\phi(\mathbf{x})$:

$$p(y \mid \mathbf{x}, \theta) = (y \mid \beta^T \phi(\mathbf{x}), \sigma^2)$$

For $\mathbf{x} = (1, x_1, x_2, x_3)$, say, we could create a ϕ that includes higher order terms, including cross-terms, e.g.

$$\phi(\mathbf{x}) = (1, x_1, x_1^2, x_2, x_2^2, x_1 x_2, x_3, x_3^2, x_1 x_3, \ldots)$$

A key point here is that while this function is not linear in the *features*, $\mathbf{x}$, it is still linear in the *parameters*, β and thus is still called linear regression.

Such a modification, using a transformation function ϕ, is known as a **basis function expansion** and can be used to generalise linear regression to many non-linear data settings.

We won't discuss this much further in this article as there are many other more sophisticated supervised learning techniques for capturing non-linearities. We've already discussed one such technique, Support Vector Machines with the "kernel trick", at length in this article.

Maximum Likelihood Estimation

In this section we are going to see how optimal linear regression coefficients, that is the β parameter components, are chosen to best fit the data. In the univariate case this is often known as "finding the line of best fit". However,

we are in a multivariate case, as our feature vector $\mathbf{x} \in \mathbb{R}^{p+1}$. Hence we are "finding the p-dimensional hyperplane of best fit"!

The main mechanism for finding parameters of statistical models is known as **maximum likelihood estimation** (MLE). I introduced it briefly in the article on Deep Learning and the Logistic Regression. Here I will expand upon it further.

Likelihood and Negative Log Likelihood

The basic idea is that if the data were to have been *generated* by the model, what parameters were most likely to have been used? That is, what is the probability of seeing the data $\mathcal{D}$, given a specific set of parameters θ? Once again, this is a conditional probability density problem. We are seeking the values of θ that maximise $p(\mathcal{D} \mid \theta)$. This CPD is known as the **likelihood**, and you might recall seeing instances of it in the introductory article on Bayesian statistics.

This problem can be formulated as hunting for the mode of $p(\mathcal{D} \mid \theta)$, which is given by $\hat{\theta}$. For reasons of computational ease we instead try and maximise the natural logarithm of the CPD rather than the CPD itself:

$$\hat{\theta} = \text{argmax}_\theta \log p(\mathcal{D} \mid \theta)$$

In linear regression problems we need to make the assumption that the feature vectors are all **independent and identically distributed** (iid). This makes it far simpler to solve the **log-likelihood** problem, using properties of natural logarithms. Since we will be differentiating these values it is far easier to differentiate a sum than a product, hence the logarithm:

$$
\begin{aligned}
l(\theta) &:= \log p(\mathcal{D} \mid \theta) \\
&= \log \left(\prod_{i=1}^{N} p(y_i \mid \mathbf{x}_i, \theta) \right) \\
&- \sum_{i \ 1}^{N} \log p(y_i \mid \mathbf{x}_i, \theta)
\end{aligned}
$$

As I also mentioned in the article on Deep Learning/Logistic Regression, for reasons of increased computational ease, it is often easier to minimise the negative of the log-likelihood rather than maximise the log-likelihood itself. Hence, we can "stick a minus sign in front of the log-likelihood" to give us the **negative log-likelihood** (NLL):

$$
\mathrm{NLL}(\theta) = -\sum_{i=1}^{N} \log p(y_i \mid \mathbf{x}_i, \theta)
$$

This is the function we need to minimise. By doing so we will derive the **ordinary least squares** estimate for the β coefficients.

Ordinary Least Squares

Our goal here is to derive the optimal set of β coefficients that are "most likely" to have generated the data for our training problem. These coefficients will allow us to form a hyperplane of "best fit" through the training data. The process we will follow is given by:

1. Use the definition of the normal distribution to expand the negative log likelihood function
2. Utilise the properties of logarithms to reformulate this in terms of the Residual Sum of Squares (RSS), which is equivalent to the sum of each residual across all observations
3. Rewrite the residuals in matrix form, creating the data matrix X, which is $N \times (p+1)$ dimensional, and formulate the RSS as a matrix equation
4. Differentiate this matrix equation with respect to (w.r.t) the parameter vector β and set the equation to zero (with some assumptions on X)
5. Solve the subsequent equation for β to receive $\hat{\beta}_{\text{OLS}}$, the **ordinary least squares** (OLS) estimate.

The next section will closely follow the treatments of [2] and [3]. The first step is to expand the NLL using the formula for a normal distribution:

$$
\begin{aligned}
\mathrm{NLL}(\theta) &= -\sum_{i\ 1}^{N} \log p(y_i \mid \mathbf{x}_i, \theta) \\
&= -\sum_{i=1}^{N} \log \left[\left(\frac{1}{2\pi\sigma^2} \right)^{\frac{1}{2}} \exp\left(-\frac{1}{2\sigma^2} (y_i - \beta^T \mathbf{x}_i)^2 \right) \right] \\
&- -\sum_{i=1}^{N} \frac{1}{2} \log\left(\frac{1}{2\pi\sigma^2} \right) - \frac{1}{2\sigma^2} (y_i - \beta^T \mathbf{x}_i)^2 \\
&= -\frac{N}{2} \log\left(\frac{1}{2\pi\sigma^2} \right) - \frac{1}{2\sigma^2} \sum_{i-1}^{N} (y_i - \beta^T \mathbf{x}_i)^2 \\
&= -\frac{N}{2} \log\left(\frac{1}{2\pi\sigma^2} \right) - \frac{1}{2\sigma^2} \mathrm{RSS}(\beta)
\end{aligned}
$$

Where $\mathrm{RSS}(\beta) := \sum_{i=1}^{N} (y_i - \beta^T \mathbf{x}_i)^2$ is the **Residual Sum of Squares**, also known as the **Sum of Squared Errors** (SSE).

Since the first term in the equation is a constant we simply need to concern ourselves with minimising the RSS, which will be sufficient for producing the optimal parameter estimate.

To simply the notation we can write this latter term in matrix form. By defining the $N \times (p+1)$ matrix X we can write the RSS term as:

$$
\mathrm{RSS}(\beta) = (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta)
$$

At this stage we now want to differentiate this term w.r.t. the parameter variable β:

$$
\frac{\partial RSS}{\partial \beta} = -2\mathbf{X}^T (\mathbf{y} - \mathbf{X}\beta)
$$

There is an extremely key assumption to make here. We need $\mathbf{X}^T\mathbf{X}$ to be positive-definite, which is only the case if there are more observations than there are dimensions. If this is not the case (which is extremely common in high-dimensional settings) then it is not possible to find a *unique* set of β coefficients and thus the following matrix equation will not hold.

Under the assumption of a positive-definite $\mathbf{X}^T\mathbf{X}$ we can set the differentiated equation to zero and solve for β:

$$\mathbf{X}^T(\mathbf{y} - \mathbf{X}\beta) = 0$$

The solution to this matrix equation provides $\hat{\beta}_{\text{OLS}}$:

$$\hat{\beta}_{\text{OLS}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

Next Steps

Now that we have considered the MLE procedure for producing the OLS estimates we are in a position to discuss what happens when we are in a high-dimensional setting (as is often the case with real world data) and thus our matrix $\mathbf{X}^T\mathbf{X}$ has no inverse. In this instance we need to use subset selection and shrinkage techniques to reduce the dimensionality of the problem. This will be the subject of the next article.

Linear Regression

A unification of Maximum Likelihood Estimation and minimizing the sum of squares.

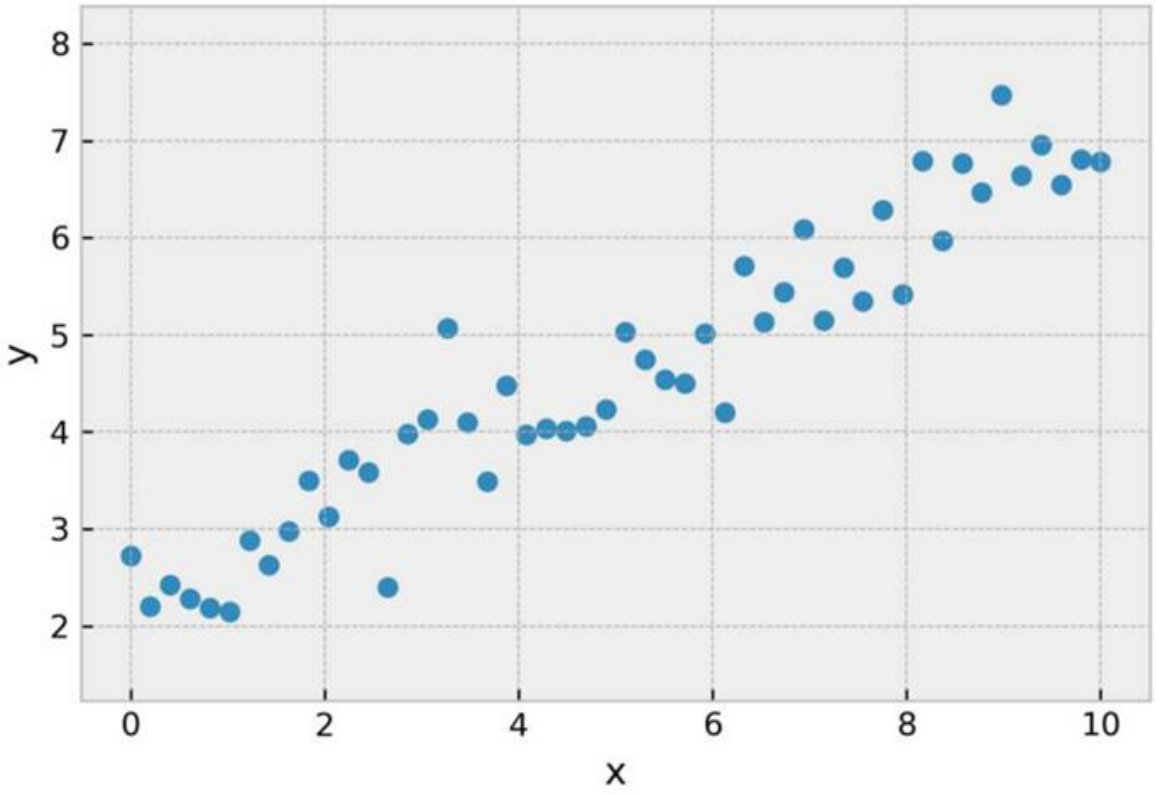
Introduction

I recently wrote about maximum likelihood estimation in my ongoing series on the fundamentals of machine learning:

In that post, we learned what it means to "model" data, and then how to use MLE to find the parameters of our model. In this post, we're going to dive into linear regression, one of the most important models in statistics, and learn how to frame it in terms of MLE. The solution is a beautiful piece of mathematics, which like most MLE models, is rich with intuition. I'll be assuming you've got a handle on the vocabulary I've covered in the other series (probability densities, conditional probabilities, likelihood function, iid data, etc.). If you see something here that you're uncomfortable with, check out the Probability and MLE posts from that series for clarity.

The Model

We use linear regression when our data has a linear relationship between the independent variables (our features) and the dependent variable (our target). In the MLE post, we saw some data that looked similar too this:



We observed that there appears to be a linear relationship between x and y, but it's not perfect. We think of these imperfections as coming from some error or noise process. Imagine drawing a line right through the cloud of points. The error for each point would be the distance from the point to our line. We'd like to explicitly include those errors in our model. One method of doing this, is to assume the errors are distributed from a Gaussian distribution with a mean of 0 and some unknown variance σ^2. The Gaussian seems like a good choice, because our errors look like they're symmetric about were the line would be, and that small errors are more likely than large errors. We write our linear model with Gaussian noise like this:

$$\epsilon \sim N(0, \sigma^2)$$
$$y = \theta_1 x + \theta_0 + \epsilon$$

Linear model with Gaussian noise term.

The error term is draw from our Gaussian, and then our observed y is then calculated by adding the error to the output of the linear equation. This model has three parameters: the slope and intercept of our line and the variance of the noise distribution. Our main goal is to find the best parameters for the slope and intercept of our line.

Likelihood Function

To apply maximum likelihood, we first need to derive the likelihood function. First, let's rewrite our model from above as a single conditional distribution given x:

$$y \sim N(\theta_1 x + \theta_0, \sigma^2)$$

Given x, y is drawn from a Gaussian centered on our line.

This is equivalent to pushing our x through the equation of the line and then adding noise from the 0 mean Gaussian.

Now, we can write the conditional distribution of y given x in terms of this Gaussian. This is just the equation of a Gaussian distribution's probability density function, with our linear equation in place of the mean:

$$f(y|x;\theta_0,\theta_1,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(y-(\theta_1 x+\theta_0))^2}{2\sigma^2}}$$

PDF of y given x and our linear model.

The semicolon in the conditional distribution acts just like a comma, but it's a useful notation for separating our observed data from the parameters.

Each point is independent and identically distributed (iid), so we can write the likelihood function with respect to all of our observed points as the product of each individual probability density. Since σ^2 is the same for each data point, we can factor out the term of the Gaussian which doesn't include x or y from the product:

$$L_X(\theta_0, \theta_1, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \prod_{(x,y)\in X} e^{\frac{-(y-(\theta_1 x+\theta_0))^2}{2\sigma^2}}$$

Likelihood for our collection of data X.

(Note: Thanks to Tongxin for catching an error here. The term before the product should be raised to the number of data points because I factored it from the product.)

Log-Likelihood:

The next step in MLE, is to find the parameters which maximize this function. To make our equation simpler, let's take the log of our likelihood. Recall, that maximizing the log-likelihood is the same as maximizing the likelihood since the log is monotonic. The natural log cancels out with the exponential, turns products into sums of logs, and division into subtraction of logs; so our log-likelihood looks much simpler:

$$l_X(\theta_0, \theta_1, \sigma^2) = \log[\frac{1}{\sqrt{2\pi\sigma^2}} \prod_{(x,y)\in X} e^{\frac{-(y-(\theta_1 x+\theta_0))^2}{2\sigma^2}}]$$

$$= \log(\frac{1}{\sqrt{2\pi\sigma^2}}) + \sum_{(x,y)\in X} \log(e^{\frac{-(y-(\theta_1 x+\theta_0))^2}{2\sigma^2}})$$

$$= \log(\frac{1}{\sqrt{2\pi\sigma^2}}) + \sum_{(x,y)\in X} \frac{-(y-(\theta_1 x+\theta_0))^2}{2\sigma^2}$$

$$= \log(1) - \log(\sqrt{2\pi\sigma^2}) - \frac{1}{2\sigma^2} \sum_{(x,y)\in X} [y - (\theta_1 x + \theta_0)]^2$$

$$= -\log(\sqrt{2\pi\sigma^2}) - \frac{1}{2\sigma^2} \sum_{(x,y)\in X} [y - (\theta_1 x + \theta_0)]^2$$

Derivation of log-likelihood for our model.

Sum of Squared Errors:

To clean things up a bit more, let's write the output of our line as a single value:

$$\hat{y} = \theta_1 x + \theta_0$$

Estimate of y from our line.

Now our log-likelihood can be written as:

$$l_X(\theta_0, \theta_1, \sigma^2) = -\log(\sqrt{2\pi\sigma^2}) - \frac{1}{2\sigma^2} \sum (y - \hat{y})^2$$

Simplified log-likelihood equation.

To remove the negative signs, let's recall that maximizing a number is the same thing as minimizing the negative of the number. So instead of maximizing the likelihood, let's minimize the negative log-likelihood:

$$-l_X(\theta_0, \theta_1, \sigma^2) = \log(\sqrt{2\pi\sigma^2}) + \frac{1}{2\sigma^2} \sum (y - \hat{y})^2$$

Minimize the negative log-likelihood.

Our ultimate goal is to find the parameters of our line. To minimize the negative log-likelihood with respect to the linear parameters (the θs), we can imagine that our variance term is a fixed constant.

Removing any constant's which don't include our θs won't alter the solution. Therefore, we can throw out any constant terms and elegantly write what we're trying to minimize as:

$$\sum(y - \hat{y})^2$$

Sum of squared error.

The maximum likelihood estimate for our linear model is the line which minimizes the sum of squared errors! This is a beautiful result, and you'll see that minimizing squared errors crops up everywhere in machine learning and statistics.

Solving for Parameters

We've concluded that the maximum likelihood estimates for our slope and intercept can be found by minimizing the sum of squared errors. Let's expand out our minimization objective and use i as our index over our n data points:

$$\begin{aligned}
\text{SSE} &= \sum_i^n (y_i - \hat{y}_i)^2 \\
&= \sum_i^n [y_i - (\theta_1 x_i + \theta_0)]^2 \\
&= \sum_i^n (y_i - \theta_1 x_i - \theta_0)^2
\end{aligned}$$

The square in the SSE formula makes it quadratic with a single minimum. The minimum can be found by taking the derivative with respect to each of the parameters, setting it equal to 0, and solving for the parameters in turn.

The Intercept:

Let's start by solving for the intercept. Taking the partial derivative with respect to the intercept and working through gives us:

$$\frac{\partial}{\partial \theta_0} \text{SSE} = -2 \sum_i^n (y_i - \theta_1 x_i - \theta_0)$$

$$= -2 \sum_i^n y_i + 2\theta_1 \sum_i^n x_i + 2n\theta_0$$

$$= -2n\bar{y} + 2\theta_1 n\bar{x} + 2n\theta_0$$

Derivative of SSE with respect to the intercept of our line.

The horizontal bars over the variables indicate the mean of those variables. We used the fact that the sum over the values of a variable is equal to the mean of those values multiplied by how many values we have. Setting the derivative equal to 0 and solving for the intercept gives us:

$$\theta_0 = \bar{y} - \theta_1 \bar{x}$$

MLE for intercept.

This is a pretty neat result. It's the equation of the line, with the means of the x's and y's in place of those variables. The intercept still depends on the slope, so we'll need to find that next.

The Slope:

We start by taking the partial derivative of the SSE with respect to our slope. We plug in our solution for the intercept and use algebra to isolate the slope term:

$$\frac{\partial}{\partial \theta_1} \text{SSE} = -2 \sum_i^n (y_i - \theta_1 x_i - \theta_0) x_i$$

$$= -2\sum_i^n x_i y_i + 2\theta_1 \sum_i^n x_i^2 + 2\theta_0 \sum_i^n x_i$$

$$= -2\sum_i^n x_i y_i + 2\theta_1 \sum_i^n x_i^2 + 2(\bar{y} - \theta_1 \bar{x}) \sum_i^n x_i$$

$$= -2\sum_i^n x_i y_i + 2\theta_1 \sum_i^n x_i^2 + 2\bar{y} \sum_i^n x_i - 2\theta_1 \bar{x} \sum_i^n x_i$$

$$= 2\theta_1 \sum_i^n x_i^2 - 2\theta_1 \bar{x} \sum_i^n x_i + 2\bar{y} \sum_i^n x_i - 2\sum_i^n x_i y_i$$

$$= 2\theta_1 (\sum_i^n x_i^2 - \bar{x} \sum_i^n x_i) + 2\bar{y} \sum_i^n x_i - 2\sum_i^n x_i y_i$$

Derivative of SSE with respect to the slope of our line.

Setting this equal to 0 and solving for the slope gives us:

$$\theta_1 = \frac{\sum_i^n x_i y_i - n\bar{y}\bar{x}}{\sum_i^n x_i^2 - n\bar{x}^2}$$

While we're technically done, we can use some fancier algebra to rewrite this without having to use the n:

$$\theta_1 = \frac{\sum_i^n (x_i - \bar{x})(y_i - \bar{y})}{\sum_i^n (x_i - \bar{x})^2}$$

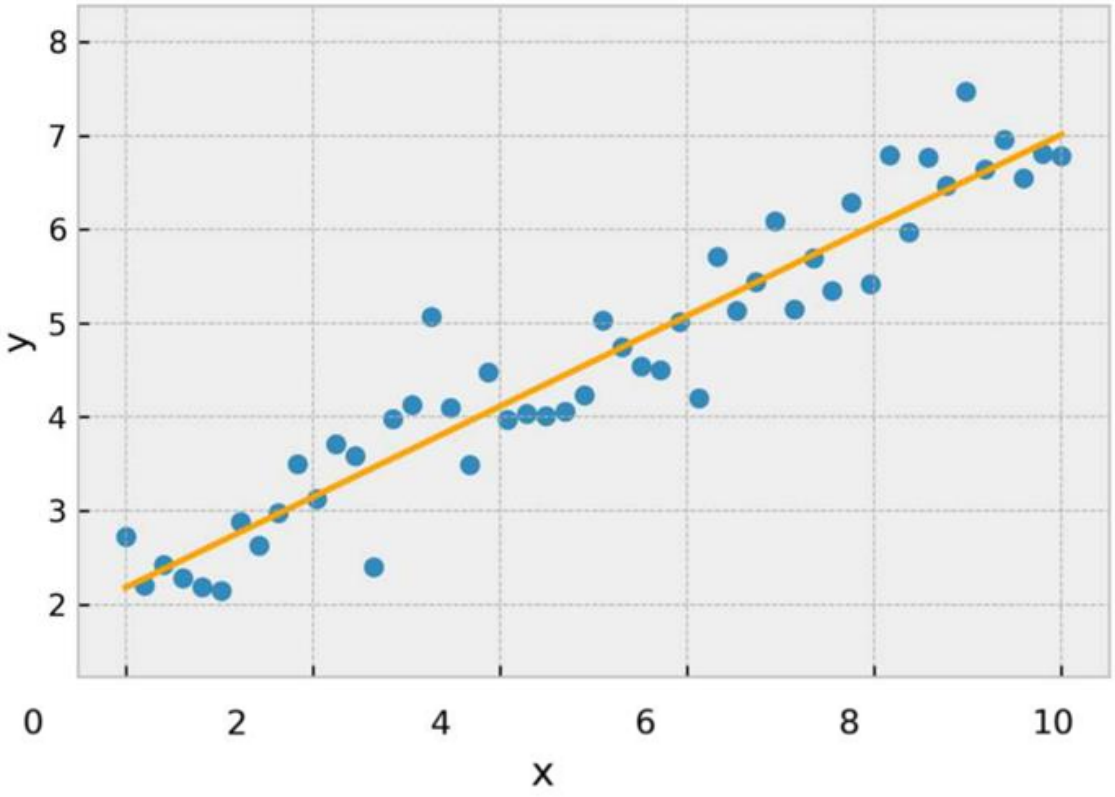
MLE estimate of the slope.

Putting it all together:

We can use these derived equations to write a simple function in python for solving the parameters for any line given at least two points:

```
def find_line(xs, ys):
    """Calculates the slope and intercept"""

    # number of points
    n = len(xs)

    # calculate means
    x_bar = sum(xs)/n
    y_bar = sum(ys)/n

    # calculate slope
    num = 0
    denom = 0
    for i in range(n):
        num += (xs[i]-x_bar)*(ys[i]-y_bar)
        denom += (xs[i]-x_bar)**2
    slope = num/denom

    # calculate intercept
    intercept = y_bar - slope*x_bar

    return slope, intercept
```

Using this code, we can fit a line to our original data (see below). This is the maximum likelihood estimator for our data. The line minimizes the sum of squared errors, which is why this method of linear regression is often called ordinary least squares.



MLE solution to our Linear Regression model.

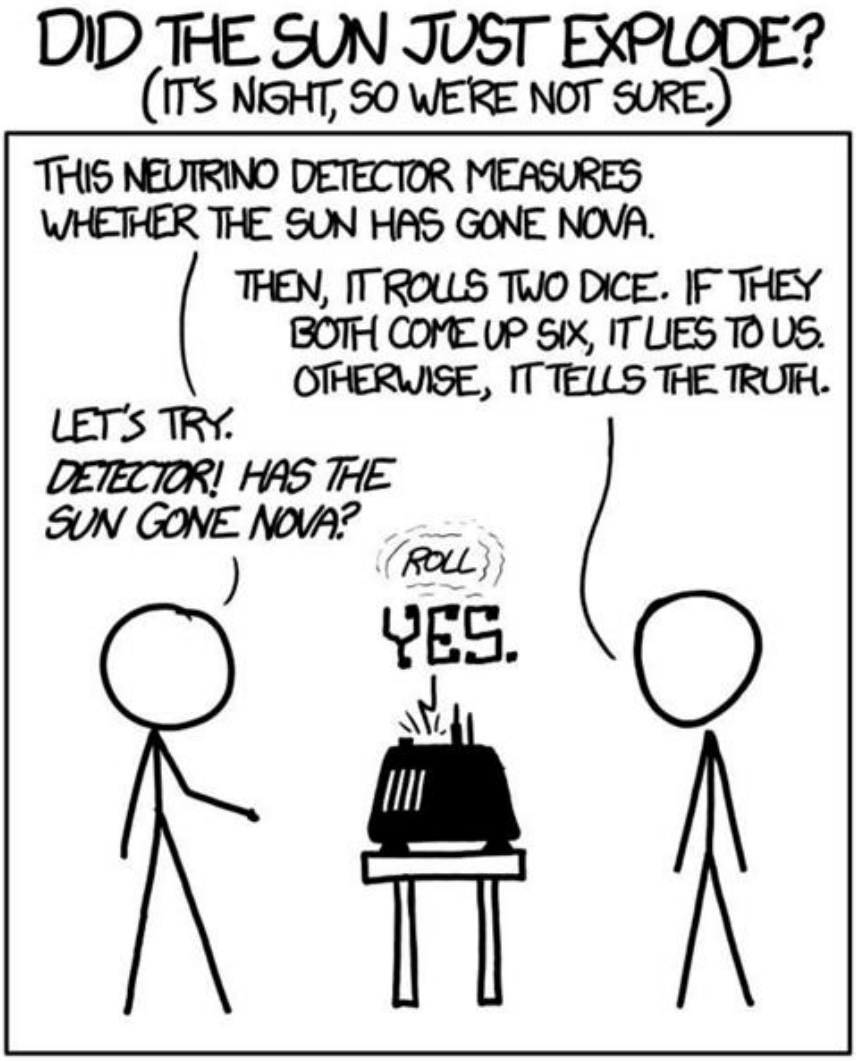
A Gentle Introduction to Maximum Likelihood Estimation

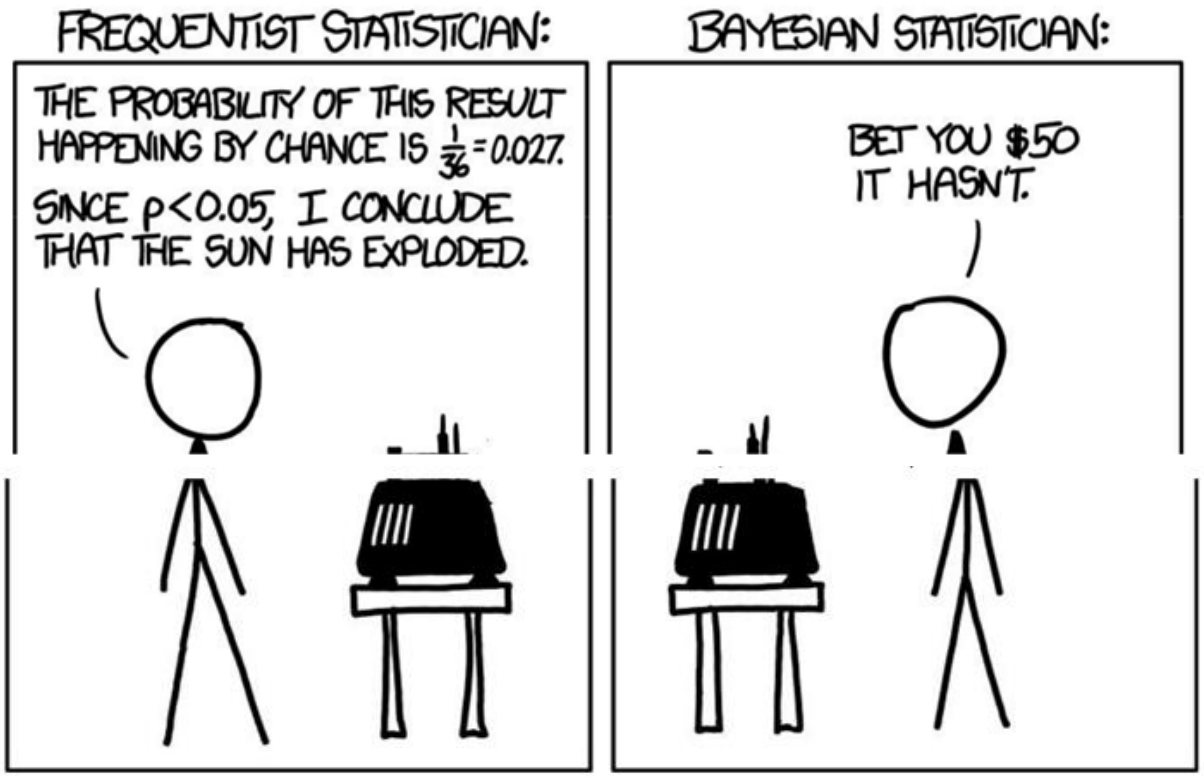
In statistics, ***maximum likelihood estimation*** *(**MLE**) is a method of estimating the parameters of a statistical model given observations, by finding the parameter values that maximize the likelihood of making the observations given the parameters. MLE can be seen as a special case of the maximum a posteriori estimation (MAP) that assumes a uniform prior distribution of the parameters, or as a variant of the MAP that ignores the prior and which therefore is unregularized.*

To spare you the wrestling required to understand and incorporate MLE into your data science workflow, ethos, and projects, I've compiled this guide. Below, we will:

- Set a probabilistic context to MLE
- Delve into the math required
- Look at how MLE works in Python
- Explore best practices in data science with MLE

Frequentists vs. Bayesians





This is funny (if you follow this strange domain of humor), and mostly right about the differences between the two camps. Not minding that our Sun going into nova is not really a repeatable experiment — sorry, frequentists! — we can generalize that for real observations and studies, both camps will usually reach similar conclusions, but differ greatly when the study design or data starts to get tricky.

Which are the best parameters/coefficients for my model?

And interestingly, you can use either school of though to explain why MLE works! Because, while MLE gives a spot estimate — which is common to frequentist outputs — it can be considered a special case of maximum a posteriori (MAP) estimation, where we use a naïve prior and never bother to update it.

Setting Up Our Problem

To approach MLE today, let's come from the Bayesian angle, and use Bayes Theorem to frame our question as such:

```
P(β|y) = P(y|β) x P(3) / P(y)
```

Or, in English:

```
posterior = likelihood x prior / evidence
```

We can effectively ignore the `prior` and the `evidence` because — given the Wiki definition of a uniform prior distribution — all coefficient values are equally likely. And probability of all data values (assume continuous) are equally likely, and basically zero.

So, in actual English: the probability of some specific coefficients given I'm seeing some results, relates to framing the question the exact opposite way. Which helps, cause that question is WAY easier to solve.

Probability and Likelihood

We'll now introduce the concept of likelihood, or `L` in our code henceforth. To grasp the distinction, I'll tag in excerpts from Randy Gallistel's excellent post:

The distinction between probability and likelihood is fundamentally important: Probability attaches to possible results; likelihood attaches to hypotheses.

Possible results are mutually exclusive and exhaustive. Suppose we ask a subject to predict the outcome of each of 10 tosses of a coin. There are only 11 possible results (0 to 10 correct predictions). The actual result will always be one and o one of the possible results. Thus, the probabilities that attach to the possible results must sum to 1.

Hypotheses, unlike results, are neither mutually exclusive nor exhaustive. Suppose that the first subject we test predicts 7 of the 10 outcomes correctly. I might hypothesize that the subject just guessed, and you might hypothesize that the subject may *be somewhat clairvoyant, by which you mean that the subject may be expected to correctly predict the results at slightly greater than chance rates over the long run. These are different hypotheses, but they are not mutually exclusive, because you hedged when you said "*may *be." You thereby allowed your hypothesis to include mine. In technical terminology, my hypothesis is nested within yours. Someone else might hypothesize that the subject is strongly clairvoyant and that the observed result underestimates the probability that her next prediction will be correct. Another person could hypothesize something else altogether. There is no limit to the hypotheses one might entertain.*

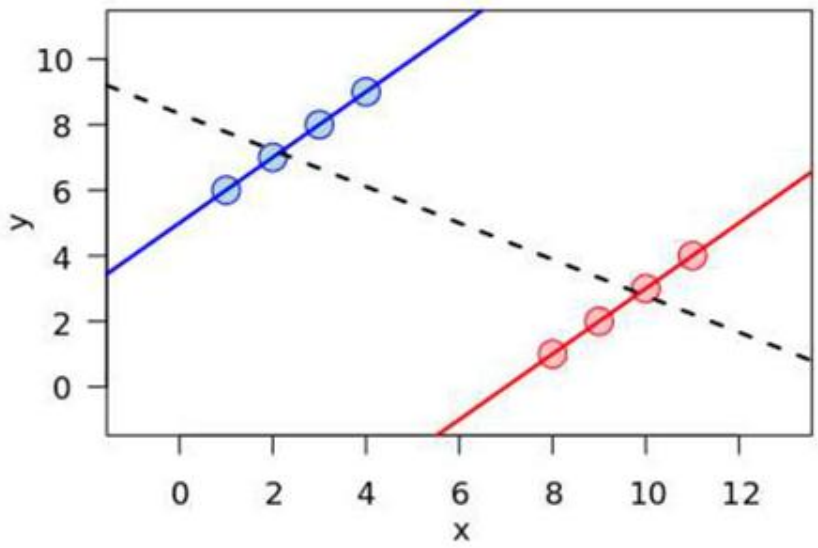
The set of hypotheses to which we attach likelihoods is limited by our capacity to dream them up. In practice, we can rarely be confident that we have imagined all the possible hypotheses. Our concern is to estimate the extent to which the experimental results affect the relative likelihood of the hypotheses we and others currently entertain. Because we generally do not entertain the full set of alternative hypotheses and because some are nested within others, the likelihoods that we attach to our hypotheses do not have any meaning in and of themselves; only the relative likelihoods — that is, the ratios of two likelihoods — have meaning.

Amazing! Thank you, Randy!

MLE is Frequentist, but can be motivated from a Bayesian perspective:

- Frequentists can claim MLE because it's a **point-wise estimate** (not a distribution) and it assumes **no prior distribution** (technically, uninformed or uniform).
- Also, MLE's do not give the 95% probability region for the true parameter value.
- However, MLE is a special form of MAP, and uses the concept of likelihood, which is central to the Bayesian philosophy.

BEWARE the assumption of naïve or uniform priors!! You may mis-attribute the data toward a model that is highly unlikely. You may fall victim to Simpson's Paradox, as below. You could easily be tricked by a small sample size.



Simpson's Paradox

All of the above are problems that Frequentists and data scientists must deal with or be aware of, so there's nothing inherently worse about MLE.

Back to Our Problem

So if $p(y|\beta)$ is equivalent to $L(\beta|y)$, then $p(y_1, y_2, \ldots, y_n|\beta)$ is equivalent to $L(\beta|y_1, y_2, \ldots, y_n)$. Also, remember that we can multiply independent probabilities, like so:

$$p(A,B) = p(A)p(B)$$

We are getting close! Here's our current setup:

$$L(\beta|y1, y2, \ldots, yn) = p(y1|\beta)p(y2|\beta), \ldots, p(yn|\beta) = \prod p(yi|\beta)$$

That part at the right looks like something we can maximize:

$$\max_{\beta} \left\{ \prod_{i} p(y_i|\beta) \right\}$$

Initial Cost Function

But we can do even better! How about using the natural log to turn our product function into a sum function? Logs are monotonic transformations, so we'll simplify our computation but maintain our optimal result.

$$\max_{\beta} \left\{ \ln \left\{ \prod_{i} p(y_i|\beta) \right\} \right\}$$

Our final cost function looks like this:

$$\max_{\beta} \left\{ \sum_{i} \ln \{ p(y_i | \beta) \} \right\}$$

To keep things simple from here, let's assume we have a regression problem, so our outcome is continuous. MLE works great for classification problems with discrete outcomes, but we have to use different distribution functions, depending on how many classes we have, etc.

Now, remembering that a central assumption of models like Ordinary Least Squares (OLS) is that the residuals are normally distributed around mean zero, our fitted OLS model literally becomes the embodiment of a maximum expectation of `y`. And our probability distribution is… Normal!

$$y_i \sim N(x_i \beta, \sigma^2)$$

y is normally distributed around our ŷ

Because computers are *much better* than us at computing the probabilities, we'll turn to Python from here!

MLE in Python

Implementing MLE in your data science modeling pipeline can be quite simple, with a variety of approaches. Below is one approach you can steal to get started.

Setup

MLE is easy if you import the right packages:

```
# import libraries
import numpy as np, pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
from scipy.optimize import minimize
import scipy.stats as stats

import pymc3 as pm3
import numdifftools as ndt
import statsmodels.api as sm
from statsmodels.base.model import GenericLikelihoodModel
%matplotlib inline
```

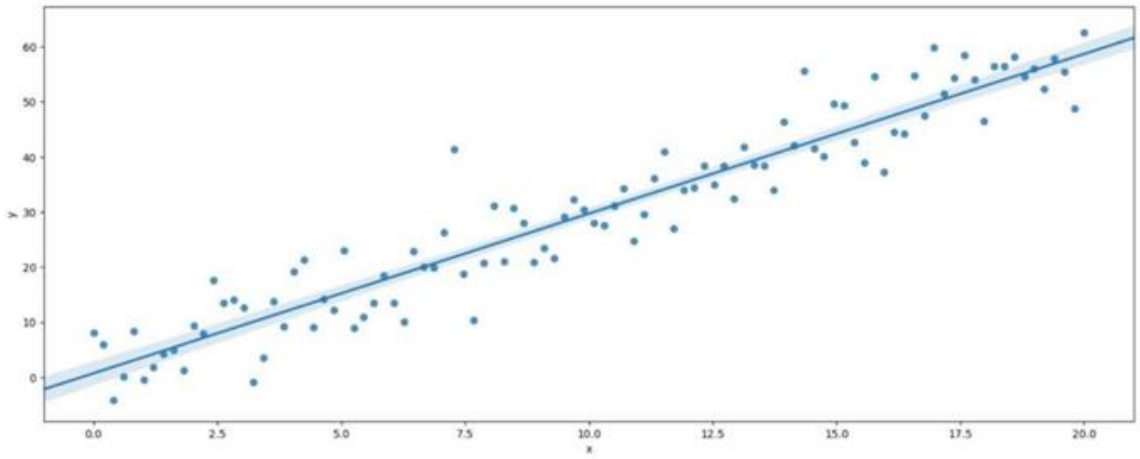
From there, we will generate data that follows a normally distributed errors around a ground truth function:

```
# generate data
N = 100
  = np.linspace(0,20,N)
ϵ = np.random.normal(loc = 0.0, scale = 5.0, size = N)
y = 3*x + ϵ

df = pd.DataFrame({'y':y, 'x':x})
df['constant'] = 1
```

Finally, let's visualize using Seaborn's regplot:

```
# plot
sns.regplot(df.x, df.y);
```

I get the below, and you should see something similar. But, keep in mind there's randomness here and we didn't use a seed:



Scatter plot with OLS line and confidence intervals

Modeling OLS with Statsmodels

Since we created regression-like, continuous data, we will use `sm.OLS` to calculate the best coefficients and Log-likelihood (LL) as a benchmark.

```
# split features and target
X = df[['constant', 'x']]

# fit model and summarize
sm.OLS(y,X).fit().summary()
```

I get this, and will record the fitted model's coefficients:

constant	0.7362	1.089	0.676	0.500	-1.424	2.896
x	2.8986	0.094	30.825	0.000	2.712	3.085

Omnibus:	5.436	**Durbin-Watson:**	1.992
Prob(Omnibus):	0.066	**Jarque-Bera (JB):**	4.966
Skew:	0.417	**Prob(JB):**	0.0835
Kurtosis:	3.705	**Cond. No.**	23.1

Notice `constant` is close to zero, and `beta` for feature `x` is close to 3, per the ground truth generator we used.

Maximizing LL to solve for Optimal Coefficients

From here, we'll use a combination of packages and custom functions to see if we can calculate the same OLS results above using MLE methods.

Because `scipy.optimize` has only a `minimize` method, we'll minimize the negative of the log-likelihood. This is even what they recommend! Math trickery is often faster and easier than re-inventing the wheel!

We can build a simple function that does everything in one pass for regression outputs:

```
# define likelihood function
def MLERegression(params):
 intercept, beta, sd = params[0], params[1], params[2] # inputs are
guesses at our parameters
 yhat = intercept + beta*x # predictions

# next, we flip the Bayesian question
# compute PDF of observed values normally distributed around mean
```

```
(yhat)
# with a standard deviation of sd
 negLL = -np.sum( stats.norm.logpdf(y, loc=yhat, scale=sd) )

# return negative LL
 return(negLL)
```

ow that we have a cost function, let's initialize and minimize it:

```
# let's start with some random coefficient guesses and optimize
guess = np.array([5,5,2])

results = minimize(MLERegression, guess, method = 'Nelder-Mead',
 options={'disp': True})

---------------------------------------------------------------------
Optimization terminated successfully.
         Current function value: 311.060386
         Iterations: 111
         Function evaluations: 195
```

Let's check the results:

```
  results # this gives us verbosity around our minimization
  # notice our final key and associated values…

  ---------------------------------------------------------------------
  final_simplex: (array([[0.45115297, 3.03667376, 4.86925122],
         [0.45123459, 3.03666955, 4.86924261],
         [0.45116379, 3.03667852, 4.86921688],
         [0.45119056, 3.03666796, 4.8692127 ]]), array([300.18758478,
  300.18758478, 300.18758478, 300.18758479]))
             fun: 300.18758477994425
         message: 'Optimization terminated successfully.'
            nfev: 148
             nit: 80
          status: 0
         success: True
               x: array([0.45115297, 3.03667376, 4.86925122])
```

And we can clean up further:

```
# drop results into df and round to match statsmodels
resultsdf = pd.DataFrame({'coef':results['x']})
resultsdf.index=['constant','x','sigma']
np.round(resultsdf.head(2), 4)

# do our numbers match the OLS model?
-----------------------------------------------------------------------
```

You'll note that OLS and MLE match up nicely! Your results will differ, again, as we're not using random seeds.

Best Practices for MLE

Before we go any further, this might be a good moment to reinforce our trust in MLE. As our regression baseline, we know that Ordinary Least Squares — by definition — is the best linear unbiased estimator for continuous outcomes that have normally distributed residuals and meet the other assumptions of linear regression. Is using MLE to find our coefficients as robust?

Yes!

- MLE is consistent with OLS.
- With infinite data, it will estimate the optimal β, *and approximate it well* for small but robust datasets.
- MLE is efficient; no consistent estimator has lower asymptotic mean squared error than MLE.

So it looks like it fully replicates what OLS does. Then… why use MLE instead of OLS?

Because!

- MLE is generalizable for regression and classification!
- MLE is efficient; no consistent estimator has lower asymptotic error than MLE if you're using the right distribution.

We can think of MLE as a modular way of fitting models by optimizing a probabilistic cost function!

Four major steps in applying MLE:

1. Define the likelihood, ensuring you're using the correct distribution for your regression or classification problem.
2. Take the natural log and reduce the product function to a sum function.
3. Maximize — or minimize the negative of — the objective function.
4. Verify that uniform priors are a safe assumption! Otherwise, you could attribute the data to a generating function or model of the world that fails the Law of Parsimony.

There's much more in the MLE space, including categorical distributions, using Bayesian statistics packages like `PyMC3` , etc. But we'll stop here for today.

Maximum Likelihood Estimation (Generic models)

This tutorial explains how to quickly implement new maximum likelihood models in `statsmodels`. We give two examples:

1. Probit model for binary dependent variables
2. Negative binomial model for count data

The `GenericLikelihoodModel` class eases the process by providing tools such as automatic numeric differentiation and a unified interface to `scipy` optimization functions. Using `statsmodels`, users can fit new MLE models simply by "plugging-in" a log-likelihood function.

Example 1: Probit model

[1]:

```
import numpy as np
from scipy import stats
import statsmodels.api as sm
from statsmodels.base.model import GenericLikelihoodModel
```

The `Spector` dataset is distributed with `statsmodels`. You can access a vector of values for the dependent variable (`endog`) and a matrix of regressors (`exog`) like this:

[2]:

```
data = sm.datasets.spector.load_pandas()
exog = data.exog
endog = data.endog
print(sm.datasets.spector.NOTE)
print(data.exog.head())
```

```
::

    Number of Observations - 32

    Number of Variables - 4

    Variable name definitions::

        Grade - binary variable indicating whether or not a student's grade
                improved.  1 indicates an improvement.
        TUCE  - Test score on economics test
        PSI   - participation in program
        GPA   - Student's grade point average

    GPA  TUCE  PSI
0  2.66  20.0  0.0
1  2.89  22.0  0.0
2  3.28  24.0  0.0
3  2.92  12.0  0.0
4  4.00  21.0  0.0
```

Them, we add a constant to the matrix of regressors:

```
exog = sm.add_constant(exog, prepend=True)
```

To create your own Likelihood Model, you simply need to overwrite the loglike method.

