Modelling and machine learning

(adapted from 2nd and 4th year Computer Science courses in data science and machine learning)

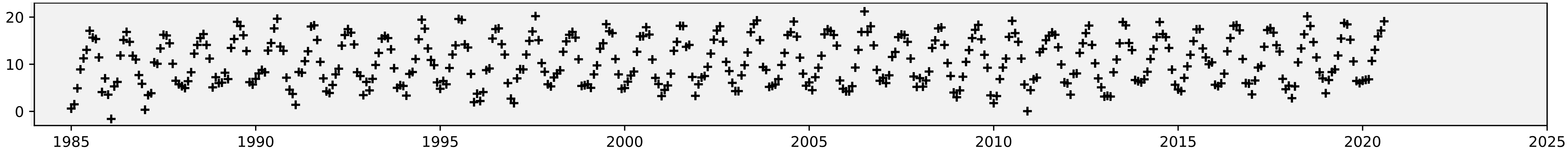
Dr Damon Wischik
Computer Science, Cambridge University

Monthly average temperatures in Cambridge, UK

The UK weather office provides monthly readings from 37 weather stations around the country. Let's look at Cambridge, from 1990.

station	yyyy	mm	t	af	rain	sun	tmin	tmax	temp
Cambridge	1985	1	1985.00	23	37.3	40.7	-2.2	3.4	0.6
Cambridge	1985	2	1985.08	13	14.6	79	-1.9	4.9	1.5
Cambridge	1985	3	1985.16	10	45.8	97.8	1.1	8.7	4.9

⋮



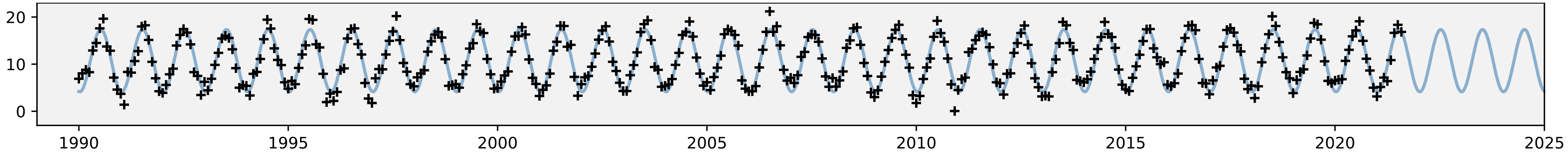
QUESTION. What model / formula would you suggest to fit this dataset?

```
def temp_model(t, …):
    return …
```

https://www.metoffice.gov.uk/research/climate/maps-and-data/historic-station-data

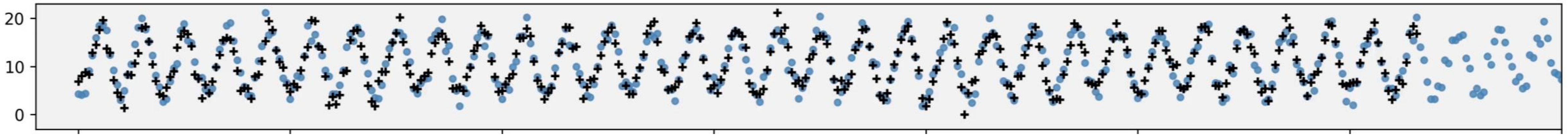
A SCIENTIST'S DETERMINISTIC MODEL

```
def temp_model(t, α=6.62, ϕ=-0.27, c=10.74):
    return c + α * np.sin(2*π*(t+ϕ))
```



A DATA SCIENTIST'S PROBABILITY MODEL

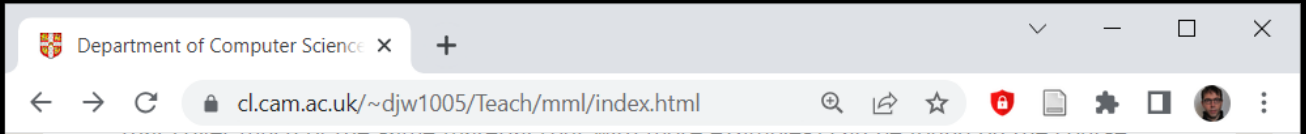
```
def rtemp(t, α=6.62, ϕ=-0.27, c=10.74, σ=1.43):
    pred = c + α * np.sin(2*π*(t+ϕ))
    return np.random.normal(loc=pred, scale=σ)
```



ALL OF MACHINE LEARNING:

1. Write out a probability model
2. Fit the model from data

Course website
[Search for “Damon Wischik” and follow the link to “Modelling and machine learning summer course”]



website for IB Data Science.

- **Prerequisites**
  You should be familiar with basic probability and calculus. Make sure you can answer these exercises, then check your answers. The programming assignments assume you are familiar with Python and numpy. In case you aren't, here's a Python tutorial, a numpy tutorial, and a matplotlib tutorial.

Assignments

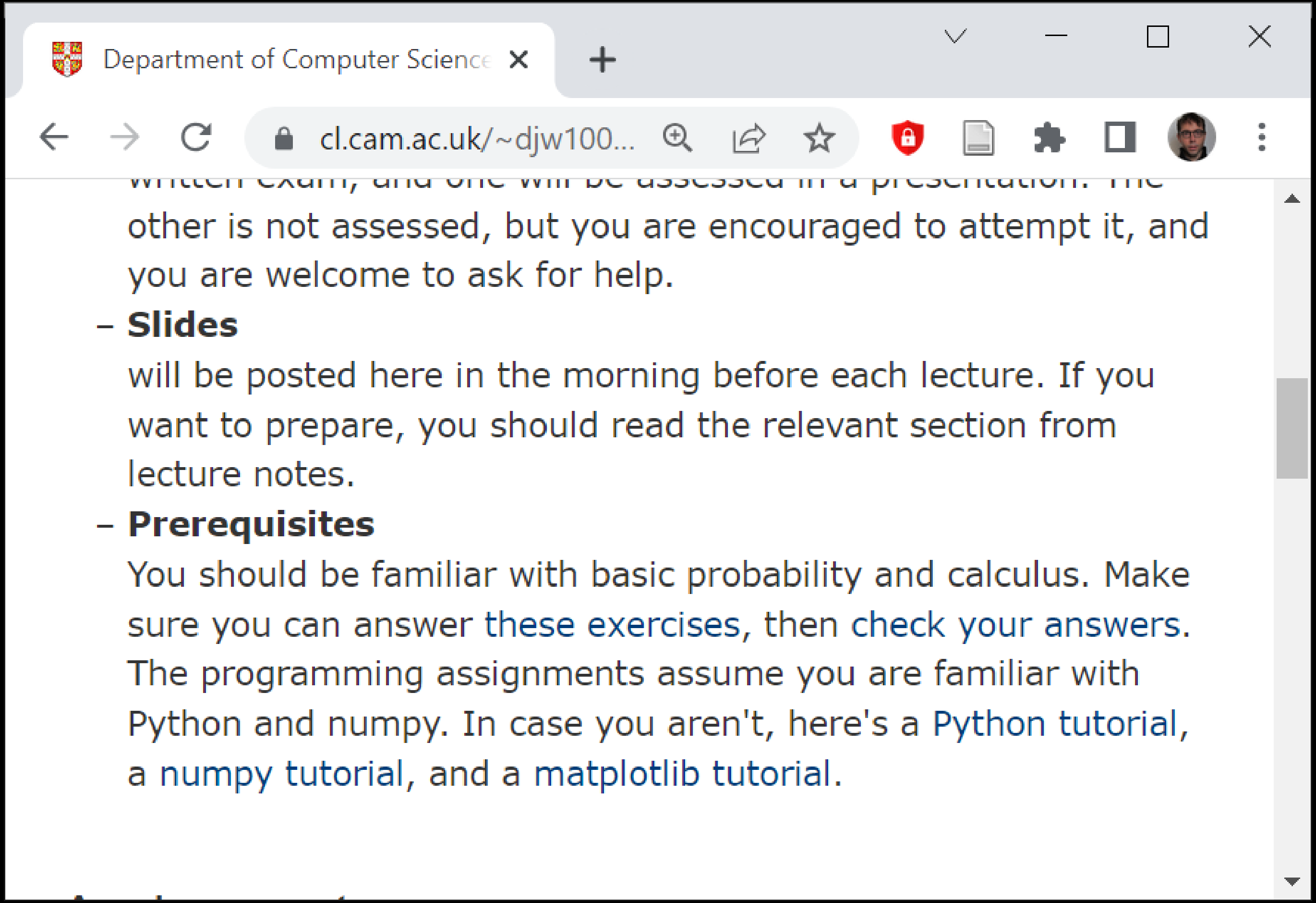
- Open-book written exam on 10 August
- Programming project: particle filter [Assessed by presentation on 16 August.]
- Advanced coursework: variational autoencoder [Optional, not assessed.]

Schedule

Session 1 [slides] (Tue 25 July, 09:00–12:00; Seminar Room, CUED)	1, 1.1, 1.2. What is a probability model? 1.3, 1.4. Maximum likelihood estimation; numerical optimization 1.5. Maths notation for specifying models 1.6, 1.7. Generative versus supervised modelling
Session 2 (Wed 26 July, 09:00–12:00; Seminar Room, CUED)	3.1, 3.2. Classification with a neural network 3.3. Optimization using pytorch 2. Regression: linear models and feature design 4. Assessing model fit
Session 3 (Thu 27 July, 09:00–12:00; Teaching Room, CUED)	5.1, 5.2. Bayes's rule 6.1. Monte Carlo integration 6.2. Bayes's rule via Monte Carlo 3.4. Latent variable generative models
Session 4	6.3, 6.4, 6.5. Variational autoencoders

- Schedule
- Slides [uploaded the night before]
- Assignments
- Code snippets

Prerequisites



other is not assessed, but you are encouraged to attempt it, and you are welcome to ask for help.

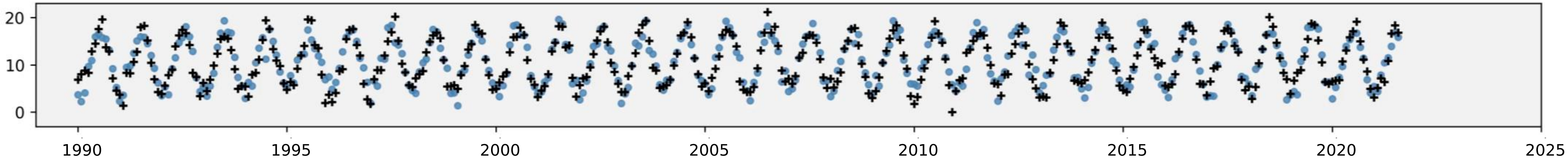
- **Slides**
  will be posted here in the morning before each lecture. If you want to prepare, you should read the relevant section from lecture notes.
- **Prerequisites**
  You should be familiar with basic probability and calculus. Make sure you can answer these exercises, then check your answers. The programming assignments assume you are familiar with Python and numpy. In case you aren't, here's a Python tutorial, a numpy tutorial, and a matplotlib tutorial.

- Basic probability
- Calculus, optimization
- Python, numpy

If you don’t get this elementary, but mildly unnatural, mathematics of elementary probability into your repertoire, then you go through a long life like a one-legged man in an ass kicking contest.

Charles Munger, business partner of Warren Buffett

1.1. How to specify a probability model

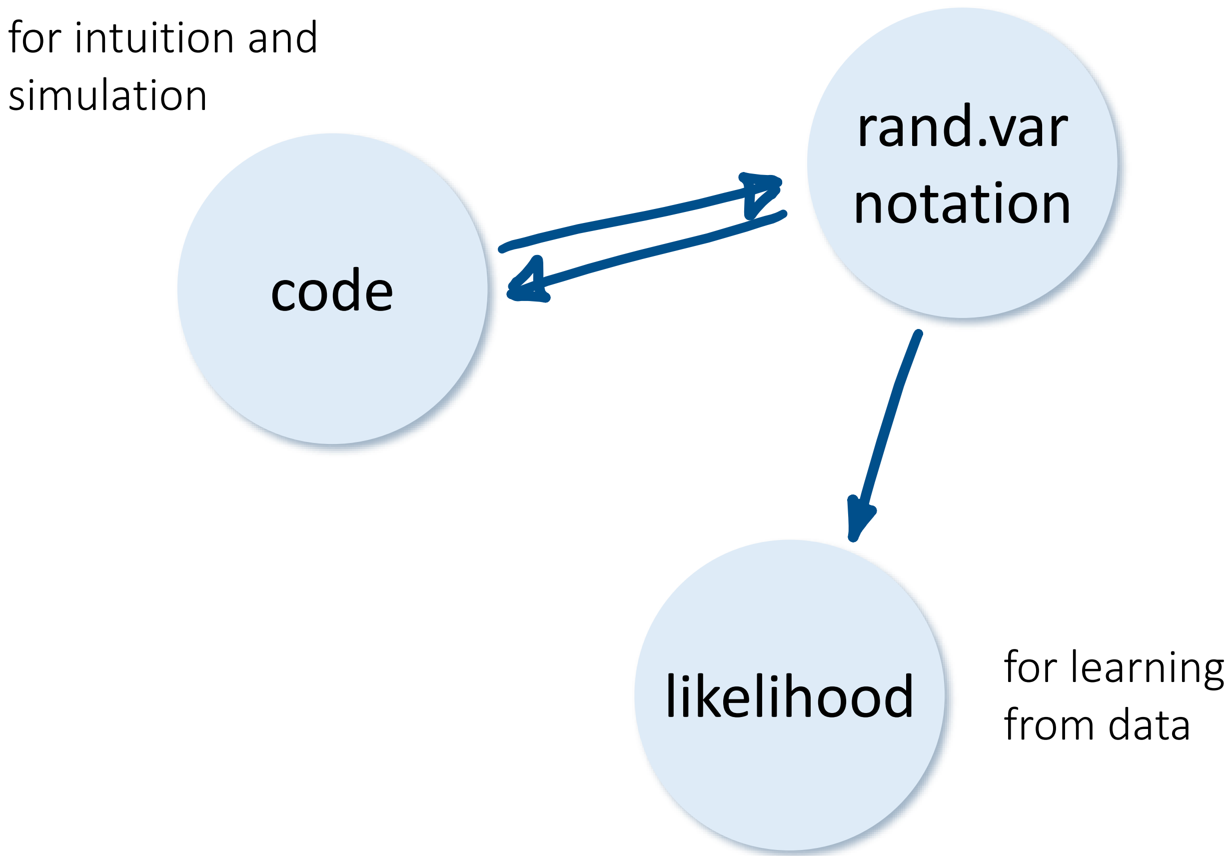



```
def rtemp(t, α=10, ϕ=-0.25, c=11, γ=0.035, σ=2):
    pred = c + α * np.sin(2*π*(t+ϕ)) + γ*t
    return np.random.normal(loc=pred, scale=σ)
```

```
df = pandas.read_csv(...)
Temp = rtemp(df.t)
```

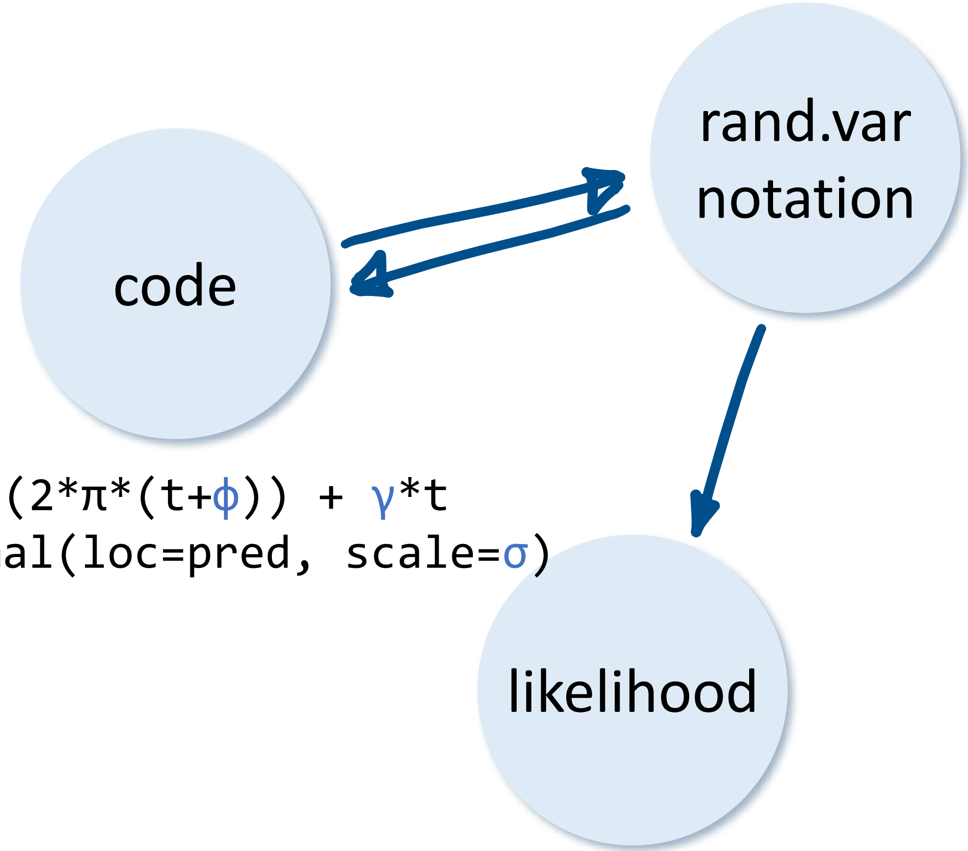
When I run this, what type of object is Temp?