[4]:

```
class MyProbit(GenericLikelihoodModel):
    def loglike(self, params):
        exog = self.exog
        endog = self.endog
        q = 2 * endog - 1
        return stats.norm.logcdf(q*np.dot(exog, params)).sum()
```

Estimate the model and print a summary:

[5]:

```
sm_probit_manual = MyProbit(endog, exog).fit()
print(sm_probit_manual.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.400588
         Iterations: 292
         Function evaluations: 494
                              MyProbit Results
==============================================================================
Dep. Variable:                  GRADE   Log-Likelihood:                -12.819
Model:                        MyProbit   AIC:                             33.64
Method:            Maximum Likelihood   BIC:                             39.50
Date:                Fri, 13 Mar 2020
Time:                        13:48:00
No. Observations:                  32
Df Residuals:                      28
Df Model:                           3
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -7.4523      2.542     -2.931      0.003     -12.435      -2.469
GPA            1.6258      0.694      2.343      0.019       0.266       2.986
TUCE           0.0517      0.084      0.617      0.537      -0.113       0.216
PSI            1.4263      0.595      2.397      0.017       0.260       2.593
==============================================================================
```

Compare your Probit implementation to `statsmodels`' "canned" implementation:

[6]:

```
sm_probit_canned = sm.Probit(endog, exog).fit()
```

```
Optimization terminated successfully.
         Current function value: 0.400588
         Iterations 6
```

[7]:

```
print(sm_probit_canned.params)
print(sm_probit_manual.params)
```

```
const   -7.452320
GPA      1.625810
TUCE     0.051729
PSI      1.426332
dtype: float64
[-7.45233176  1.62580888  0.05172971  1.42631954]
```

[8]:

```
print(sm_probit_canned.cov_params())
print(sm_probit_manual.cov_params())
```

```
          const       GPA      TUCE       PSI
const  6.464166 -1.169668 -0.101173 -0.594792
GPA   -1.169668  0.481473 -0.018914  0.105439
TUCE  -0.101173 -0.018914  0.007038  0.002472
PSI   -0.594792  0.105439  0.002472  0.354070
[[ 6.46416778e+00 -1.16966617e+00 -1.01173181e-01 -5.94789009e-01]
 [-1.16966617e+00  4.81472117e-01 -1.89134591e-02  1.05438228e-01]
 [-1.01173181e-01 -1.89134591e-02  7.03758403e-03  2.47189233e-03]
 [-5.94789009e-01  1.05438228e-01  2.47189233e-03  3.54069514e-01]]
```

$$\mathcal{L}(\beta_j; y, \alpha) = \sum_{i=1}^{n} y_i ln \left(\frac{\alpha exp(X_i'\beta)}{1 + \alpha exp(X_i'\beta)} \right) - \frac{1}{\alpha} ln(1 + \alpha exp(X_i'\beta)) + ln\Gamma(y_i + 1/\alpha) - ln\Gamma(y_i + 1) - ln\Gamma(1/\alpha)$$

with a matrix of regressors X, a vector of coefficients β, and the negative binomial heterogeneity parameter α.

Using the `nbinom` distribution from `scipy`, we can write this likelihood simply as:

[9]:

```
import numpy as np
from scipy.stats import nbinom
```

[10]:

```
def _ll_nb2(y, X, beta, alph):
    mu = np.exp(np.dot(X, beta))
    size = 1/alph
    prob = size/(size+mu)
    ll = nbinom.logpmf(y, size, prob)
    return ll
```

New Model Class

We create a new model class which inherits from `GenericLikelihoodModel`:

[11]:

```
from statsmodels.base.model import GenericLikelihoodModel
```

[12]:

```
class NBin(GenericLikelihoodModel):
    def __init__(self, endog, exog, **kwds):
        super(NBin, self).__init__(endog, exog, **kwds)

    def nloglikeobs(self, params):
        alph = params[-1]
        beta = params[:-1]
        ll = _ll_nb2(self.endog, self.exog, beta, alph)
        return -ll

    def fit(self, start_params=None, maxiter=10000, maxfun=5000, **kwds):
        # we have one additional parameter and we need to add it for summary
        self.exog_names.append('alpha')
        if start_params == None:
            # Reasonable starting values
            start_params = np.append(np.zeros(self.exog.shape[1]), .5)
            # intercept
            start_params[-2] = np.log(self.endog.mean())
        return super(NBin, self).fit(start_params=start_params,
                                     maxiter=maxiter, maxfun=maxfun,
                                     **kwds)
```

Two important things to notice:

- `nloglikeobs`: This function should return one evaluation of the negative log-likelihood function per observation in your dataset (i.e. rows of the endog/X matrix).
- `start_params`: A one-dimensional array of starting values needs to be provided. The size of this array determines the number of parameters that will be used in optimization.

That's it! You're done!

Usage Example

The Medpar [https://raw.githubusercontent.com/vincentarelbundock/Rdatasets/doc/COUNT/medpar.html] dataset is hosted in CSV format at the Rdatasets repository [https://raw.githubusercontent.com/vincentarelbundock/Rdatasets]. We use the `read_csv` function from the Pandas library [https://pandas.pydata.org] to load the data in memory. We then print the first few columns:

[13]:

```
import statsmodels.api as sm
```

[14]:

```
medpar = sm.datasets.get_rdataset("medpar", "COUNT", cache=True).data

medpar.head()
```

[14]:

	los	hmo	white	died	age80	type	type1	type2	type3	provnum
0	4	0	1	0	0	1	1	0	0	30001
1	9	1	1	0	0	1	1	0	0	30001
2	3	1	1	1	1	1	1	0	0	30001
3	9	0	1	0	0	1	1	0	0	30001
4	1	0	1	1	1	1	1	0	0	30001

The model we are interested in has a vector of non-negative integers as dependent variable (`los`), and 5 regressors: `Intercept`, `type2`, `type3`, `hmo`, `white`.

For estimation, we need to create two variables to hold our regressors and the outcome variable. These can be ndarrays or pandas objects.

[15]:

```
y = medpar.los
X = medpar[["type2", "type3", "hmo", "white"]].copy()
X["constant"] = 1
```

Then, we fit the model and extract some information:

[16]:

```
mod = NBin(y, X)
res = mod.fit()
```

```
Optimization terminated successfully.
         Current function value: 3.209014
         Iterations: 805
         Function evaluations: 1238
```

Extract parameter estimates, standard errors, p-values, AIC, etc.:

[17]:

```
print('Parameters: ', res.params)
print('Standard errors: ', res.bse)
print('P-values: ', res.pvalues)
print('AIC: ', res.aic)
```

```
Parameters:  [ 0.2212642   0.70613942 -0.06798155 -0.12903932  2.31026565  0.44575147]
Standard errors:  [0.05859259 0.07613047 0.05326096 0.0685414  0.06794696 0.01981542]
P-values:  [1.22298084e-005 1.76979047e-020 2.01819053e-001 5.97481232e-002
 2.15207253e-253 4.62688811e-112]
AIC:  9604.95320583016
```

As usual, you can obtain a full list of available information by typing `dir(res)`. We can also look at the summary of the estimation results.

[18]:

```
print(res.summary())
```

```
                             NBin Results
==============================================================================
Dep. Variable:                    los   Log-Likelihood:                -4797.5
Model:                           NBin   AIC:                             9605.
Method:            Maximum Likelihood   BIC:                             9632.
Date:                Fri, 13 Mar 2020
Time:                        13:48:01
No. Observations:                1495
Df Residuals:                    1490
Df Model:                           4
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
type2          0.2213      0.051      4.373      0.000       0.122       0.320
type3          0.7061      0.076      9.275      0.000       0.557       0.855
hmo           -0.0680      0.053     -1.276      0.202      -0.172       0.036
white         -0.1290      0.069     -1.883      0.060      -0.263       0.005
constant       2.3103      0.068     34.001      0.000       2.177       2.443
alpha          0.4458      0.020     22.495      0.000       0.407       0.485
==============================================================================
```

Testing

We can check the results by using the statsmodels implementation of the Negative Binomial model, which uses the analytic score function and Hessian.

[19]:

```
res_nbin = sm.NegativeBinomial(y, X).fit(disp=0)
print(res_nbin.summary())
```

```
                     NegativeBinomial Regression Results
==============================================================================
Dep. Variable:                    los   No. Observations:                 1495
Model:               NegativeBinomial   Df Residuals:                     1490
Method:                          MLE   Df Model:                            4
Date:                Fri, 13 Mar 2020   Pseudo R-squ.:                0.01215
Time:                        13:48:01   Log-Likelihood:                -4797.5
converged:                      True   LL-Null:                       -4856.5
Covariance Type:            nonrobust   LLR p-value:                1.404e-24
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
type2          0.2212      0.051      4.373      0.000       0.122       0.320
type3          0.7062      0.076      9.276      0.000       0.557       0.855
hmo           -0.0680      0.053     -1.276      0.202      -0.172       0.036
white         -0.1291      0.069     -1.883      0.060      -0.263       0.005
constant       2.3103      0.068     34.001      0.000       2.177       2.443
alpha          0.4457      0.020     22.495      0.000       0.407       0.485
==============================================================================
```

[20]:

```
print(res_nbin.params)
```

```
type2       0.221218
type3       0.706173
hmo        -0.067987
white      -0.129053
constant    2.310279
alpha       0.445748
dtype: float64
```

[21]:

```
print(res_nbin.bse)
```

```
type2       0.050592
type3       0.076131
hmo         0.053261
white       0.068541
constant    0.067947
alpha       0.019815
dtype: float64
```

Or we could compare them to results obtained using the MASS implementation for R:

```
url = 'https://raw.githubusercontent.com/vincentarelbundock/Rdatasets/csv/COUNT/medpar.csv'
medpar = read.csv(url)
f = los~factor(type)+hmo+white

library(MASS)
mod = glm.nb(f, medpar)
coef(summary(mod))
                Estimate Std. Error   z value      Pr(>|z|)
(Intercept)   2.31027893 0.06744676 34.253370 3.885556e-257
factor(type)2 0.22124898 0.05045746  4.384861  1.160597e-05
factor(type)3 0.70615882 0.07599849  9.291748  1.517751e-20
hmo           -0.06795522 0.05321375 -1.277024  2.015939e-01
white         -0.12906544 0.06836272 -1.887951  5.903257e-02
```

Numerical precision

The `statsmodels` generic MLE and `R` parameter estimates agree up to the fourth decimal. The standard errors, however, agree only up to the second decimal. This discrepancy is the result of imprecision in our Hessian numerical estimates. In the current context, the difference between `MASS` and `statsmodels` standard error estimates is substantively irrelevant, but it highlights the fact that users who need very precise estimates may not always want to rely on default settings when using numerical derivatives. In such cases, it is better to use analytical derivatives with the `LikelihoodModel` class.

07: Regularization

The problem of overfitting

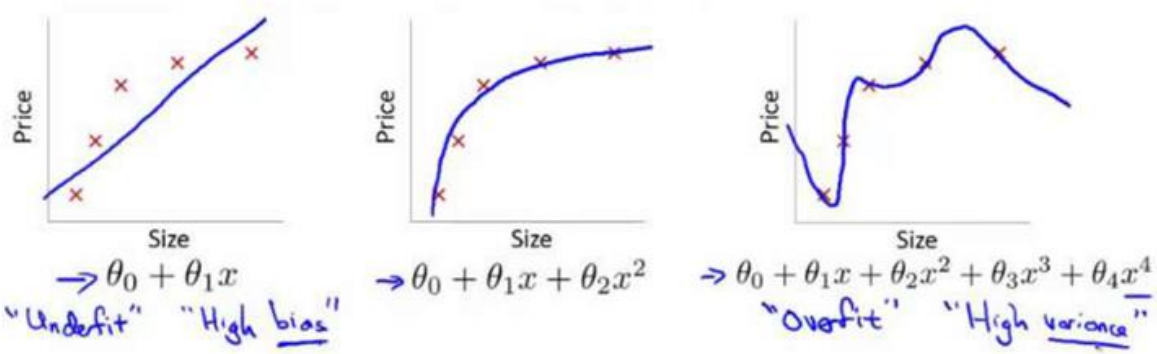
So far we've seen a few algorithms - work well for many applications, but can suffer from the problem of overfitting

- What is overfitting?
- What is regularization and how does it help

Overfitting with linear regression

- Using our house pricing example again
  - Fit a linear function to the data - not a great model
    - This is **underfitting** - also known as **high bias**
    - Bias is a historic/technical one - if we're fitting a straight line to the data we have a strong preconception that there should be a linear fit
      - In this case, this is not correct, but a straight line can't help being straight!
  - Fit a quadratic function
    - Works well
  - Fit a 4th order polynomial
    - Now curve fit's through all five examples
      - Seems to do a good job fitting the training set
      - But, despite fitting the data we've provided very well, this is actually not such a good model
    - This is **overfitting** - also known as **high variance**

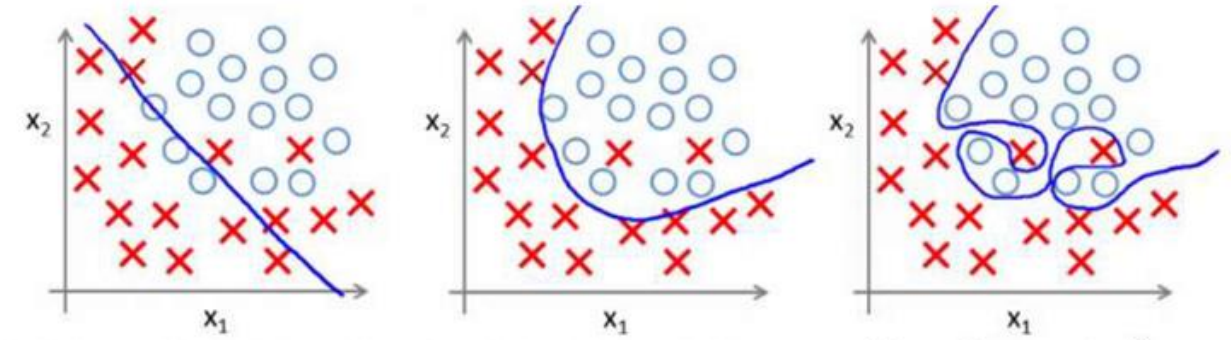
- Algorithm has high variance
  - High variance - if fitting high order polynomial then the hypothesis can basically fit any data
  - Space of hypothesis is too large



- To recap, if we have too many features then the learned hypothesis may give a cost function of exactly zero
  - But this tries too hard to fit the training set
  - Fails to provide a *general* solution - **unable to generalize** (apply to new examples)

Overfitting with logistic regression

- Same thing can happen to logistic regression
  - Sigmoidal function is an underfit
  - But a high order polynomial gives and overfitting (high variance hypothesis)



UNDERFITTING (high bias)

OVERFITTING (high variance)

Addressing overfitting

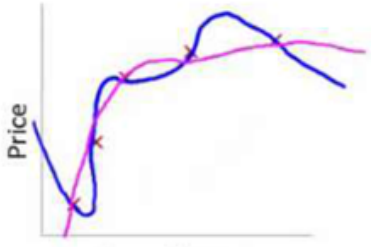
- Later we'll look at identifying when overfitting and underfitting is occurring
- Earlier we just plotted a higher order function - saw that it looks "too curvy"
  - Plotting hypothesis is one way to decide, but doesn't always work
  - Often have lots of a features - here it's not just a case of selecting a degree polynomial, but also harder to plot the data and visualize to decide what features to keep and which to drop
  - If you have lots of features and little data - overfitting can be a problem
- How do we deal with this?
  - 1) **Reduce number of features**
    - Manually select which features to keep
    - Model selection algorithms are discussed later (good for reducing number of features)
    - But, in reducing the number of features we lose some information
      - Ideally select those features which minimize data loss, but even so, some info is lost
  - 2) **Regularization**
    - Keep all features, but reduce magnitude of parameters θ
    - Works well when we have a lot of features, each of which contributes a bit to predicting y

Cost function optimization for regularization

- Penalize and make some of the θ parameters really small
  - e.g. here θ_3 and θ_4

$$\min_\theta \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + 1000\,\theta_3^2 + 1000\,\theta_4^2$$

- The addition in blue is a modification of our cost function to help penalize θ_3 and θ_4
  - So here we end up with θ_3 and θ_4 being close to zero (because the constants are massive)
  - So we're basically left with a quadratic function



$$\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4$$

- In this example, we penalized two of the parameter values
  - More generally, regularization is as follows

- Regularization
  - Small values for parameters corresponds to a simpler hypothesis (you effectively get rid of some of the terms)
  - A simpler hypothesis is less prone to overfitting
- Another example
  - Have 100 features $x_1, x_2, \ldots, x_{100}$
  - Unlike the polynomial example, we don't know what are the high order terms
    - How do we pick the ones to pick to shrink?
  - With regularization, take cost function and modify it to shrink all the parameters
    - Add a term at the end
      - This regularization term shrinks every parameter
      - By convention you don't penalize θ_0 - minimization is from θ_1 onwards

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2\right]$$

$\theta_1, \theta_2, \theta_3, \ldots, \theta_{100}$

- In practice, if you include θ_0 has little impact
- λ is the **regularization parameter**
  - Controls a trade off between our two goals
    - 1) Want to fit the training set well
    - 2) Want to keep parameters small
  - With our example, using the **regularized objective** (i.e. the cost function with the regularization term) you get a much smoother curve which fits the data and gives a much better hypothesis
    - If λ is very large we end up penalizing ALL the parameters (θ_1, θ_2 etc.) so all the parameters end up being close to zero
      - If this happens, it's like we got rid of all the terms in the hypothesis
        - This results here is then underfitting
      - So this hypothesis is too biased because of the absence of any parameters (effectively)
  - So, λ should be chosen carefully - not too big...
    - We look at some automatic ways to select λ later in the course

Regularized linear regression

- Previously, we looked at two algorithms for linear regression
  - Gradient descent
  - Normal equation
- Our linear regression with regularization is shown below

$$J(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda\sum_{j=1}^{n}\theta_j^2\right]$$

$$\min_\theta J(\theta)$$

- Previously, gradient descent would repeatedly update the parameters θ_j, where j = 0,1,2...n simultaneously
  - Shown below

Repeat $\{$

$$\theta_0 := \theta_0 - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha \quad \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

$$(j = 1, 2, 3, \ldots, n)\ \}$$

- We've got the θ_0 update here shown explicitly
  - This is because for regularization we don't penalize θ_0 so treat it slightly differently
- How do we regularize these two rules?
  - Take the term and add $\lambda/m * \theta_j$
    - Sum for every θ (i.e. j = 0 to n)
  - This gives regularization for gradient descent
- We can show using calculus that the equation given below is the partial derivative of the regularized J(θ)

$$\theta_j := \theta_j - \alpha\left[\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} + \frac{\lambda}{m}\theta_j\right]$$

$$(j = 1, 2, 3, \ldots, n)$$

$\frac{\partial}{\partial\theta_j} J(\theta)$ regularized

- The update for θ_j
  - θ_j gets updated to
    - $\theta_j - \alpha$ * [a big term which also depends on θ_j]
- So if you group the θ_j terms together

$$\theta_j := \theta_j(1 - \alpha\frac{\lambda}{m}) - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

- The term
$$(1 - \alpha\frac{\lambda}{m})$$
  - Is going to be a number less than 1 usually
  - Usually learning rate is small and m is large
    - So this typically evaluates to (1 - a small number)
    - So the term is often around 0.99 to 0.95
- This in effect means θ_j gets multiplied by 0.99
  - Means the squared norm of θ_j a little smaller
  - The second term is exactly the same as the original gradient descent

Regularization with the normal equation

- Normal equation is the other linear regression model
  - Minimize the J(θ) using the normal equation
  - To use regularization we add a term (+ λ [n+1 x n+1]) to the equation
    - [n+1 x n+1] is the n+1 identity matrix

$$\Theta = \left(X^T X + \lambda \begin{bmatrix} 0 & & & \\ & 1 & & \\ & & 1 & \\ & & & \ddots \\ & & & & 1 \end{bmatrix}_{(n+1)\times(n+1)} \right)^{-1} X^T y$$

e.g. if n = 2 $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

Regularization for logistic regression

- We saw earlier that logistic regression can be prone to overfitting with lots of features
- Logistic regression cost function is as follows;

$$J(\theta) = -\left[\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)} + (1 - y^{(i)}) \log (1 - h_\theta(x^{(i)}))\right]$$

- To modify it we have to add an extra term

$$+ \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

- This has the effect of penalizing the parameters θ_1, θ_2 up to θ_n
  - Means, like with linear regression, we can get what appears to be a better fitting lower order hypothesis
- How do we implement this?
  - Original logistic regression with gradient descent function was as follows

$$\theta_j := \theta_j - \alpha \quad \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$
$$(j = 0, 1, 2, 3, \ldots, n)$$

- Again, to modify the algorithm we simply need to modify the update rule for θ_1, onwards
  - Looks cosmetically the same as linear regression, except obviously the hypothesis is very different

$$\theta_j := \theta_j(1 - \alpha\tfrac{\lambda}{m}) - \alpha\tfrac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)}$$

Advanced optimization of regularized linear regression

- As before, define a costFunction which takes a θ parameter and gives jVal and gradient back

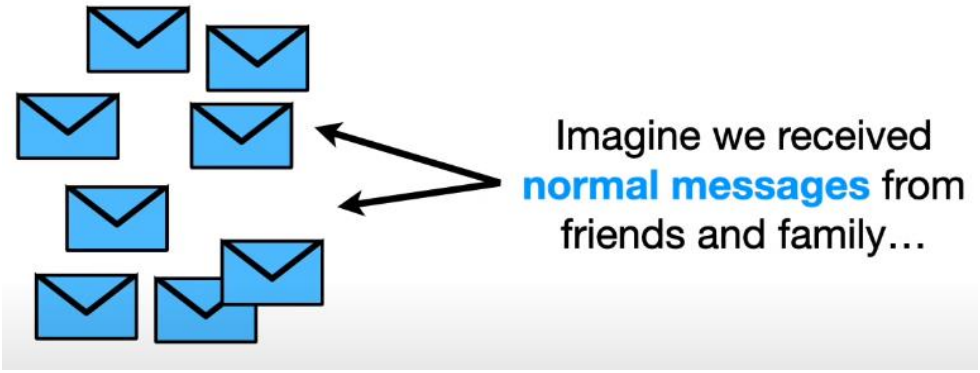
```
function [jVal, gradient] = costFunction(theta)
    jVal = [ code to compute J(θ)] ;

    gradient(1) = [code to compute ∂/∂θ0 J(θ) ] ;

    gradient(2) = [code to compute ∂/∂θ1 J(θ) ] ;

    gradient(3) = [code to compute ∂/∂θ2 J(θ) ] ;
      :
    gradient(n+1) = [code to compute ∂/∂θn J(θ) ] ;
```

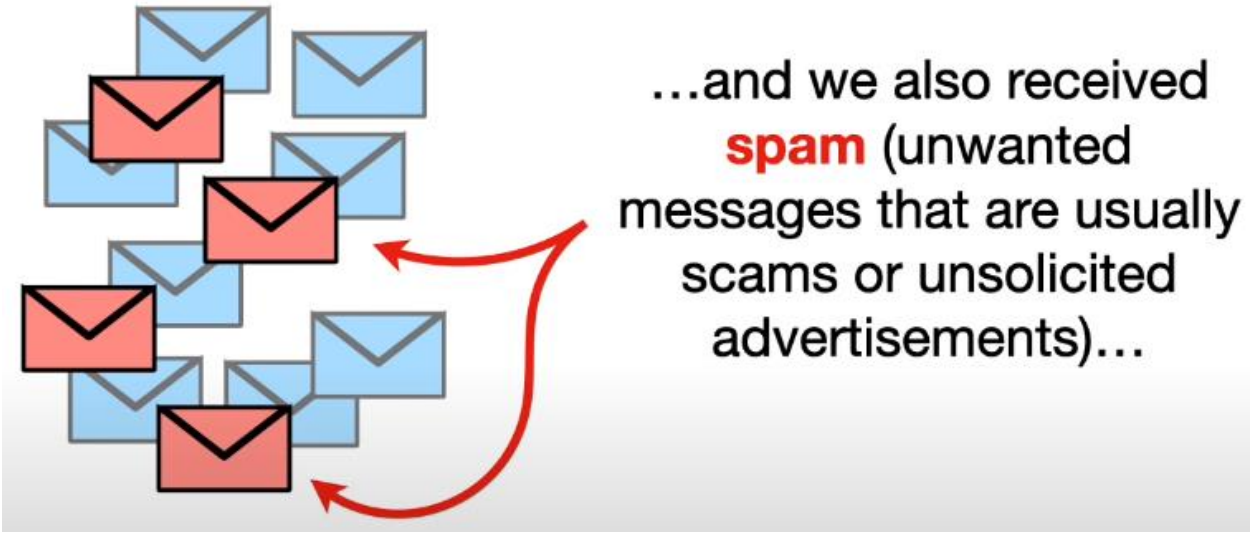
- use `fminunc`
  - Pass it an `@costfunction` argument
  - Minimizes in an optimized manner using the cost function
- `jVal`
  - Need code to compute J(θ)
    - Need to include regularization term
- Gradient
  - Needs to be the partial derivative of J(θ) with respect to θ_i
  - Adding the appropriate term here is also necessary

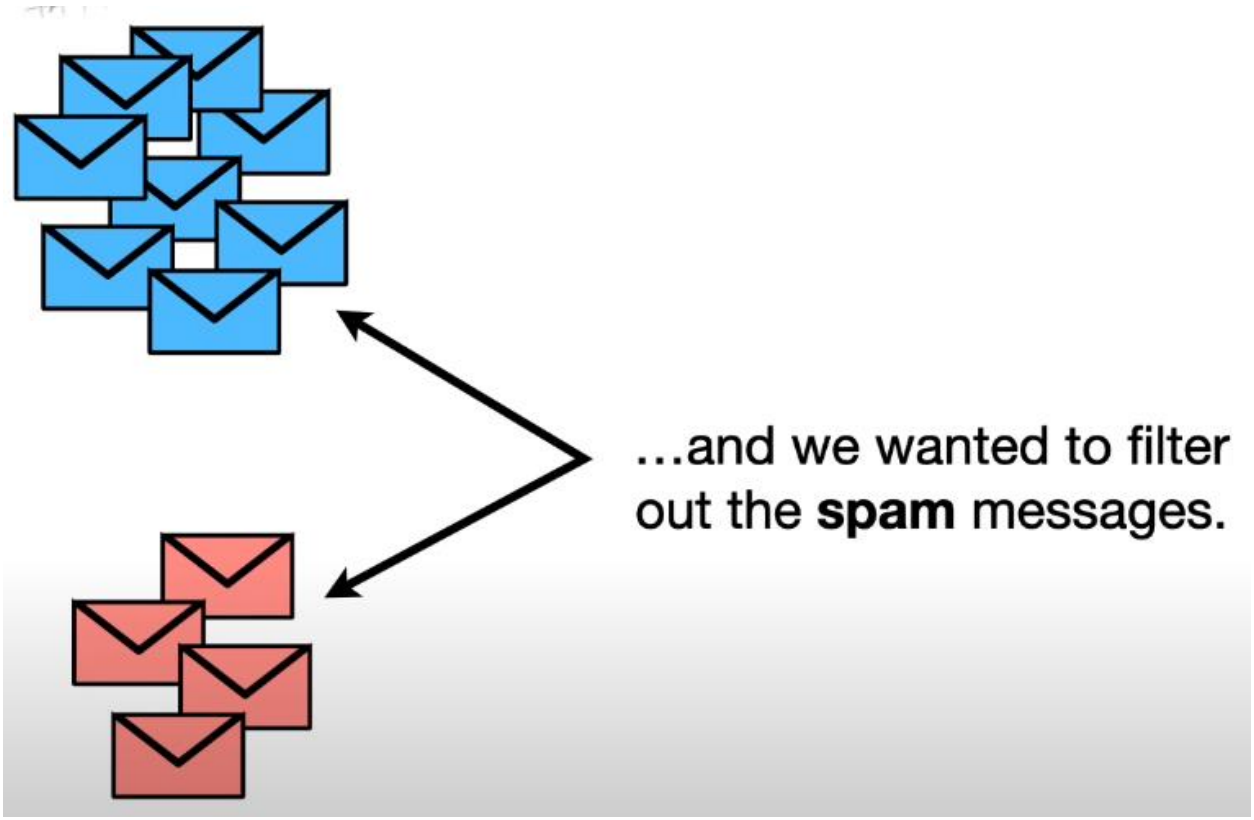
```
function [jVal, gradient] = costFunction(theta)
      jVal = [ code to compute J(θ)] ;
                J(θ) = [ -1/m Σ_{i=1}^{m} y^(i) log(h_θ(x^(i)) + (1 - y^(i)) log 1 - h_θ(x^(i)) ] + λ/2m Σ_{j=1}^{n} θ_j^2
      gradient(1) = [code to compute ∂/∂θ_0 J(θ) ] ;
                1/m Σ_{i=1}^{m} (h_θ(x^(i)) - y^(i)) x_0^(i)
      gradient(2) = [code to compute ∂/∂θ_1 J(θ) ] ;
                1/m Σ_{i=1}^{m} (h_θ(x^(i)) - y^(i)) x_1^(i) + λ/m θ_1
      gradient(3) = [code to compute ∂/∂θ_2 J(θ) ] ;
        :       1/m Σ_{i=1}^{m} (h_θ(x^(i)) - y^(i)) x_2^(i) + λ/m θ_2
        :
      gradient(n+1) = [code to compute ∂/∂θ_n J(θ) ] ;
```

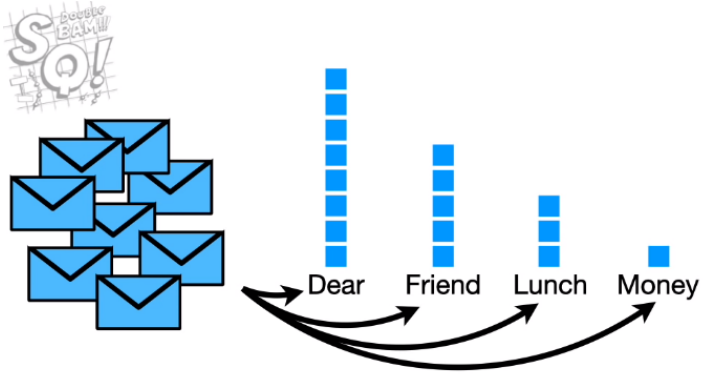
- Ensure summation doesn't extend to to the lambda term!
  - It doesn't, but, you know, don't be daft!

Naïve Bayes

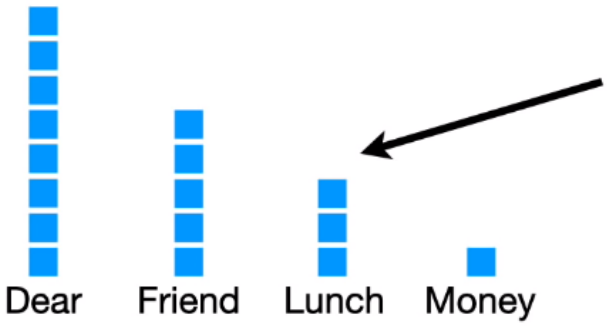

Imagine we received
normal messages from
friends and family…


…and we also received
spam (unwanted
messages that are usually
scams or unsolicited
advertisements)…

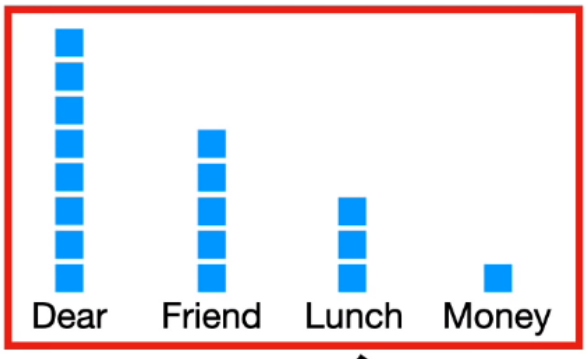




So, the first thing we do is make a **histogram** of all the words that occur in the **normal messages** from friends and family.



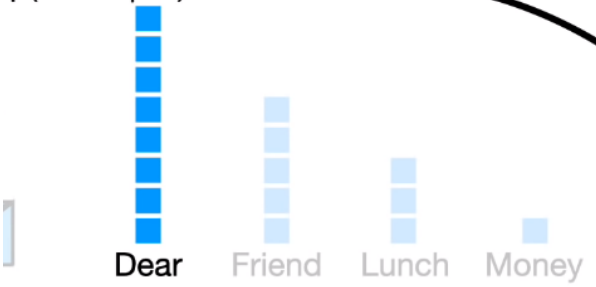
We can use the histogram to calculate the probabilities of seeing each word, given that it was in a **normal message**.



…divided by **17**, the total number of words in all of the **normal messages**.

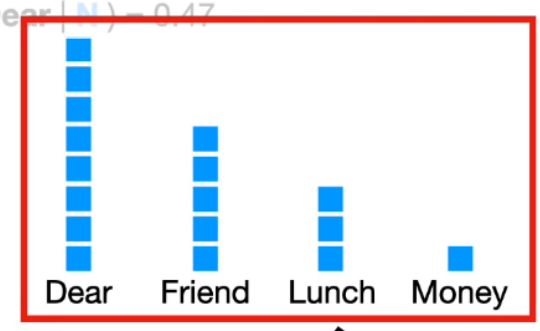
$$p(\textbf{Dear} \mid \textbf{Normal}) = \frac{8}{17}$$

$$p(\textbf{Dear} \mid \textbf{N}) = 0.47$$



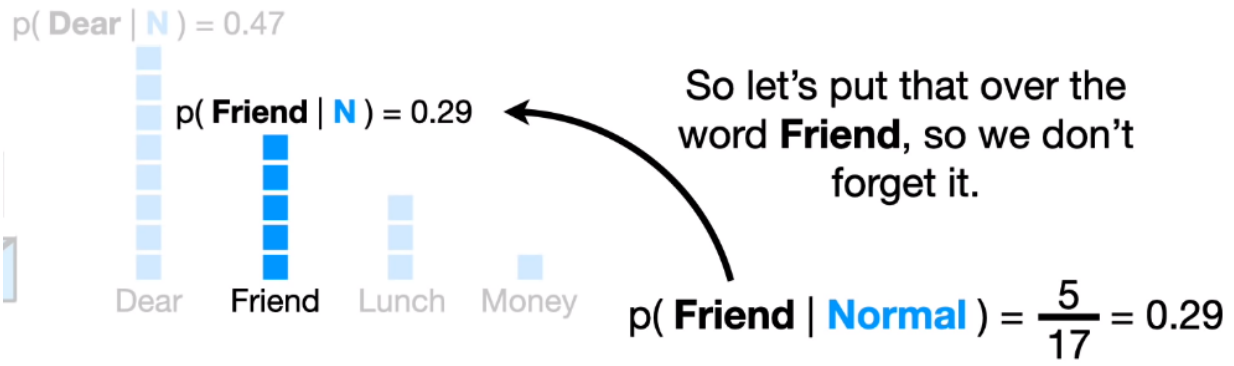
So let's put that over the word **Dear**, so we don't forget it.

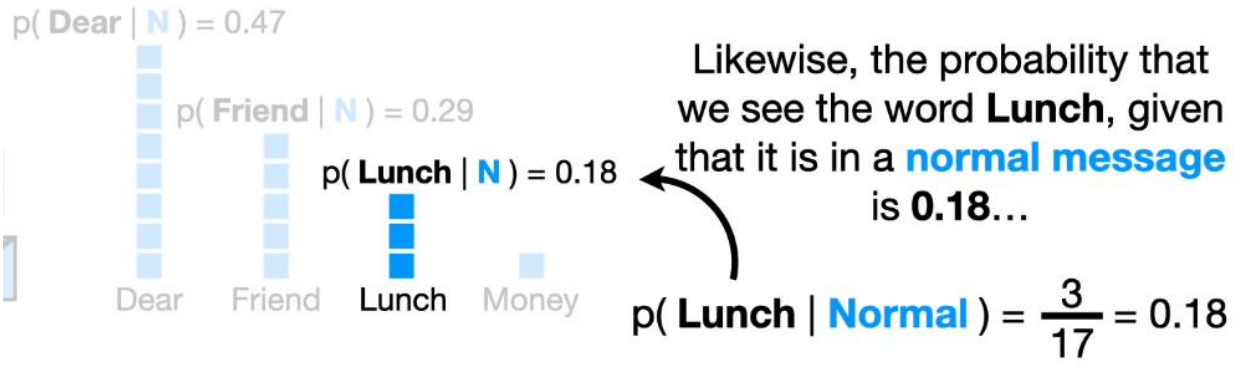
$$p(\textbf{Dear} \mid \textbf{Normal}) = \frac{8}{17} = 0.47$$

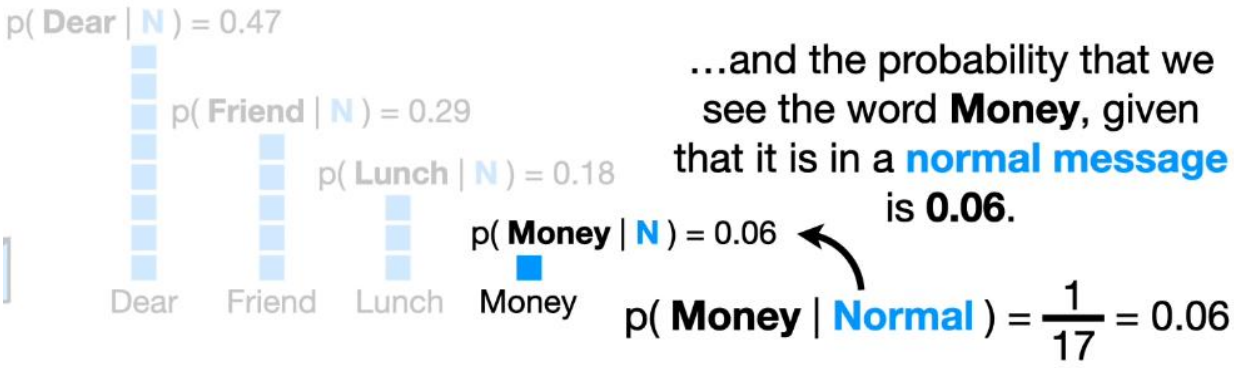


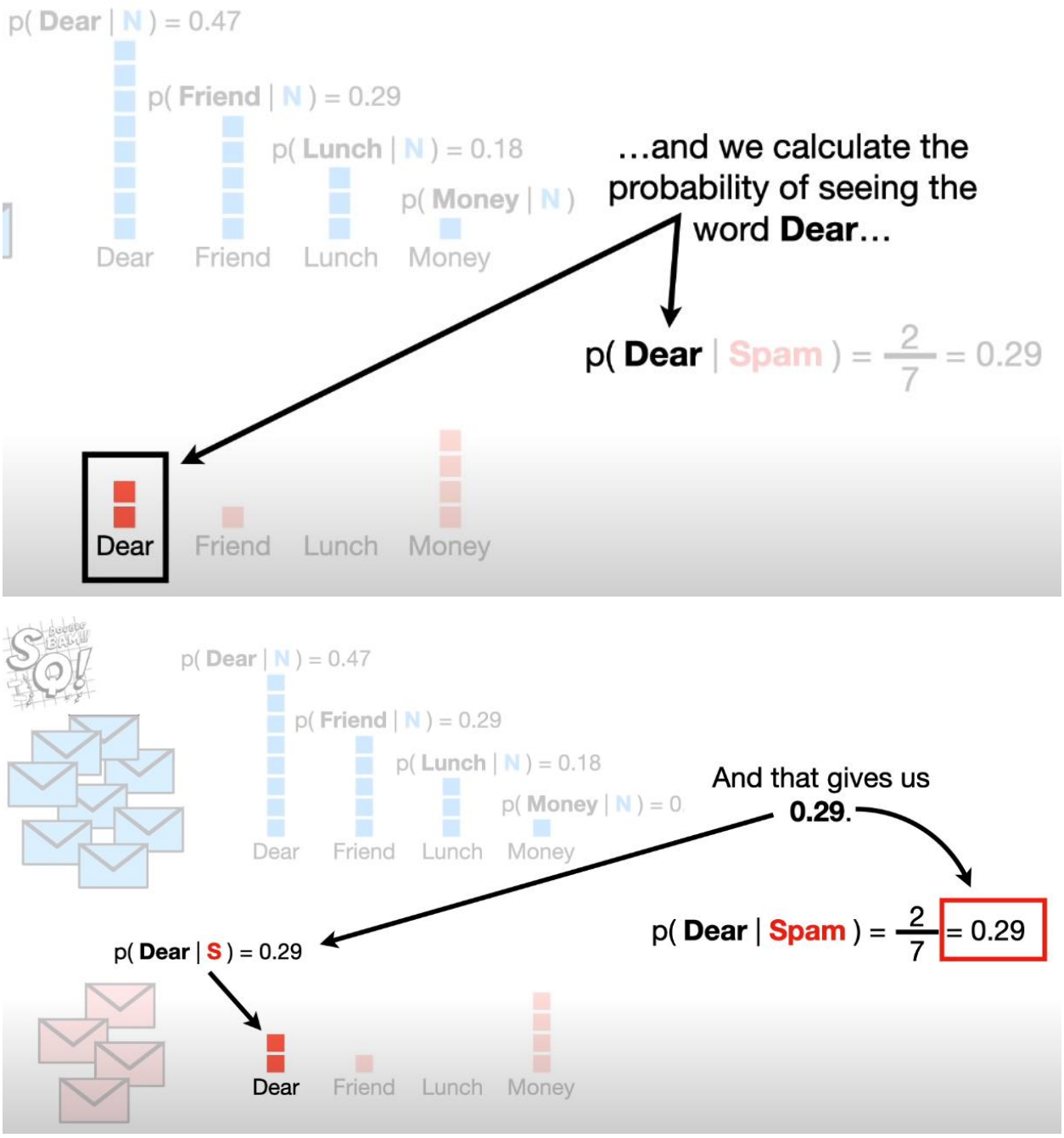
…divided by **17**, the total number of words in all of the **normal messages**.