Three views of a probability model



Three views of a probability model

$$\text{Temp}_i \sim \alpha \sin\big(2\pi(t_i + \varphi)\big) + c + \gamma t_i + \text{Normal}(0, \sigma^2),$$
$$i \in \{1, \dots, n\}$$




```
def rtemp(t, α,ϕ,c,γ,σ):
    pred = c + α * np.sin(2*π*(t+ϕ)) + γ*t
    return np.random.normal(loc=pred, scale=σ)
```

```python
def ry():
    x = random.random()
    y = x ** 2
    return y
```

$X \sim U[0,1]$

$Y = X^2$

```python
def ri(a,b):
    x = random.random()
    i = math.floor(a*x+b)
    return i
```

$X \sim U[0,1]$

$I = \lfloor aX + b \rfloor$

```python
x = random.random()
y = x**2
```

$X \sim U[0,1]$

$Y = X^2$

```python
def rz():
    x₁ = random.random()
    x₂ = random.random()
    return x₁ * math.log(x₂)
```

$X_1, X_2 \sim U[0,1]$

$Z = X_1 \log X_2$

Assume our random variables are independent unless specified otherwise.

```python
def rmyrandpair():
    x1 = random.random()
    x2 = random.random()
    y,z = (x1+x2, x1*x2)
    return (y,z)
```

$(Y, Z) \sim \text{Myrandpair}$

```python
λ = 3
x₁ = random.uniform(0,λ)
x₂ = random.uniform(0,λ)
```

$X_1, X_2 \sim U[0, \lambda]$

```
x = random.random()
y = 1 - x
```

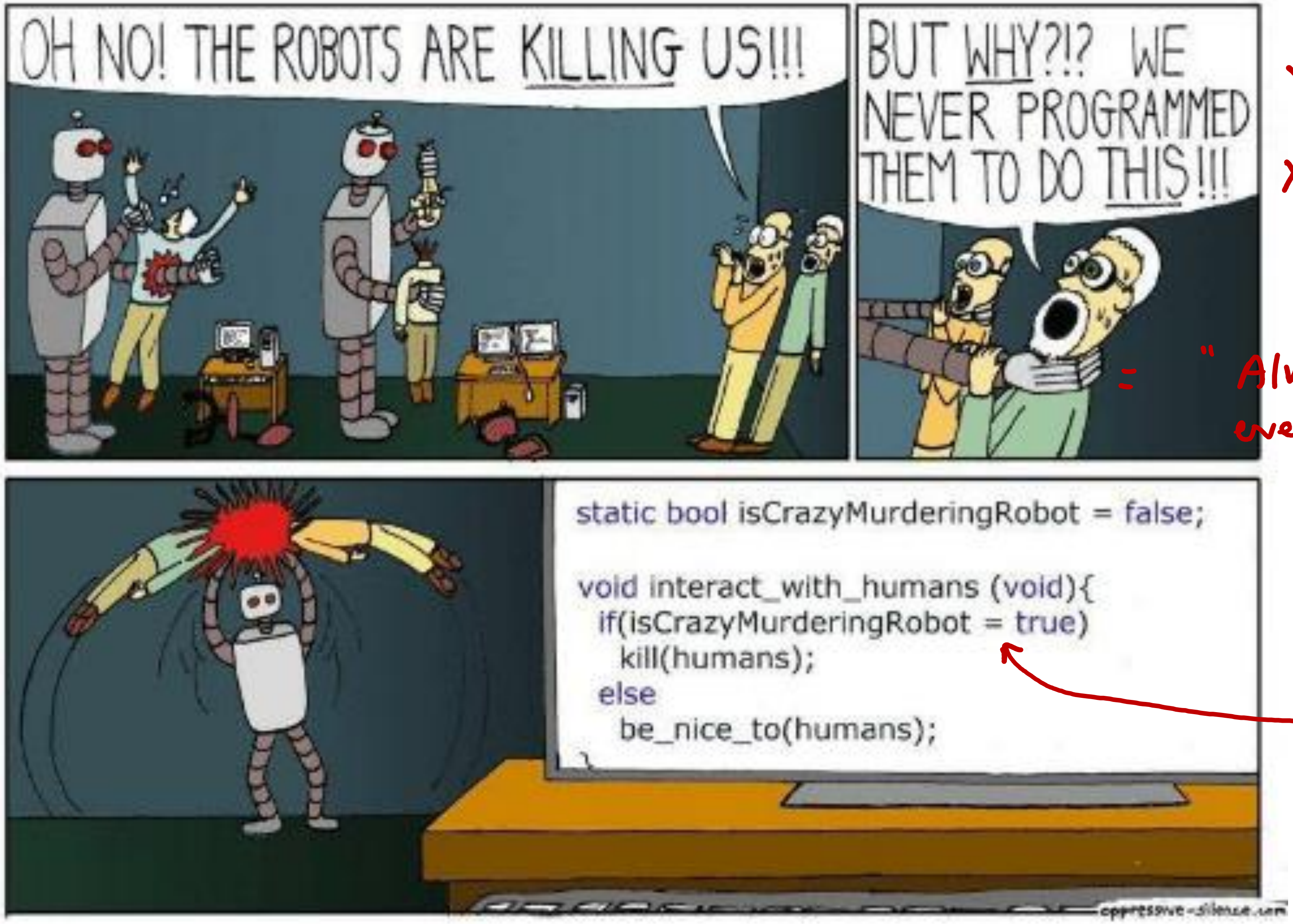
$X \sim U[0,1]$
$Y = 1 - X$

$\sim$ "has the same distribution as"
"has the same histogram"

$X \sim U[0,1]$

$Y \sim U[0,1]$

$X \sim Y$



$=$ "Always has the same value, every time I run the code"

$Y = 1 - X$

$X + Y = 1$

$==$ test for equality

$=$ assign

```
x = random.random()
y = np.random.normal(
        loc=x, scale=0.1)
```

$X \sim U[0,1]$

$Y \sim N(X, 0.1^2)$

```
def rtemp(t, α=10, ϕ=-0.25, c=11, γ=0.035, σ=2):
    pred = α*np.sin(2*π*(t+ϕ)) + c + γ*t
    return np.random.normal(loc=pred, scale=σ)

df = pandas.read_csv(...)  # data frame, 380 rows
Temp = rtemp(df.t)         # vector of 380 random temperatures
```

$$\text{Temp}_i \sim \alpha \sin\big(2\pi(t_i + \varphi)\big) + c + \gamma t_i + \text{Normal}(0, \sigma^2), \qquad i \in \{1, \dots, n\}$$

There are n=380 equations here.

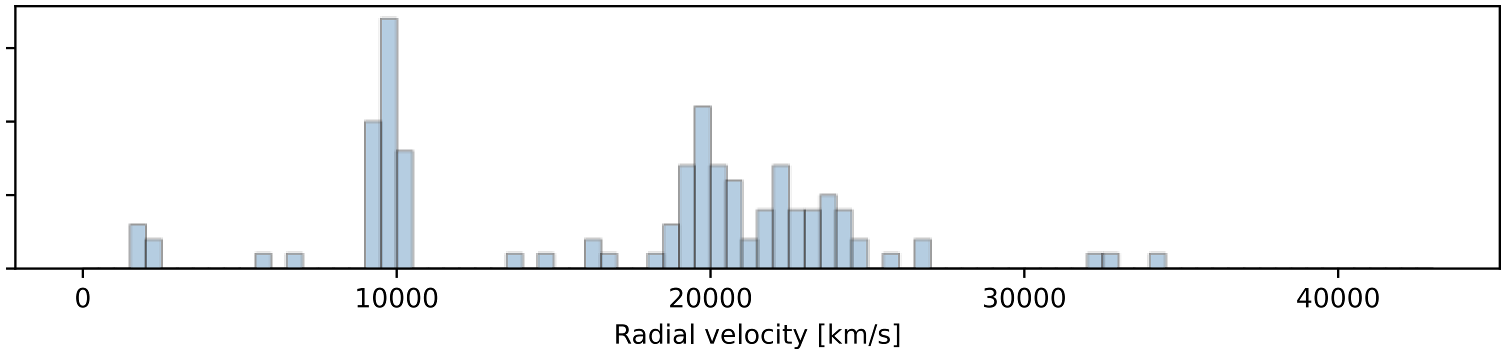
Take all of these N(0,σ²) random variables to be independent

$$\text{Temp}_i = \alpha \sin\big(2\pi(t_i + \varphi)\big) + c + \gamma t_i + \varepsilon_i, \qquad \varepsilon_i \sim \text{Normal}(0, \sigma^2), \qquad i \in \{1, \dots, n\}$$

Speeds of galaxies in the Corona Borealis region

Postman, Huchra, Geller (1986)

A histogram of radial velocities of 120 galaxies

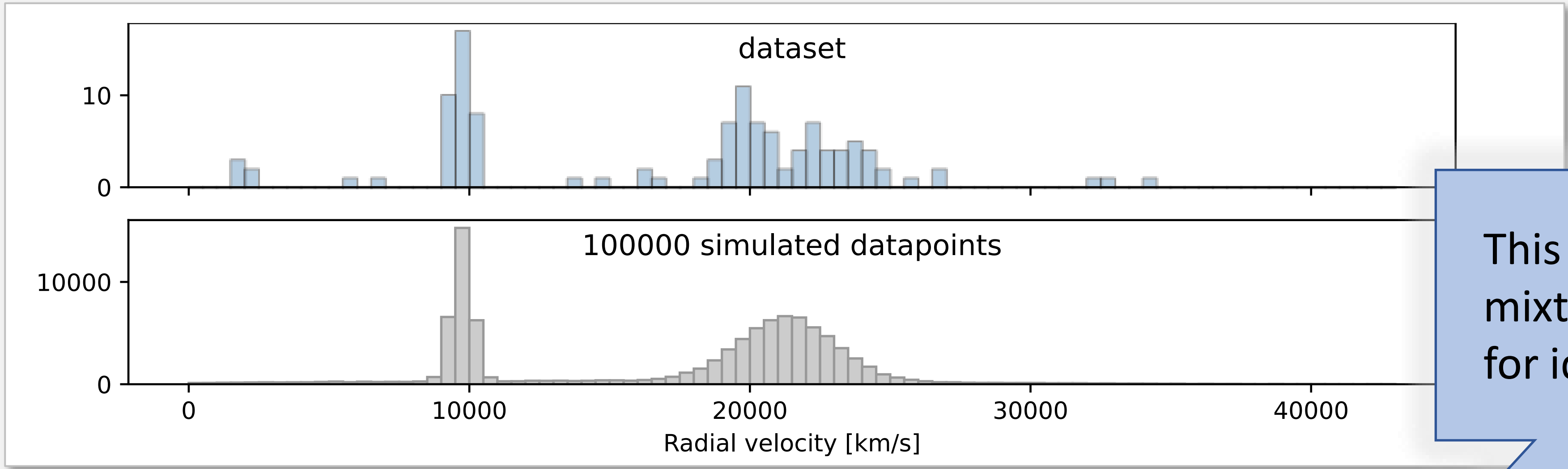


How would you complete this code?

```
def rgalaxy(...):
    # TODO: return a random galaxy speed
```

Speeds of galaxies in the Corona Borealis region

Postman, Huchra, Geller (1986)



"Gaussian Mixture Model"

This is called a Gaussian mixture model. It's handy for identifying clusters.

$$K = \begin{cases} 1 & \text{w.p. } p_1 \\ 2 & \text{w.p. } p_2 \\ 3 & \text{w.p. } p_3 \end{cases} \qquad K \sim \text{Cat}(p)$$

$$X \sim N(\mu_K, \sigma_K^2)$$

```
def rgalaxy(p,μ,σ):
    k = np.random.choice([1,2,3], p=p)
    μi,σi = μ[k-1], σ[k-1]
    x = np.random.normal(loc=μi, scale=σi)
    return x

def rgalaxies(size, p,μ,σ):
    return [rgalaxy(p,μ,σ) for _ in range(size)]

p = [0.28, 0.54, 0.18]
μ = [9740, 21300, 15000]
σ = [340, 1700, 10600]
```

1.2. The standard numerical random variables that you should know:

DISCRETE RANDOM VARIABLES

Binomial $X \sim \text{Bin}(n, p)$	$\mathbb{P}(X = x) = \binom{n}{x} p^x (1-p)^{n-x}$ $x \in \{0,1,\dots,n\}$	For count data, e.g. number of heads in n coin tosses
Poisson $X \sim \text{Pois}(\lambda)$	$\mathbb{P}(X = x) = \dfrac{\lambda^x e^{-\lambda x}}{x!}$ $x \in \{0,1,\dots\}$	For count data, e.g. number of buses passing a spot
Categorical $X \sim \text{Cat}([p_1,\dots,p_k])$	$\mathbb{P}(X = x) = p_x$ $x \in \{1,\dots,k\}$	For picking one of a fixed number of choices

CONTINUOUS RANDOM VARIABLES

Uniform $X \sim U[a, b]$	$\text{pdf}(x) = \dfrac{1}{b-a}$ $x \in [a, b]$	A uniformly-distributed floating point value
Normal / Gaussian $X \sim N(\mu, \sigma^2)$	$\text{pdf}(x) = \dfrac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2}$ $x \in \mathbb{R}$	For data about magnitudes, e.g. temperature or height
Pareto $X \sim \text{Pareto}(\alpha)$	$\text{pdf}(x) = \alpha\, x^{-(\alpha+1)}$ $x \geq 1$	For data about "cascade" magnitudes, e.g. forest fires
Exponential $X \sim \text{Exp}(\lambda)$	$\text{pdf}(x) = \lambda\, e^{-\lambda x}$ $x > 0$	For waiting times, e.g. time until next bus
Beta $X \sim \text{Beta}(a, b)$	$\text{pdf}(x) \propto x^{a-1}(1-x)^{b-1}$ $x \in (0,1)$	Arises in Bayesian inference

Section 1.3

1. Write out a probability model
2. Fit the model from data

using Maximum Likelihood Estimation

Maximum Likelihood Estimation

Suppose our probability model has unknown parameters which we'd like to estimate.

- The *likelihood* is the probability of seeing the data that we actually saw.
- The likelihood depends on our model's parameters.
- Let's simply pick the parameters that maximize the likelihood.

Exercise 1.3.1 (Coin tosses)

Suppose we take a biased coin, and tossed it $n = 10$ times, and observe $x = 6$ heads. Let's use the probability model

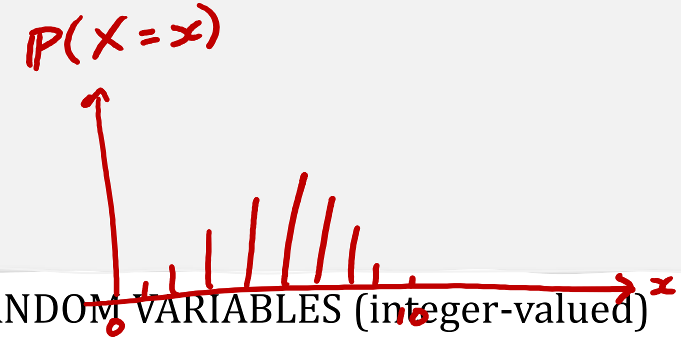
$$X \sim \text{Binom}(n, p)$$

where p is the probability of heads. Estimate p.