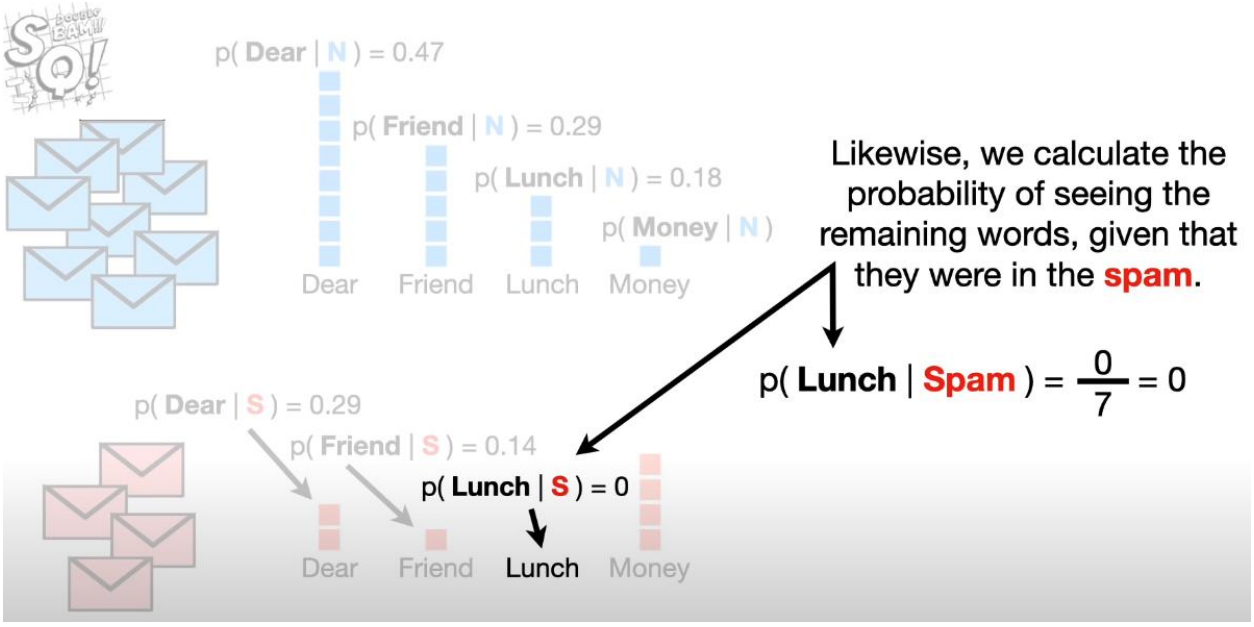
$$p(\textbf{Friend} \mid \textbf{Normal}) = \frac{5}{17} = 0.29$$

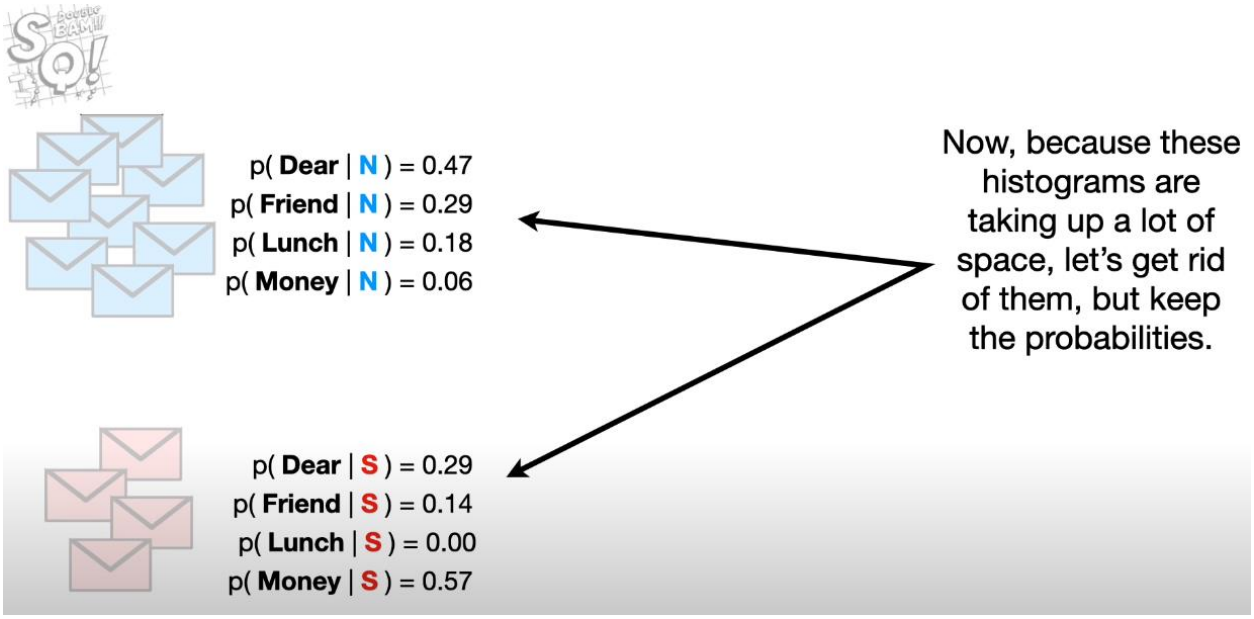

p(Dear | N) = 0.47
p(Friend | N) = 0.29
Dear
Friend
Lunch
Money
So let's put that over the word Friend, so we don't forget it.
p(Friend | Normal) = 5/17 = 0.29

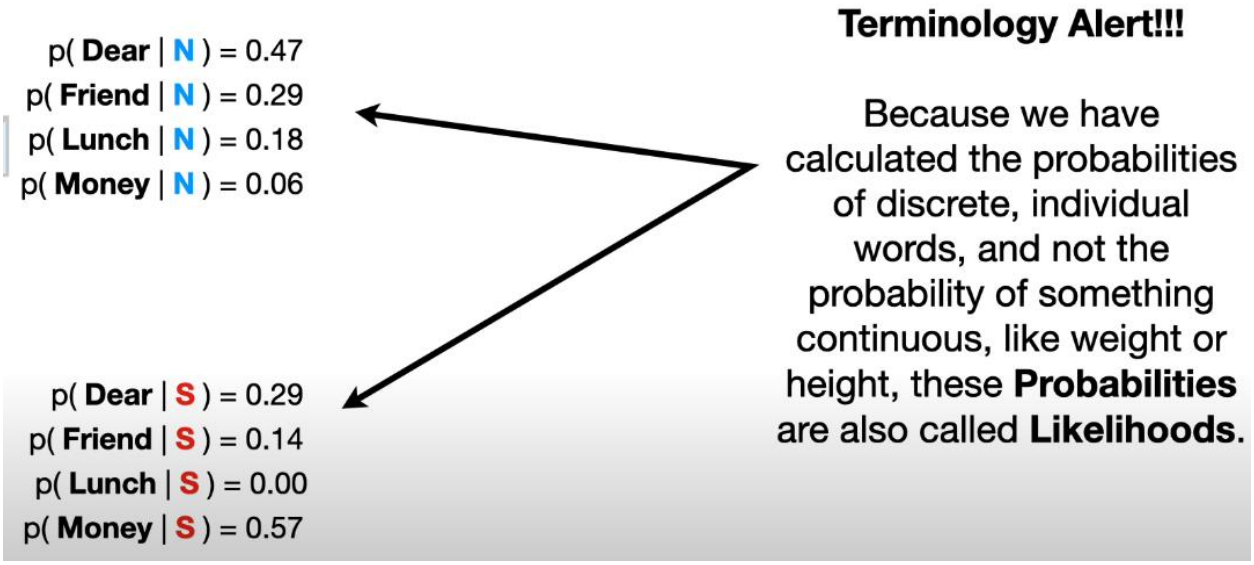

p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
Dear
Friend
Lunch
Money
Likewise, the probability that we see the word Lunch, given that it is in a normal message is 0.18...
p(Lunch | Normal) = 3/17 = 0.18

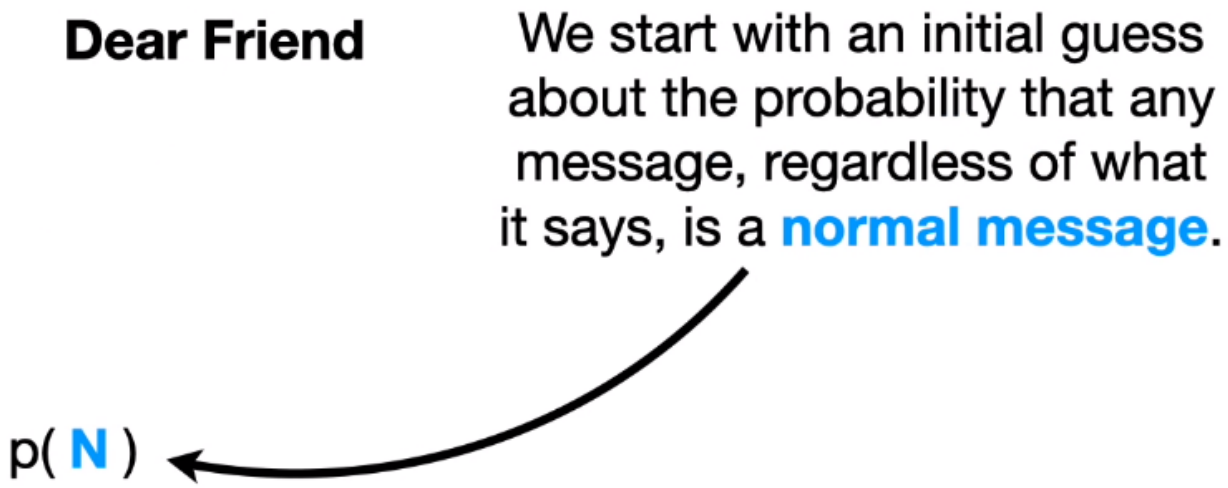

p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
Dear
Friend
Lunch
Money
...and the probability that we see the word Money, given that it is in a normal message is 0.06.
p(Money | Normal) = 1/17 = 0.06

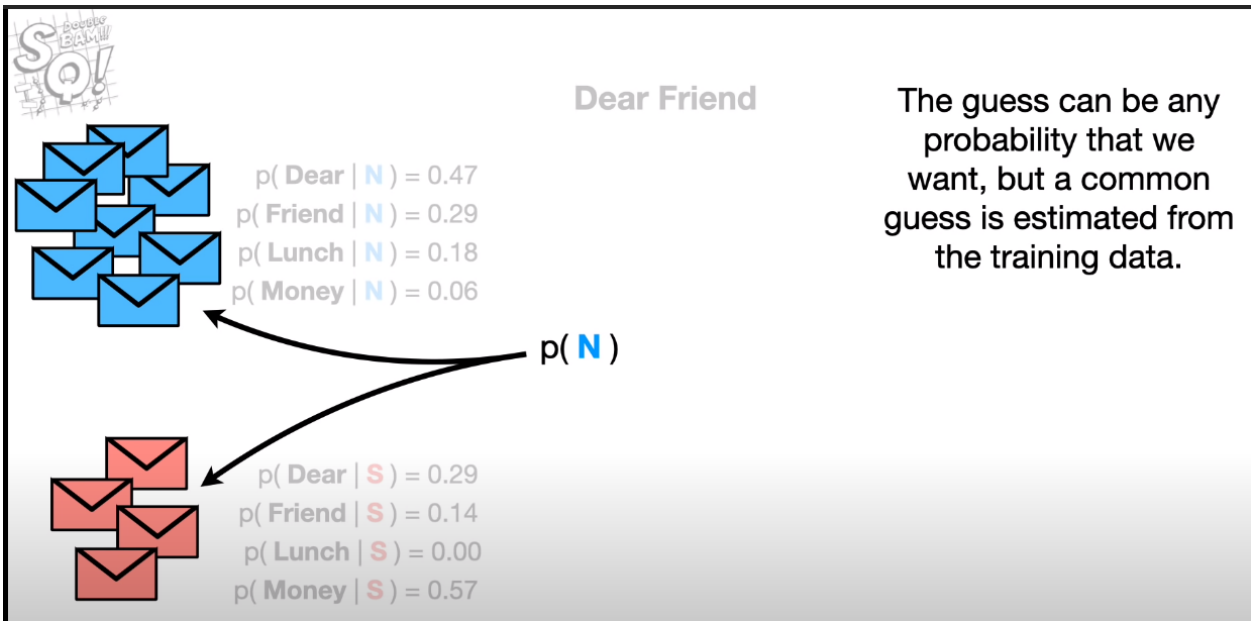

p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N)
Dear
Friend
Lunch
Money
…and we calculate the probability of seeing the word Dear…
p(Dear | Spam) = 2/7 = 0.29
Dear
Friend
Lunch
Money
So!
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0
Dear
Friend
Lunch
Money
And that gives us 0.29.
p(Dear | Spam) = 2/7 = 0.29
p(Dear | S) = 0.29
Dear
Friend
Lunch
Money


So!
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N)
Dear
Friend
Lunch
Money
Likewise, we calculate the probability of seeing the remaining words, given that they were in the spam.
p(Lunch | Spam) = 0/7 = 0
p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0
Dear
Friend
Lunch
Money

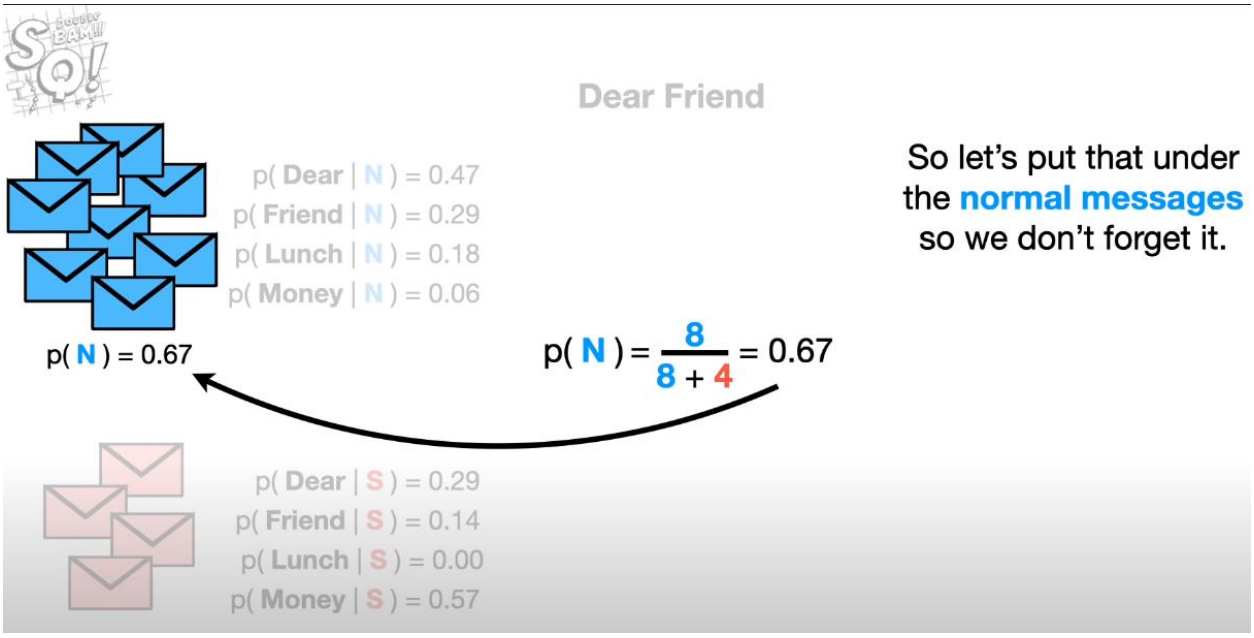

So!
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
Now, because these histograms are taking up a lot of space, let's get rid of them, but keep the probabilities.
p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57


p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57
Terminology Alert!!!
Because we have calculated the probabilities of discrete, individual words, and not the probability of something continuous, like weight or height, these Probabilities are also called Likelihoods.


Dear Friend
We start with an initial guess about the probability that any message, regardless of what it says, is a normal message.
p(N)

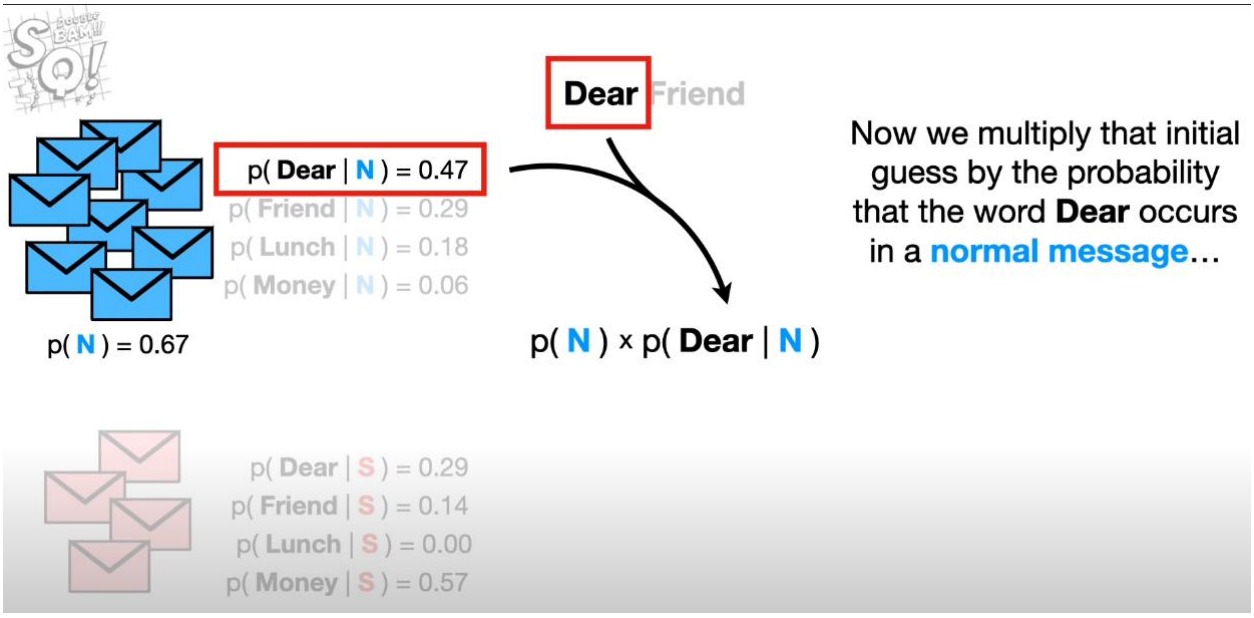


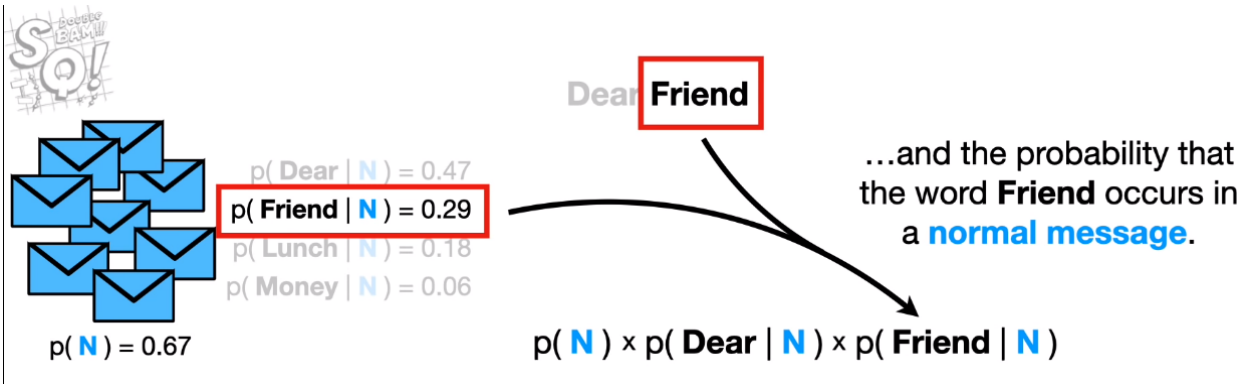
The guess can be any probability that we want, but a common guess is estimated from the training data.

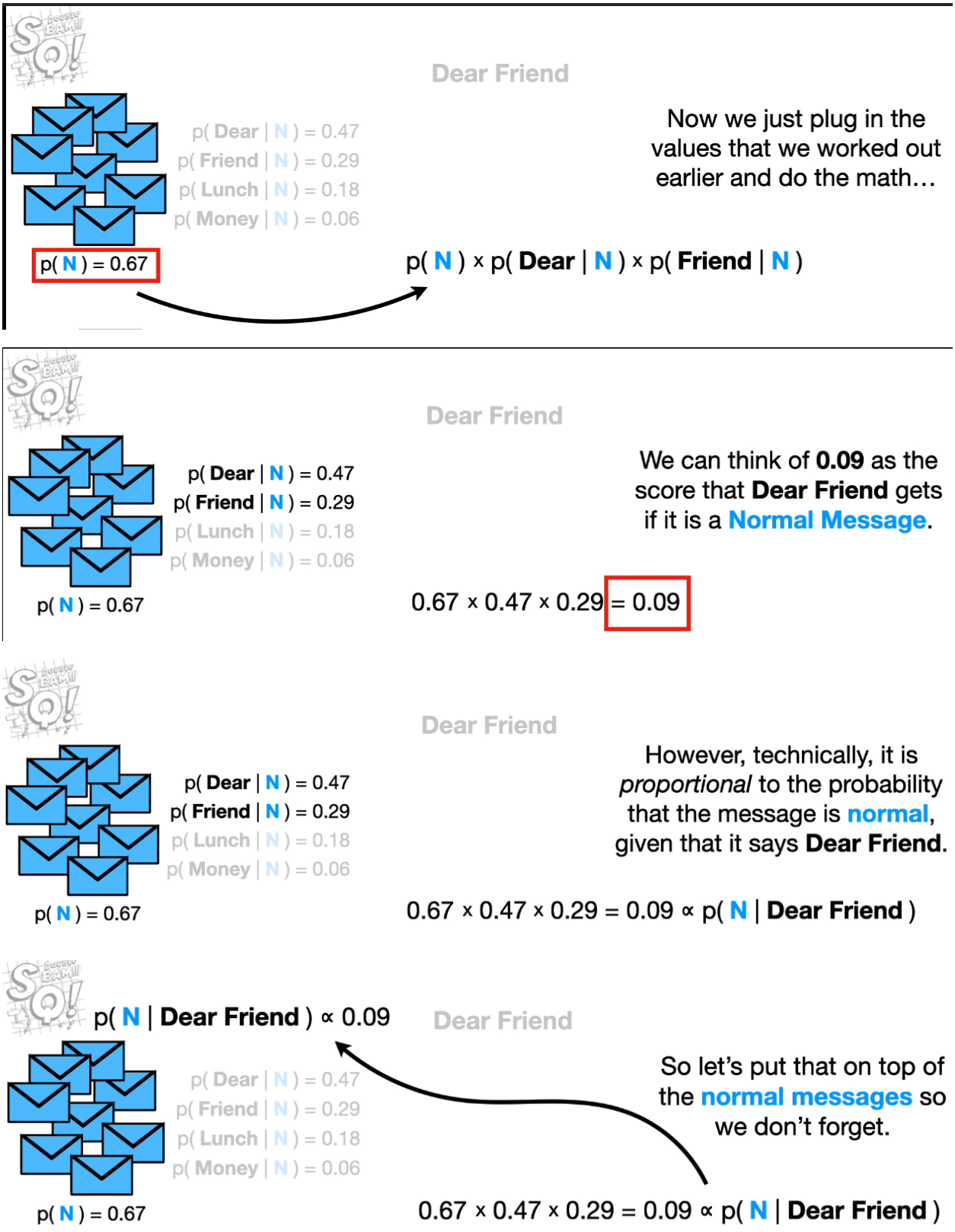


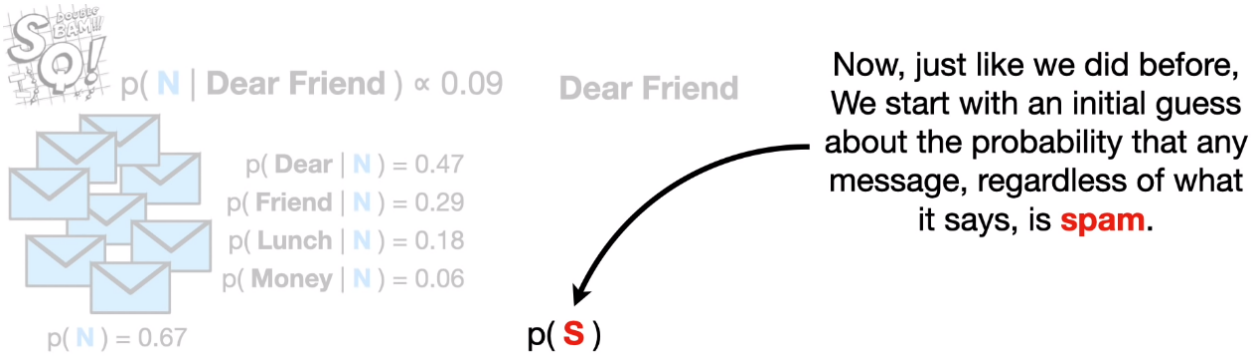
So let's put that under the **normal messages** so we don't forget it.

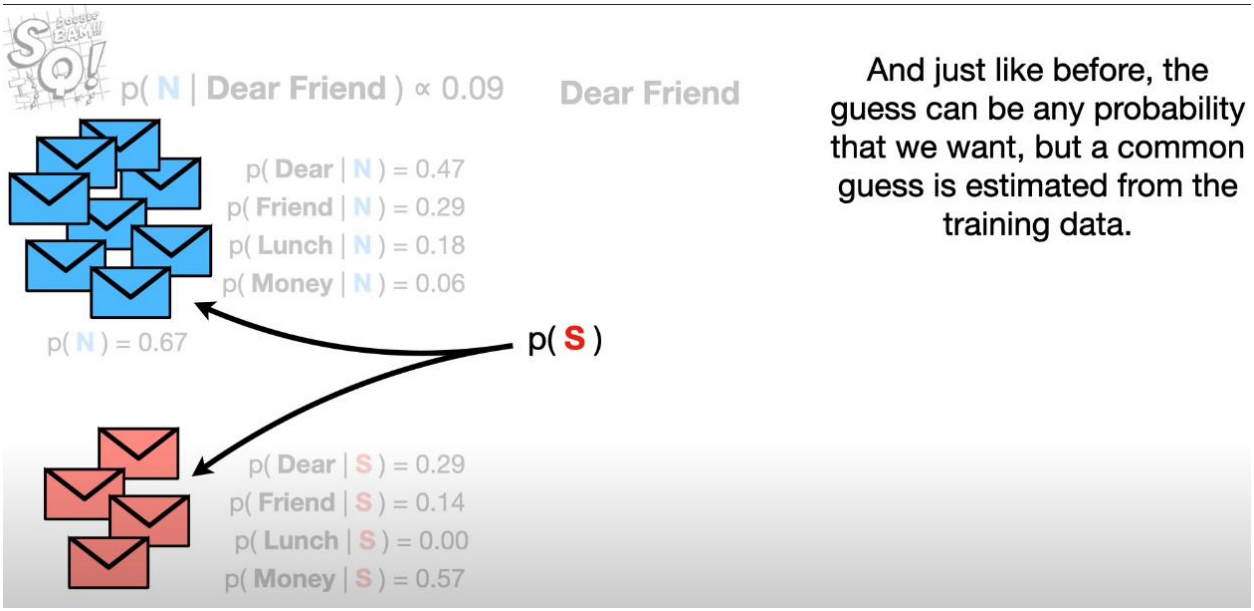
p(N) ← The initial guess that we observe a Normal messages is called a **Prior Probability.**

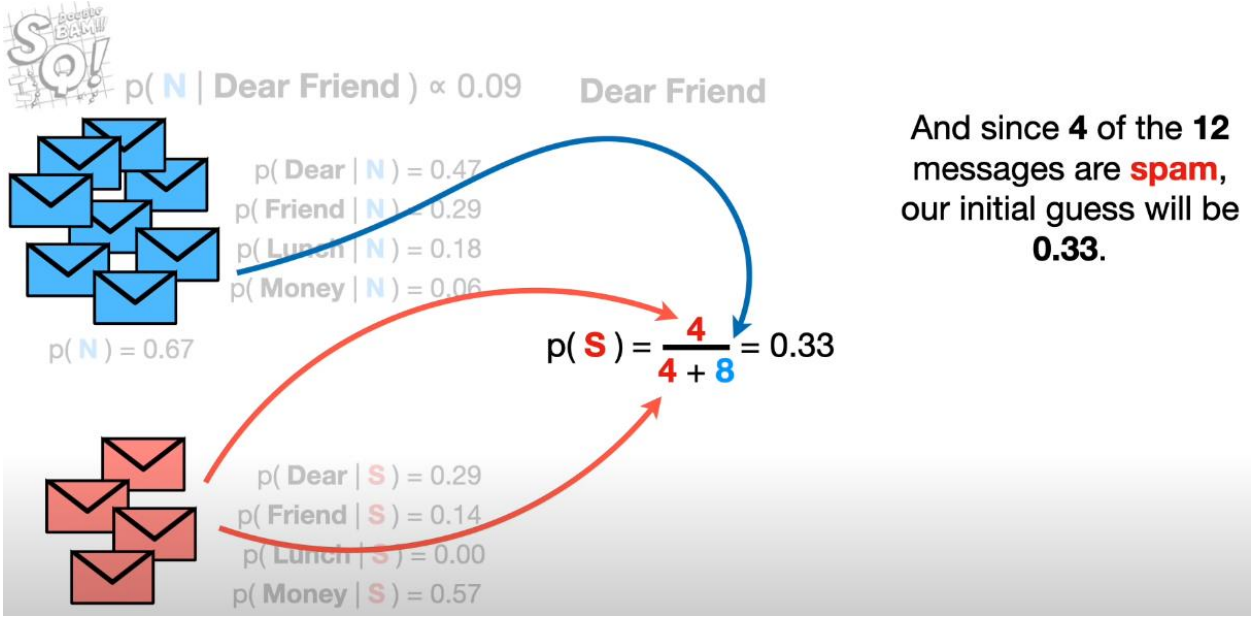


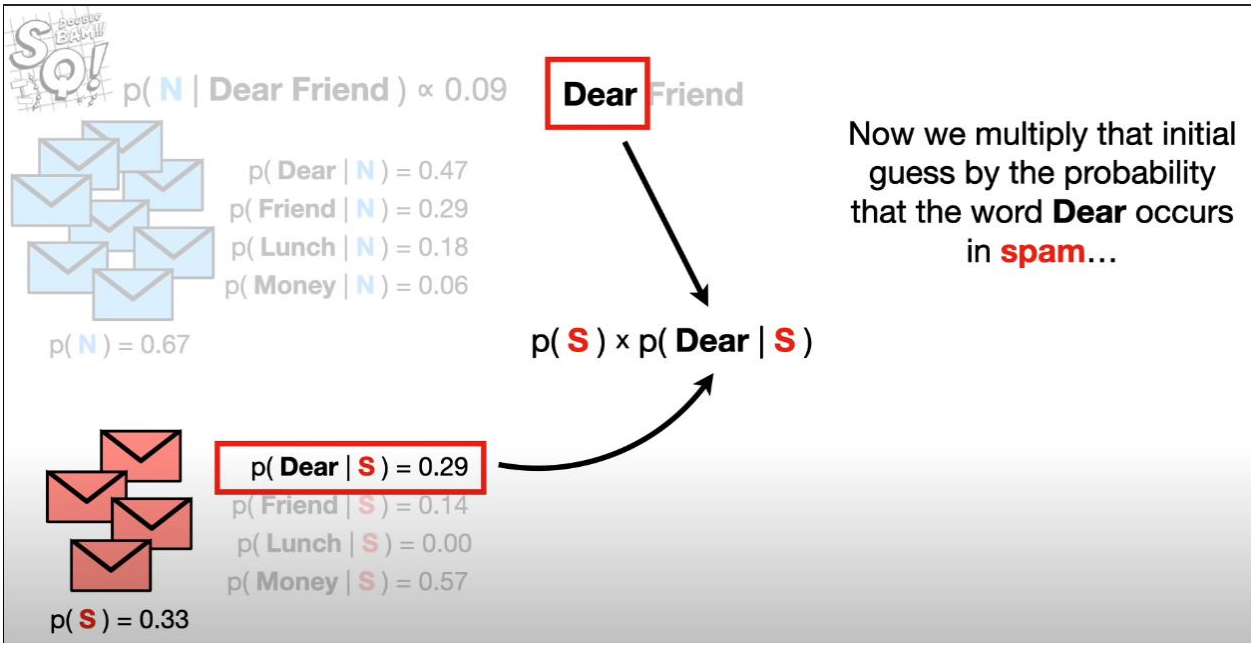


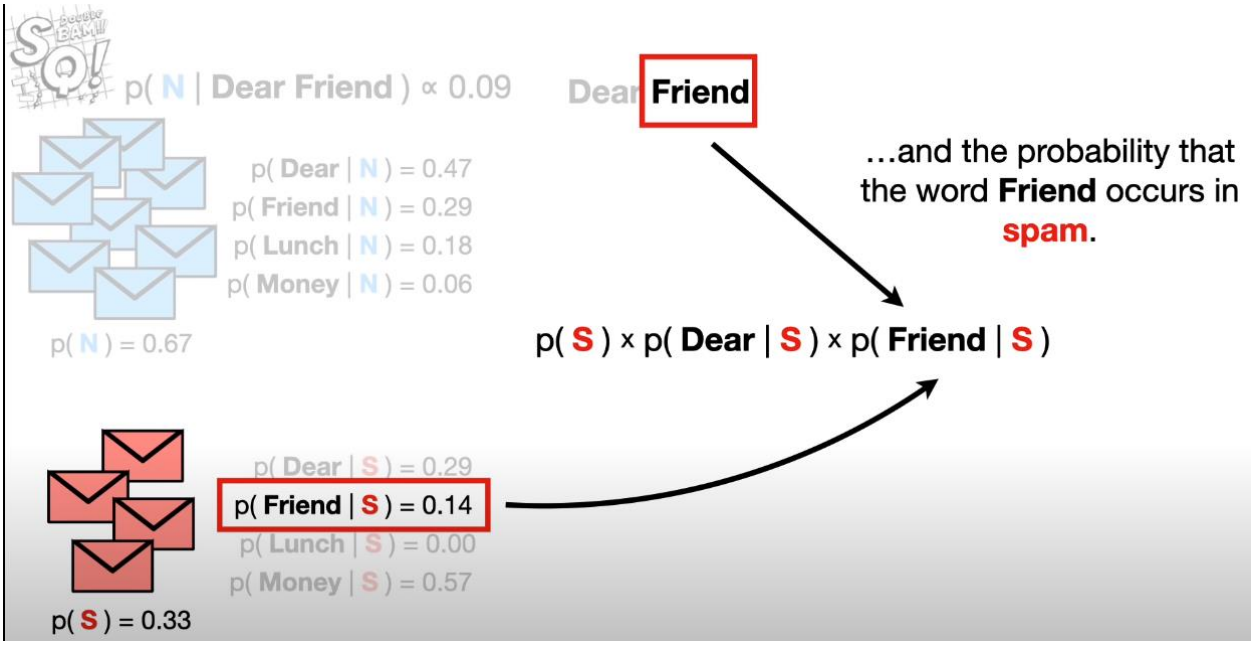

So!
Dear Friend
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
p(N) = 0.67
Now we just plug in the values that we worked out earlier and do the math…
p(N) x p(Dear | N) x p(Friend | N)
So!
Dear Friend
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
p(N) = 0.67
We can think of 0.09 as the score that Dear Friend gets if it is a Normal Message.
0.67 x 0.47 x 0.29 = 0.09
So!
Dear Friend
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
p(N) = 0.67
However, technically, it is proportional to the probability that the message is normal, given that it says Dear Friend.
0.67 x 0.47 x 0.29 = 0.09 ∝ p(N | Dear Friend)
So!
p(N | Dear Friend) ∝ 0.09
Dear Friend
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
p(N) = 0.67
So let's put that on top of the normal messages so we don't forget.
0.67 x 0.47 x 0.29 = 0.09 ∝ p(N | Dear Friend)

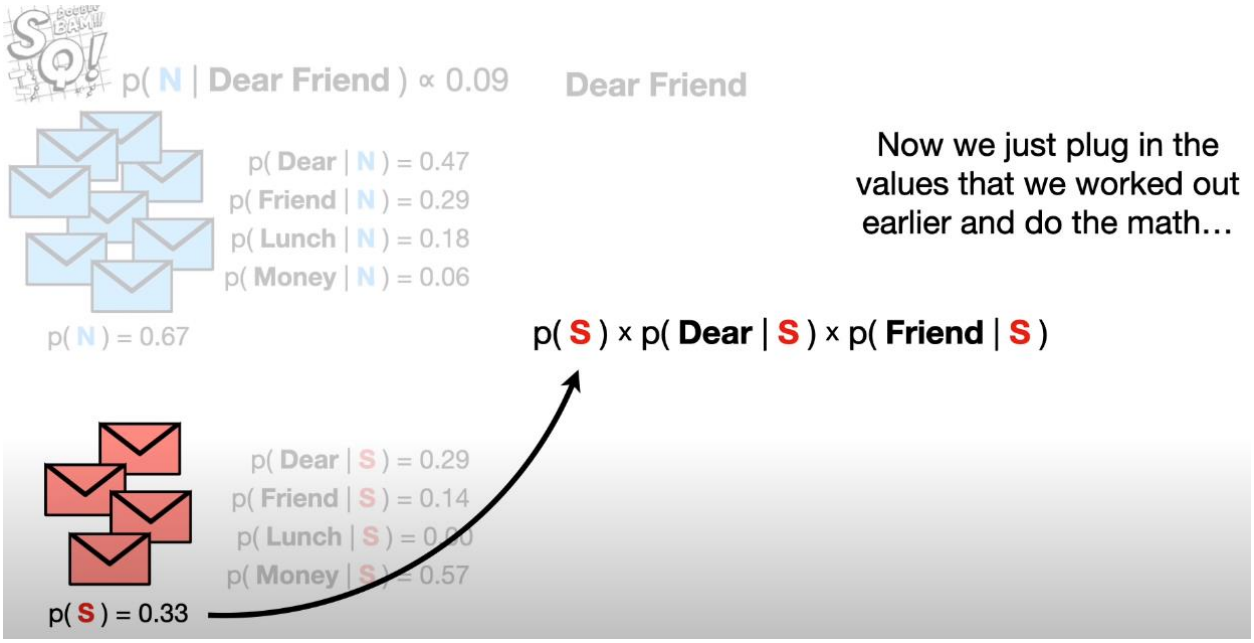

p(N | Dear Friend) ∝ 0.09
Dear Friend
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
p(N) = 0.67
p(S)
Now, just like we did before, We start with an initial guess about the probability that any message, regardless of what it says, is spam.


p(N | Dear Friend) ∝ 0.09
Dear Friend
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
p(N) = 0.67
p(S)
p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57
And just like before, the guess can be any probability that we want, but a common guess is estimated from the training data.

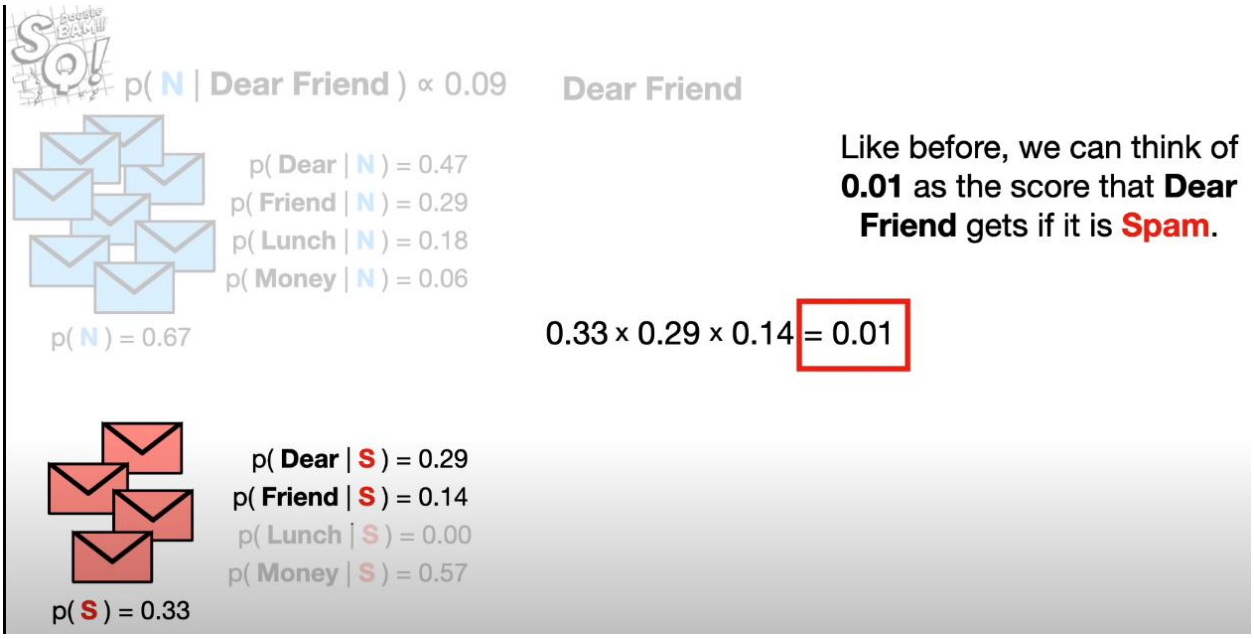

p(N | Dear Friend) ∝ 0.09
Dear Friend
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
p(N) = 0.67
p(S) = 4 / (4 + 8) = 0.33
p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57
And since 4 of the 12 messages are spam, our initial guess will be 0.33.

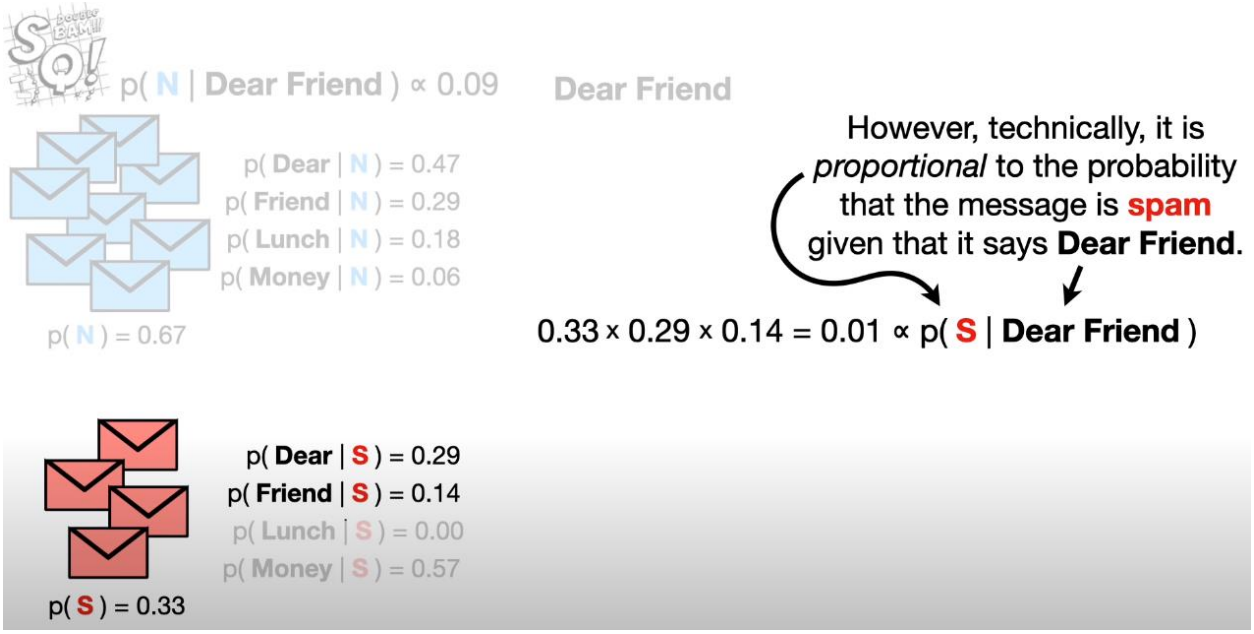

p(N | Dear Friend) ∝ 0.09
Dear Friend
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
p(N) = 0.67
Now we multiply that initial guess by the probability that the word Dear occurs in spam…
p(S) × p(Dear | S)
p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57
p(S) = 0.33

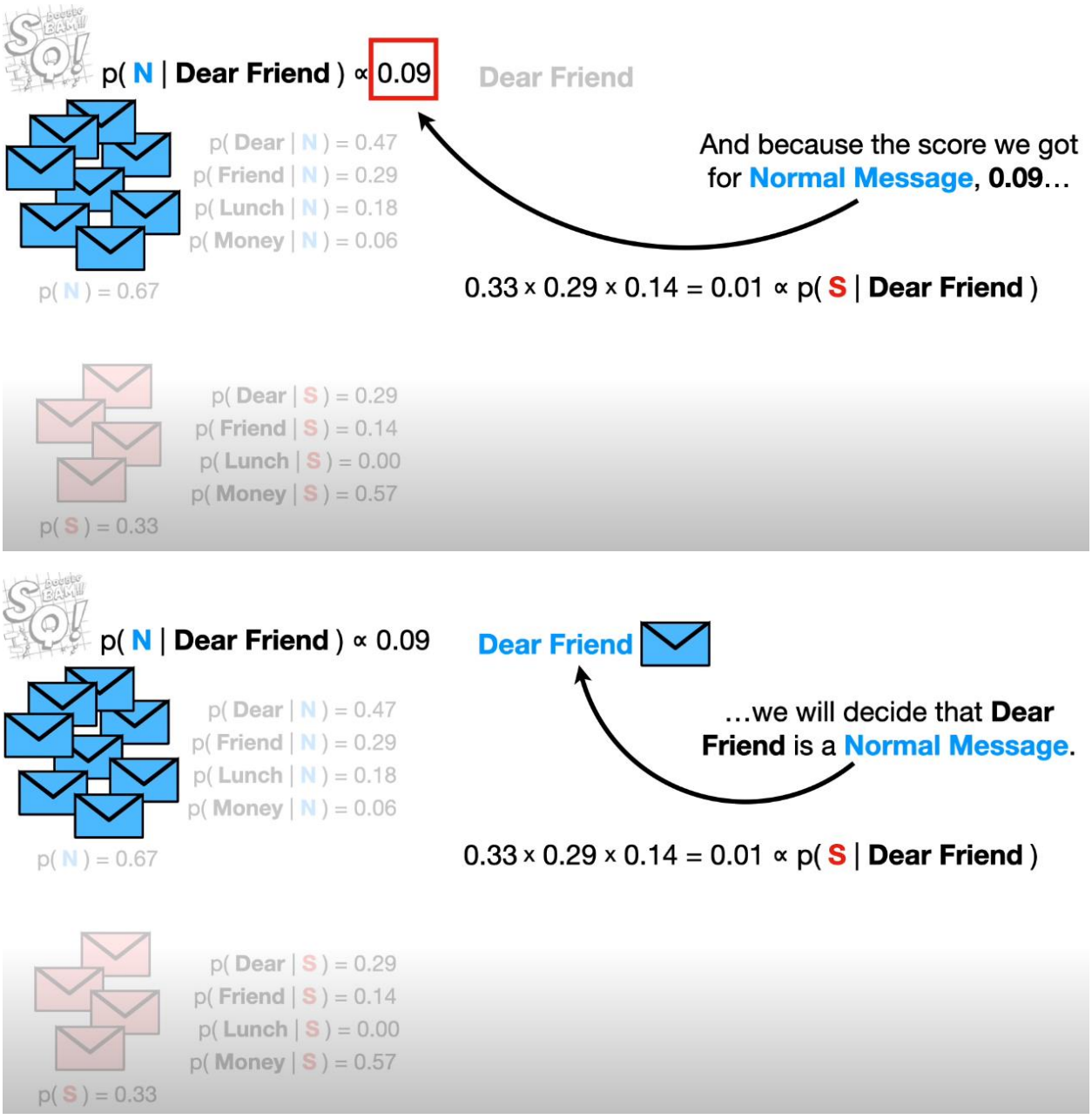

p(N | Dear Friend) ∝ 0.09
Dear Friend
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
p(N) = 0.67
…and the probability that the word Friend occurs in spam.
p(S) × p(Dear | S) × p(Friend | S)
p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57
p(S) = 0.33


p(N | Dear Friend) ∝ 0.09
Dear Friend
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
p(N) = 0.67
Now we just plug in the values that we worked out earlier and do the math…
p(S) × p(Dear | S) × p(Friend | S)
p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57
p(S) = 0.33