The likelihood,

Likelihood of the observed data: $\mathbb{P}(X = x) = \binom{n}{x} p^x (1-p)^{n-x}$

i.e. the probability of seeing data x



DISCRETE RANDOM VARIABLES (integer-valued)

Binomial $\mathbb{P}(X = x) = \binom{n}{x} p^x (1 - p)^{n-x}$

$X \sim \text{Bin}(n, p)$ $x \in \{0,1, \dots, n\}$

`np.random.binomial(n,p)`

$\binom{n}{x} = \frac{n!}{x!\,(n-x)!}$

Maximum Likelihood Estimation:

Parameter that maximizes it: choose the value of p to make this likelihood as large as possible. (at $x = 6$).

$$\log lik = \log \binom{n}{x} + x \log p + (n-x) \log(1-p)$$

$$\frac{d}{dp} \log lik = \frac{x}{p} - \frac{n-x}{1-p}$$

$$\frac{d}{dp} \log lik = 0 \Rightarrow \hat{p} = \frac{x}{n}$$

Exercise 1.3.1 (Coin tosses)

Suppose we take a biased coin, and tossed it $n = 10$ times, and observe $x = 6$ heads. Let's use the probability model

$$X \sim \text{Binom}(n, p)$$

where p is the probability of heads. Estimate p.

Log likelihood of the observed data:

$$lik = \mathbb{P}(X = x) = \binom{n}{x} p^x (1-p)^{n-x}$$

Parameter that maximizes it:

Maximum Likelihood Estimation

Suppose our probability model has unknown parameters which we'd like to estimate.

- The *likelihood* is the probability of seeing the data that we actually saw.
- The likelihood depends on our model's parameters.
- Let's simply pick the parameters that maximize the likelihood.

- If the data consists of many datapoints, and our model says they're all independent, the likelihood of the dataset is the product of the likelihoods of the individual datapoints.

Exercise 1.3.2 (Exponential sample)
Let the dataset be a list of real numbers, $x_1, \ldots, x_n$, all > 0.
Use the probability model that says they're all independent $\text{Exp}(\lambda)$ random variables, where λ is unknown. Estimate λ.

Log likelihood of the observed data:

$$\text{lik}(x_1, \ldots, x_n) = (\lambda e^{-\lambda x_1}) \times \cdots \times (\lambda e^{-\lambda x_n})$$

CONTINUOUS RANDOM VARIABLES (real-valued)

Exponential	$\text{pdf}(x) = \lambda e^{-\lambda x}$
$X \sim \text{Exp}(\lambda)$	$x > 0$
`np.random.exponential(scale=1/λ)`	

Parameter that maximizes it:

Exercise (Using indicator functions to handle boundaries)

We throw a k-sided dice, and get the answer 10.

Estimate k, using the probability model

$$\mathbb{P}(\text{throw } x) = \frac{1}{k}, \qquad x \in \{1, \dots, k\}$$

INDICATOR FUNCTIONS

The indicator function 1_A is simply

$$1_A = \begin{cases} 1 \text{ if statement } A \text{ is true} \\ 0 \text{ if statement } A \text{ is false} \end{cases}$$

Exercise 1.3.4 (Predictive models)

Consider a dataset of January temperatures, one record per year. Let t_i be the year for record $i = 1, \dots, n$, and let y_i be the temperature. Using the probability model

$$Y_i \sim \text{Normal}(\alpha + \gamma t_i, \sigma^2)$$

estimate γ, the annual rate of temperature change.

When there are multiple unknowns, we have to maximize over all of them simultaneously (even if we only care about one of them)

The question doesn't tell us the values for these parameters, so treat them as unknown parameters to be estimated.

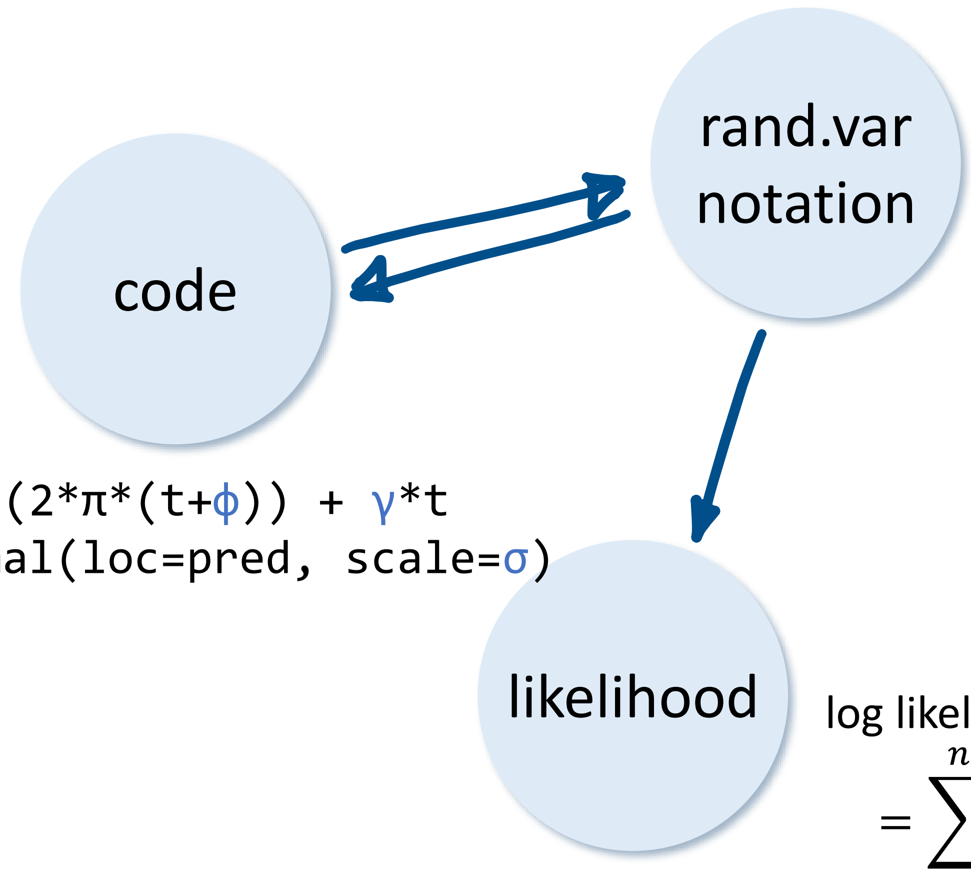
lik (dataset ; α, γ, σ)

$$\left.\begin{aligned} \frac{\partial}{\partial \alpha} \text{lik} &= 0 \\ \frac{\partial}{\partial \gamma} \text{lik} &= 0 \\ \frac{\partial}{\partial \sigma} \text{lik} &= 0 \end{aligned}\right\}$$

solve simultaneously.

Three views of a probability model

$$\text{Temp}_i \sim \alpha \sin\big(2\pi(t_i + \varphi)\big) + c + \gamma t_i + \text{Normal}(0, \sigma^2),$$
$$i \in \{1, \ldots, n\}$$




```
def rtemp(t, α,ϕ,c,γ,σ):
    pred = c + α * np.sin(2*π*(t+ϕ)) + γ*t
    return np.random.normal(loc=pred, scale=σ)
```

log likelihood of observations ($\text{temp}_1, \ldots, \text{temp}_n$)

$$= \sum_{i=1}^{n} \log\left[\frac{1}{\sqrt{2\pi\sigma^2}} e^{-(\text{temp}_i - \text{pred}_i)^2/2\sigma^2}\right]$$

Section 1.4

1. Write out a probability model
2. Fit the model from data
   using Maximum Likelihood Estimation
   with numerical optimization

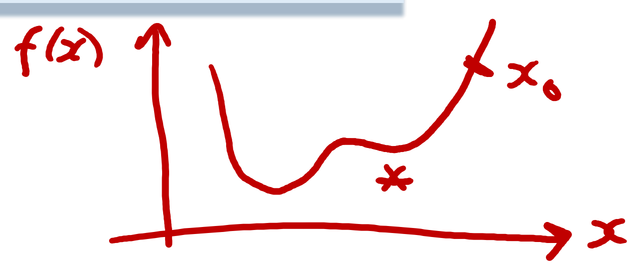
Numerical optimization with Python / scipy

To find the minimum of a function $f: \mathbb{R}^K \to \mathbb{R}$,

```
import scipy.optimize

def f(x):
    return …

x0 = […]  # initial guess
x̂ = scipy.optimize.fmin(f, x0)
```



- There is no scipy.optimize.fmax
- See the documentation to control number of iterations, …
- This function finds a local minimum, perhaps not a global minimum, so choose x_0 wisely

Exercise (Softmax transformation)

Find the maximum of

$$f(p_1, p_2, p_3) = 0.2\log p_1 + 0.5\log p_2 + 0.3\log p_3$$

over $p_1, p_2, p_3 \in (0,1)$ such that $p_1 + p_2 + p_3 = 1$.

Cunning trick: instead of finding the maximum over (p_1, p_2, p_3) such that $p_1+p_2+p_3=1$, we'll instead find the maximum over $(s_1, s_2, s_3) \in \mathbb{R}^3$

and set $p_i = \dfrac{e^{s_i}}{e^{s_1}+e^{s_2}+e^{s_3}}$

This forces $p_i \in (0,1)$, $p_1+p_2+p_3=1$.

```
def f(p):
    p₁,p₂,p₃ = p
    return 0.2*np.log(p₁) + 0.5*np.log(p₂) + 0.3*np.log(p₃)

def softmax(s):
    p = np.exp(s)
    return p / np.sum(p)

ŝ = scipy.optimize.fmin(lambda s: -f(softmax(s)), [0,0,0])
softmax(ŝ)
```

This "softmax" transformation is common in ML.

```
Optimization terminated successfully. Current function value: 1.02965. Iterations: 63.
Function evaluations: 120
array([0.19999474, 0.49999912, 0.30000614])
```

Software 1.0 is code we write. Software 2.0 is code written by the optimization based on an evaluation criterion (such as “classify this training data correctly”). It is likely that any setting where the program is not obvious but one can repeatedly evaluate the performance of it (e.g. — did you classify some images correctly? do you win games of Go?) will be subject to this transition, because the optimization can find much better code than what a human can write.

1. Write out a probability model
2. Fit the model from data
   Find the formula for the likelihood
   using Maximum Likelihood Estimation
   usually with numerical optimization

" The likelihood for X of x "

The *likelihood function* for a random variable X is written $\Pr_X(x)$ and defined as

$$\Pr_X(x) = \mathbb{P}(X = x) \quad \text{in the case where } X \text{ is discrete}$$

and as

$$\Pr_X(x) = \mathrm{pdf}(x) \quad \text{in the case where } X \text{ is continuous with prob. density function } \mathrm{pdf}(x)$$

For parameterized random variables, write

$\Pr_X(x\,;\theta)$ or $\Pr_X(x \mid \theta)$ or $\Pr_X(x)$

Transforms of random variables:

$\Pr_{X+Y}(0.2)$ or $\Pr_{X^2}(z)$

I call RNG for X, and I call the RNG for Y, and I add the two outputs together. What's the chance I got 0.2?

Pairs of random variables:

$\Pr_{X,Y}(x, y)$

The $\Pr_X(x)$ notation keeps track of

- the random variable X
- an observation x

$\Pr_{X,Y}(x, y)$ is called the *joint likelihood* of X and Y

$\Pr_{X,Y}(x, y) = \mathbb{P}(X = x \text{ and } Y = y)$
for discrete random variables

$\Pr_{X,Y}(x, y) =$ <something similar/>
for continuous random variables

Independent random variables:

$$\Pr_{X,Y}(x,y) = \Pr_X(x)\,\Pr_Y(y)$$

Independent identically-distributed (IID) sample from X:

$$\Pr(x_1,\ldots,x_n) = \Pr_X(x_1) \times \cdots \times \Pr_X(x_n)$$

Sequential generation of X then Y:

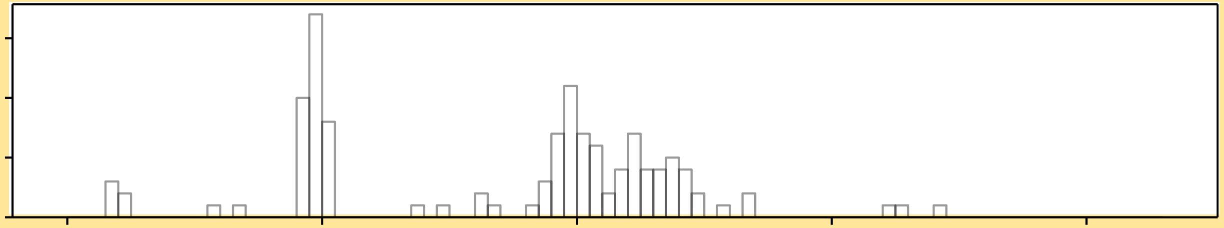
$$\Pr_{X,Y}(x,y) = \Pr_X(x)\,\Pr_Y(y\,;x)$$

$X \sim N(\mu_k, \sigma_k^2)$

$$\Pr_{K,X}(k,x) = \Pr_K(k)\,\Pr_X(x\,;k) = p_k \frac{1}{\sqrt{2\pi\sigma_k^2}} e^{-(x-\mu_k)^2/2\sigma_k^2}$$

Exercise. Write down the joint likelihood $\Pr_{K,X}(k,x)$ for

```
def rgalaxy(p,μ,σ):
    k = np.random.choice([1,2,3], p=p)
    return np.random.normal(loc=μ[k-1], scale=σ[k-1])
```



Maximum Likelihood Estimation, again

If we've seen an outcome x, and we've proposed a probability model X, and if its distribution involves some unknown parameters θ,

the *maximum likelihood estimator* for θ is

$$\hat{\theta} = \arg\max_{\theta} \Pr_X(x\,;\theta)$$

Could be discrete or continuous.
Could be a single observation, or a dataset.
The point of the likelihood notation is to be able to write down a single equation and cover all these cases.

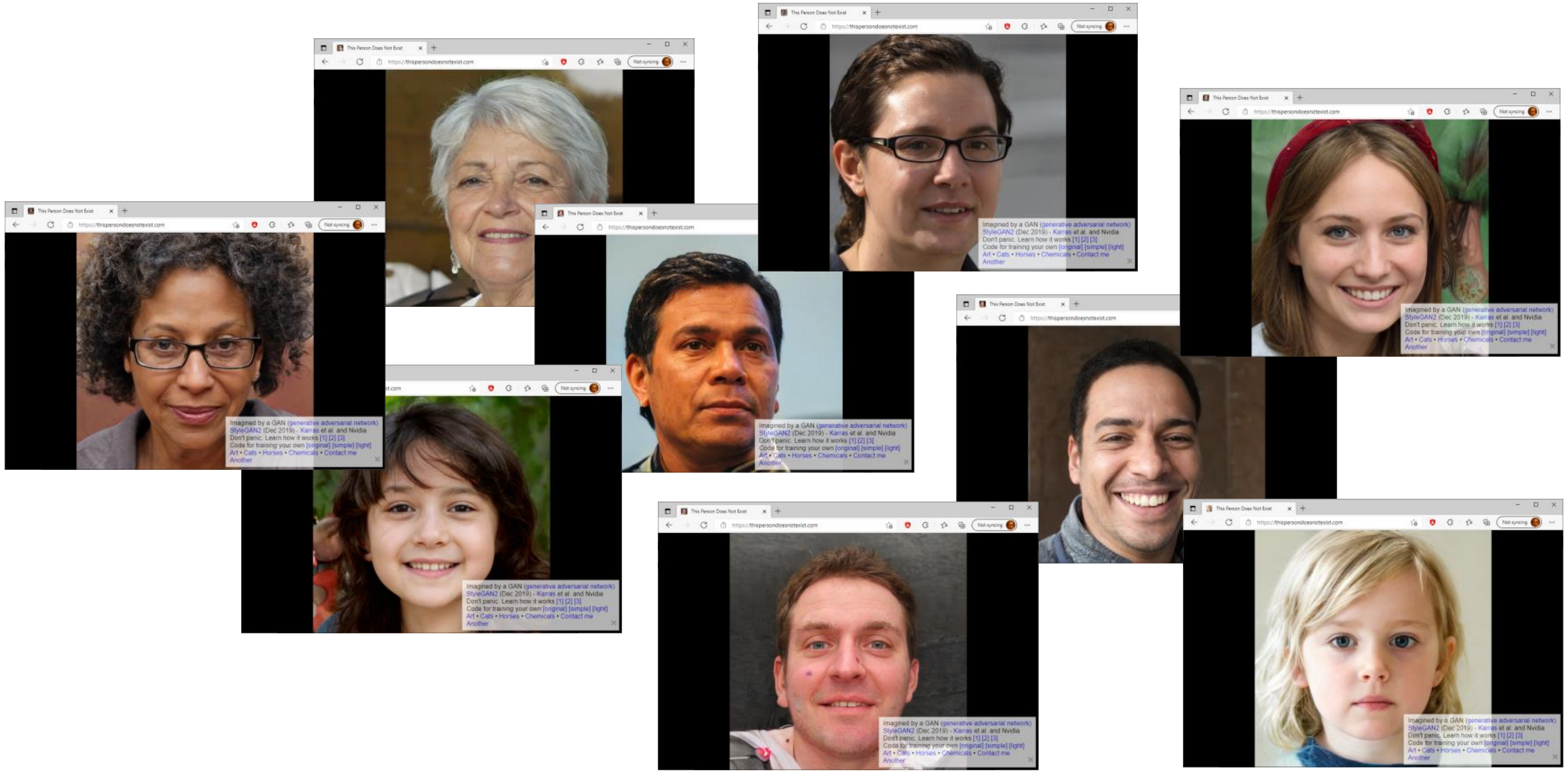
Brain teaser

Let $X \sim \text{Bin}(n = 2, p = 0.9)$. What is $\Pr_X(X)$?