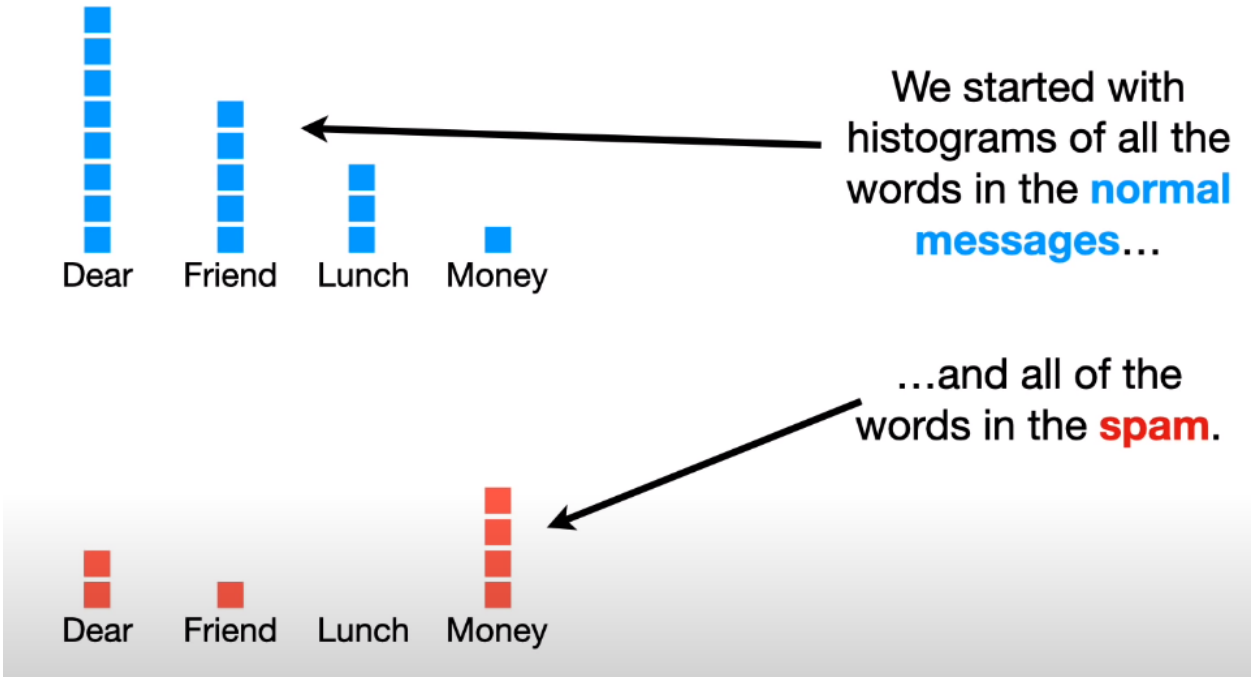

p(N | Dear Friend) ∝ 0.09
Dear Friend
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
p(N) = 0.67
…and we get 0.01.
0.33 × 0.29 × 0.14 = 0.01
p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57
p(S) = 0.33

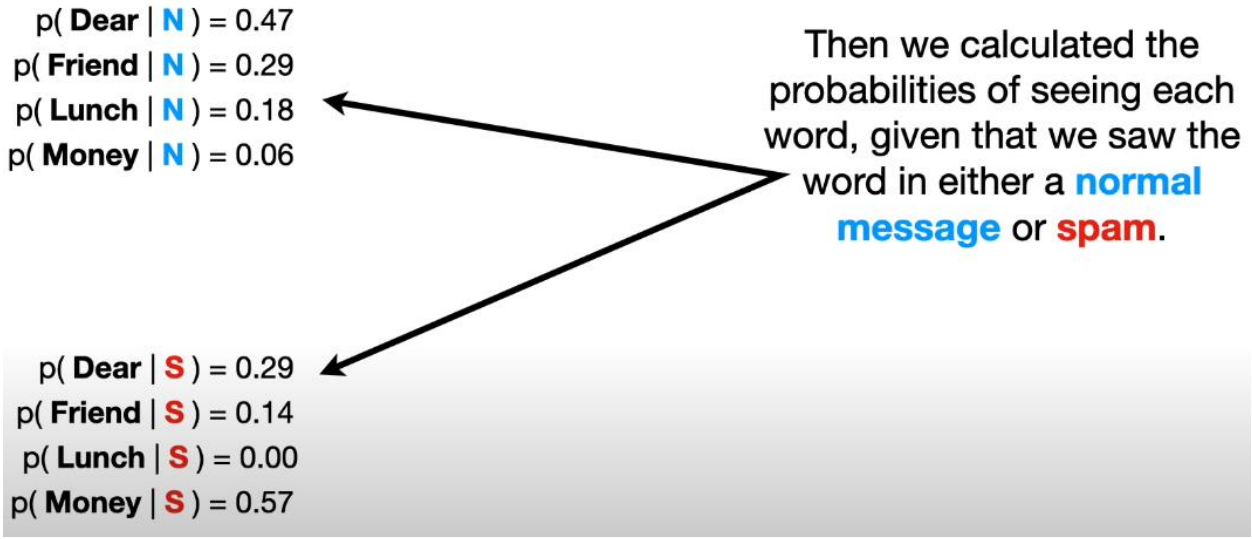

p(N | Dear Friend) ∝ 0.09
Dear Friend
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
p(N) = 0.67
Like before, we can think of 0.01 as the score that Dear Friend gets if it is Spam.
0.33 x 0.29 x 0.14 = 0.01
p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57
p(S) = 0.33

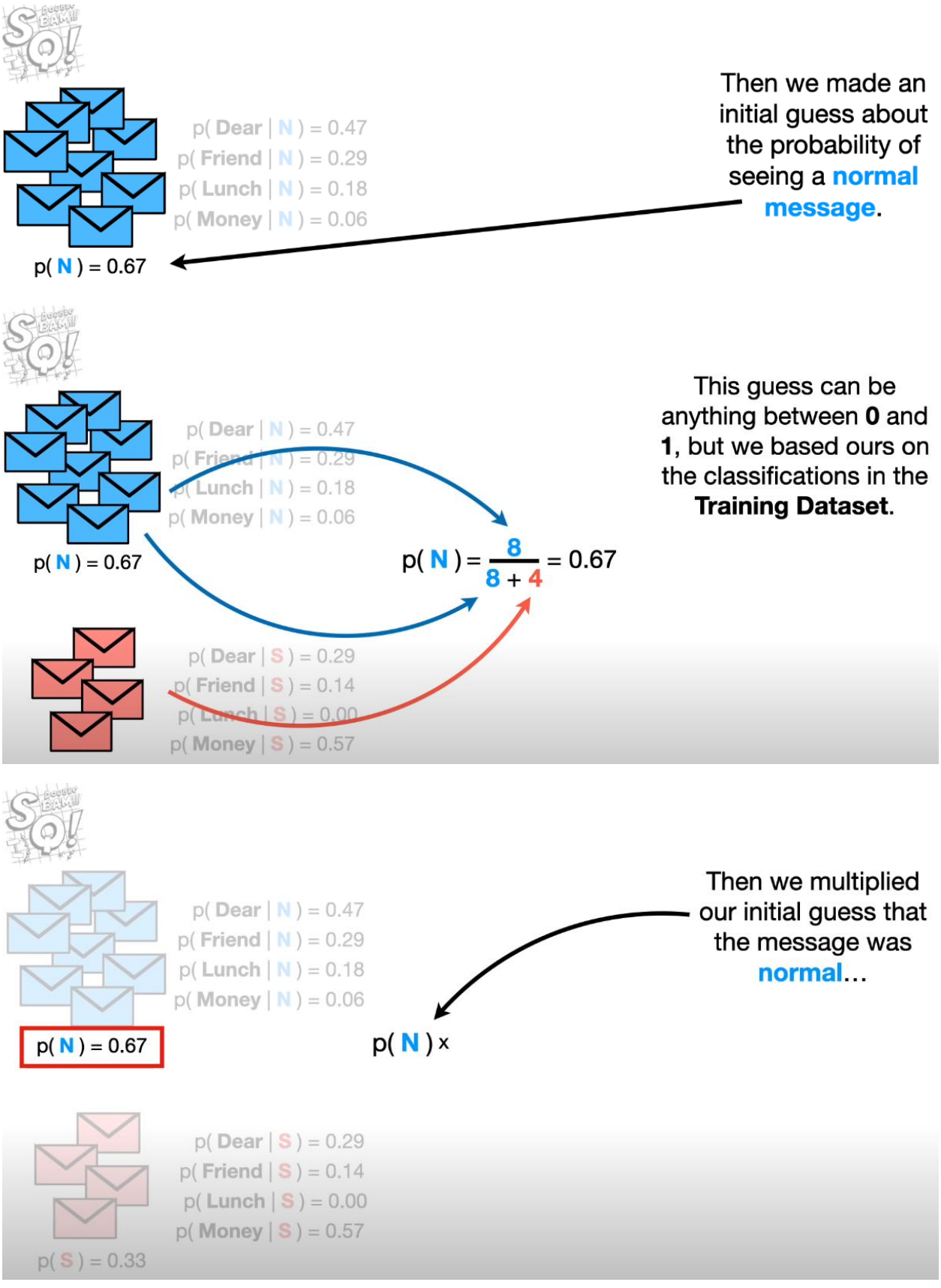

p(N | Dear Friend) ∝ 0.09
Dear Friend
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
p(N) = 0.67
However, technically, it is proportional to the probability that the message is spam given that it says Dear Friend.
0.33 x 0.29 x 0.14 = 0.01 ∝ p(S | Dear Friend)
p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57
p(S) = 0.33

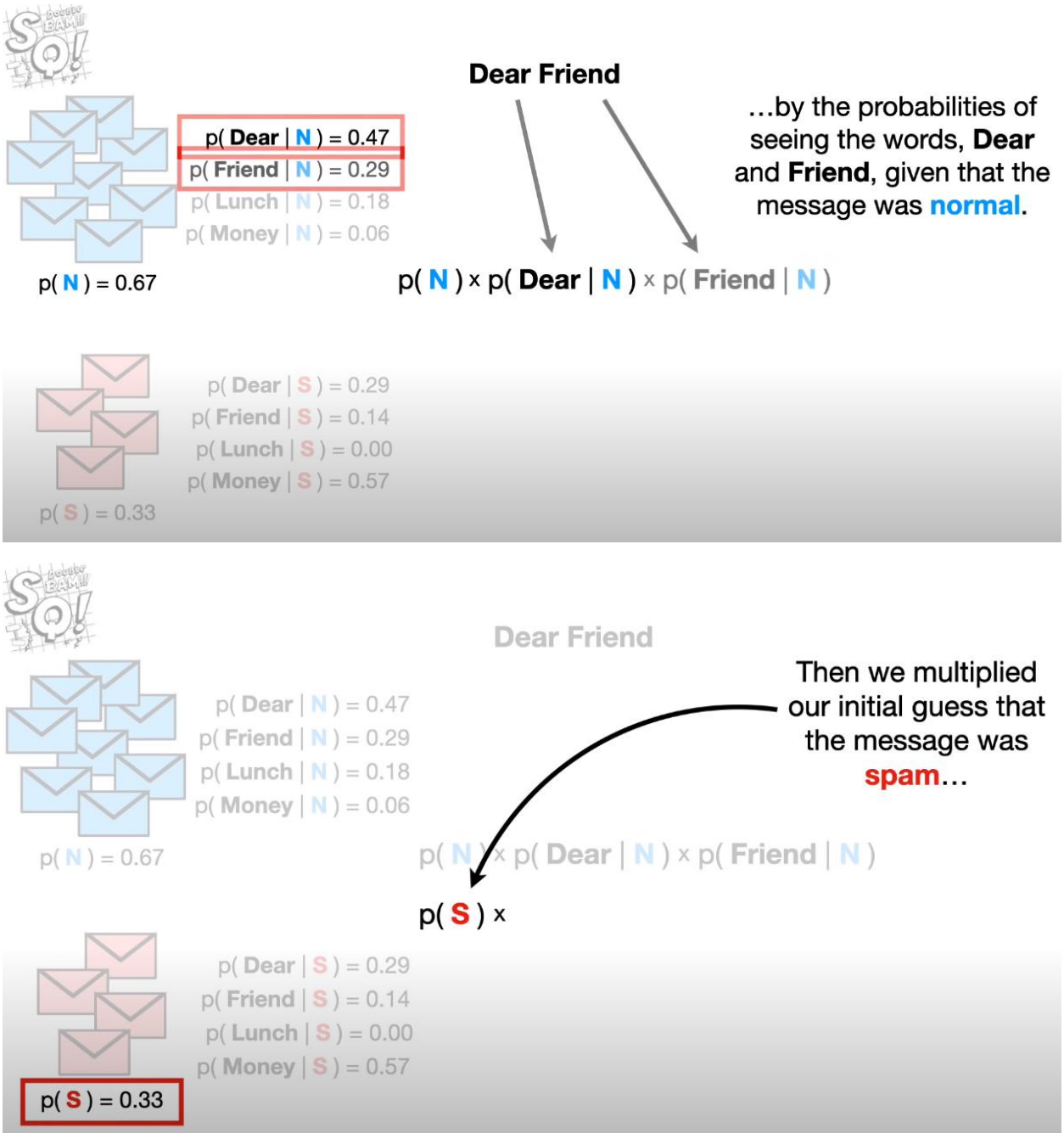


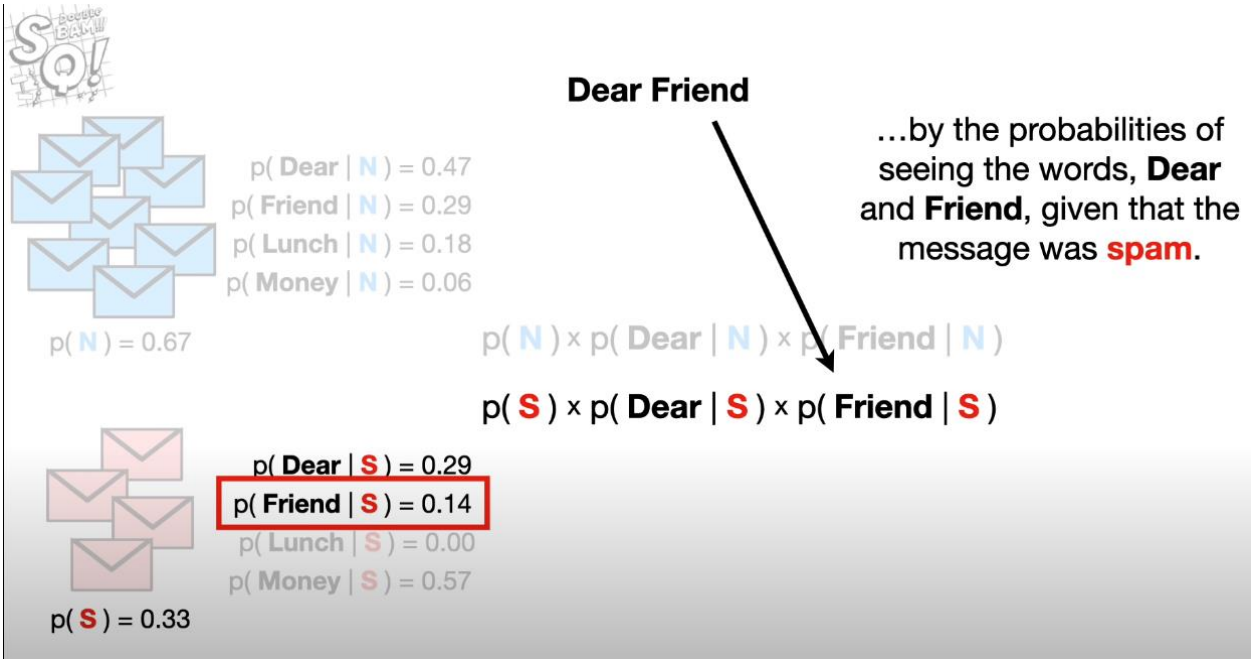
Now, before we move on to a slightly more complex situation, let's review what we've done so far.

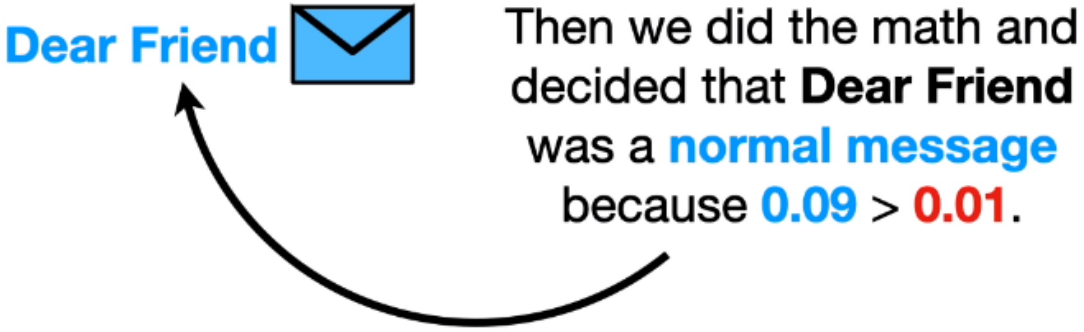

We started with histograms of all the words in the normal messages…
Dear
Friend
Lunch
Money
…and all of the words in the spam.
Dear
Friend
Lunch
Money


p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
Then we calculated the probabilities of seeing each word, given that we saw the word in either a normal message or spam.
p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57


So!
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
p(N) = 0.67
Then we made an initial guess about the probability of seeing a normal message.
So!
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
p(N) = 0.67
p(N) = 8 / (8 + 4) = 0.67
This guess can be anything between 0 and 1, but we based ours on the classifications in the Training Dataset.
p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57
So!
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
p(N) = 0.67
p(N) x
Then we multiplied our initial guess that the message was normal…
p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57
p(S) = 0.33


So!
Dear Friend
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
p(N) = 0.67
…by the probabilities of seeing the words, Dear and Friend, given that the message was normal.
p(N) x p(Dear | N) x p(Friend | N)
p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57
p(S) = 0.33
So!
Dear Friend
Then we multiplied our initial guess that the message was spam…
p(N) x p(Dear | N) x p(Friend | N)
p(S) x
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
p(N) = 0.67
p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57
p(S) = 0.33





$$p(\text{N}) \times p(\text{Dear} \mid \text{N}) \times p(\text{Friend} \mid \text{N}) = 0.09$$

$$p(\text{S}) \times p(\text{Dear} \mid \text{S}) \times p(\text{Friend} \mid \text{S}) = 0.01$$

Now that we understand the basics of how **Naive Bayes Classification** works…

$p(\mathbf{N}) \times p(\mathbf{Dear} \mid \mathbf{N}) \times p(\mathbf{Friend} \mid \mathbf{N}) = 0.09$

$p(\mathbf{S}) \times p(\mathbf{Dear} \mid \mathbf{S}) \times p(\mathbf{Friend} \mid \mathbf{S}) = 0.01$

…let's look at a slightly more complicated example.

Lunch Money Money Money Money

) = 0.47
) = 0.29
) = 0.18
) = 0.06

This time, let's try to classify this message.

Lunch **Money Money Money Money**

) = 0.47
) = 0.29
) = 0.18
) = 0.06

NOTE: This message countians the word **Money** four times…

Lunch **Money Money Money Money**

p(**Dear** | **N**) = 0.47
p(**Friend** | **N**) = 0.29
p(**Lunch** | **N**) = 0.18
p(**Money** | **N**) = 0.06

…and since the probability of seeing the word **Money** is much higher in **spam** **(0.56)** than in **normal messages** **(0.06)**…

p(**Dear** | **S**) = 0.29
p(**Friend** | **S**) = 0.14
p(**Lunch** | **S**) = 0.00
p(**Money** | **S**) = 0.57

Lunch Money Money Money Money

p(**Dear** | **N**) = 0.47
p(**Friend** | **N**) = 0.29
p(**Lunch** | **N**) = 0.18
p(**Money** | **N**) = 0.06

p(**N**) = 0.67

…then it seems reasonable to predict that this message will end up being **spam**.

p(**Dear** | **S**) = 0.29
p(**Friend** | **S**) = 0.14
p(**Lunch** | **S**) = 0.00
p(**Money** | **S**) = 0.57

p(**S**) = 0.33

Lunch Money Money Money Money

p(**Dear** | **N**) = 0.47
p(**Friend** | **N**) = 0.29
p(**Lunch** | **N**) = 0.18
p(**Money** | **N**) = 0.06

So let's do the math!

p(**Dear** | **S**) = 0.29
p(**Friend** | **S**) = 0.14
p(**Lunch** | **S**) = 0.00
p(**Money** | **S**) = 0.57

Calculating the score for a **normal message** works just like before…

Lunch Money Money Money Money

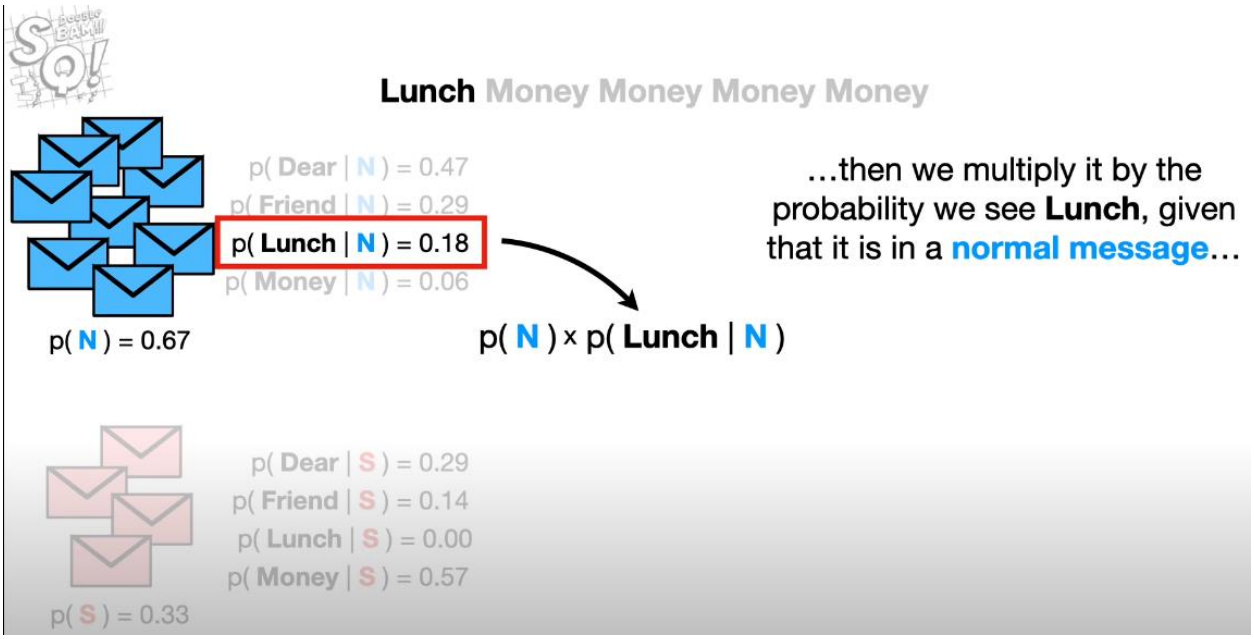
p(**Dear** | **N**) = 0.47
p(**Friend** | **N**) = 0.29
p(**Lunch** | **N**) = 0.18
p(**Money** | **N**) = 0.06

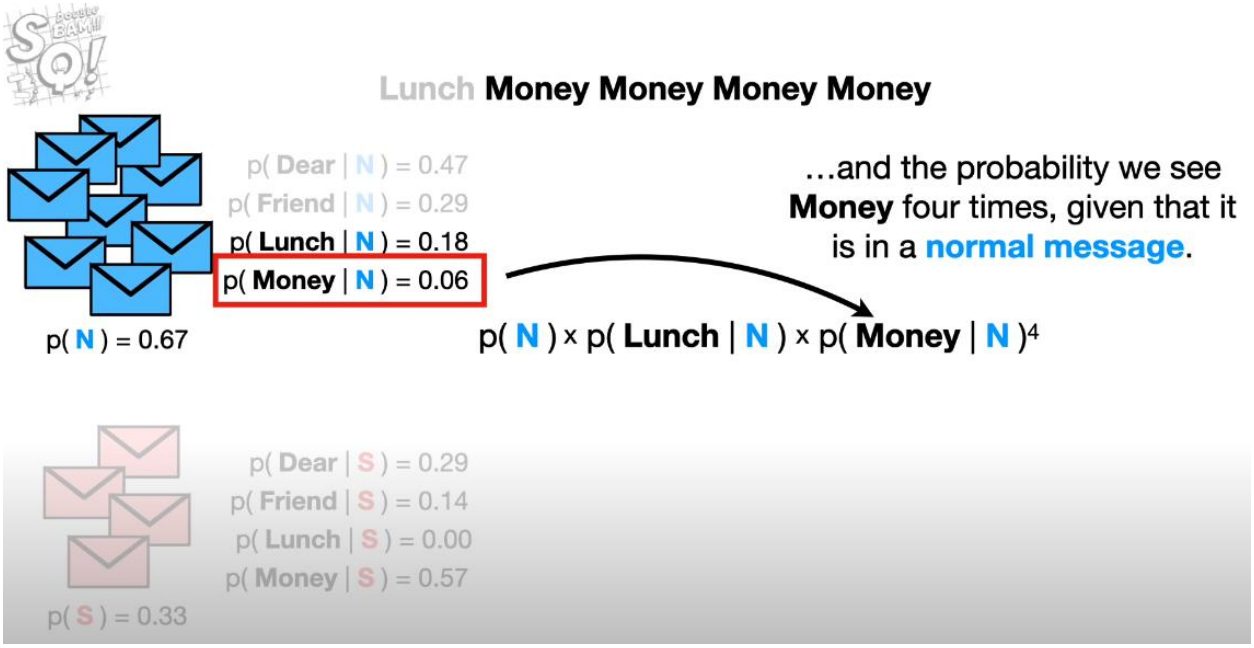
p(**N**) = 0.67 → p(**N**)

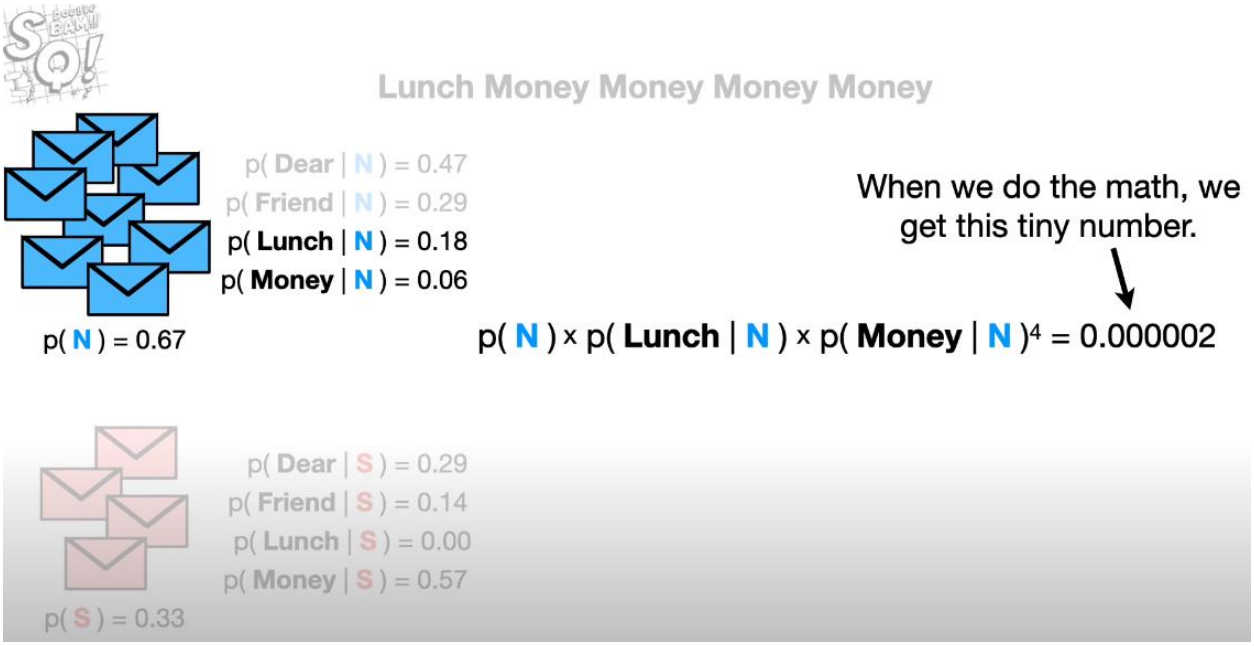
We start with the guess…

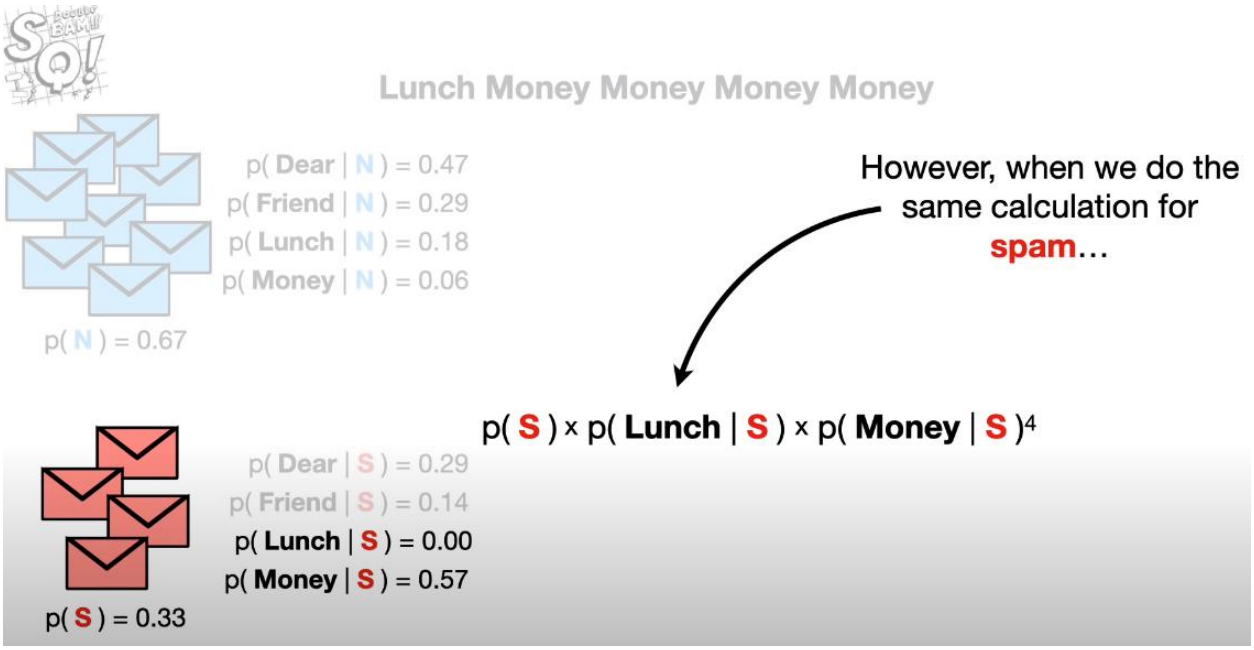
p(**Dear** | **S**) = 0.29
p(**Friend** | **S**) = 0.14
p(**Lunch** | **S**) = 0.00
p(**Money** | **S**) = 0.57

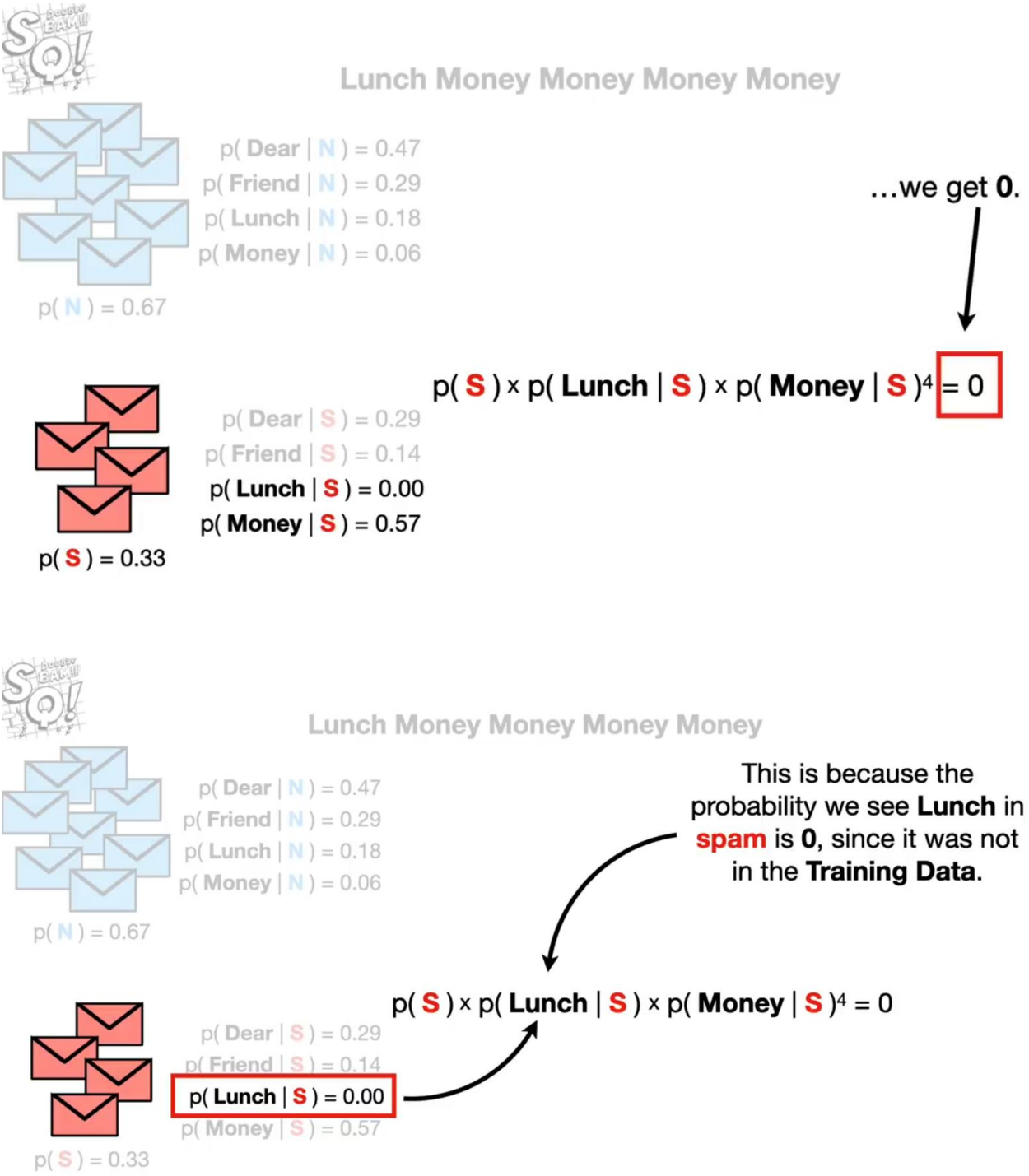
p(**S**) = 0.33

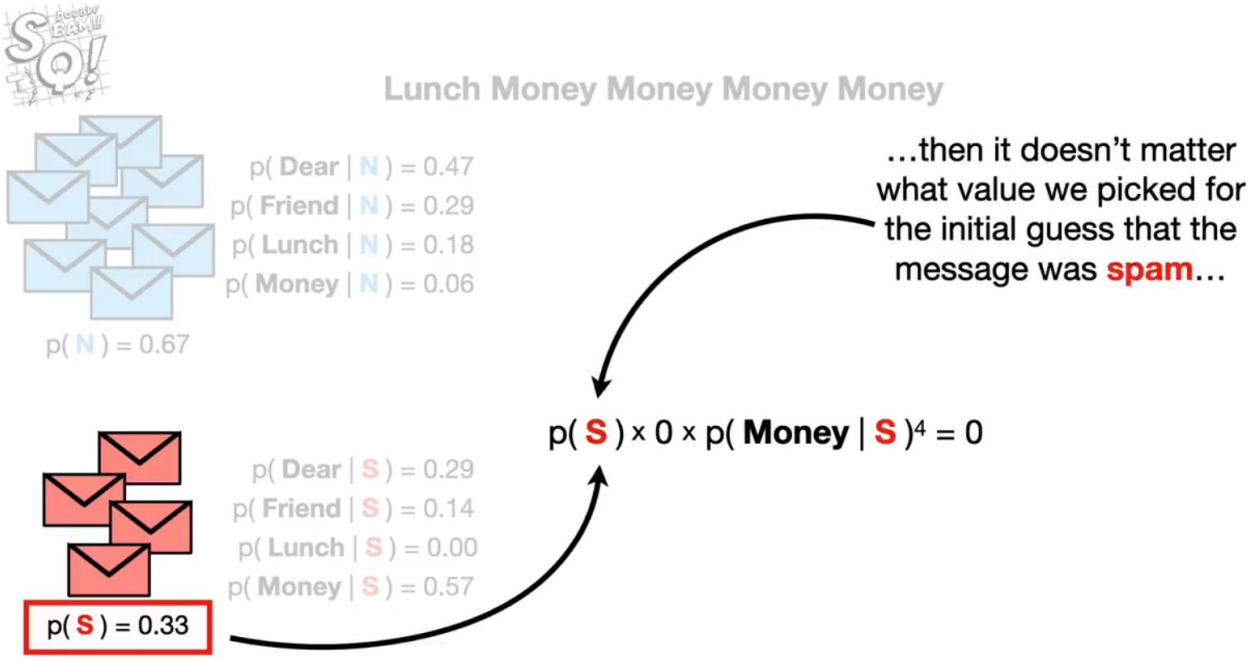

Lunch Money Money Money Money
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
p(N) = 0.67
…then we multiply it by the probability we see Lunch, given that it is in a normal message…
p(N) x p(Lunch | N)
p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57
p(S) = 0.33

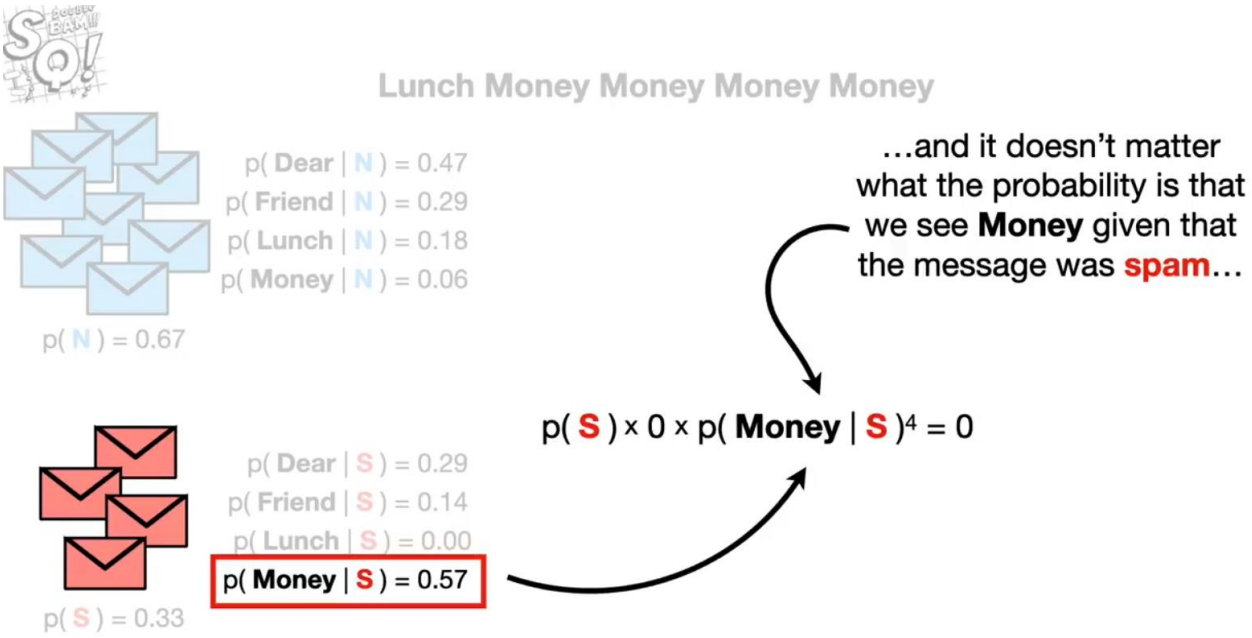

Lunch Money Money Money Money
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
p(N) = 0.67
…and the probability we see Money four times, given that it is in a normal message.
p(N) x p(Lunch | N) x p(Money | N)4
p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57
p(S) = 0.33

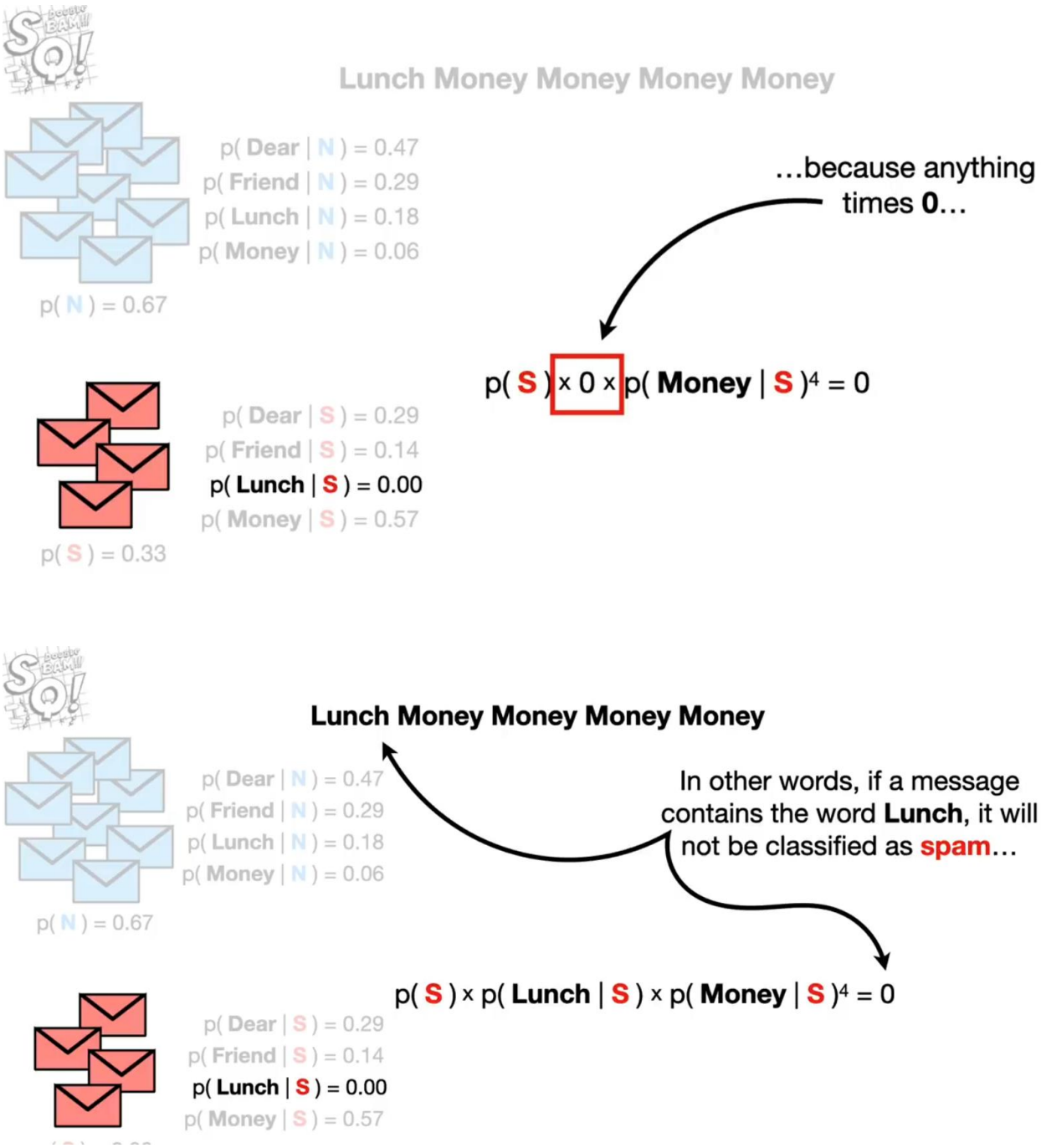

Lunch Money Money Money Money
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
p(N) = 0.67
When we do the math, we get this tiny number.
p(N) x p(Lunch | N) x p(Money | N)4 = 0.000002
p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57
p(S) = 0.33


Lunch Money Money Money Money
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
p(N) = 0.67
However, when we do the same calculation for spam…
p(S) x p(Lunch | S) x p(Money | S)4
p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57
p(S) = 0.33


Lunch Money Money Money Money
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
p(N) = 0.67
...we get 0.
p(S) x p(Lunch | S) x p(Money | S)4 = 0
p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57
p(S) = 0.33
Lunch Money Money Money Money
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
p(N) = 0.67
This is because the probability we see Lunch in spam is 0, since it was not in the Training Data.
p(S) x p(Lunch | S) x p(Money | S)4 = 0
p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57
p(S) = 0.33


Lunch Money Money Money Money
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
p(N) = 0.67
…then it doesn't matter what value we picked for the initial guess that the message was spam…
p(S) x 0 x p(Money | S)4 = 0
p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57
p(S) = 0.33


Lunch Money Money Money Money
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
p(N) = 0.67
…and it doesn't matter what the probability is that we see Money given that the message was spam…
p(S) x 0 x p(Money | S)4 = 0
p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57
p(S) = 0.33


So!
Lunch Money Money Money Money
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
p(N) = 0.67
…because anything times 0…
p(S) x 0 x p(Money | S)4 = 0
p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57
p(S) = 0.33
So!
Lunch Money Money Money Money
In other words, if a message contains the word Lunch, it will not be classified as spam…
p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06
p(N) = 0.67
p(S) x p(Lunch | S) x p(Money | S)4 = 0
p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57

Lunch Money Money Money Money

p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06

p(N) = 0.67

…and that means we will always classify the messages with **Lunch** in them as **normal**, no matter how how many times we see the word **Money**.

$$p(\text{N}) \times p(\text{Lunch} \mid \text{N}) \times p(\text{Money} \mid \text{N})^4 = 0.000002$$

$$p(\text{S}) \times p(\text{Lunch} \mid \text{S}) \times p(\text{Money} \mid \text{S})^4 = 0$$

p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57

p(S) = 0.33

Lunch Money Money Money Money

p(Dear | N) = 0.47
p(Friend | N) = 0.29
p(Lunch | N) = 0.18
p(Money | N) = 0.06

p(N) = 0.67

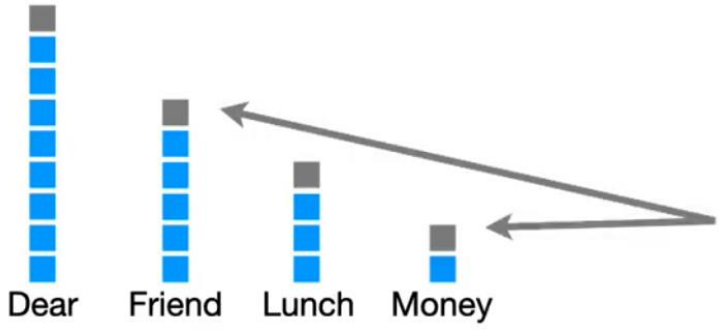
And that's a problem.

$$p(\text{N}) \times p(\text{Lunch} \mid \text{N}) \times p(\text{Money} \mid \text{N})^4 = 0.000002$$

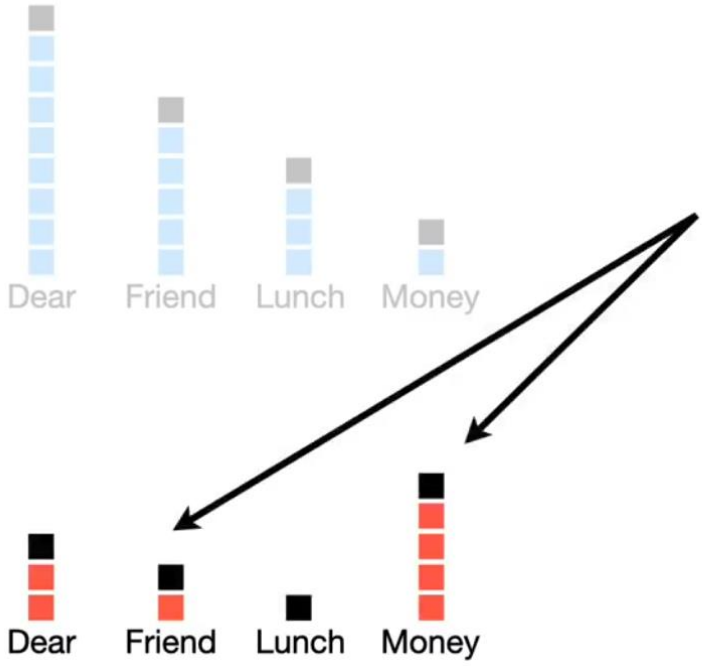
$$p(\text{S}) \times p(\text{Lunch} \mid \text{S}) \times p(\text{Money} \mid \text{S})^4 = 0$$

p(Dear | S) = 0.29
p(Friend | S) = 0.14
p(Lunch | S) = 0.00
p(Money | S) = 0.57

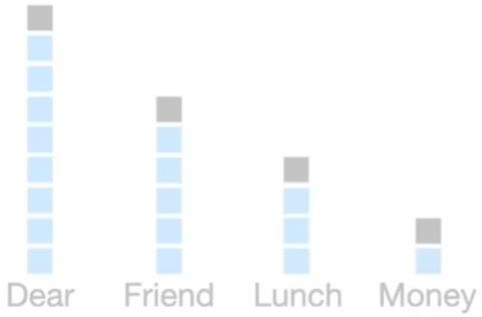
p(S) = 0.33



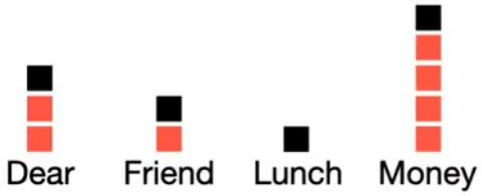
To work around this problem, people usually add **1** count, represented by a **black box**, to each word in the histograms.