$\Pr_X(x)$

1.6. GENERATIVE MODELLING

Given a dataset $x_1, \dots, x_n$ can we design a probability model that might have generated it?



1.6. GENERATIVE MODELLING

Given a dataset $x_1, \dots, x_n$ can we design a probability model that might have generated it?

1. Choose a distribution with tuneable parameters, call it X. We want $x_1, \dots, x_n$ to look like independent samples from X.
2. Write out the likelihood of the dataset
   $\Pr(x_1, \dots, x_n\,; \theta) = \Pr_X(x_1; \theta) \times \cdots \times \Pr_X(x_n; \theta)$
3. Fit the model using maximum likelihood estimation

$\log \Pr(x_1, \dots, x_n) = \sum_{i=1}^{n} \log \Pr(x_i)$

Exercise 1.6.1 (Fitting a Normal distribution)
Given a numerical dataset $x_1, \dots, x_n$, fit a $\mathrm{Normal}(\mu, \sigma^2)$ distribution, where μ and σ are unknown.

Model for a single observation

$$X \sim N(\mu, \sigma^2)$$

Likelihood for a single observation

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

Log likelihood of the dataset

$$\log \Pr(x_1, \dots, x_n; \mu, \sigma) = \log\left[\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n e^{-\sum_{i=1}^n (x_i-\mu)^2/2\sigma^2}\right]$$

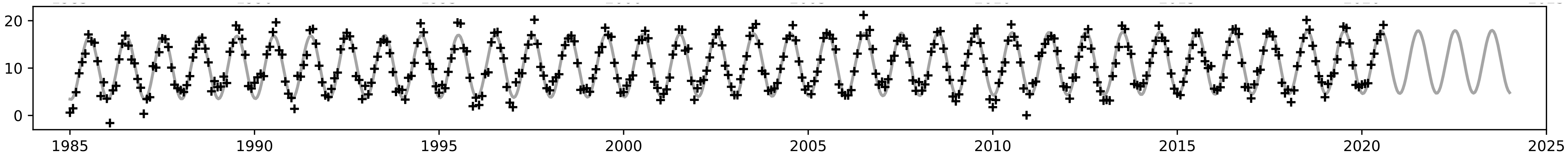
$$= -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^n (x_i-\mu)^2$$

Maximize over unknown parameters

1.7. SUPERVISED LEARNING

station	yyyy	mm	t	af	rain	sun	tmin	tmax	temp
Cambridge	1985	1	1985.00	23	37.3	40.7	-2.2	3.4	0.6
Cambridge	1985	2	1985.08	13	14.6	79	-1.9	4.9	1.5
Cambridge	1985	3	1985.16	10	45.8	97.8	1.1	8.7	4.9

⋮



How have temperatures been changing? What will they be in the future?

i.e. how can I PREDICT temp GIVEN t?

https://www.metoffice.gov.uk/research/climate/maps-and-data/historic-station-data

1.7. SUPERVISED LEARNING

station	yyyy	mm	t	af	rain	sun	tmin	tmax	temp
Cambridge	1985	1	1985.00	23	37.3	40.7	-2.2	3.4	0.6
Cambridge	1985	2	1985.08	13	14.6	79	-1.9	4.9	1.5
Cambridge	1985	3	1985.16	10	45.8	97.8	1.1	8.7	4.9
⋮									

called the PREDICTOR variable,
or the FEATURE,
or the COVARIATE

called the RESPONSE,
or the LABEL variable

How have temperatures been changing? What will they be in the future?

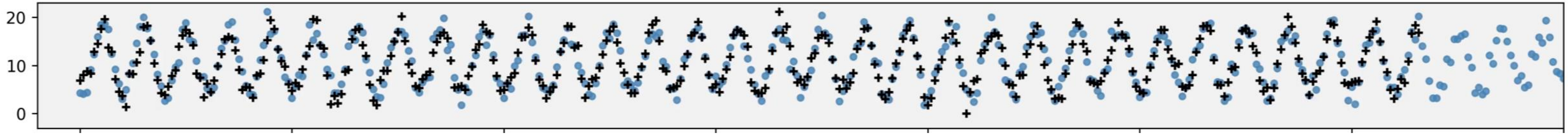
i.e. how can I PREDICT temp GIVEN t?

https://www.metoffice.gov.uk/research/climate/maps-and-data/historic-station-data

1.7. SUPERVISED LEARNING

station	yyyy	mm	t	af	rain	sun	tmin	tmax	temp
Cambridge	1985	1	1985.00	23	37.3	40.7	-2.2	3.4	0.6
Cambridge	1985	2	1985.08	13	14.6	79	-1.9	4.9	1.5
Cambridge	1985	3	1985.16	10	45.8	97.8	1.1	8.7	4.9

⋮



How have temperatures been changing? What will they be in the future?

~~i.e. how can I PREDICT temp GIVEN t?~~

i.e. what's a good PROBABILITY MODEL for temp GIVEN t?

https://www.metoffice.gov.uk/research/climate/maps-and-data/historic-station-data

Given a dataset $(x_1, y_1), \ldots, (x_n, y_n)$ where y_i is the label in record i and x_i is the predictor variable or variables,

Supervised learning

1. Choose a probability distribution for the label, which depends on one or more unknown parameters θ as well as on the predictors. Let its likelihood be

$$\Pr_Y(y\,; x, \theta)$$

2. Model the dataset as independent observations of Y drawn from this distribution, i.e. let the likelihood of the dataset be
$$\Pr(y_1, \ldots, y_n\,; x_1, \ldots, x_n, \theta) = \Pr_Y(y_1; x_1, \theta) \times \cdots \times \Pr_Y(y_n; x_n, \theta)$$

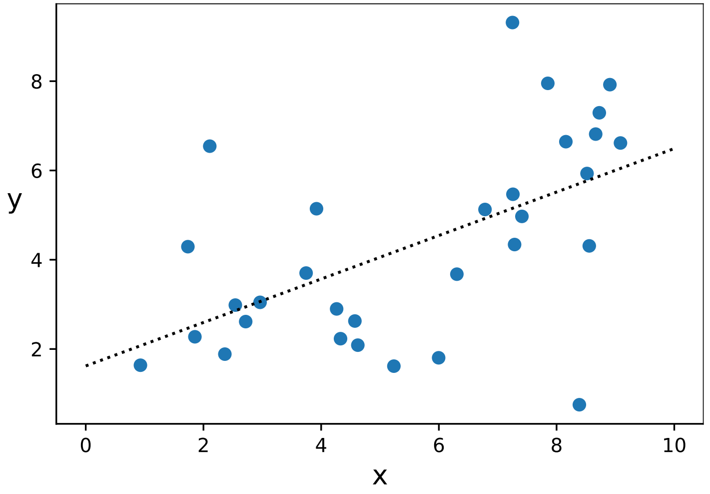
3. Estimate θ using maximum likelihood estimation

Exercise (Straight-line fit)

Given a labelled dataset $(x_1, y_1), \ldots, (x_n, y_n)$ consisting of pairs of numbers, fit the model

$$Y_i \sim a + b\, x_i + \text{Normal}(0, \sigma^2)$$

where σ is given and a and b are parameters to be estimated.