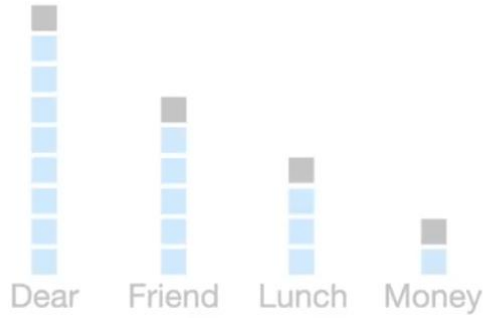


To work around this problem, people usually add **1** count, represented by a **black box**, to each word in the histograms.

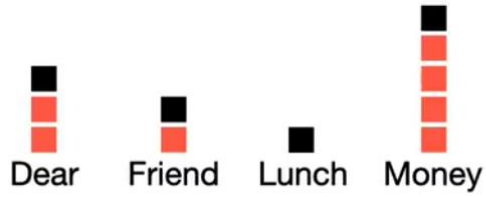


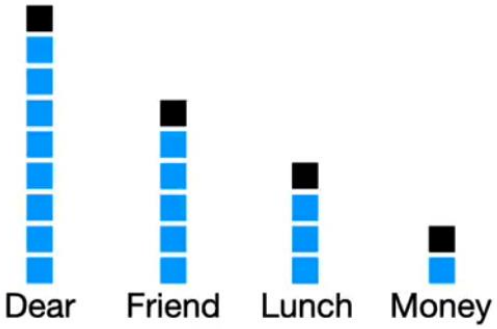
NOTE: The number of counts we add to each word is typically referred to with the Greek letter ***α* (alpha)**.



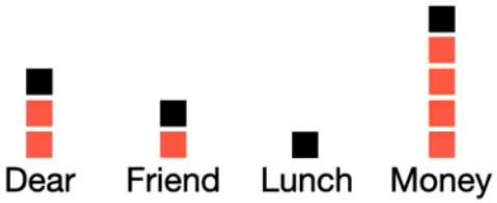


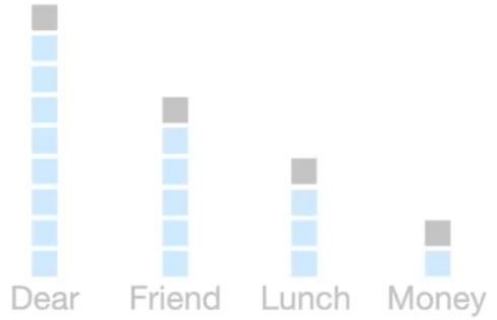
In this case, ***α* = 1**, but we could have set it to anything.





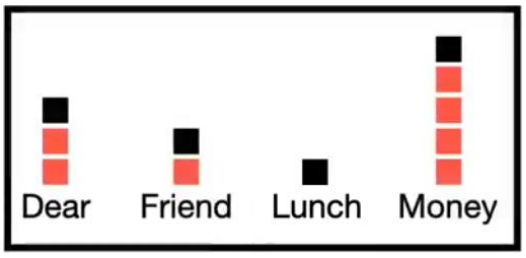
Anyway, now when we calculate the probabilities of observing each word, we never get **0**.

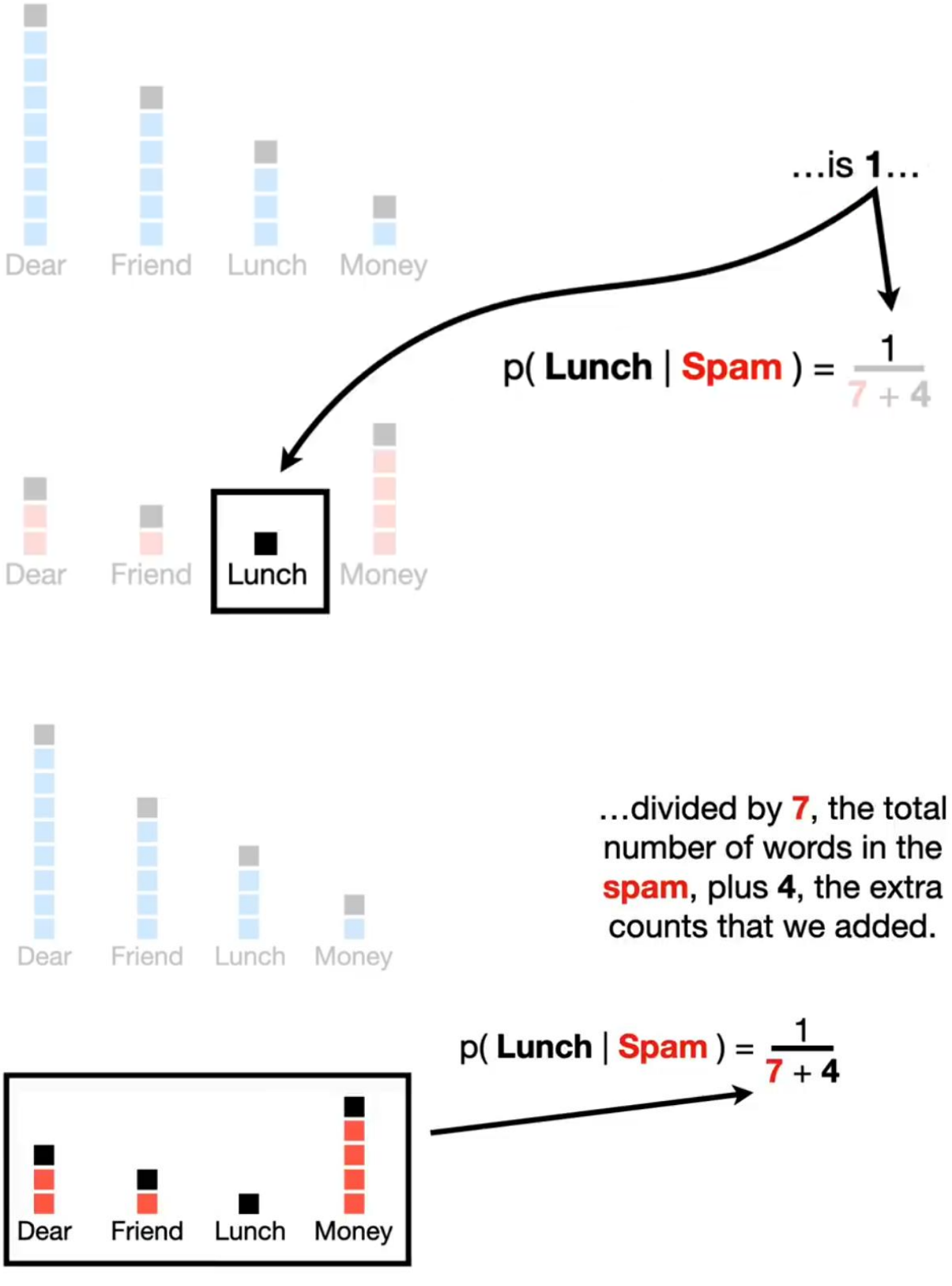


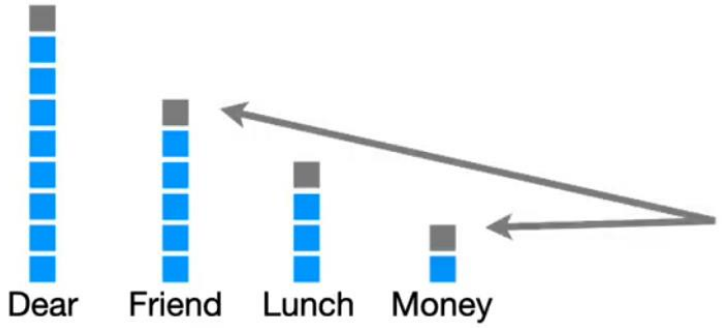


For example, the probability of seeing **Lunch**, given that it is in **spam**…

p(**Lunch** | **Spam**)




Dear
Friend
Lunch
Money
...is 1...
p(Lunch | Spam) = 1 / 7 + 4
Dear
Friend
Lunch
Money
Dear
Friend
Lunch
Money
...divided by 7, the total number of words in the spam, plus 4, the extra counts that we added.
p(Lunch | Spam) = 1 / 7 + 4
Dear
Friend
Lunch
Money

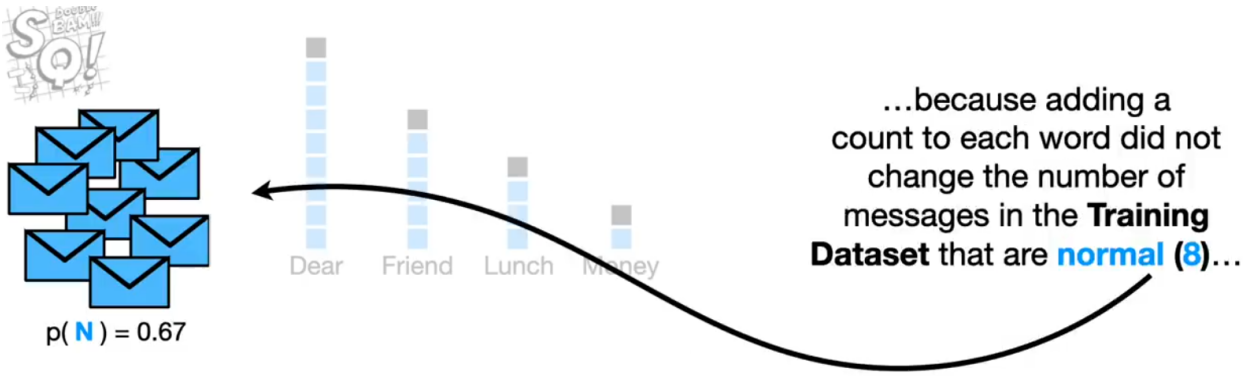


To work around this problem, people usually add **1** count, represented by a **black box**, to each word in the histograms.

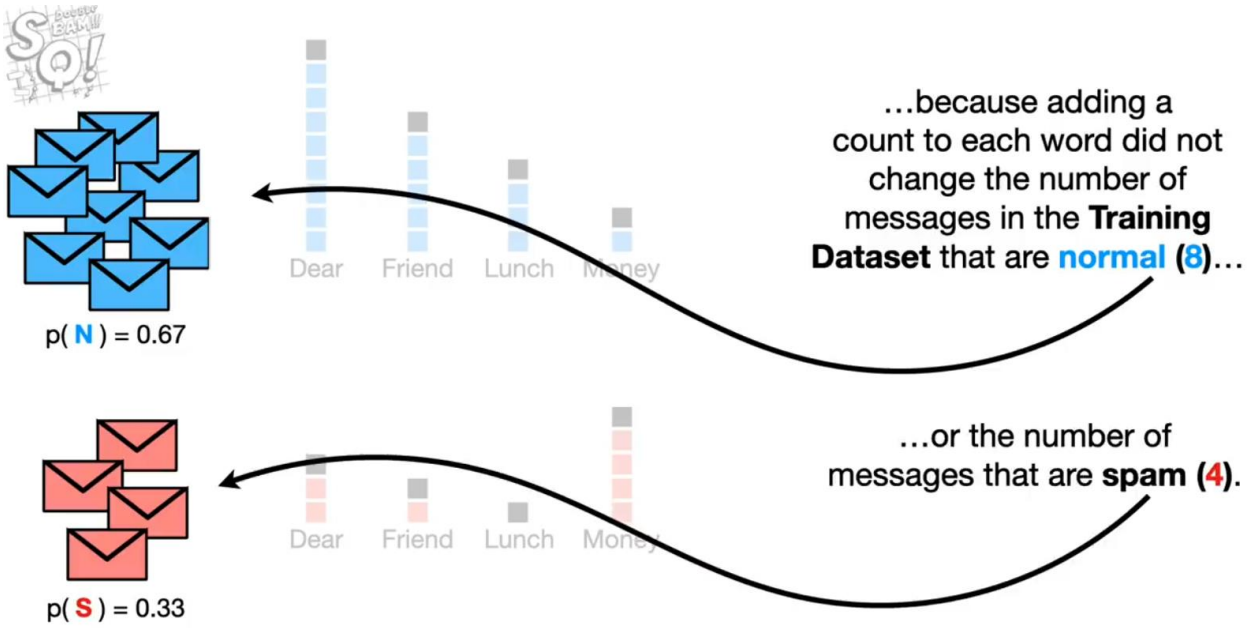
And that gives us **0.09**.

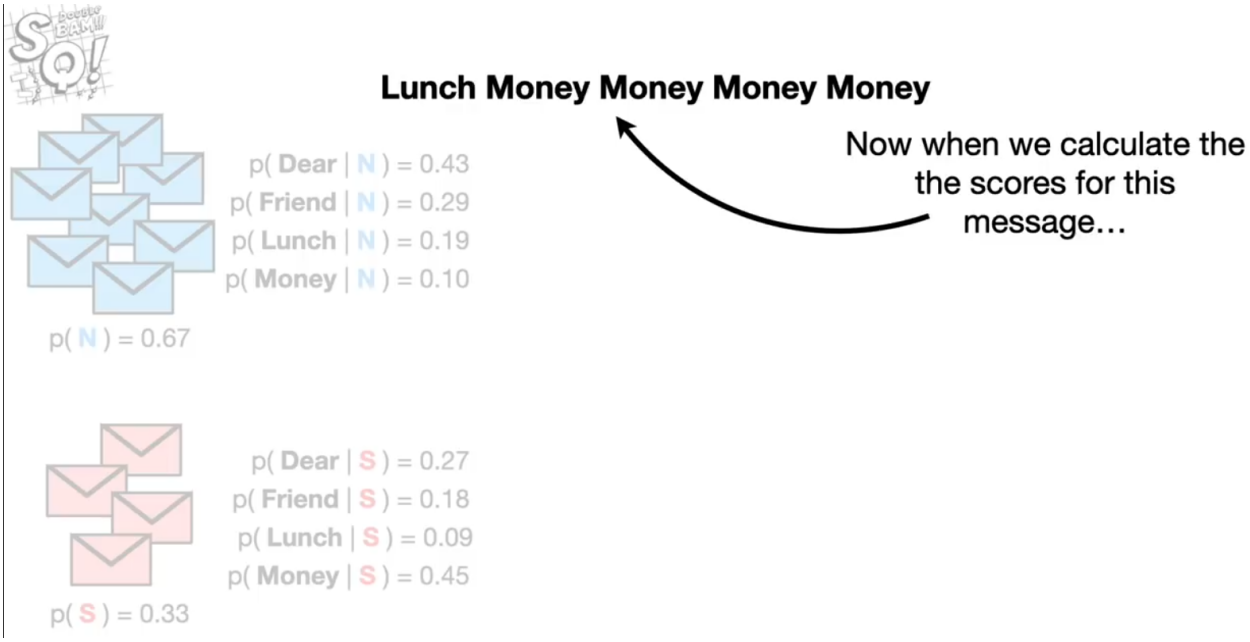
$$p(\text{Lunch} \mid \text{Spam}) = \frac{1}{7+4} = 0.09$$

NOTE: Adding counts (**black boxes**) to each word does not change our initial guess that a message is **normal**, **p(N)**…



…because adding a count to each word did not change the number of messages in the **Training Dataset** that are **normal** (**8**)…


Dear
Friend
Lunch
Money
p(N) = 0.67
…because adding a count to each word did not change the number of messages in the Training Dataset that are normal (8)…
Dear
Friend
Lunch
Money
p(S) = 0.33
…or the number of messages that are spam (4).


Lunch Money Money Money Money
Now when we calculate the the scores for this message…
p(Dear | N) = 0.43
p(Friend | N) = 0.29
p(Lunch | N) = 0.19
p(Money | N) = 0.10
p(N) = 0.67
p(Dear | S) = 0.27
p(Friend | S) = 0.18
p(Lunch | S) = 0.09
p(Money | S) = 0.45
p(S) = 0.33

Lunch Money Money Money Money

p(Dear | N) = 0.43
p(Friend | N) = 0.29
p(Lunch | N) = 0.19
p(Money | N) = 0.10

…but now when we calculate the value for **spam**, we get a value **> 0**.

$$p(N) \times p(\text{Lunch} \mid N) \times p(\text{Money} \mid N)^4 = 0.00001$$

$$p(S) \times p(\text{Lunch} \mid S) \times p(\text{Money} \mid S)^4 = 0.00122$$

p(Dear | S) = 0.27
p(Friend | S) = 0.18
p(Lunch | S) = 0.09
p(Money | S) = 0.45

Lunch Money Money Money Money

p(Dear | N) = 0.43
p(Friend | N) = 0.29
p(Lunch | N) = 0.19
p(Money | N) = 0.10

And since the value for **spam** is greater than the one for a **normal message**…

$$p(N) \times p(\text{Lunch} \mid N) \times p(\text{Money} \mid N)^4 = 0.00001$$

$$p(S) \times p(\text{Lunch} \mid S) \times p(\text{Money} \mid S)^4 = 0.00122$$

p(Dear | S) = 0.27
p(Friend | S) = 0.18
p(Lunch | S) = 0.09
p(Money | S) = 0.45

Lunch Money Money Money Money

p(**Dear** | **N**) = 0.43
p(**Friend** | **N**) = 0.29
p(**Lunch** | **N**) = 0.19
p(**Money** | **N**) = 0.10

…we classify the message as **spam**.

$$p(\mathbf{N}) \times p(\mathbf{Lunch} \mid \mathbf{N}) \times p(\mathbf{Money} \mid \mathbf{N})^4 = 0.00001$$

$$p(\mathbf{S}) \times p(\mathbf{Lunch} \mid \mathbf{S}) \times p(\mathbf{Money} \mid \mathbf{S})^4 = 0.00122$$

p(**Dear** | **S**) = 0.27
p(**Friend** | **S**) = 0.18
p(**Lunch** | **S**) = 0.09
p(**Money** | **S**) = 0.45

Now let's talk about why **Naive Bayes** is ***naive***.

p(**N**) = 0.67

p(**Dear** | **N**) = 0.43
p(**Friend** | **N**) = 0.29
p(**Lunch** | **N**) = 0.19
p(**Money** | **N**) = 0.10

The thing that makes **Naive Bayes** so ***naive*** is that it treats all word orders the same.

p(**Dear** | **N**) = 0.43
p(**Friend** | **N**) = 0.29
p(**Lunch** | **N**) = 0.19
p(**Money** | **N**) = 0.10

p(**N**) = 0.67

For example, the **Normal** message score for the phrase, **Dear Friend**…

Score for **Dear Friend** =

p(**N**) × p(**Dear** | **N**) × p(**Friend** | **N**) = 0.08

p(**Dear** | **N**) = 0.43
p(**Friend** | **N**) = 0.29
p(**Lunch** | **N**) = 0.19
p(**Money** | **N**) = 0.10

p(**N**) = 0.67

…is the exact same as the score for **Friend Dear**.

Score for **Dear Friend** =

p(**N**) × p(**Dear** | **N**) × p(**Friend** | **N**) = 0.08

Score for **Friend Dear** =

p(**N**) × p(**Friend** | **N**) × p(**Dear** | **N**) = 0.08

p(**Dear** | **N**) = 0.43
p(**Friend** | **N**) = 0.29
p(**Lunch** | **N**) = 0.19
p(**Money** | **N**) = 0.10

p(**N**) = 0.67

In other words, regardless of how the words are ordered, we get **0.08**.

Score for **Dear Friend** =

p(**N**) × p(**Dear** | **N**) × p(**Friend** | **N**) = 0.08

Score for **Friend Dear** =

p(**N**) × p(**Friend** | **N**) × p(**Dear** | **N**) = 0.08

Treating all word orders equal is very different from how you and I communicate.

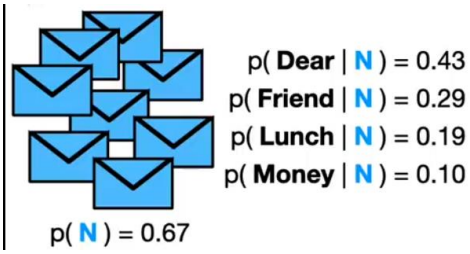
Score for **Dear Friend** =

p(N) × p(**Dear** | N) × p(**Friend** | N) = 0.08

Score for **Friend Dear** =

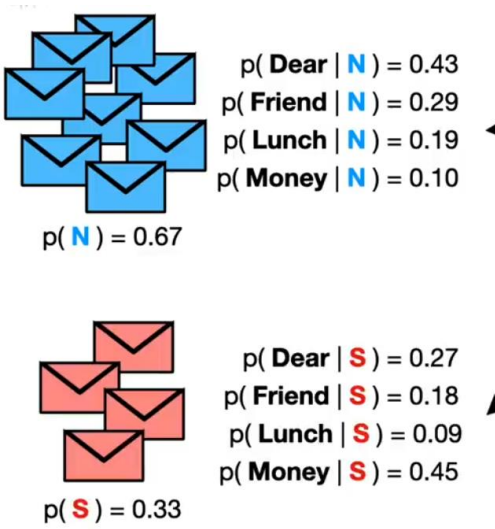
p(N) × p(**Friend** | N) × p(**Dear** | N) = 0.08

Every language has grammar rules and common phrases, but **Naive Bayes** ignores all of that stuff.



Instead, **Naive Bayes** treats language like it just a bag full of words and each message is random handful of them.

Naive Bayes ignores all the rules because keeping track of every single reasonable phrase in a language would be impossible.



That said, even though **Naive Bayes** is ***naive***, it tends to perform surprisingly well when separating **Normal** messages from **Spam**.

In **Machine Learning Lingo**, we'd say that by ignoring relationships among words, **Naive Bayes** has high **bias**...

...but because it works well in practice, **Naive Bayes** has low **variance**.

Gaussian Distribution

The Gaussian Distribution is pretty common in the case of continuous probability distribution. The distribution is frequently used in statistics and it is generally required in natural or social sciences to showcase the real-valued random variables. The probability density formula for Gaussian Distribution in mathematics is given as below –

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

Where,
x is the variable
μ is the mean
sigma is the standard deviation

You must be wondering what is the usage of Gaussian functions in statistics. They are used to describe the normal distributions and signal processing for images. They are also required in the heat transfer and diffusion of equations to define the wire transformation.

The normal distribution (or Gaussian distribution), also referred as bell curve, is very useful due to the central limit theorem. Normal distribution states which are average of random variables converge in distribution to the normal and are normally distributed when the number of random variables is large.

The probability density of the normal distribution can be computed as

$$f\left(x|\mu,\sigma^2\right) = \frac{1}{\sqrt{2\sigma^2\pi}} e^{-(x-\mu)^2/2\sigma^2}, \tag{8.3}$$

where μ is the mean of the distribution, σ is the standard deviation, and σ^2 is the variance.

Gaussian Distribution Formula

Gaussian distribution is very common in a continuous probability distribution. The Gaussian distributions are important in statistics and are often used in the natural and social sciences to represent real-valued random variables. Check out the Gaussian distribution formula below.

Formula of Gaussian Distribution

The probability density function formula for Gaussian distribution is given by,

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

Where,

x is the variable

μ is the mean

σ is the standard deviation

Solved Examples

Question 1: Calculate the probability density function of Gaussian distribution using the following data. x = 2, μ = 5 and σ = 3

Solution:

From the question it is given that, x = 2, μ = 5 and σ = 3

Probability density function formula of Gaussian distribution is,

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

$f(2, 5, 3) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-(2-5)^2}{18}}$

$=[1/(3 \times 2.51)] \times 0.6065$

$= 0.1328 \times 0.6065$

$= 0.0805$

(1) Calculate the probability density function of Gaussian distribution using the following data. x = 5, μ = 7 and σ = 4.

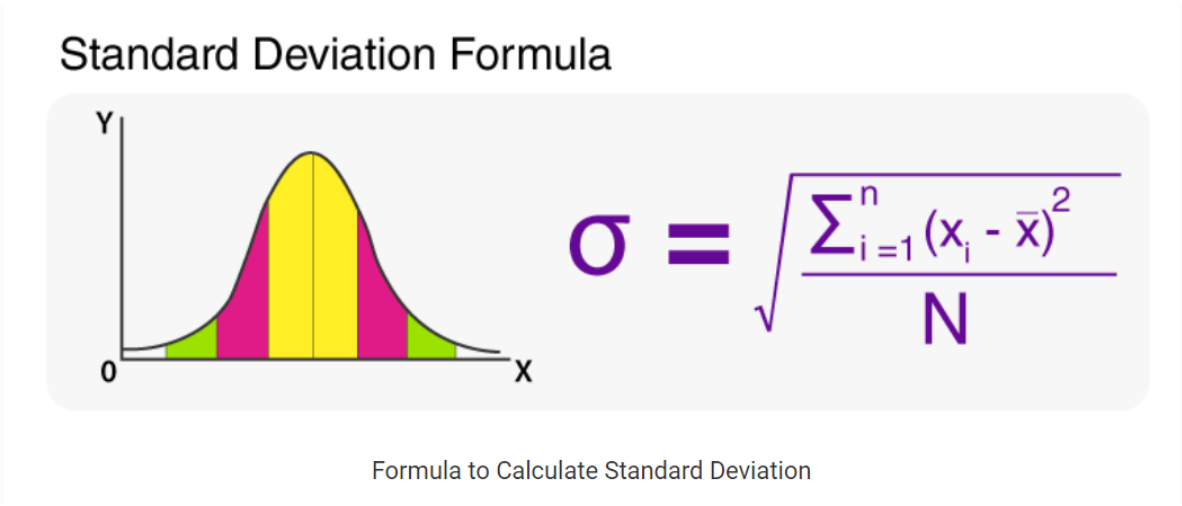
(2) During a survey, 6 students were asked how many hours per day they study on an average? Their answers were as follows: 2, 6, 5, 3, 2, 3. Evaluate the standard deviation.

Standard Deviation Formula

Standard deviation formula is used to find the values of a particular data that is dispersed. In simple words, the standard deviation is defined as the deviation of the values or data from an average mean. Lower standard deviation concludes that the values are very close to their average. Whereas higher values mean the values are far from the mean value. It should be noted that the standard deviation value can never be negative.

Standard Deviation is of two types:

1. Population Standard Deviation
2. Sample Standard Deviation



Formula to Calculate Standard Deviation

Formulas for Standard Deviation

Population Standard Deviation Formula	$\sigma = \sqrt{\frac{\sum(X-\mu)^2}{n}}$
Sample Standard Deviation Formula	$s = \sqrt{\frac{\sum(X-\bar{X})^2}{n-1}}$

Notations for Standard Deviation

- σ = Standard Deviation
- x_i = Terms Given in the Data
- $\bar{x}$ = Mean
- n = Total number of Terms

Standard Deviation Formula Based on Discrete Frequency Distribution

For discrete frequency distribution of the type:

x: $x_1, x_2, x_3, \ldots x_n$ and

f: $f_1, f_2, f_3, \ldots f_n$

The formula for standard deviation becomes:

$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{n} f_i(x_i - \bar{x})^2}$$

Here, N is given as:

$N = {}^{n}\sum_{i=1} f_i$

Standard Deviation Formula for Grouped Data

There is another standard deviation formula which is derived from the variance. This formula is given as:

$$\sigma = \frac{1}{N}\sqrt{\sum_{i=i}^{n} f_i x_i^2 - (\sum_{i=1}^{n} f_i x_i)^2}$$

Also Check: Difference Between Variance and Standard Deviation

Question: During a survey, 6 students were asked how many hours per day they study on an average? Their answers were as follows: 2, 6, 5, 3, 2, 3. Evaluate the standard deviation.

Solution:

Find the mean of the data:

$\frac{(2+6+5+3+2+3)}{6}$ = 3.5

Step 2: Construct the table:

x_1	$x_1 - \bar{x}$	$(x_1 - \bar{x})^2$
2	-1.5	2.25
6	2.5	6.25
5	1.5	2.25
3	-0.5	0.25
2	-1.5	2.25
3	-0.5	0.25
= 13.5		

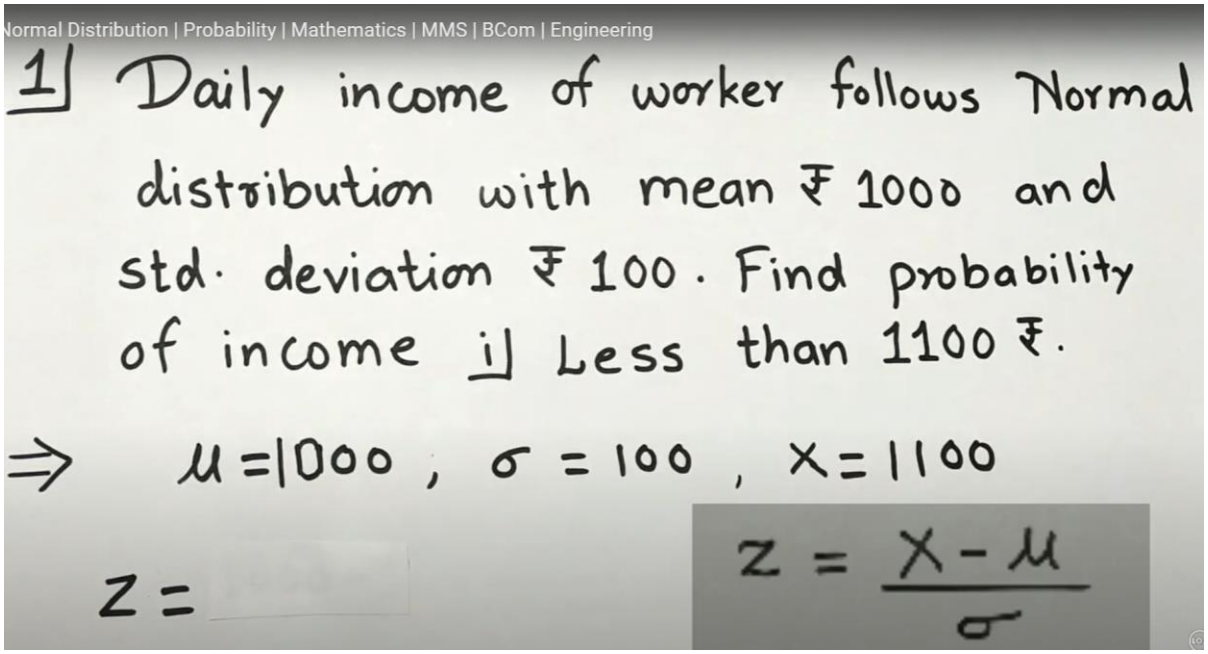
Step 3: Now, use the Standard Deviation formula

Sample Standard Deviation = $s = \sqrt{\frac{\sum(X-\bar{X})^2}{n-1}}$

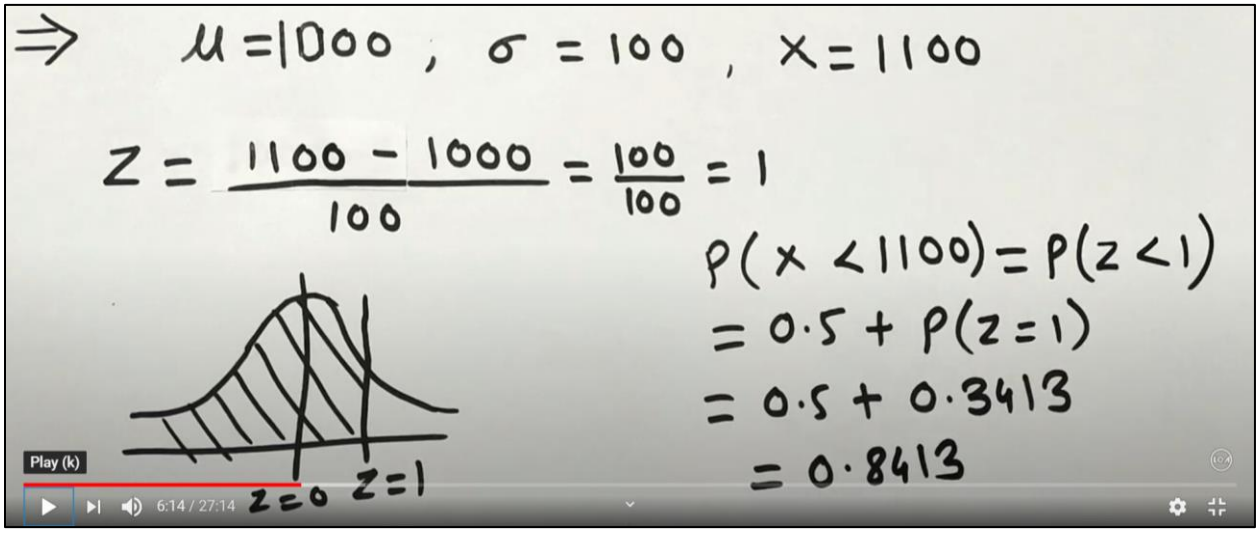
=√(13.5/[6-1])

=√[2.7]

=1.643


Normal Distribution | Probability | Mathematics | MMS | BCom | Engineering
1] Daily income of worker follows Normal distribution with mean ₹ 1000 and std. deviation ₹ 100. Find probability of income i] Less than 1100 ₹.
⇒ μ = 1000 , σ = 100 , X = 1100
Z =
Z = (X − μ) / σ

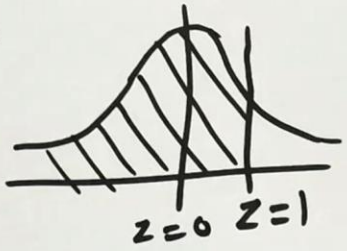
Ans:


⇒ μ = 1000 , σ = 100 , X = 1100
Z = (1100 − 1000)/100 = 100/100 = 1
P(X < 1100) = P(Z < 1)
= 0.5 + P(Z = 1)
= 0.5 + 0.3413
= 0.8413
Play (k)
6:14 / 27:14
Z = 0
Z = 1

std. deviation ₹ 100. Find probability of income i) Less than 1100 ₹.

$\Rightarrow \quad \mu = 1000, \quad \sigma = 100, \quad X = 1100$

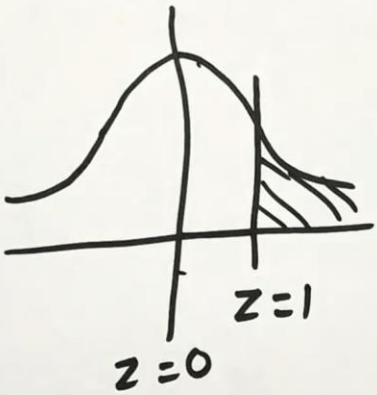
$$Z = \frac{1100 - 1000}{100} = \frac{100}{100} = 1$$



$$P(x < 1100) = P(z < 1)$$
$$= 0.5 + P(z = 1)$$
$$= 0.5 + 0.3413$$
$$= 0.8413 //$$

ii) More than ₹ 1100.

$Z = 1$

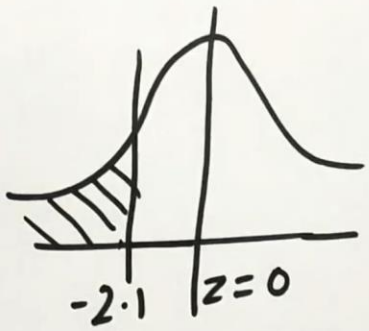


$$P(x > 1100)$$
$$= P(z > 1)$$
$$= 0.5 - P(z = 1)$$
$$= 0.5 - 0.3413$$
$$= 0.1587 //$$

iii) Less than ₹ 790.

$(P(z=2.1) = 0.4821)$

$$z = \frac{790 - 1000}{100} = -2.1$$



$$P(x < 790)$$
$$= P(z < -2.1)$$
$$= 0.5 - P(z = -2.1)$$
$$= 0.5 - 0.4821$$
$$= 0.0179$$

2) 2000 students appeared for exam. Distribution of marks is assumed to be normal with mean 30, std. deviation 6.25. How many students will get marks i) bet^n 20 and 40. $(P(z=1.6) = 0.4452)$

$\Rightarrow$

2] 2000 students appeared for exam. Distribution of marks is assumed to be normal with mean 30, std. deviation 6.25. How many students will get marks i] betⁿ 20 and 40. $(P(z=1.6) = 0.4452)$

$\Rightarrow$ $\mu = 30$, $\sigma = 6.25$,

$x_1 = 20$ $\qquad$ $x_2 = 40$

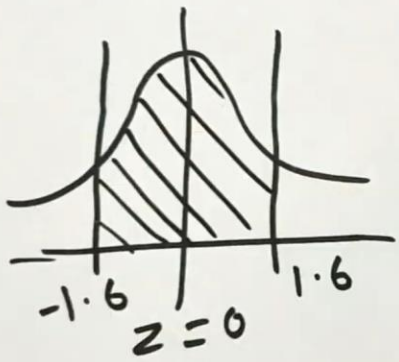
Distribution of marks is assumed to be normal with mean 30, std. deviation 6.25. How many students will get marks i] betⁿ 20 and 40. $(P(z=1.6) = 0.4452)$

$\Rightarrow$ $\mu = 30$, $\sigma = 6.25$,

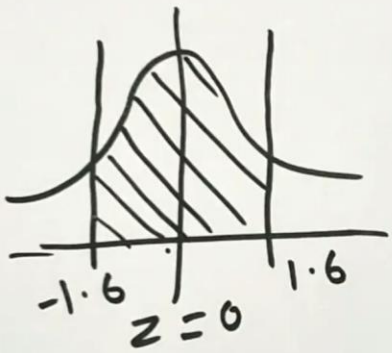
$x_1 = 20$ $\qquad$ $x_2 = 40$

$$Z_1 = \frac{20-30}{6.25} = -1.6 \qquad Z_2 = \frac{40-30}{6.25} = 1.6$$

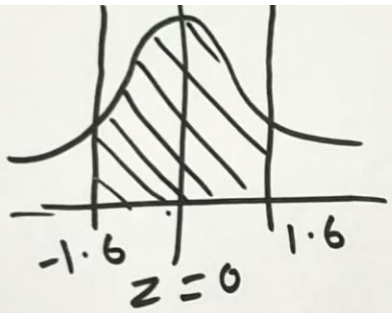
$Z_1 = -1.6$, $Z_2 = 1.6$



$$P(20 < X < 40)$$
$$= P(-1.6 < Z < 1.6)$$



$$P(20 < X < 40)$$
$$= P(-1.6 < Z < 1.6)$$
$$= P(z = -1.6) + P(z = 1.6)$$
$$= 0.4452 + 0.4452$$
$$= 0.8904 //$$



$= P(-1.6 < Z < 1.6)$

$= P(z = -1.6) + P(z = 1.6)$

$= 0.4452 + 0.4452$

$= 0.8904$ //

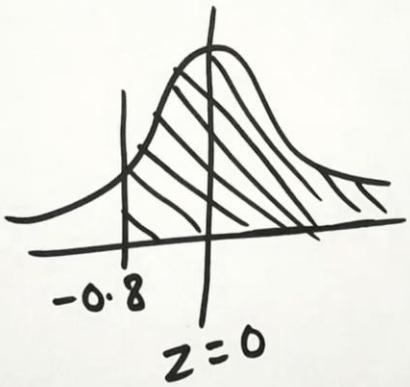
No. of students bet^n 20 to 40

$= 2000 \times 0.8904 = 1780.8$

≈ 1781

ii) More than 25.

$x = 25, \quad z = \frac{25 - 30}{6.25} = -0.8$



$P(x > 25)$

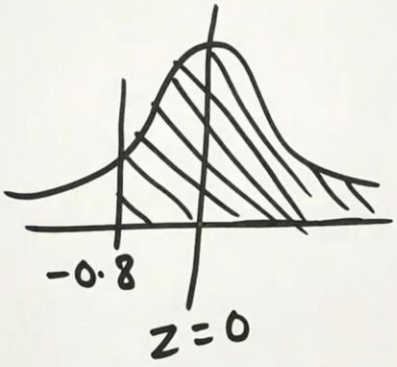
$= P(z > -0.8)$

$= 0.5 + P(z = -0.8)$

$= 0.5 + 0.2881$

$= 0.7881$

6.25



$P(x > 25)$

$= P(z > -0.8)$

$= 0.5 + P(z = -0.8)$

$= 0.5 + 0.2881$

$= 0.7881$

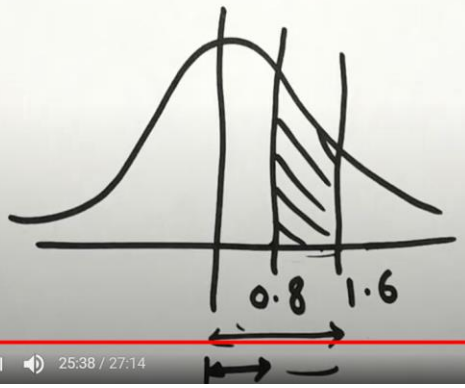
No. of students more than 25

$= 2000 \times 0.7881 = 1576.2$

≈ 1576 //

iii) Betⁿ 35 & 40

$Z_1 = \frac{35-30}{6.25} = 0.8$, $Z_2 = \frac{40-30}{6.25} = 1.6$



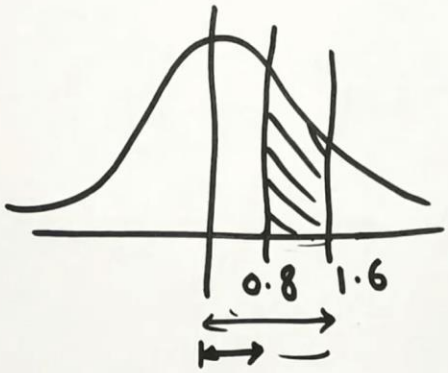
$P(35 < x < 40)$

$= P(z = 1.6) - P(z = 0.8)$

$= 0.4452 - 0.2881$

$= 0.157$

$Z_1 = \frac{35-30}{6.25} = 0.8$, $Z_2 = \frac{40-30}{6.25} = 1.6$



$P(35 < x < 40)$

$= P(z = 1.6) - P(z = 0.8)$

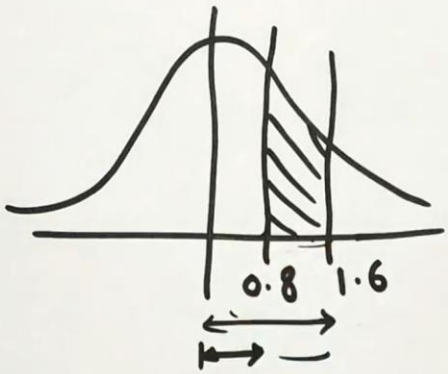
$= 0.4452 - 0.2881$

$= 0.1571$

No. of students bet^n 35 to 40

$= 2000 \times 0.1571 = 314.2$

$Z_1 = \frac{35-30}{6.25} = 0.8$, $Z_2 = \frac{40-30}{6.25} = 1.6$



$P(35 < x < 40)$

$= P(z = 1.6) - P(z = 0.8)$

$= 0.4452 - 0.2881$

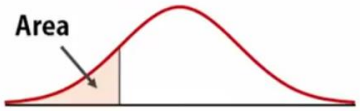
$= 0.1571$

No. of students bet^n 35 to 40

$= 2000 \times 0.1571 = 314.2 \approx 314$ //

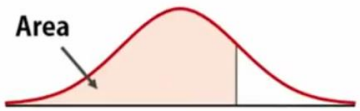
Normal Distributions
Calculating Probabilities

Negative *z*



z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
-2.5	0.0062	0.0060	0.0059	0.0057	0.0055	0.0054	0.0052	0.0051	0.0049	0.0048
-2.4	0.0082	0.0080	0.0078	0.0075	0.0073	0.0071	0.0069	0.0068	0.0066	0.0064
-2.3	0.0107	0.0104	0.0102	0.0099	0.0096	0.0094	0.0091	0.0089	0.0087	0.0084
-2.2	0.0139	0.0136	0.0132	0.0129	0.0125	0.0122	0.0119	0.0116	0.0113	0.0110
-2.1	0.0179	0.0174	0.0170	0.0166	0.0162	0.0158	0.0154	0.0150	0.0146	0.0143
-2.0	0.0228	0.0222	0.0217	0.0212	0.0207	0.0202	0.0197	0.0192	0.0188	0.0183
-1.9	0.0287	0.0281	0.0274	0.0268	0.0262	0.0256	0.0250	0.0244	0.0239	0.0233
-1.8	0.0359	0.0351	0.0344	0.0336	0.0329	0.0322	0.0314	0.0307	0.0301	0.0294

Positive *z*



z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	0.5000	0.5040	0.5080	0.5120	0.5160	0.5199	0.5239	0.5279	0.5319	0.5359
0.1	0.5398	0.5438	0.5478	0.5517	0.5557	0.5596	0.5636	0.5675	0.5714	0.5753
0.2	0.5793	0.5832	0.5871	0.5910	0.5948	0.5987	0.6026	0.6064	0.6103	0.6141
0.3	0.6179	0.6217	0.6255	0.6293	0.6331	0.6368	0.6406	0.6443	0.6480	0.6517
0.4	0.6554	0.6591	0.6628	0.6664	0.6700	0.6736	0.6772	0.6808	0.6844	0.6879
0.5	0.6915	0.6950	0.6985	0.7019	0.7054	0.7088	0.7123	0.7157	0.7190	0.7224
0.6	0.7257	0.7291	0.7324	0.7357	0.7389	0.7422	0.7454	0.7486	0.7517	0.7549
0.7	0.7580	0.7611	0.7642	0.7673	0.7704	0.7734	0.7764	0.7794	0.7823	0.7852
0.8	0.7881	0.7910	0.7939	0.7967	0.7995	0.8023	0.8051	0.8078	0.8106	0.8133
0.9	0.8159	0.8186	0.8212	0.8238	0.8264	0.8289	0.8315	0.8340	0.8365	0.8389

Normal Distributions
Calculating Probabilities

Negative *z*

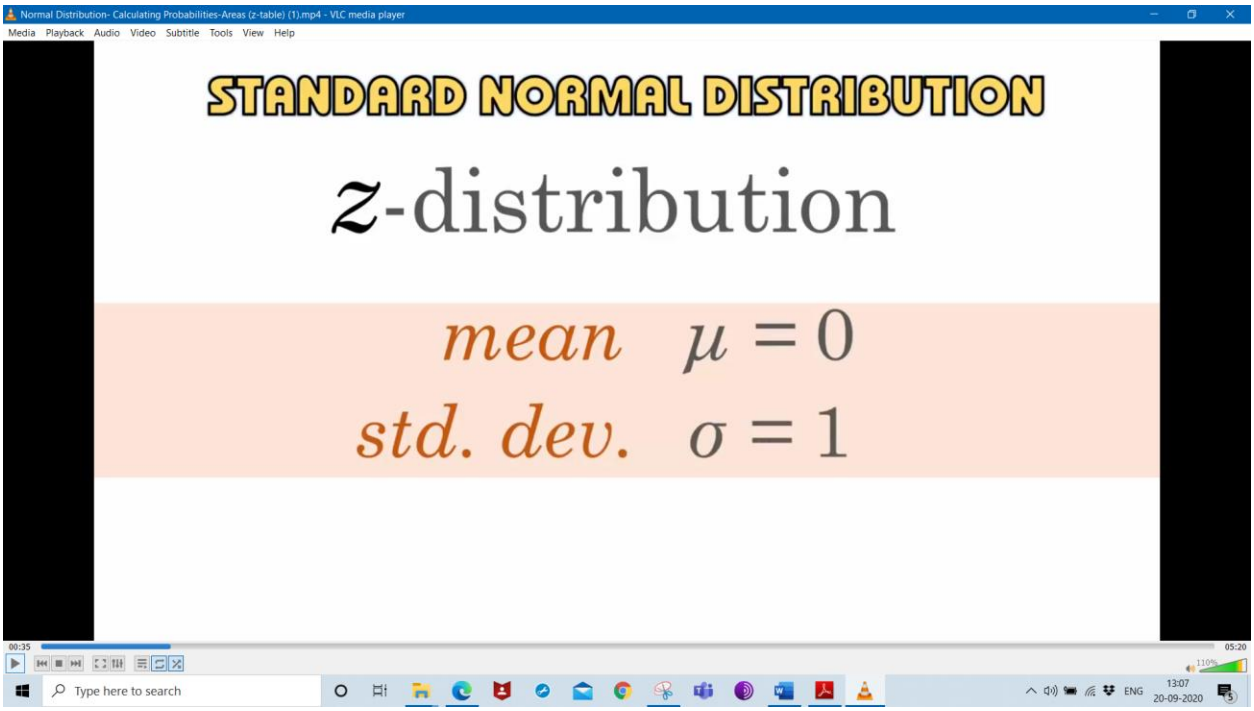
Area

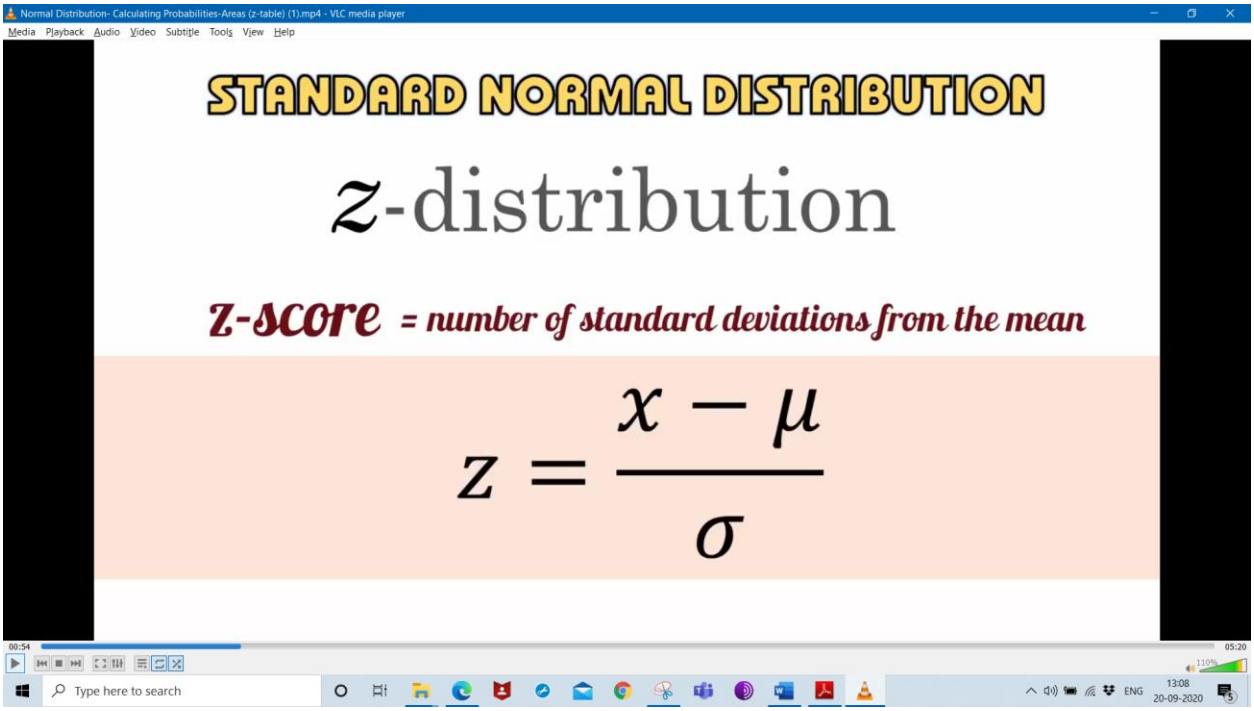
Positive *z*

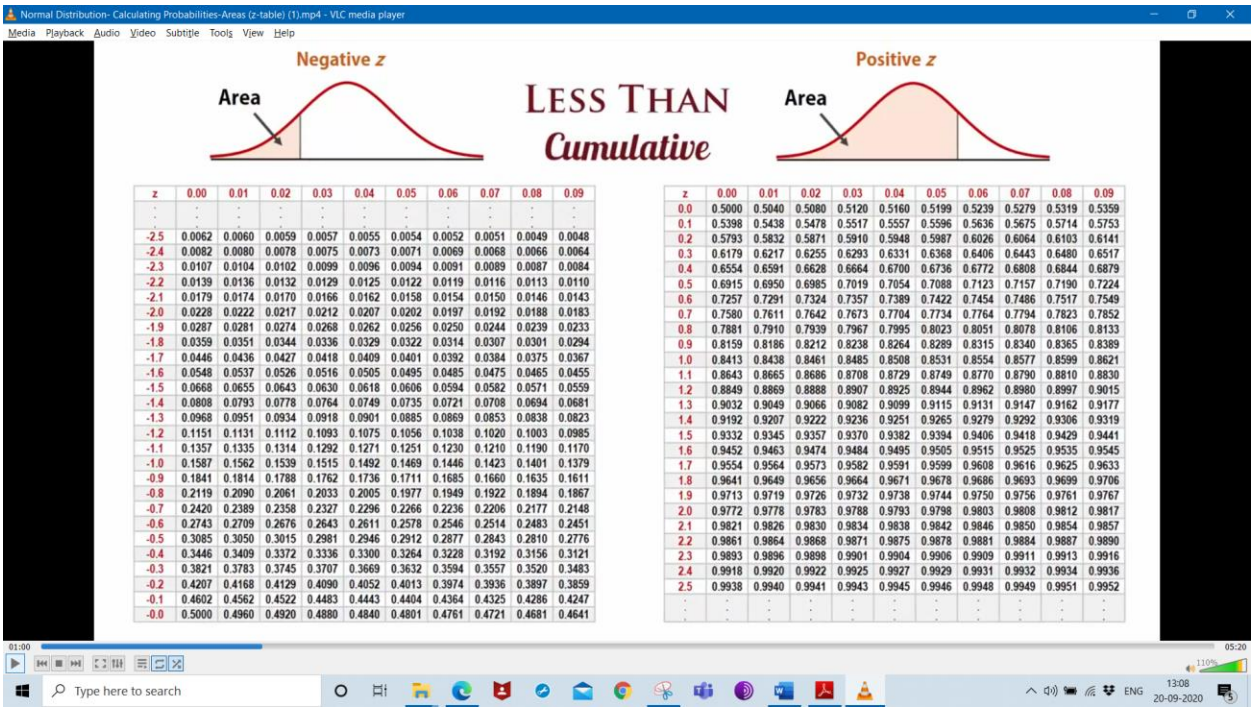
Area

z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
-2.5	0.0062	0.0060	0.0059	0.0057	0.0055	0.0054	0.0052	0.0051	0.0049	0.0048
-2.4	0.0082	0.0080	0.0078	0.0075	0.0073	0.0071	0.0069	0.0068	0.0066	0.0064
-2.3	0.0107	0.0104	0.0102	0.0099	0.0096	0.0094	0.0091	0.0089	0.0087	0.0084
-2.2	0.0139	0.0136	0.0132	0.0129	0.0125	0.0122	0.0119	0.0116	0.0113	0.0110
-2.1	0.0179	0.0174	0.0170	0.0166	0.0162	0.0158	0.0154	0.0150	0.0146	0.0143
-2.0	0.0228	0.0222	0.0217	0.0212	0.0207	0.0202	0.0197	0.0192	0.0188	0.0183
-1.9	0.0287	0.0281	0.0274	0.0268	0.0262	0.0256	0.0250	0.0244	0.0239	0.0233
-1.8	0.0359	0.0351	0.0344	0.0336	0.0329	0.0322	0.0314	0.0307	0.0301	0.0294

z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	0.5000	0.5040	0.5080	0.5120	0.5160	0.5199	0.5239	0.5279	0.5319	0.5359
0.1	0.5398	0.5438	0.5478	0.5517	0.5557	0.5596	0.5636	0.5675	0.5714	0.5753
0.2	0.5793	0.5832	0.5871	0.5910	0.5948	0.5987	0.6026	0.6064	0.6103	0.6141
0.3	0.6179	0.6217	0.6255	0.6293	0.6331	0.6368	0.6406	0.6443	0.6480	0.6517
0.4	0.6554	0.6591	0.6628	0.6664	0.6700	0.6736	0.6772	0.6808	0.6844	0.6879
0.5	0.6915	0.6950	0.6985	0.7019	0.7054	0.7088	0.7123	0.7157	0.7190	0.7224
0.6	0.7257	0.7291	0.7324	0.7357	0.7389	0.7422	0.7454	0.7486	0.7517	0.7549
0.7	0.7580	0.7611	0.7642	0.7673	0.7704	0.7734	0.7764	0.7794	0.7823	0.7852
0.8	0.7881	0.7910	0.7939	0.7967	0.7995	0.8023	0.8051	0.8078	0.8106	0.8133
0.9	0.8159	0.8186	0.8212	0.8238	0.8264	0.8289	0.8315	0.8340	0.8365	0.8389

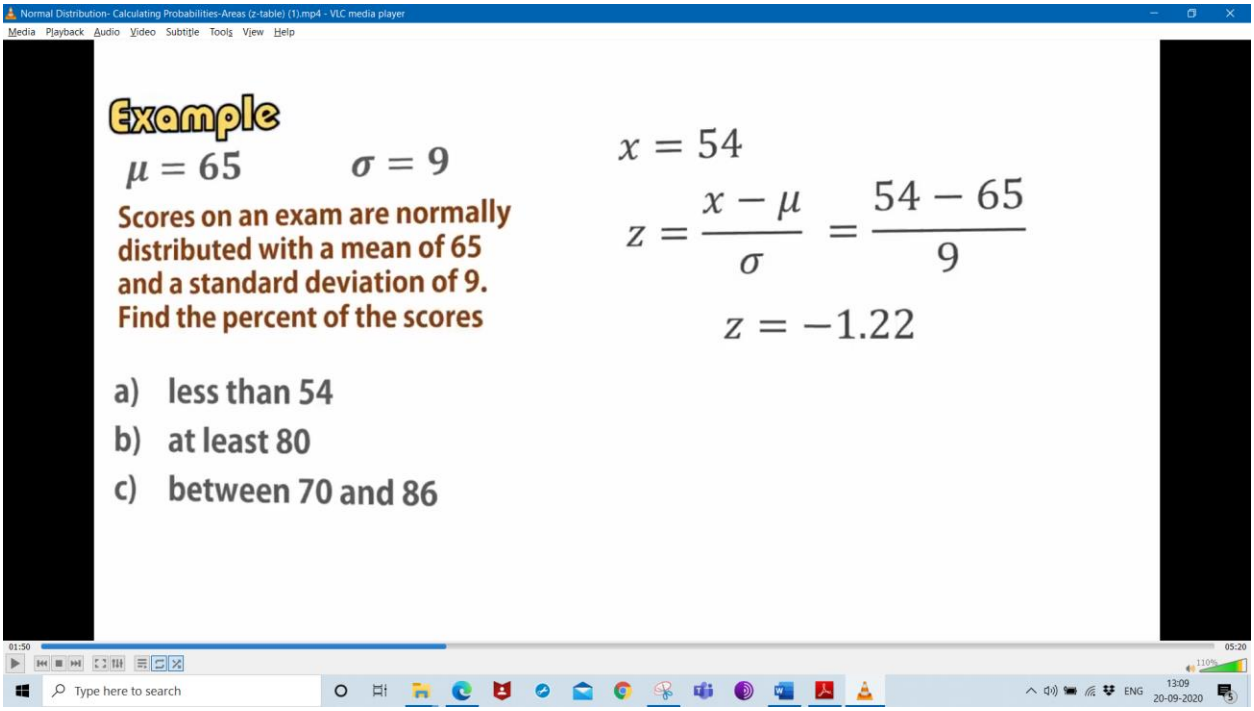

Normal Distribution- Calculating Probabilities-Areas (z-table) (1).mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
STANDARD NORMAL DISTRIBUTION
z-distribution
mean $\mu = 0$
std. dev. $\sigma = 1$
Type here to search
ENG
13:07
20-09-2020

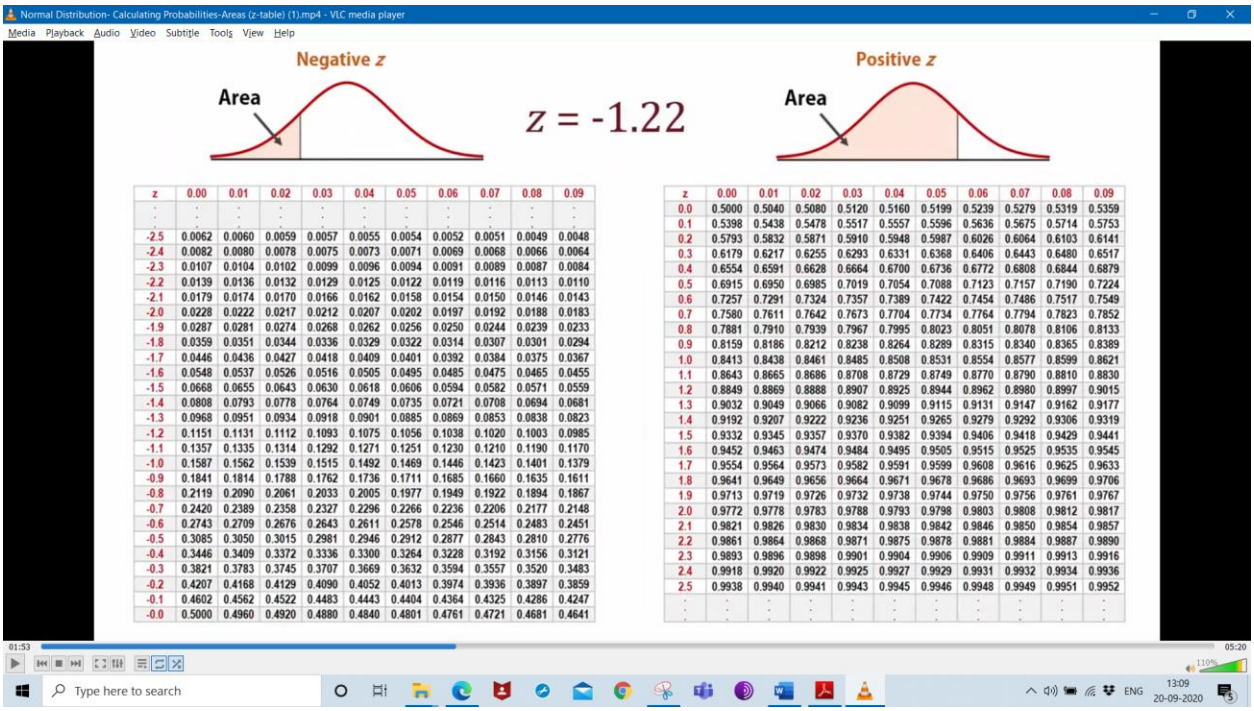

Normal Distribution- Calculating Probabilities-Areas (z-table) (1).mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
STANDARD NORMAL DISTRIBUTION
z-distribution
z-score = number of standard deviations from the mean
$z = \frac{x - \mu}{\sigma}$
Type here to search
ENG
13:08
20-09-2020



z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
-2.5	0.0062	0.0060	0.0059	0.0057	0.0055	0.0054	0.0052	0.0051	0.0049	0.0048
-2.4	0.0082	0.0080	0.0078	0.0075	0.0073	0.0071	0.0069	0.0068	0.0066	0.0064
-2.3	0.0107	0.0104	0.0102	0.0099	0.0096	0.0094	0.0091	0.0089	0.0087	0.0084
-2.2	0.0139	0.0136	0.0132	0.0129	0.0125	0.0122	0.0119	0.0116	0.0113	0.0110
-2.1	0.0179	0.0174	0.0170	0.0166	0.0162	0.0158	0.0154	0.0150	0.0146	0.0143
-2.0	0.0228	0.0222	0.0217	0.0212	0.0207	0.0202	0.0197	0.0192	0.0188	0.0183
-1.9	0.0287	0.0281	0.0274	0.0268	0.0262	0.0256	0.0250	0.0244	0.0239	0.0233
-1.8	0.0359	0.0351	0.0344	0.0336	0.0329	0.0322	0.0314	0.0307	0.0301	0.0294
-1.7	0.0446	0.0436	0.0427	0.0418	0.0409	0.0401	0.0392	0.0384	0.0375	0.0367
-1.6	0.0548	0.0537	0.0526	0.0516	0.0505	0.0495	0.0485	0.0475	0.0465	0.0455
-1.5	0.0668	0.0655	0.0643	0.0630	0.0618	0.0606	0.0594	0.0582	0.0571	0.0559
-1.4	0.0808	0.0793	0.0778	0.0764	0.0749	0.0735	0.0721	0.0708	0.0694	0.0681
-1.3	0.0968	0.0951	0.0934	0.0918	0.0901	0.0885	0.0869	0.0853	0.0838	0.0823
-1.2	0.1151	0.1131	0.1112	0.1093	0.1075	0.1056	0.1038	0.1020	0.1003	0.0985
-1.1	0.1357	0.1335	0.1314	0.1292	0.1271	0.1251	0.1230	0.1210	0.1190	0.1170
-1.0	0.1587	0.1562	0.1539	0.1515	0.1492	0.1469	0.1446	0.1423	0.1401	0.1379
-0.9	0.1841	0.1814	0.1788	0.1762	0.1736	0.1711	0.1685	0.1660	0.1635	0.1611
-0.8	0.2119	0.2090	0.2061	0.2033	0.2005	0.1977	0.1949	0.1922	0.1894	0.1867
-0.7	0.2420	0.2389	0.2358	0.2327	0.2296	0.2266	0.2236	0.2206	0.2177	0.2148
-0.6	0.2743	0.2709	0.2676	0.2643	0.2611	0.2578	0.2546	0.2514	0.2483	0.2451
-0.5	0.3085	0.3050	0.3015	0.2981	0.2946	0.2912	0.2877	0.2843	0.2810	0.2776
-0.4	0.3446	0.3409	0.3372	0.3336	0.3300	0.3264	0.3228	0.3192	0.3156	0.3121
-0.3	0.3821	0.3783	0.3745	0.3707	0.3669	0.3632	0.3594	0.3557	0.3520	0.3483
-0.2	0.4207	0.4168	0.4129	0.4090	0.4052	0.4013	0.3974	0.3936	0.3897	0.3859
-0.1	0.4602	0.4562	0.4522	0.4483	0.4443	0.4404	0.4364	0.4325	0.4286	0.4247
-0.0	0.5000	0.4960	0.4920	0.4880	0.4840	0.4801	0.4761	0.4721	0.4681	0.4641

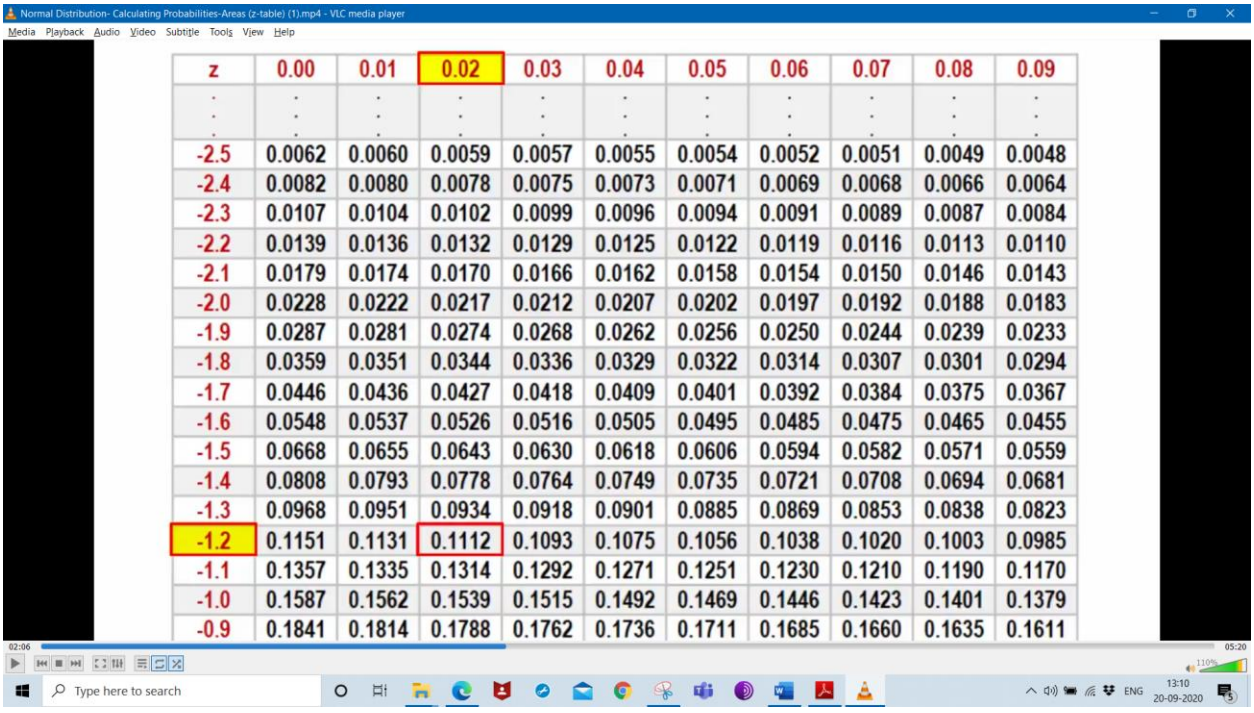
z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	0.5000	0.5040	0.5080	0.5120	0.5160	0.5199	0.5239	0.5279	0.5319	0.5359
0.1	0.5398	0.5438	0.5478	0.5517	0.5557	0.5596	0.5636	0.5675	0.5714	0.5753
0.2	0.5793	0.5832	0.5871	0.5910	0.5948	0.5987	0.6026	0.6064	0.6103	0.6141
0.3	0.6179	0.6217	0.6255	0.6293	0.6331	0.6368	0.6406	0.6443	0.6480	0.6517
0.4	0.6554	0.6591	0.6628	0.6664	0.6700	0.6736	0.6772	0.6808	0.6844	0.6879
0.5	0.6915	0.6950	0.6985	0.7019	0.7054	0.7088	0.7123	0.7157	0.7190	0.7224
0.6	0.7257	0.7291	0.7324	0.7357	0.7389	0.7422	0.7454	0.7486	0.7517	0.7549
0.7	0.7580	0.7611	0.7642	0.7673	0.7704	0.7734	0.7764	0.7794	0.7823	0.7852
0.8	0.7881	0.7910	0.7939	0.7967	0.7995	0.8023	0.8051	0.8078	0.8106	0.8133
0.9	0.8159	0.8186	0.8212	0.8238	0.8264	0.8289	0.8315	0.8340	0.8365	0.8389
1.0	0.8413	0.8438	0.8461	0.8485	0.8508	0.8531	0.8554	0.8577	0.8599	0.8621
1.1	0.8643	0.8665	0.8686	0.8708	0.8729	0.8749	0.8770	0.8790	0.8810	0.8830
1.2	0.8849	0.8869	0.8888	0.8907	0.8925	0.8944	0.8962	0.8980	0.8997	0.9015
1.3	0.9032	0.9049	0.9066	0.9082	0.9099	0.9115	0.9131	0.9147	0.9162	0.9177
1.4	0.9192	0.9207	0.9222	0.9236	0.9251	0.9265	0.9279	0.9292	0.9306	0.9319
1.5	0.9332	0.9345	0.9357	0.9370	0.9382	0.9394	0.9406	0.9418	0.9429	0.9441
1.6	0.9452	0.9463	0.9474	0.9484	0.9495	0.9505	0.9515	0.9525	0.9535	0.9545
1.7	0.9554	0.9564	0.9573	0.9582	0.9591	0.9599	0.9608	0.9616	0.9625	0.9633
1.8	0.9641	0.9649	0.9656	0.9664	0.9671	0.9678	0.9686	0.9693	0.9699	0.9706
1.9	0.9713	0.9719	0.9726	0.9732	0.9738	0.9744	0.9750	0.9756	0.9761	0.9767
2.0	0.9772	0.9778	0.9783	0.9788	0.9793	0.9798	0.9803	0.9808	0.9812	0.9817
2.1	0.9821	0.9826	0.9830	0.9834	0.9838	0.9842	0.9846	0.9850	0.9854	0.9857
2.2	0.9861	0.9864	0.9868	0.9871	0.9875	0.9878	0.9881	0.9884	0.9887	0.9890
2.3	0.9893	0.9896	0.9898	0.9901	0.9904	0.9906	0.9909	0.9911	0.9913	0.9916
2.4	0.9918	0.9920	0.9922	0.9925	0.9927	0.9929	0.9931	0.9932	0.9934	0.9936
2.5	0.9938	0.9940	0.9941	0.9943	0.9945	0.9946	0.9948	0.9949	0.9951	0.9952
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.

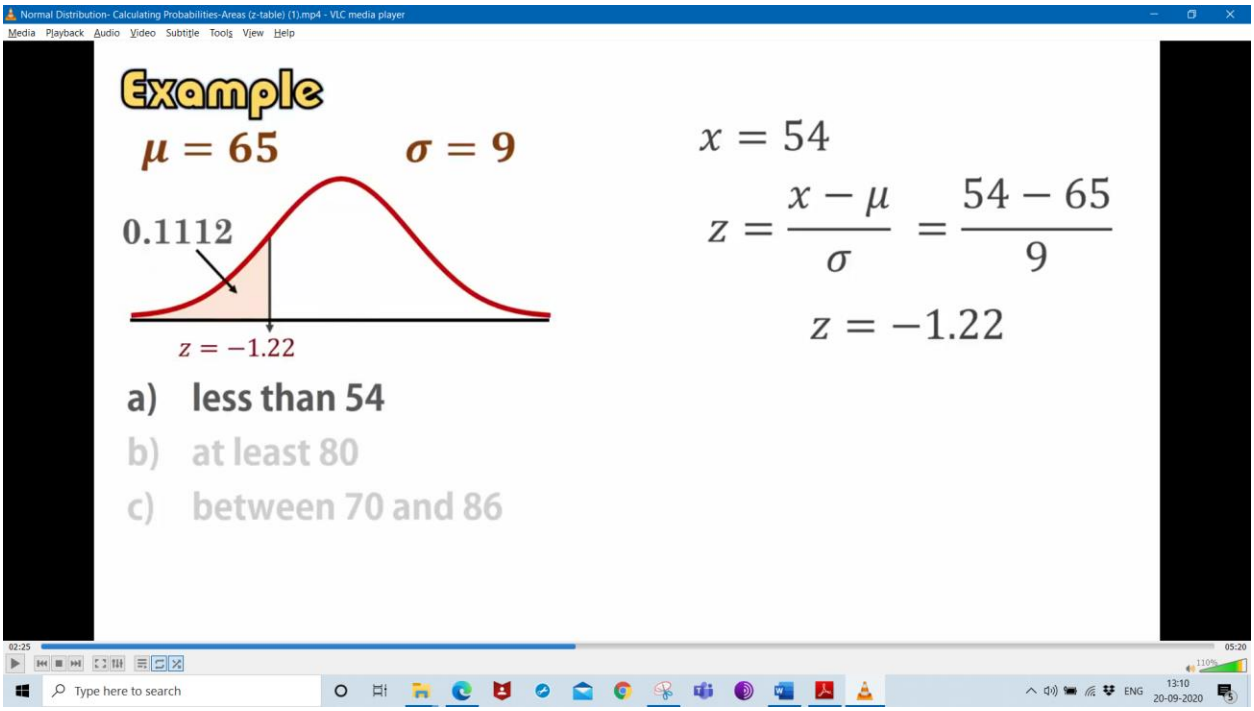


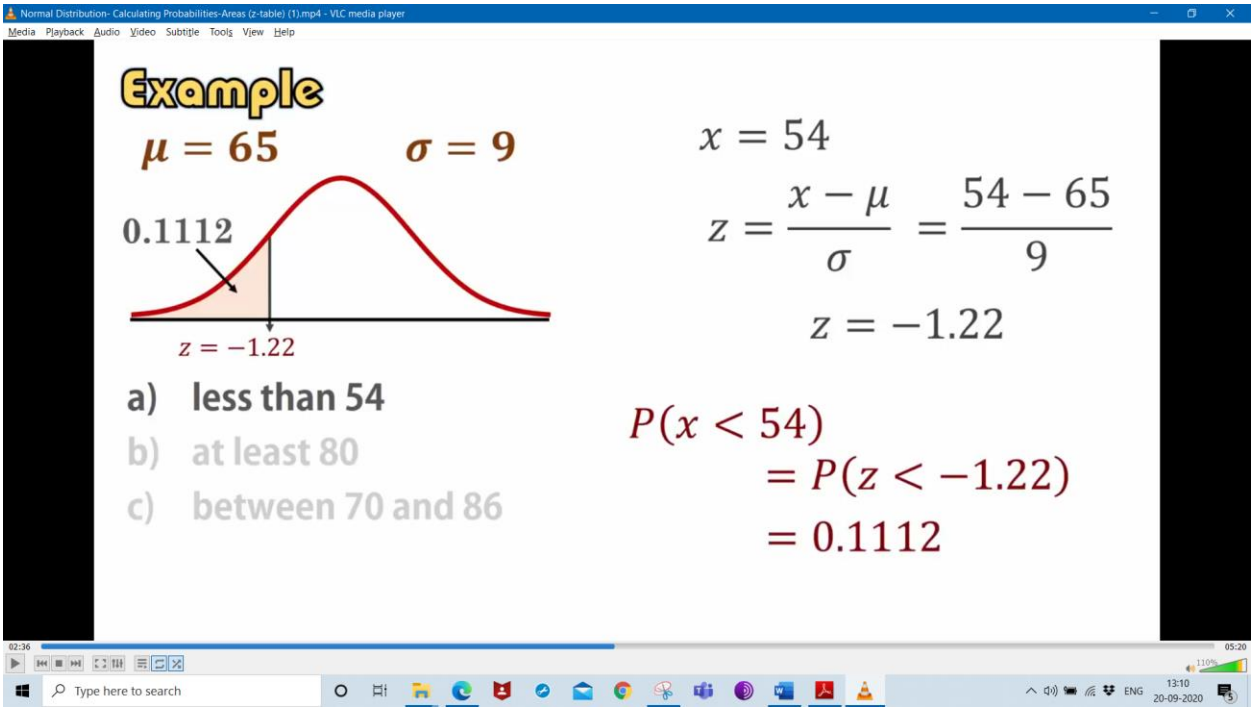


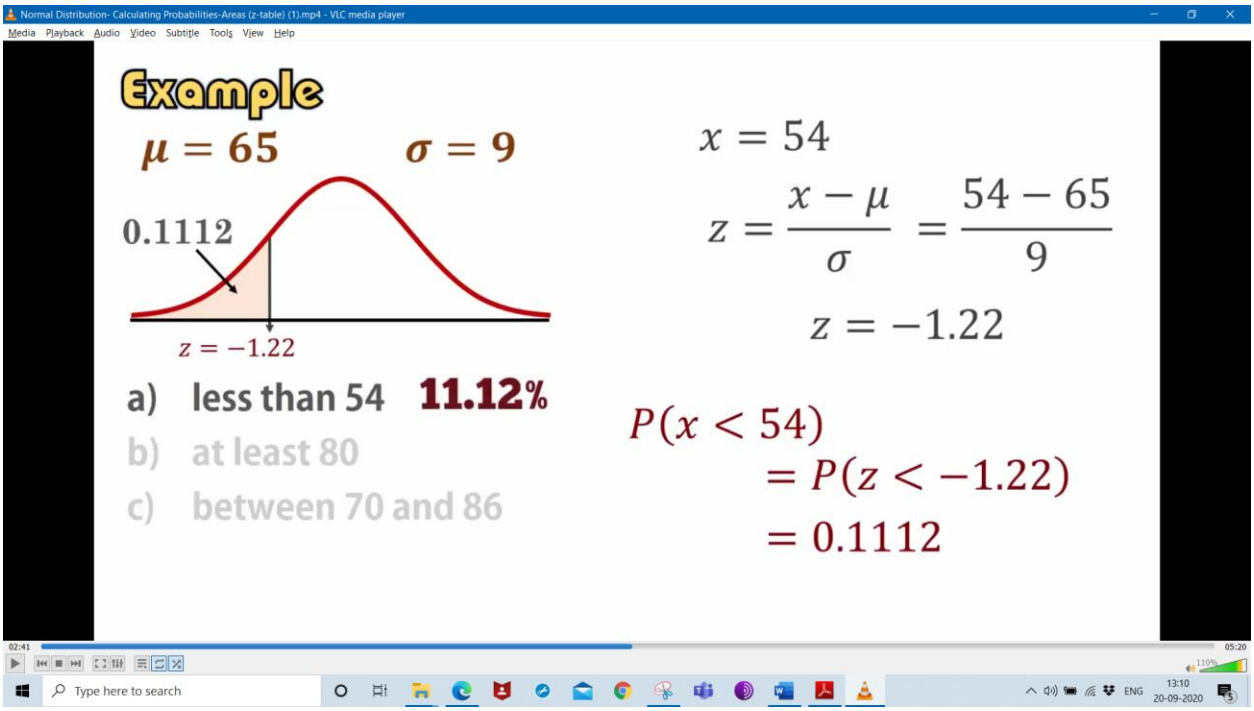
z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
-2.5	0.0062	0.0060	0.0059	0.0057	0.0055	0.0054	0.0052	0.0051	0.0049	0.0048
-2.4	0.0082	0.0080	0.0078	0.0075	0.0073	0.0071	0.0069	0.0068	0.0066	0.0064
-2.3	0.0107	0.0104	0.0102	0.0099	0.0096	0.0094	0.0091	0.0089	0.0087	0.0084
-2.2	0.0139	0.0136	0.0132	0.0129	0.0125	0.0122	0.0119	0.0116	0.0113	0.0110
-2.1	0.0179	0.0174	0.0170	0.0166	0.0162	0.0158	0.0154	0.0150	0.0146	0.0143
-2.0	0.0228	0.0222	0.0217	0.0212	0.0207	0.0202	0.0197	0.0192	0.0188	0.0183
-1.9	0.0287	0.0281	0.0274	0.0268	0.0262	0.0256	0.0250	0.0244	0.0239	0.0233
-1.8	0.0359	0.0351	0.0344	0.0336	0.0329	0.0322	0.0314	0.0307	0.0301	0.0294
-1.7	0.0446	0.0436	0.0427	0.0418	0.0409	0.0401	0.0392	0.0384	0.0375	0.0367
-1.6	0.0548	0.0537	0.0526	0.0516	0.0505	0.0495	0.0485	0.0475	0.0465	0.0455
-1.5	0.0668	0.0655	0.0643	0.0630	0.0618	0.0606	0.0594	0.0582	0.0571	0.0559
-1.4	0.0808	0.0793	0.0778	0.0764	0.0749	0.0735	0.0721	0.0708	0.0694	0.0681
-1.3	0.0968	0.0951	0.0934	0.0918	0.0901	0.0885	0.0869	0.0853	0.0838	0.0823
-1.2	0.1151	0.1131	0.1112	0.1093	0.1075	0.1056	0.1038	0.1020	0.1003	0.0985
-1.1	0.1357	0.1335	0.1314	0.1292	0.1271	0.1251	0.1230	0.1210	0.1190	0.1170
-1.0	0.1587	0.1562	0.1539	0.1515	0.1492	0.1469	0.1446	0.1423	0.1401	0.1379
-0.9	0.1841	0.1814	0.1788	0.1762	0.1736	0.1711	0.1685	0.1660	0.1635	0.1611
-0.8	0.2119	0.2090	0.2061	0.2033	0.2005	0.1977	0.1949	0.1922	0.1894	0.1867
-0.7	0.2420	0.2389	0.2358	0.2327	0.2296	0.2266	0.2236	0.2206	0.2177	0.2148
-0.6	0.2743	0.2709	0.2676	0.2643	0.2611	0.2578	0.2546	0.2514	0.2483	0.2451
-0.5	0.3085	0.3050	0.3015	0.2981	0.2946	0.2912	0.2877	0.2843	0.2810	0.2776
-0.4	0.3446	0.3409	0.3372	0.3336	0.3300	0.3264	0.3228	0.3192	0.3156	0.3121
-0.3	0.3821	0.3783	0.3745	0.3707	0.3669	0.3632	0.3594	0.3557	0.3520	0.3483
-0.2	0.4207	0.4168	0.4129	0.4090	0.4052	0.4013	0.3974	0.3936	0.3897	0.3859
-0.1	0.4602	0.4562	0.4522	0.4483	0.4443	0.4404	0.4364	0.4325	0.4286	0.4247
-0.0	0.5000	0.4960	0.4920	0.4880	0.4840	0.4801	0.4761	0.4721	0.4681	0.4641

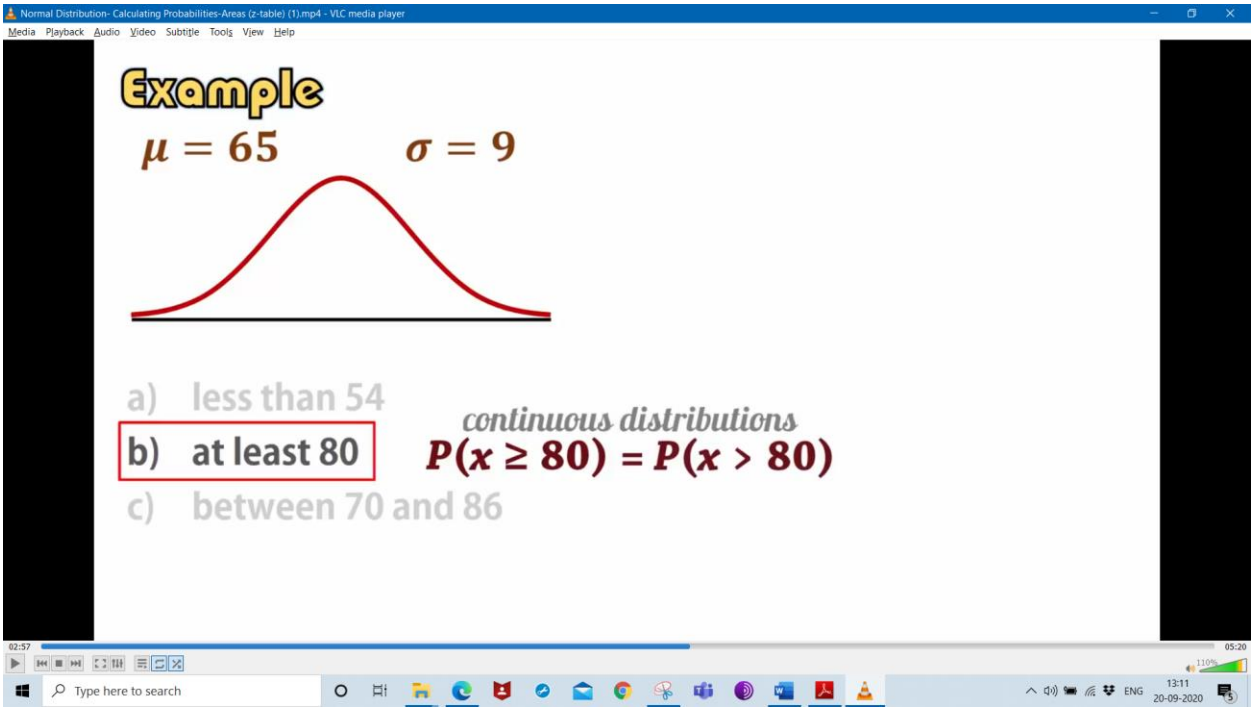
z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	0.5000	0.5040	0.5080	0.5120	0.5160	0.5199	0.5239	0.5279	0.5319	0.5359
0.1	0.5398	0.5438	0.5478	0.5517	0.5557	0.5596	0.5636	0.5675	0.5714	0.5753
0.2	0.5793	0.5832	0.5871	0.5910	0.5948	0.5987	0.6026	0.6064	0.6103	0.6141
0.3	0.6179	0.6217	0.6255	0.6293	0.6331	0.6368	0.6406	0.6443	0.6480	0.6517
0.4	0.6554	0.6591	0.6628	0.6664	0.6700	0.6736	0.6772	0.6808	0.6844	0.6879
0.5	0.6915	0.6950	0.6985	0.7019	0.7054	0.7088	0.7123	0.7157	0.7190	0.7224
0.6	0.7257	0.7291	0.7324	0.7357	0.7389	0.7422	0.7454	0.7486	0.7517	0.7549
0.7	0.7580	0.7611	0.7642	0.7673	0.7704	0.7734	0.7764	0.7794	0.7823	0.7852
0.8	0.7881	0.7910	0.7939	0.7967	0.7995	0.8023	0.8051	0.8078	0.8106	0.8133
0.9	0.8159	0.8186	0.8212	0.8238	0.8264	0.8289	0.8315	0.8340	0.8365	0.8389
1.0	0.8413	0.8438	0.8461	0.8485	0.8508	0.8531	0.8554	0.8577	0.8599	0.8621
1.1	0.8643	0.8665	0.8686	0.8708	0.8729	0.8749	0.8770	0.8790	0.8810	0.8830
1.2	0.8849	0.8869	0.8888	0.8907	0.8925	0.8944	0.8962	0.8980	0.8997	0.9015
1.3	0.9032	0.9049	0.9066	0.9082	0.9099	0.9115	0.9131	0.9147	0.9162	0.9177
1.4	0.9192	0.9207	0.9222	0.9236	0.9251	0.9265	0.9279	0.9292	0.9306	0.9319
1.5	0.9332	0.9345	0.9357	0.9370	0.9382	0.9394	0.9406	0.9418	0.9429	0.9441
1.6	0.9452	0.9463	0.9474	0.9484	0.9495	0.9505	0.9515	0.9525	0.9535	0.9545
1.7	0.9554	0.9564	0.9573	0.9582	0.9591	0.9599	0.9608	0.9616	0.9625	0.9633
1.8	0.9641	0.9649	0.9656	0.9664	0.9671	0.9678	0.9686	0.9693	0.9699	0.9706
1.9	0.9713	0.9719	0.9726	0.9732	0.9738	0.9744	0.9750	0.9756	0.9761	0.9767
2.0	0.9772	0.9778	0.9783	0.9788	0.9793	0.9798	0.9803	0.9808	0.9812	0.9817
2.1	0.9821	0.9826	0.9830	0.9834	0.9838	0.9842	0.9846	0.9850	0.9854	0.9857
2.2	0.9861	0.9864	0.9868	0.9871	0.9875	0.9878	0.9881	0.9884	0.9887	0.9890
2.3	0.9893	0.9896	0.9898	0.9901	0.9904	0.9906	0.9909	0.9911	0.9913	0.9916
2.4	0.9918	0.9920	0.9922	0.9925	0.9927	0.9929	0.9931	0.9932	0.9934	0.9936
2.5	0.9938	0.9940	0.9941	0.9943	0.9945	0.9946	0.9948	0.9949	0.9951	0.9952
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.