Data from Randall Munroe, https://xkcd.com/2048/, via https://gitlab.com/b-rowlingson

1.2. The standard numerical random variables that you should know:

DISCRETE RANDOM VARIABLES

Binomial $X \sim \mathrm{Bin}(n,p)$	$\mathbb{P}(X=x)=\binom{n}{x}p^x(1-p)^{n-x}$ $x \in \{0,1,\dots,n\}$	For count data, e.g. number of heads in n coin tosses
Poisson $X \sim \mathrm{Pois}(\lambda)$	$\mathbb{P}(X=x)=\frac{\lambda^x e^{-\lambda x}}{x!}$ $x \in \{0,1,\dots\}$	For count data, e.g. number of buses passing a spot
Categorical $X \sim \mathrm{Cat}([p_1,\dots,p_k])$	$\mathbb{P}(X=x)=p_x$ $x \in \{1,\dots,k\}$	For picking one of a fixed number of choices

CONTINUOUS RANDOM VARIABLES

Uniform $X \sim U[a,b]$	$\mathrm{pdf}(x)=\frac{1}{b-a}$ $x \in [a,b]$	A uniformly-distributed floating point value
Normal / Gaussian $X \sim N(\mu,\sigma^2)$	$\mathrm{pdf}(x)=\frac{1}{\sqrt{2\pi\sigma^2}}e^{-(x-\mu)^2/2\sigma^2}$ $x \in \mathbb{R}$	For data about magnitudes, e.g. temperature or height
Pareto $X \sim \mathrm{Pareto}(\alpha)$	$\mathrm{pdf}(x)=\alpha\, x^{-(\alpha+1)}$ $x \geq 1$	For data about "cascade" magnitudes, e.g. forest fires
Exponential $X \sim \mathrm{Exp}(\lambda)$	$\mathrm{pdf}(x)=\lambda\, e^{-\lambda x}$ $x > 0$	For waiting times, e.g. time until next bus
Beta $X \sim \mathrm{Beta}(a,b)$	$\mathrm{pdf}(x) \propto x^{a-1}(1-x)^{b-1}$ $x \in (0,1)$	Arises in Bayesian inference

1.2. The standard numerical random variables that you should know:

Useful properties of the Normal distribution:

- If we rescale a Normal, we get a Normal
  $a + b\,N(0,1) \sim a + N(0,b^2) \sim N(a,b^2)$ for constants a and b
- If we add independent Normals, we get a Normal
  $N(\mu,\sigma^2) + N(\nu,\rho^2) \sim N(\mu+\nu, \sigma^2+\rho^2)$

Normal / Gaussian
$X \sim N(\mu,\sigma^2)$

$$\text{pdf}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x-\mu)^2/2\sigma^2}$$

$x \in \mathbb{R}$

For data about magnitudes, e.g. temperature or height

The Uniform isn't as nicely behaved:

- $a + b\,U[0,1] \sim U[a, a+b]$
- $U[0,1] + U[0,1]$ is not uniform

The Binomial isn't as nicely behaved:

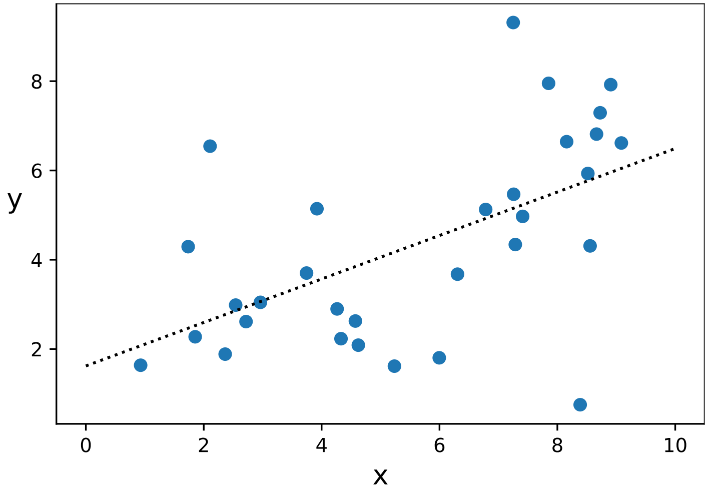
- $a + b\,\text{Bin}(n,p)$ is not Binomial
- $\text{Bin}(n_1,p) + \text{Bin}(n_2,p) \sim \text{Bin}(n_1+n_2,p)$

Exercise (Straight-line fit)

Given a labelled dataset $(x_1, y_1), \dots, (x_n, y_n)$ consisting of pairs of numbers, fit the model

$$Y_i \sim a + b\, x_i + \text{Normal}(0, \sigma^2)$$

where σ is given and a and b are parameters to be estimated.



Model for a single observation:

$$Y_i \sim a + bx_i + N(0, \sigma^2) \sim N(a + bx_i, \sigma^2)$$

Likelihood of a single observation:

$$\Pr_Y(y\,;\,x, a, b, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(y-a-bx)^2/2\sigma^2}$$

Log likelihood of the dataset:

$$\log \Pr(y_1, \dots, y_n\,;\,x_1, \dots x_n, a, b, \sigma) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - a - bx_i)^2$$

Optimize over the unknown parameters:

DISCUSSION:
Why should we use Maximum Likelihood Estimation rather than other methods (e.g. the Method of Moments) to estimate parameters?

According to my "*underwear theory of modelling*", parameters and models are private things inside the head of the modeller, and it's indecent to expose it in public. What matters is the probability distribution we're proposing, not the parameters.

- Sometimes, the same model can be written with different parameters. For example, $N(\mu, \sigma^2)$ and $N(\mu, e^s)$ result in the same distribution, just with different parameters. Maximum Likelihood Estimation will find parameters that give the best-fitting model, regardless of how we've chosen to parameterize the model.
- Sometimes, a model's parameters are unidentifiable. For example, if our model is $N(a+b, \sigma^2)$, then it's impossible to distinguish $(a=1, b=2)$ from $(a=1.5, b=1.5)$, since they give the same distribution. (This is especially relevant in neural networks, where we don't care if one neuron does a task or another neuron does it, all we care about is how the neurons work together.) Maximum Likelihood Estimation simply returns an arbitrary choice that maximizes the likelihood; other estimation methods just give up.
- All the successful methods in machine learning use maximum likelihood estimation. If it's good enough for ChatGPT and DALL-E, it's good enough for us!

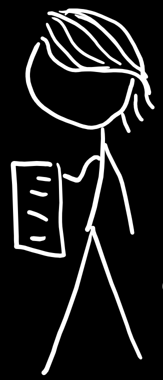
THE CONVENTIONAL VIEW OF MACHINE LEARNING

Data

image

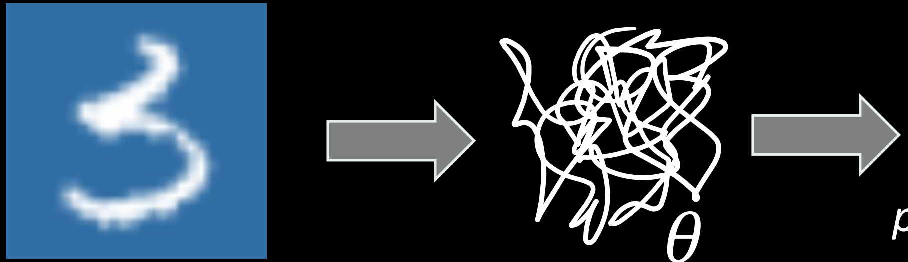
3

true label



(from manual annotation)

ML



θ

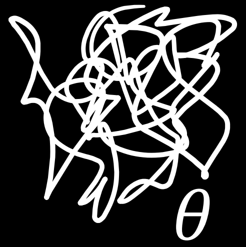
5

predicted label

Training — Measure the prediction accuracy, i.e. the fraction of images it predicts correctly
Tune the model's parameters (θ) to get good prediction accuracy on a training dataset

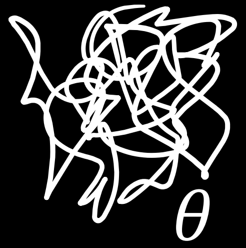
Evaluation — Evaluate your model's accuracy on a holdout set

CONVENTIONAL (ALGORITHMIC) VIEW

features →  → prediction

evaluation metric on holdout dataset:
prediction accuracy

PROBABILITY MODELLER'S VIEW

 →  →

$$\text{prediction} = \begin{cases} 5 \text{ with prob } 45\% \\ 3 \text{ with prob } 41\% \\ 6 \text{ with prob } \ \ 5\% \\ \quad \ldots \end{cases}$$

evaluation metric on holdout dataset:
the probability that our model assigns to the true label

Our job is to invent a probability model, specifying the **distribution** of temperature at a given timepoint.

PROBABILITY MODELLER'S VIEW