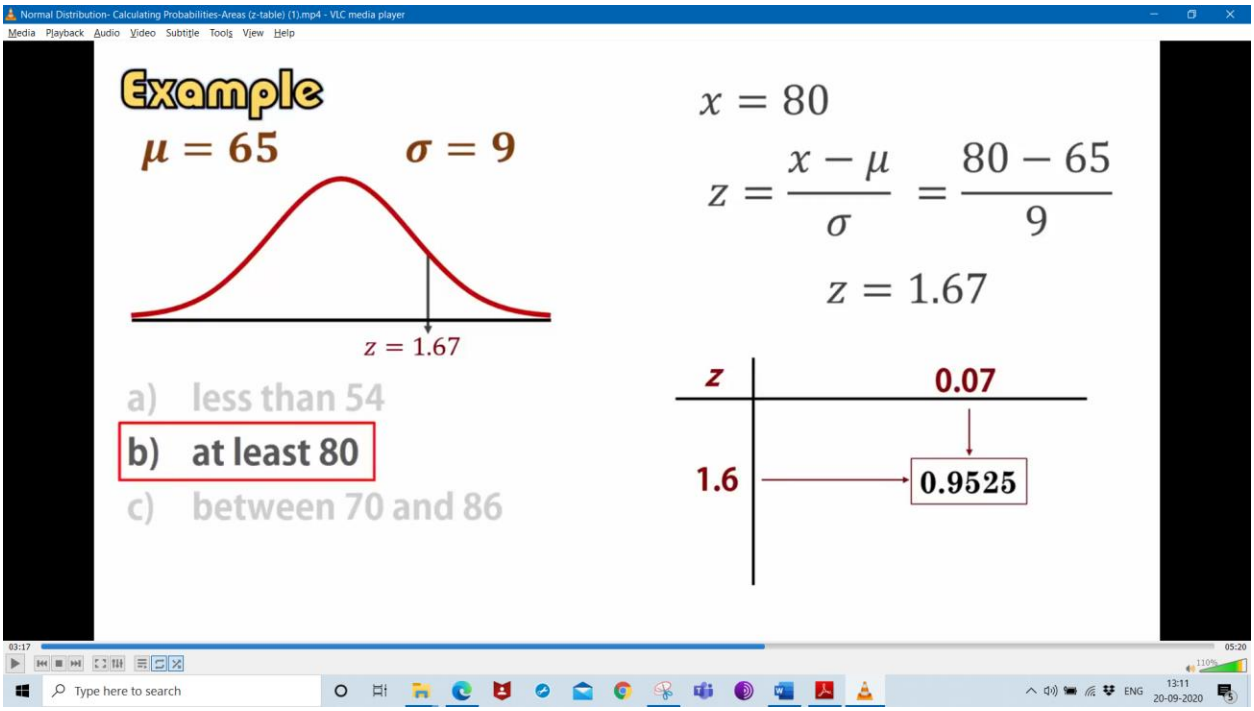


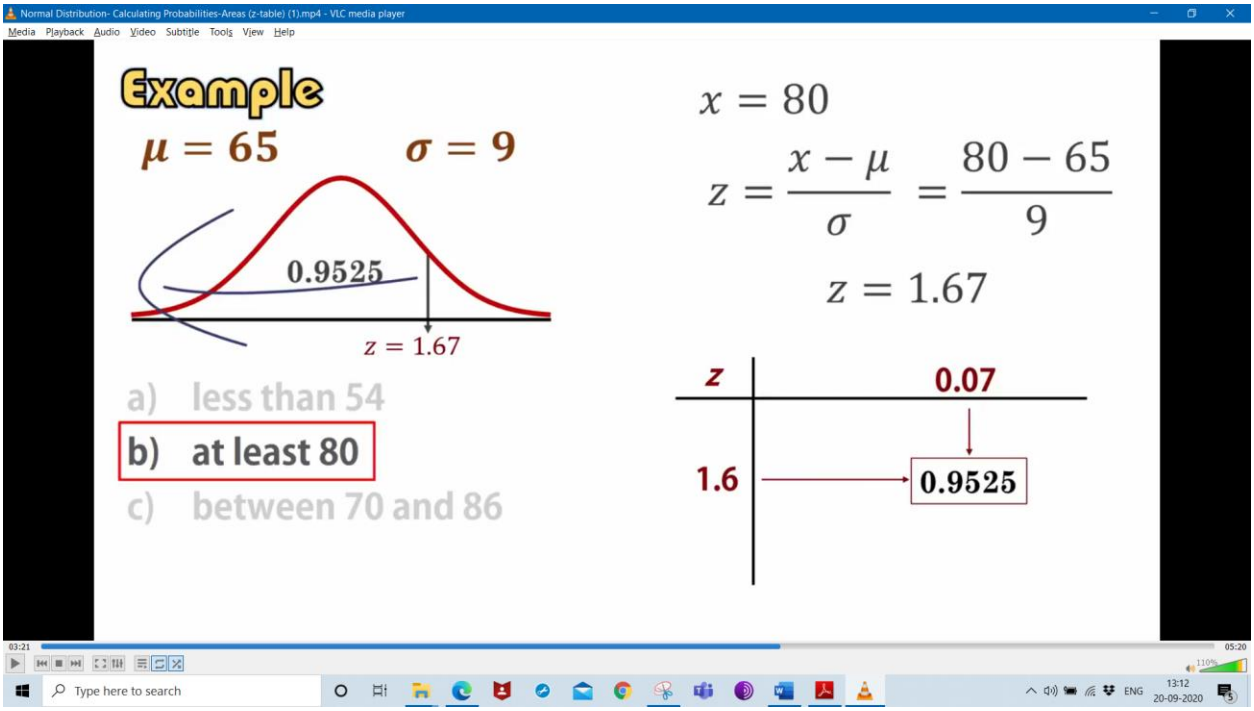

Normal Distribution- Calculating Probabilities-Areas (z-table) (1).mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Example
$\mu = 65$ $\sigma = 9$
0.1112
$z = -1.22$
a) less than 54
b) at least 80
c) between 70 and 86
$x = 54$
$z = \frac{x - \mu}{\sigma} = \frac{54 - 65}{9}$
$z = -1.22$
02:25
05:20
Type here to search
ENG
13:10
20-09-2020

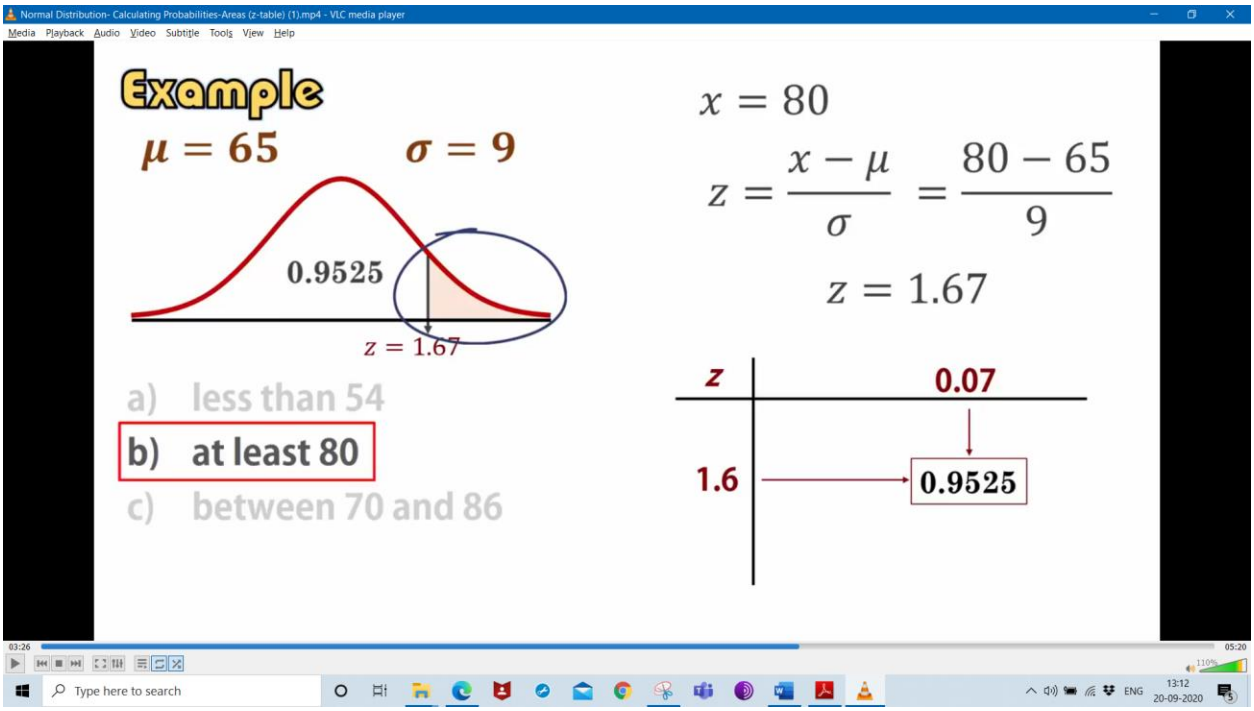

Normal Distribution- Calculating Probabilities-Areas (z-table) (1).mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Example
$\mu = 65$ $\sigma = 9$
0.1112
$z = -1.22$
a) less than 54
b) at least 80
c) between 70 and 86
$x = 54$
$z = \frac{x - \mu}{\sigma} = \frac{54 - 65}{9}$
$z = -1.22$
$P(x < 54)$
$= P(z < -1.22)$
$= 0.1112$
02:36
05:20
Type here to search
ENG
13:10
20-09-2020

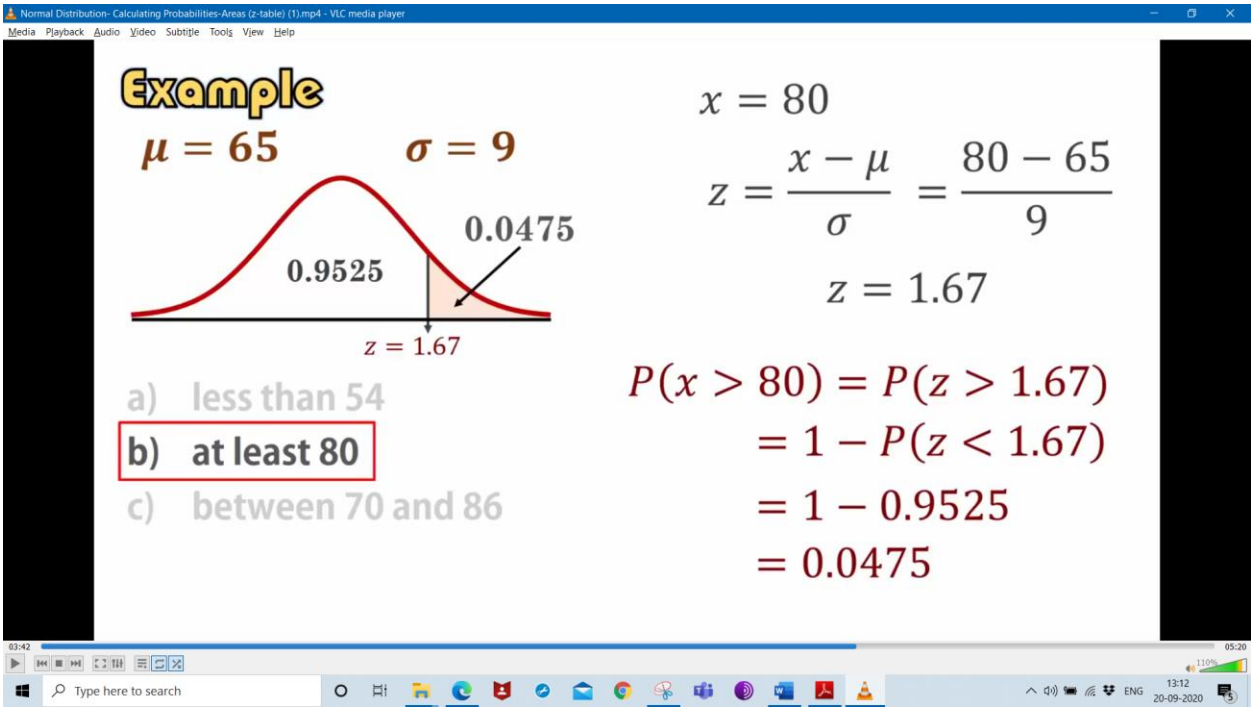

Normal Distribution- Calculating Probabilities-Areas (z-table) (1).mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Example
$\mu = 65$ $\sigma = 9$
0.1112
$z = -1.22$
a) less than 54 11.12%
b) at least 80
c) between 70 and 86
$x = 54$
$z = \frac{x - \mu}{\sigma} = \frac{54 - 65}{9}$
$z = -1.22$
$P(x < 54)$
$= P(z < -1.22)$
$= 0.1112$
Type here to search
13:10
20-09-2020

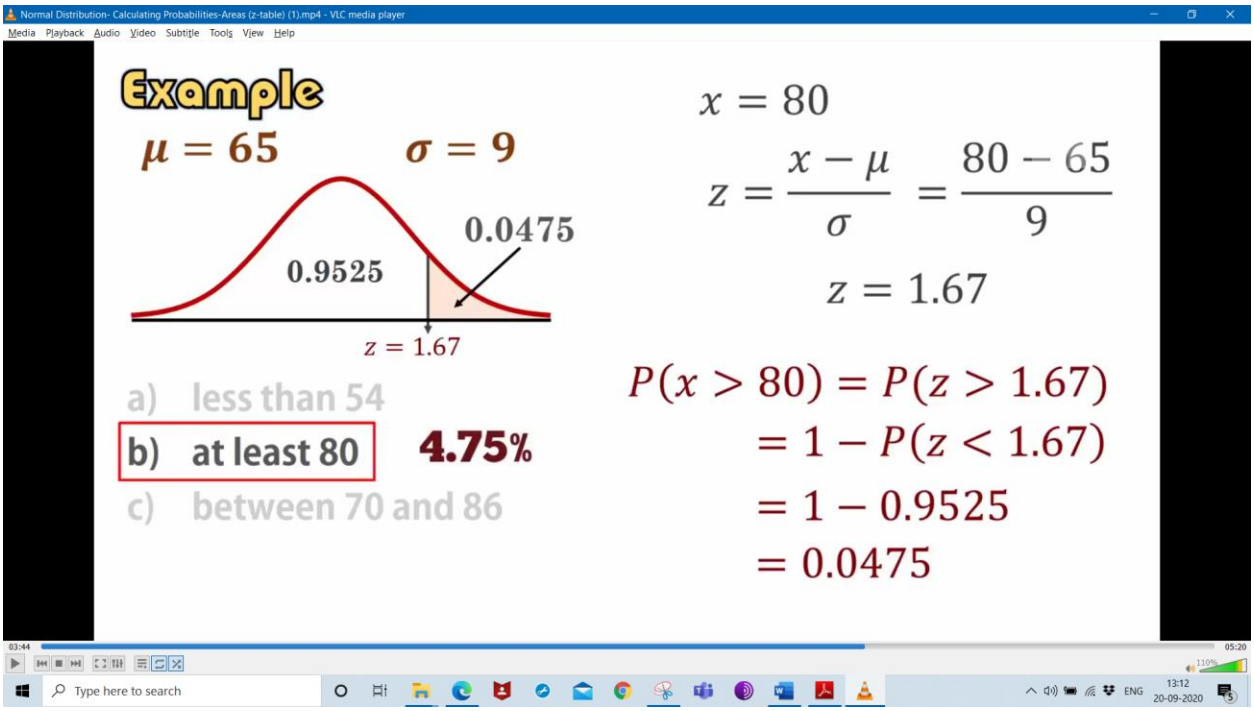

Normal Distribution- Calculating Probabilities-Areas (z-table) (1).mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Example
$\mu = 65$ $\sigma = 9$
a) less than 54
b) at least 80
c) between 70 and 86
continuous distributions
$P(x \geq 80) = P(x > 80)$
Type here to search
13:11
20-09-2020

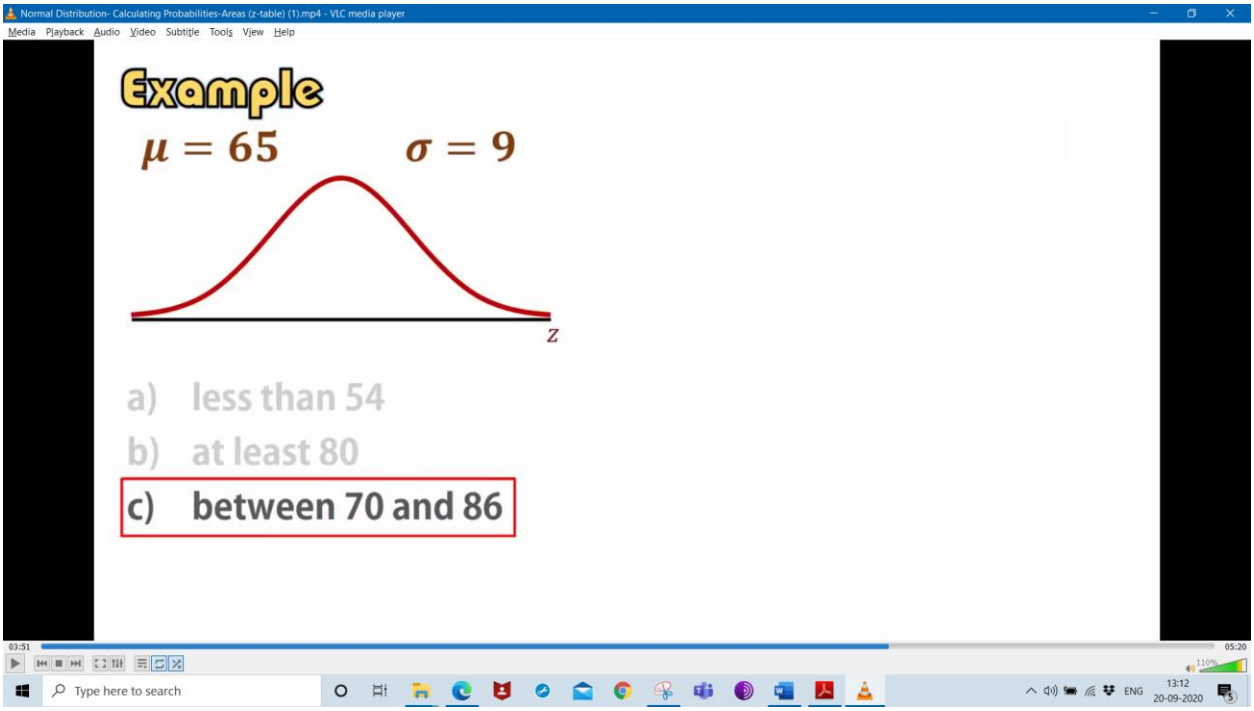

Normal Distribution- Calculating Probabilities-Areas (z-table) (1).mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Example
$\mu = 65$ $\sigma = 9$
$z = 1.67$
a) less than 54
b) at least 80
c) between 70 and 86
$x = 80$
$z = \frac{x - \mu}{\sigma} = \frac{80 - 65}{9}$
$z = 1.67$
z 0.07
1.6 0.9525
Type here to search
ENG 13:11 20-09-2020

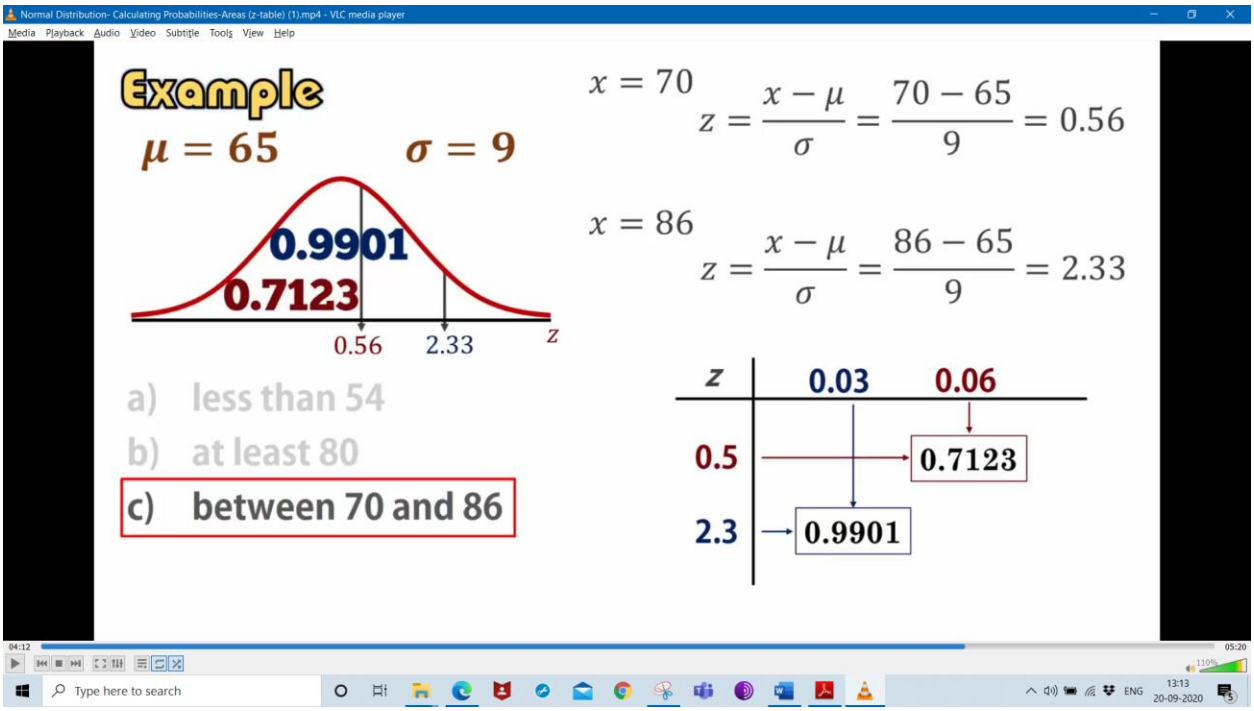

Normal Distribution- Calculating Probabilities-Areas (z-table) (1).mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Example
$\mu = 65$ $\sigma = 9$
0.9525
$z = 1.67$
a) less than 54
b) at least 80
c) between 70 and 86
$x = 80$
$z = \frac{x - \mu}{\sigma} = \frac{80 - 65}{9}$
$z = 1.67$
z 0.07
1.6 0.9525
Type here to search
ENG 13:12 20-09-2020

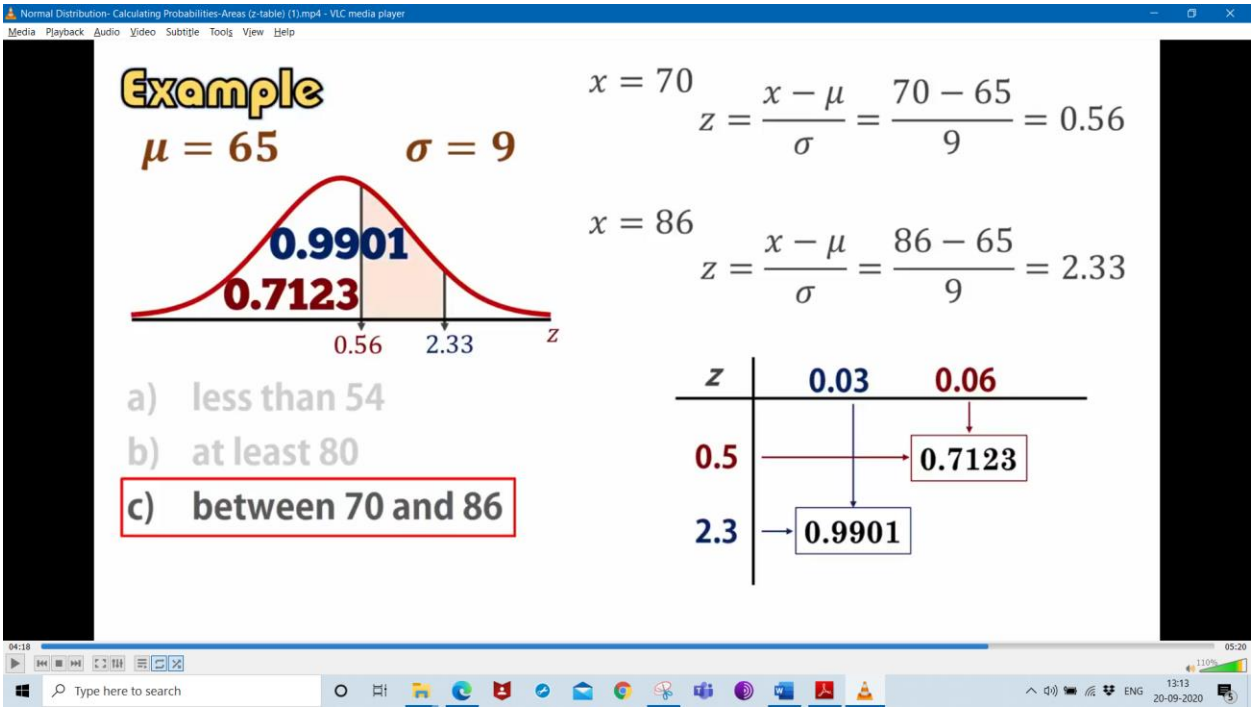

Normal Distribution- Calculating Probabilities-Areas (z-table) (1).mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Example
$\mu = 65$ $\sigma = 9$
0.9525
$z = 1.67$
a) less than 54
b) at least 80
c) between 70 and 86
$x = 80$
$z = \frac{x - \mu}{\sigma} = \frac{80 - 65}{9}$
$z = 1.67$
z 0.07
1.6 0.9525
Type here to search
ENG 13:12 20-09-2020

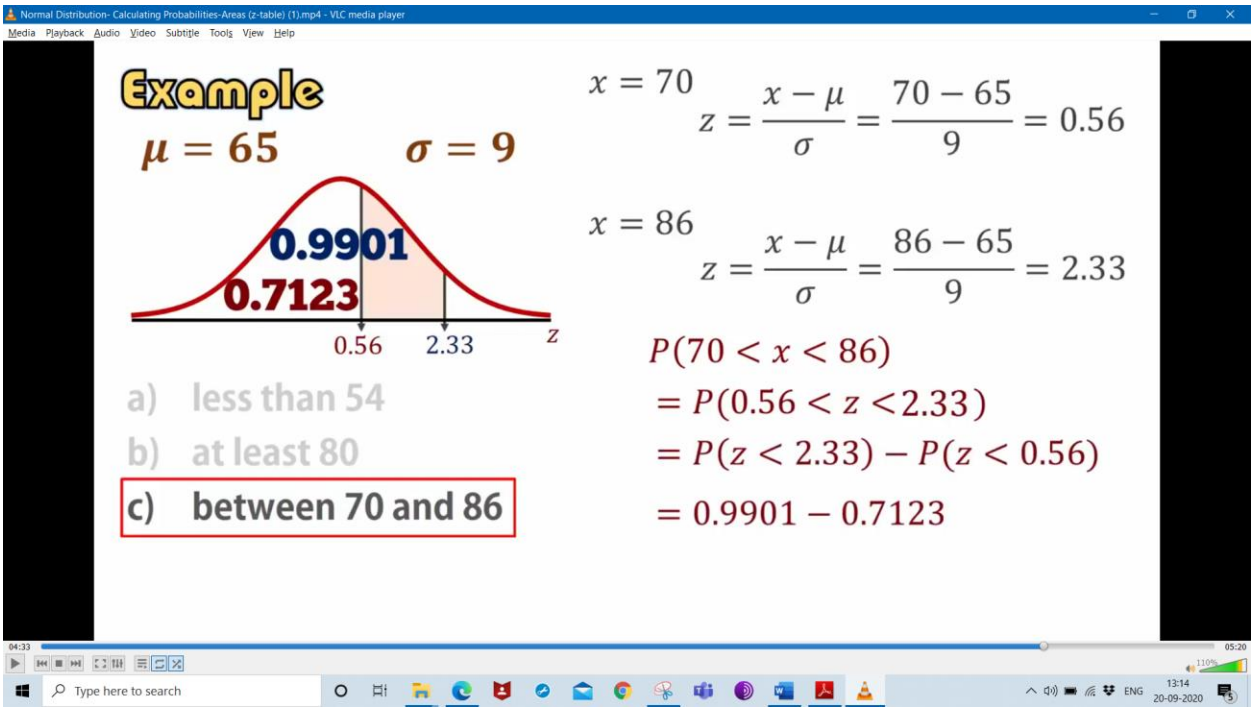

Normal Distribution- Calculating Probabilities-Areas (z-table) (1).mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Example
$\mu = 65$ $\sigma = 9$
0.0475
0.9525
$z = 1.67$
a) less than 54
b) at least 80
c) between 70 and 86
$x = 80$
$z = \frac{x - \mu}{\sigma} = \frac{80 - 65}{9}$
$z = 1.67$
$P(x > 80) = P(z > 1.67)$
$= 1 - P(z < 1.67)$
$= 1 - 0.9525$
$= 0.0475$
Type here to search
ENG 13:12 20-09-2020

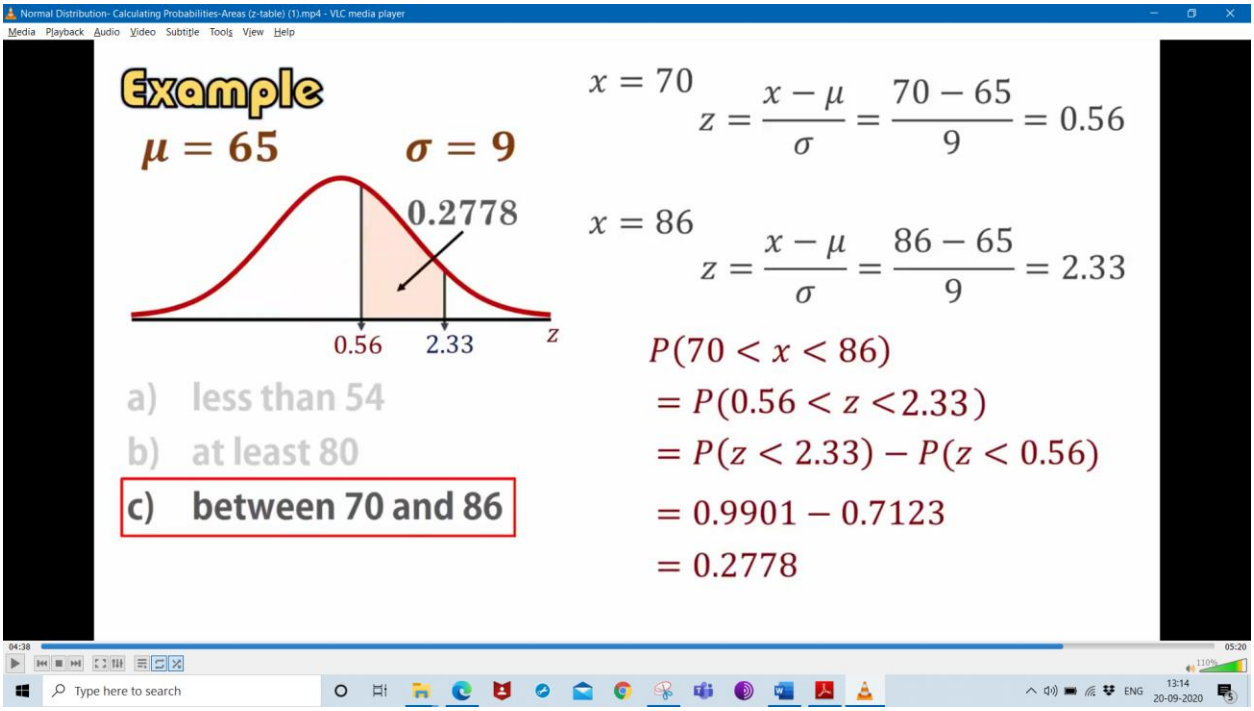

Normal Distribution- Calculating Probabilities-Areas (z-table) (1).mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Example
$\mu = 65$ $\sigma = 9$
0.0475
0.9525
$z = 1.67$
$x = 80$
$z = \frac{x - \mu}{\sigma} = \frac{80 - 65}{9}$
$z = 1.67$
a) less than 54
b) at least 80
4.75%
c) between 70 and 86
$P(x > 80) = P(z > 1.67)$
$= 1 - P(z < 1.67)$
$= 1 - 0.9525$
$= 0.0475$
Type here to search
ENG
13:12
20-09-2020

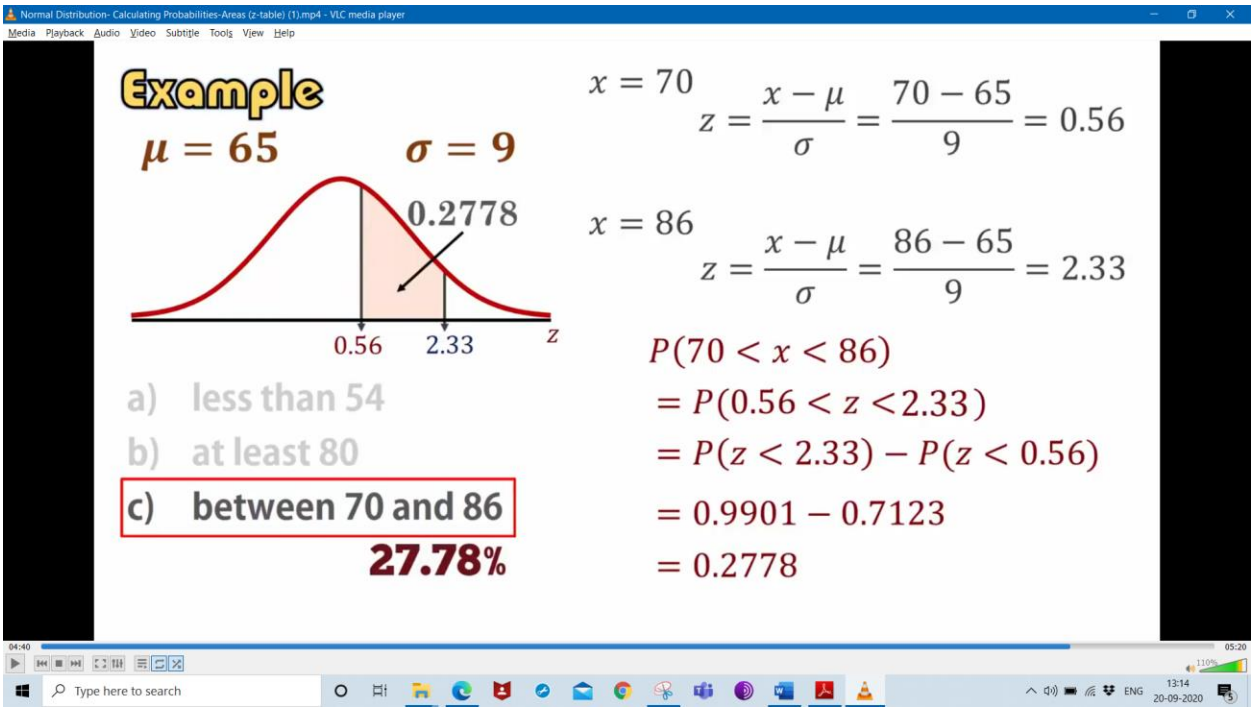

Normal Distribution- Calculating Probabilities-Areas (z-table) (1).mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Example
$\mu = 65$ $\sigma = 9$
z
a) less than 54
b) at least 80
c) between 70 and 86
Type here to search
ENG
13:12
20-09-2020

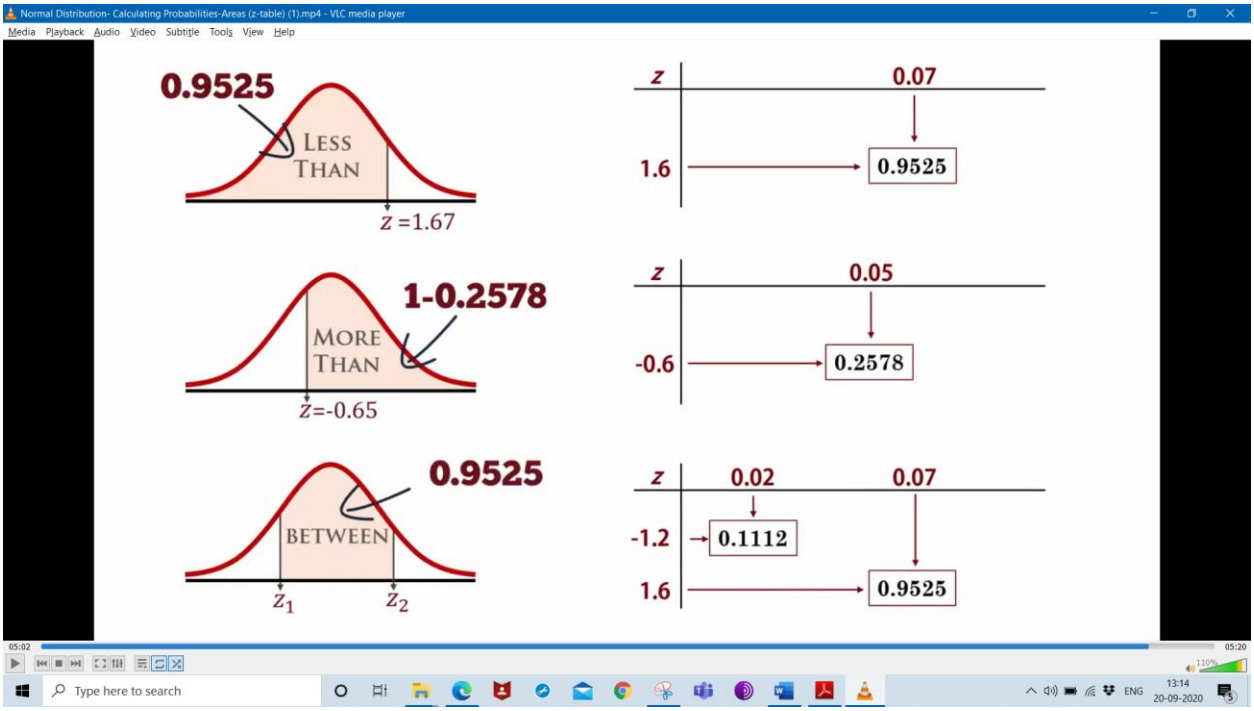

Normal Distribution- Calculating Probabilities-Areas (z-table) (1).mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Example
$\mu = 65$ $\sigma = 9$
0.9901
0.7123
0.56 2.33 z
a) less than 54
b) at least 80
c) between 70 and 86
$x = 70$ $z = \frac{x-\mu}{\sigma} = \frac{70-65}{9} = 0.56$
$x = 86$ $z = \frac{x-\mu}{\sigma} = \frac{86-65}{9} = 2.33$
z 0.03 0.06
0.5 0.7123
2.3 0.9901
Type here to search
ENG 13:13 20-09-2020

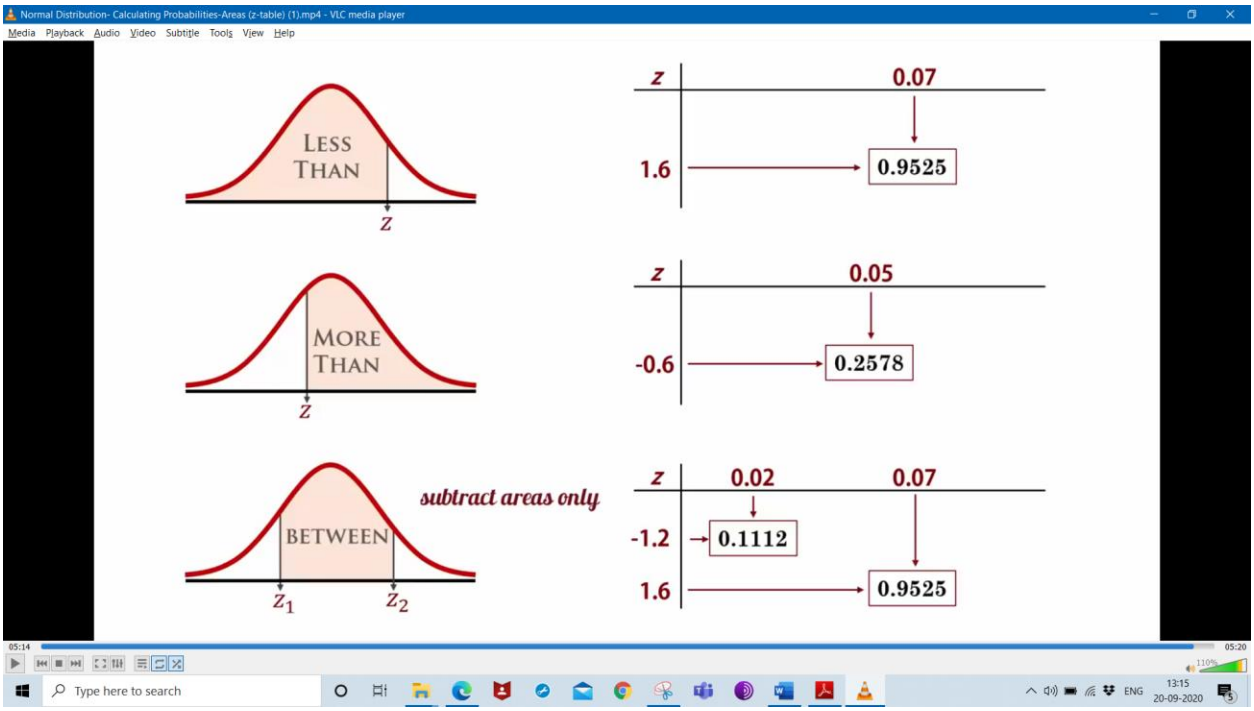

Normal Distribution- Calculating Probabilities-Areas (z-table) (1).mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Example
$\mu = 65$ $\sigma = 9$
0.9901
0.7123
0.56 2.33 z
a) less than 54
b) at least 80
c) between 70 and 86
$x = 70$ $z = \frac{x-\mu}{\sigma} = \frac{70-65}{9} = 0.56$
$x = 86$ $z = \frac{x-\mu}{\sigma} = \frac{86-65}{9} = 2.33$
z 0.03 0.06
0.5 0.7123
2.3 0.9901
Type here to search
ENG 13:13 20-09-2020


Normal Distribution- Calculating Probabilities-Areas (z-table) (1).mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Example
$\mu = 65$ $\sigma = 9$
0.9901
0.7123
0.56 2.33 z
a) less than 54
b) at least 80
c) between 70 and 86
$x = 70$ $z = \frac{x-\mu}{\sigma} = \frac{70-65}{9} = 0.56$
$x = 86$ $z = \frac{x-\mu}{\sigma} = \frac{86-65}{9} = 2.33$
$P(70 < x < 86)$
$= P(0.56 < z < 2.33)$
$= P(z < 2.33) - P(z < 0.56)$
$= 0.9901 - 0.7123$
Type here to search
ENG 13:14 20-09-2020


Normal Distribution- Calculating Probabilities-Areas (z-table) (1).mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Example
$\mu = 65$ $\sigma = 9$
0.2778
0.56 2.33 z
a) less than 54
b) at least 80
c) between 70 and 86
$x = 70$ $z = \frac{x-\mu}{\sigma} = \frac{70-65}{9} = 0.56$
$x = 86$ $z = \frac{x-\mu}{\sigma} = \frac{86-65}{9} = 2.33$
$P(70 < x < 86)$
$= P(0.56 < z < 2.33)$
$= P(z < 2.33) - P(z < 0.56)$
$= 0.9901 - 0.7123$
$= 0.2778$
Type here to search
ENG 13:14 20-09-2020


Normal Distribution- Calculating Probabilities-Areas (z-table) (1).mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Example
$\mu = 65$ $\sigma = 9$
0.2778
0.56 2.33 z
a) less than 54
b) at least 80
c) between 70 and 86
27.78%
$x = 70$ $z = \frac{x-\mu}{\sigma} = \frac{70-65}{9} = 0.56$
$x = 86$ $z = \frac{x-\mu}{\sigma} = \frac{86-65}{9} = 2.33$
$P(70 < x < 86)$
$= P(0.56 < z < 2.33)$
$= P(z < 2.33) - P(z < 0.56)$
$= 0.9901 - 0.7123$
$= 0.2778$
Type here to search
ENG 13:14 20-09-2020


Normal Distribution- Calculating Probabilities-Areas (z-table) (1).mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
0.9525
LESS THAN
z =1.67
z 0.07
1.6 0.9525
1-0.2578
MORE THAN
z=-0.65
z 0.05
-0.6 0.2578
0.9525
BETWEEN
z_1 z_2
z 0.02 0.07
-1.2 0.1112
1.6 0.9525
Type here to search
ENG 13:14 20-09-2020

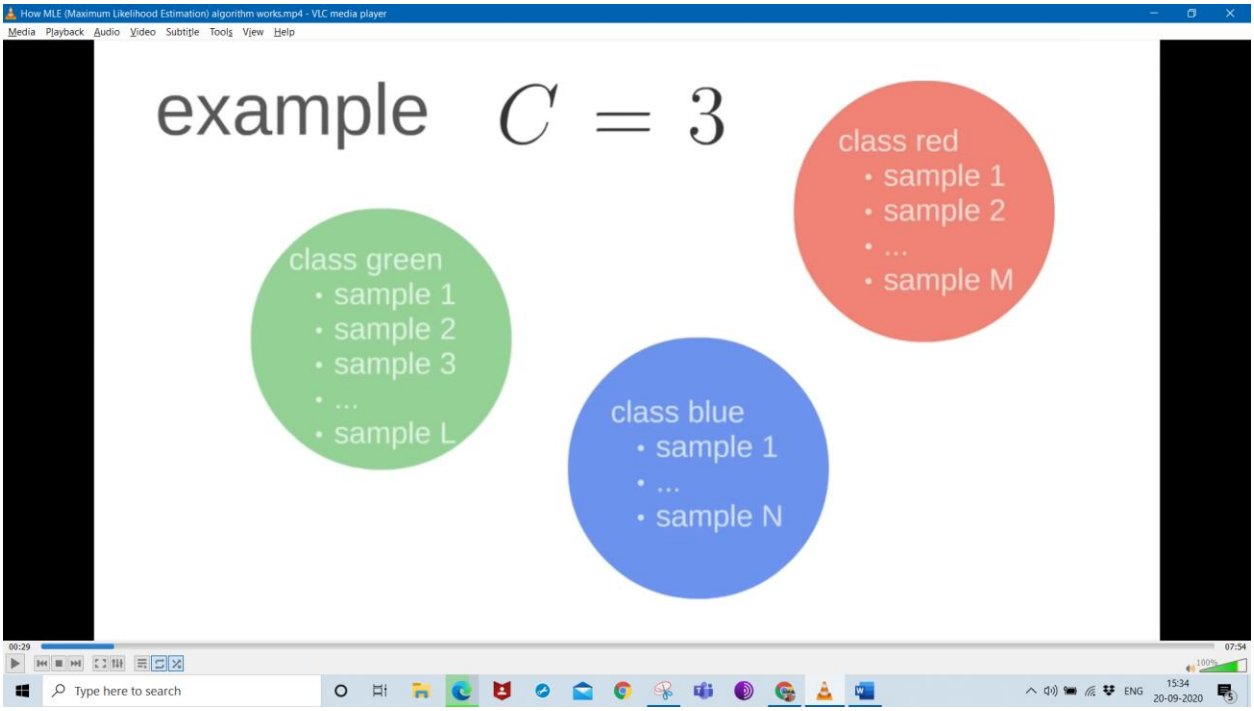

Normal Distribution- Calculating Probabilities-Areas (z-table) (1).mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
LESS THAN
z
z 0.07
1.6 0.9525
MORE THAN
z
z 0.05
-0.6 0.2578
BETWEEN
subtract areas only
z_1
z_2
z 0.02 0.07
-1.2 0.1112
1.6 0.9525
Type here to search
ENG
13:15
20-09-2020

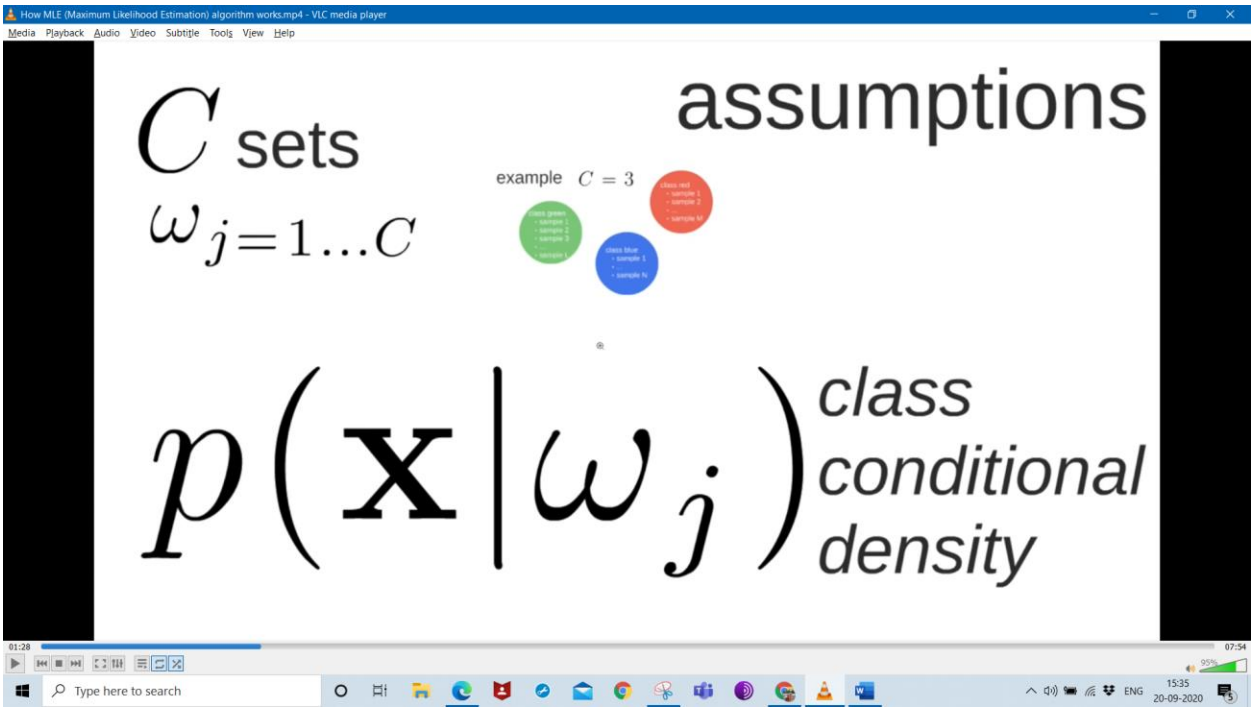
MLE Example

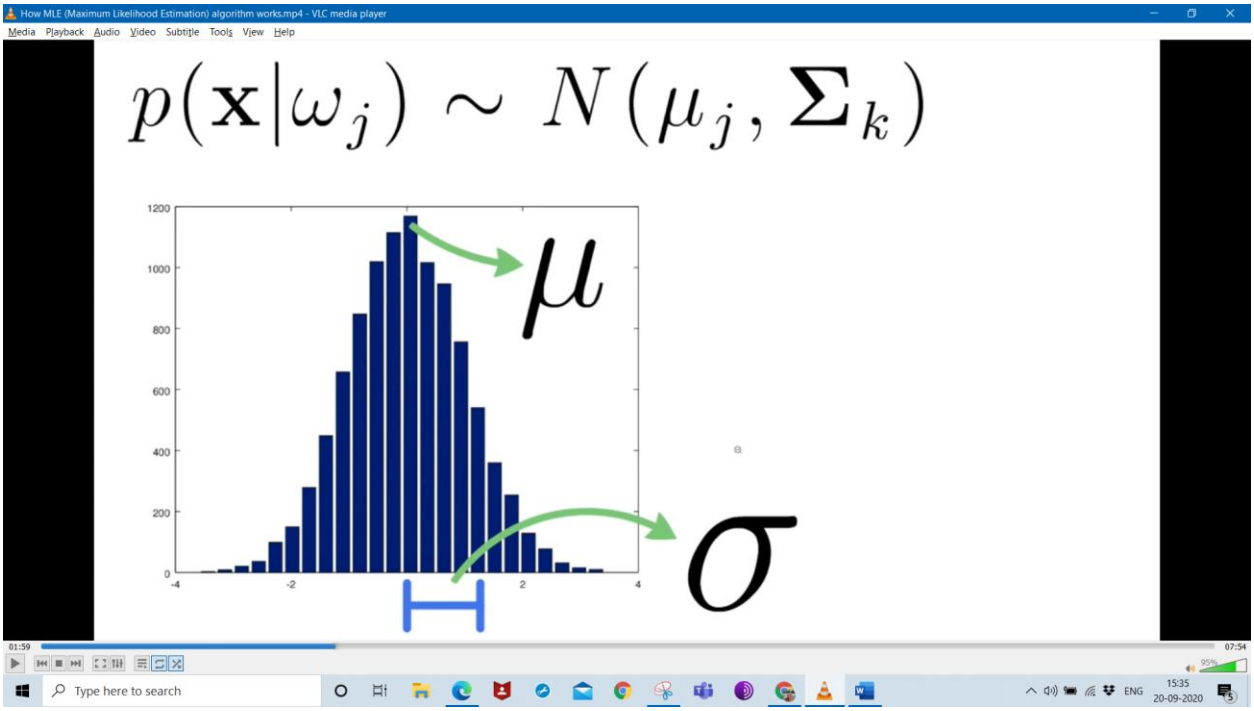
C sets

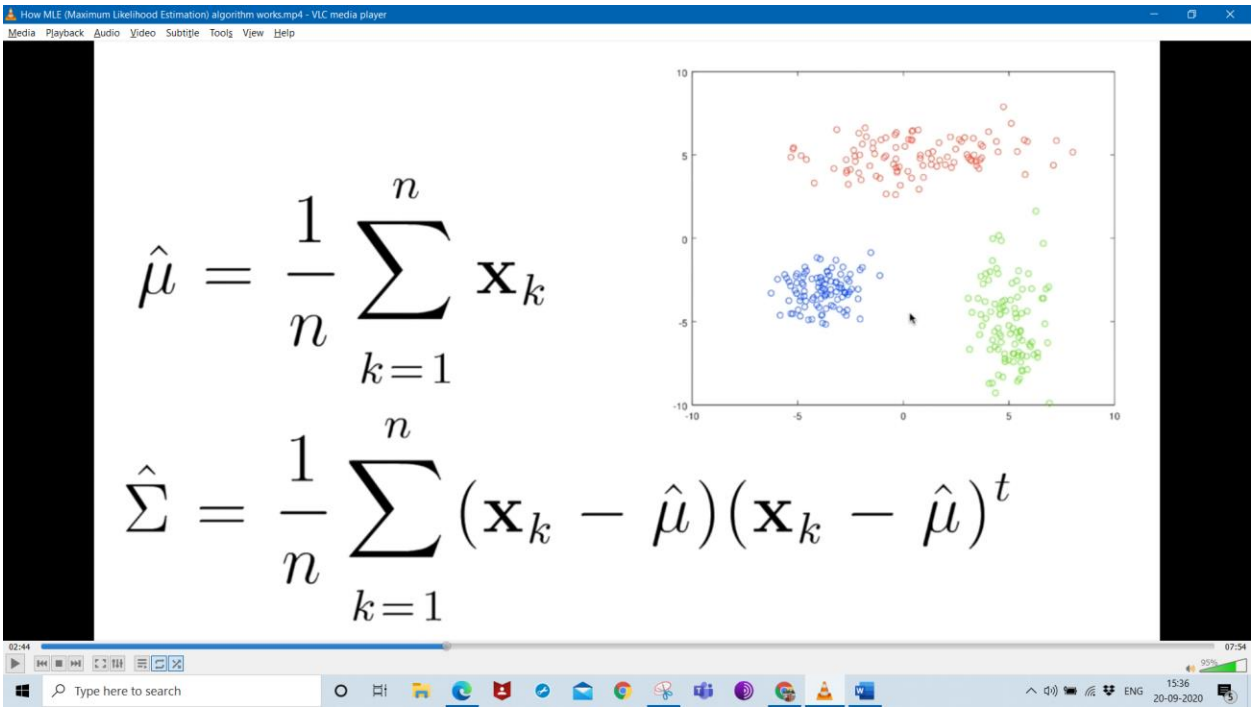
$\omega_{j=1...C}$

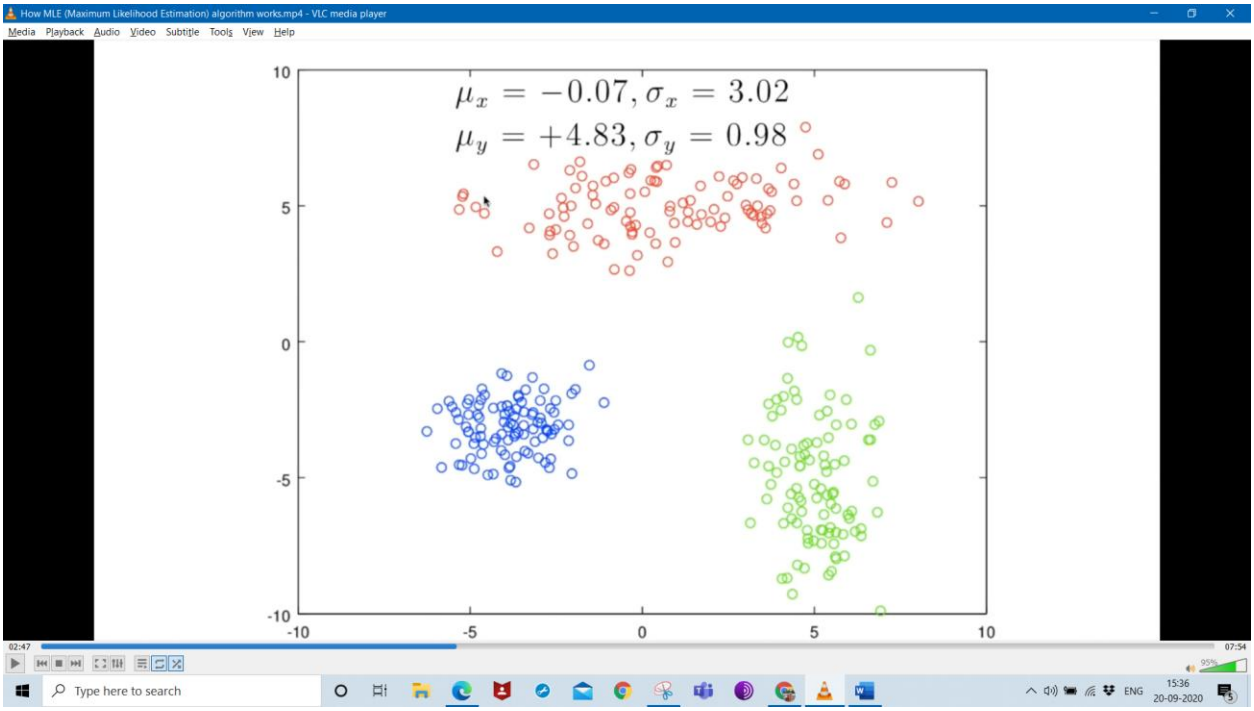
assumptions

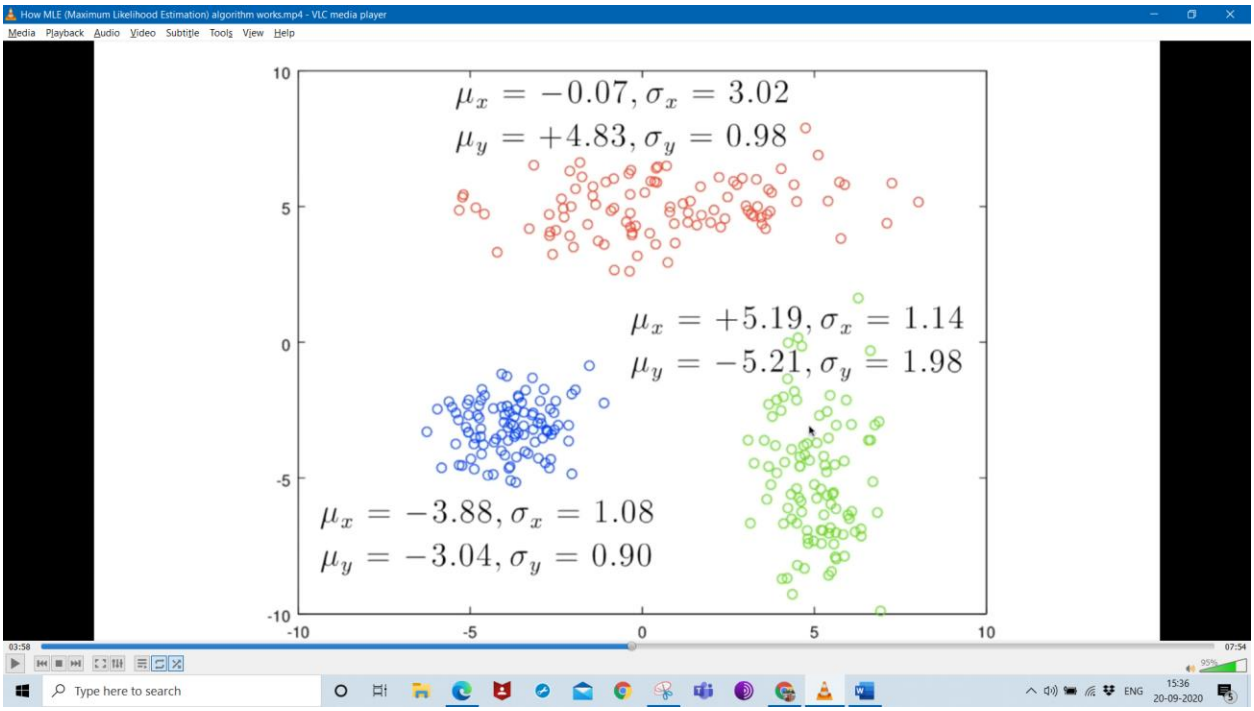


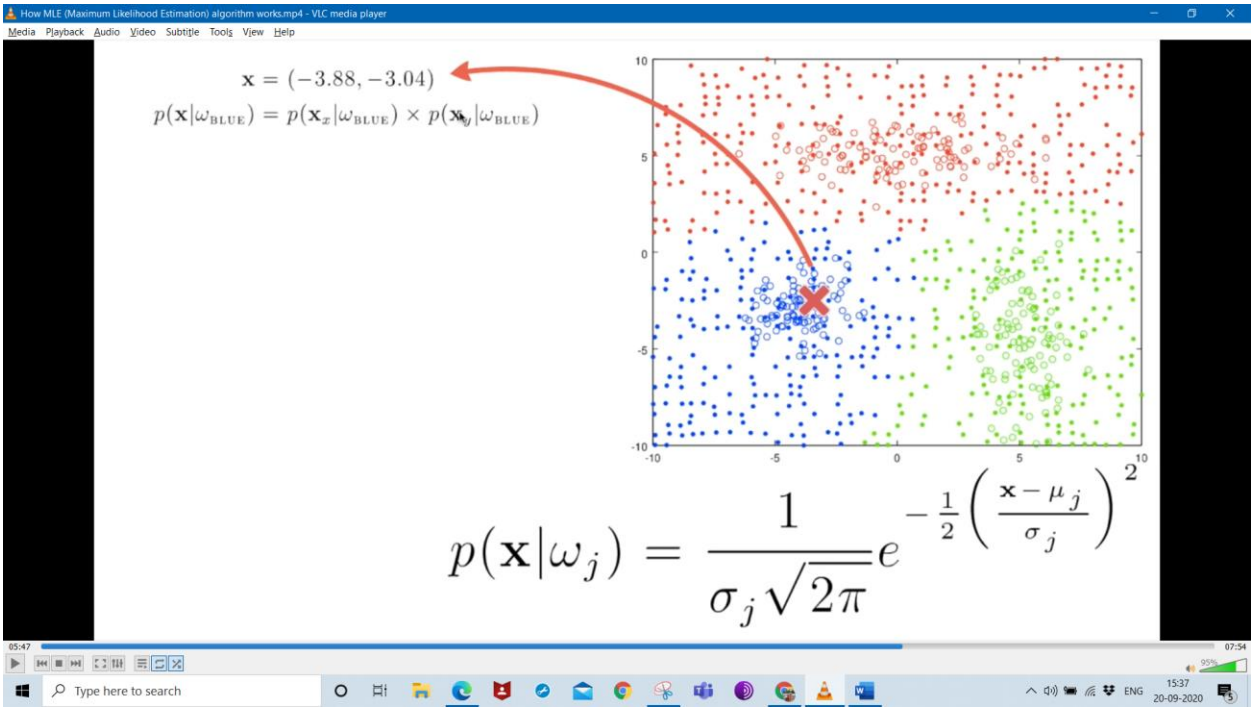

How MLE (Maximum Likelihood Estimation) algorithm works.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
C sets
$\omega_{j=1\ldots C}$
assumptions
example $C = 3$
$p(\mathbf{x}|\omega_j)$
class conditional density
01:28
07:54
Type here to search
ENG
15:35
20-09-2020

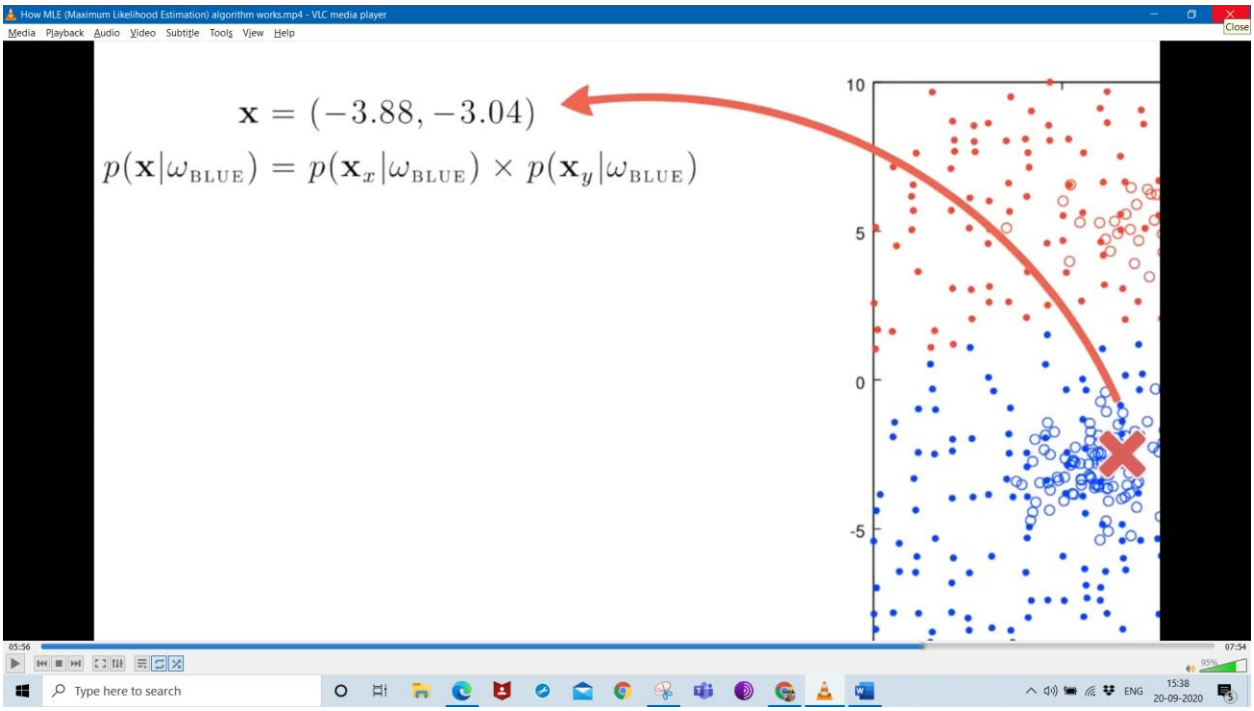

How MLE (Maximum Likelihood Estimation) algorithm works.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
$p(\mathbf{x}|\omega_j) \sim N(\mu_j, \Sigma_k)$
1200
1000
800
600
400
200
0
-4
-2
2
4
μ
σ
01:59
07:54
Type here to search
ENG
15:35
20-09-2020

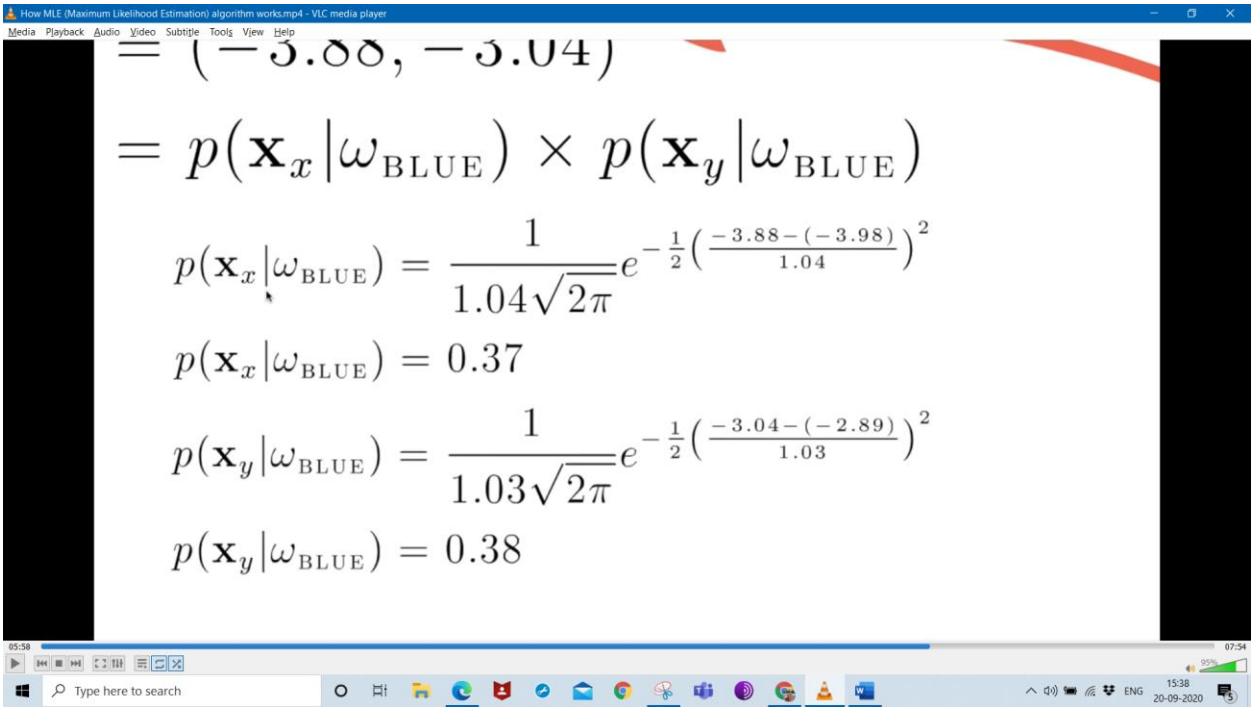

How MLE (Maximum Likelihood Estimation) algorithm works.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
$\hat{\mu} = \frac{1}{n}\sum_{k=1}^{n} \mathbf{x}_k$
$\hat{\Sigma} = \frac{1}{n}\sum_{k=1}^{n} (\mathbf{x}_k - \hat{\mu})(\mathbf{x}_k - \hat{\mu})^t$
02:44
07:54
Type here to search
ENG
15:36
20-09-2020

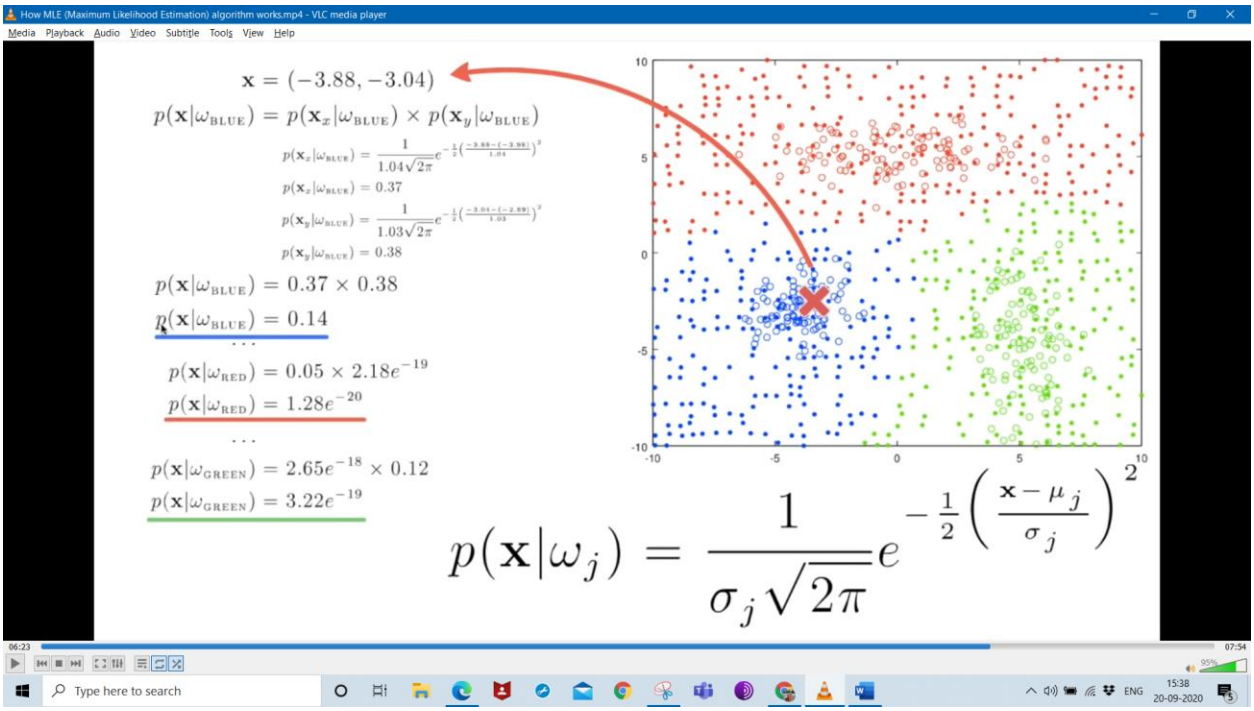

How MLE (Maximum Likelihood Estimation) algorithm works.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
$\mu_x = -0.07, \sigma_x = 3.02$
$\mu_y = +4.83, \sigma_y = 0.98$
10
5
0
-5
-10
-10
-5
0
5
10
02:47
07:54
Type here to search
ENG
15:36
20-09-2020

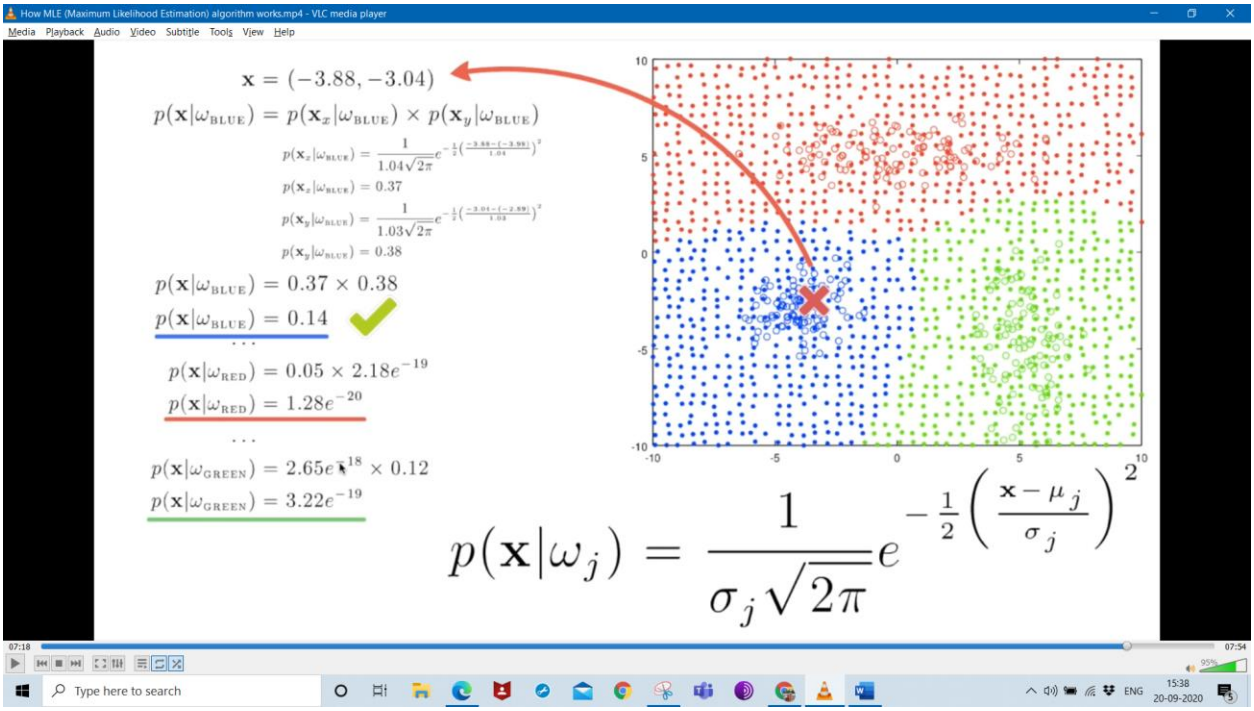

How MLE (Maximum Likelihood Estimation) algorithm works.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
$\mu_x = -0.07, \sigma_x = 3.02$
$\mu_y = +4.83, \sigma_y = 0.98$
$\mu_x = +5.19, \sigma_x = 1.14$
$\mu_y = -5.21, \sigma_y = 1.98$
$\mu_x = -3.88, \sigma_x = 1.08$
$\mu_y = -3.04, \sigma_y = 0.90$
10
5
0
-5
-10
-10
-5
0
5
10
03:58
07:54
Type here to search
ENG
15:36
20-09-2020

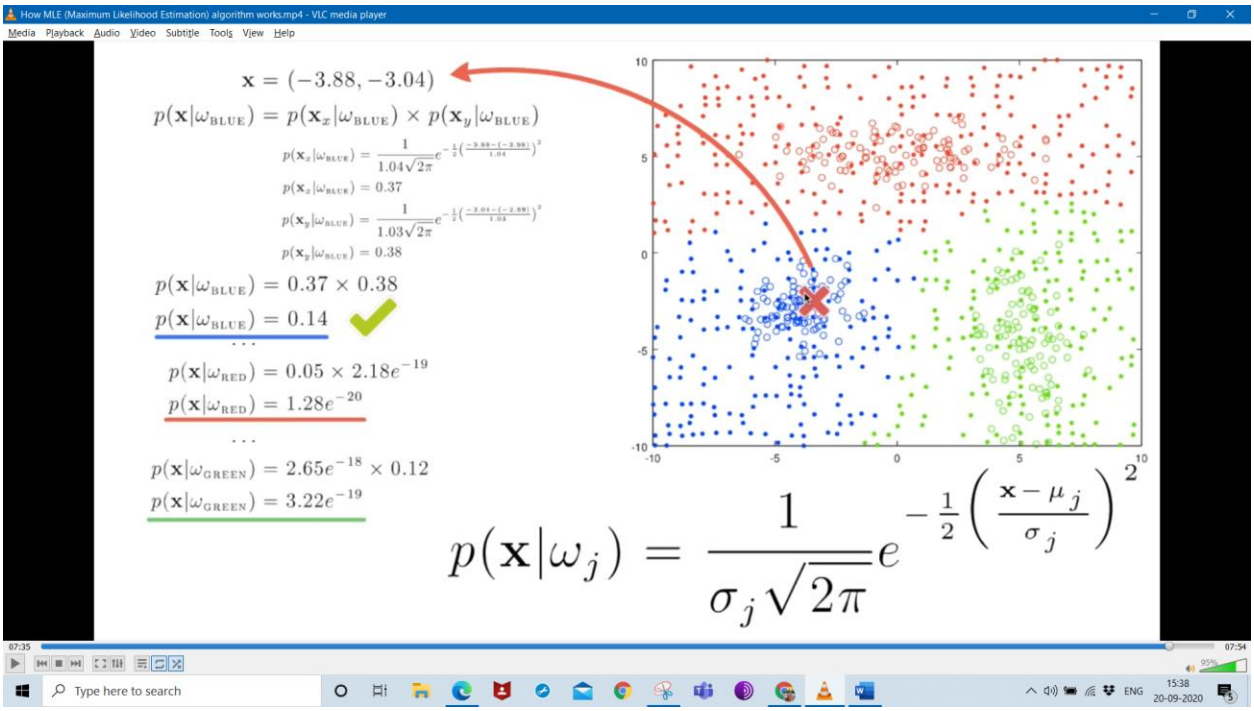

How MLE (Maximum Likelihood Estimation) algorithm works.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
$\mathbf{x} = (-3.88, -3.04)$
$p(\mathbf{x}|\omega_{\text{BLUE}}) = p(\mathbf{x}_x|\omega_{\text{BLUE}}) \times p(\mathbf{x}_y|\omega_{\text{BLUE}})$
$p(\mathbf{x}|\omega_j) = \frac{1}{\sigma_j\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\mathbf{x}-\mu_j}{\sigma_j}\right)^2}$
10
5
0
-5
-10
-10
-5
0
5
10
05:47
07:54
Type here to search
ENG
15:37
20-09-2020


How MLE (Maximum Likelihood Estimation) algorithm works.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
Close
$\mathbf{x} = (-3.88, -3.04)$
$p(\mathbf{x}|\omega_{\text{BLUE}}) = p(\mathbf{x}_x|\omega_{\text{BLUE}}) \times p(\mathbf{x}_y|\omega_{\text{BLUE}})$
10
5
0
-5
05:56
07:54
Type here to search
ENG
15:38
20-09-2020


How MLE (Maximum Likelihood Estimation) algorithm works.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
$= (-3.88, -3.04)$
$= p(\mathbf{x}_x|\omega_{\text{BLUE}}) \times p(\mathbf{x}_y|\omega_{\text{BLUE}})$
$p(\mathbf{x}_x|\omega_{\text{BLUE}}) = \frac{1}{1.04\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{-3.88-(-3.98)}{1.04}\right)^2}$
$p(\mathbf{x}_x|\omega_{\text{BLUE}}) = 0.37$
$p(\mathbf{x}_y|\omega_{\text{BLUE}}) = \frac{1}{1.03\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{-3.04-(-2.89)}{1.03}\right)^2}$
$p(\mathbf{x}_y|\omega_{\text{BLUE}}) = 0.38$
05:58
07:54
Type here to search
ENG
15:38
20-09-2020


How MLE (Maximum Likelihood Estimation) algorithm works.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
$\mathbf{x} = (-3.88, -3.04)$
$p(\mathbf{x}|\omega_{\text{BLUE}}) = p(\mathbf{x}_x|\omega_{\text{BLUE}}) \times p(\mathbf{x}_y|\omega_{\text{BLUE}})$
$p(\mathbf{x}_x|\omega_{\text{BLUE}}) = \frac{1}{1.04\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{-3.88-(-3.99)}{1.04}\right)^2}$
$p(\mathbf{x}_x|\omega_{\text{BLUE}}) = 0.37$
$p(\mathbf{x}_y|\omega_{\text{BLUE}}) = \frac{1}{1.03\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{-3.04-(-2.99)}{1.03}\right)^2}$
$p(\mathbf{x}_y|\omega_{\text{BLUE}}) = 0.38$
$p(\mathbf{x}|\omega_{\text{BLUE}}) = 0.37 \times 0.38$
$p(\mathbf{x}|\omega_{\text{BLUE}}) = 0.14$
$p(\mathbf{x}|\omega_{\text{RED}}) = 0.05 \times 2.18e^{-19}$
$p(\mathbf{x}|\omega_{\text{RED}}) = 1.28e^{-20}$
$p(\mathbf{x}|\omega_{\text{GREEN}}) = 2.65e^{-18} \times 0.12$
$p(\mathbf{x}|\omega_{\text{GREEN}}) = 3.22e^{-19}$
$p(\mathbf{x}|\omega_j) = \frac{1}{\sigma_j\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\mathbf{x}-\mu_j}{\sigma_j}\right)^2}$
06:23
07:54
Type here to search
ENG
15:38
20-09-2020


How MLE (Maximum Likelihood Estimation) algorithm works.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
$\mathbf{x} = (-3.88, -3.04)$
$p(\mathbf{x}|\omega_{\text{BLUE}}) = p(\mathbf{x}_x|\omega_{\text{BLUE}}) \times p(\mathbf{x}_y|\omega_{\text{BLUE}})$
$p(\mathbf{x}_x|\omega_{\text{BLUE}}) = \frac{1}{1.04\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{-3.88-(-3.99)}{1.04}\right)^2}$
$p(\mathbf{x}_x|\omega_{\text{BLUE}}) = 0.37$
$p(\mathbf{x}_y|\omega_{\text{BLUE}}) = \frac{1}{1.03\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{-3.04-(-2.99)}{1.03}\right)^2}$
$p(\mathbf{x}_y|\omega_{\text{BLUE}}) = 0.38$
$p(\mathbf{x}|\omega_{\text{BLUE}}) = 0.37 \times 0.38$
$p(\mathbf{x}|\omega_{\text{BLUE}}) = 0.14$
$p(\mathbf{x}|\omega_{\text{RED}}) = 0.05 \times 2.18e^{-19}$
$p(\mathbf{x}|\omega_{\text{RED}}) = 1.28e^{-20}$
$p(\mathbf{x}|\omega_{\text{GREEN}}) = 2.65e^{-18} \times 0.12$
$p(\mathbf{x}|\omega_{\text{GREEN}}) = 3.22e^{-19}$
$p(\mathbf{x}|\omega_j) = \frac{1}{\sigma_j\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\mathbf{x}-\mu_j}{\sigma_j}\right)^2}$
07:18
07:54
Type here to search
ENG
15:38
20-09-2020


How MLE (Maximum Likelihood Estimation) algorithm works.mp4 - VLC media player
Media Playback Audio Video Subtitle Tools View Help
$\mathbf{x} = (-3.88, -3.04)$
$p(\mathbf{x}|\omega_{\text{BLUE}}) = p(\mathbf{x}_x|\omega_{\text{BLUE}}) \times p(\mathbf{x}_y|\omega_{\text{BLUE}})$
$p(\mathbf{x}_x|\omega_{\text{BLUE}}) = \frac{1}{1.04\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{-3.88-(-3.99)}{1.04}\right)^2}$
$p(\mathbf{x}_x|\omega_{\text{BLUE}}) = 0.37$
$p(\mathbf{x}_y|\omega_{\text{BLUE}}) = \frac{1}{1.03\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{-3.04-(-2.99)}{1.03}\right)^2}$
$p(\mathbf{x}_y|\omega_{\text{BLUE}}) = 0.38$
$p(\mathbf{x}|\omega_{\text{BLUE}}) = 0.37 \times 0.38$
$p(\mathbf{x}|\omega_{\text{BLUE}}) = 0.14$
$p(\mathbf{x}|\omega_{\text{RED}}) = 0.05 \times 2.18e^{-19}$
$p(\mathbf{x}|\omega_{\text{RED}}) = 1.28e^{-20}$
$p(\mathbf{x}|\omega_{\text{GREEN}}) = 2.65e^{-18} \times 0.12$
$p(\mathbf{x}|\omega_{\text{GREEN}}) = 3.22e^{-19}$
$p(\mathbf{x}|\omega_j) = \frac{1}{\sigma_j\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{\mathbf{x}-\mu_j}{\sigma_j}\right)^2}$
07:35
07:54
Type here to search
ENG
15:38
20-09-2020