§1.3 Maximum likelihood estimation

All of machine learning is based on a single idea:

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

This is behind
- A-level statistics formulae
- our climate model
- ChatGPT training

i.e. estimate its parameters using Maximum Likelihood Estimation (mle)

§1

The model typically has unknown parameters.

The likelihood is the probability of seeing the data that we actually saw.

It depends on the 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

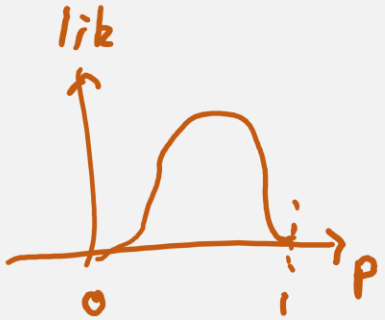
$$X \sim \text{Binom}(n, p)$$

where p is the probability of heads. Estimate p.

Likelihood of the observed data:

$$\text{lik} = P(X = x) \qquad x = 6$$

$$= \binom{n}{x} p^x (1-p)^{n-x}$$



Parameter that maximizes it:

$$\frac{d}{dp}\text{lik} = \binom{n}{x}\left[x p^{x-1}(1-p)^{n-x} - (n-x)p^x(1-p)^{n-x-1}\right]$$

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

There are standard numerical random variables that you should know:

DISCRETE RANDOM VARIABLES

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

CONTINUOUS RANDOM VARIABLES

Uniform $X \sim U[a, b]$	$\text{pdf}(x) = \frac{1}{b-a}$ $x \in [a, b]$	A uniformly-distributed floating point value
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
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

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:

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

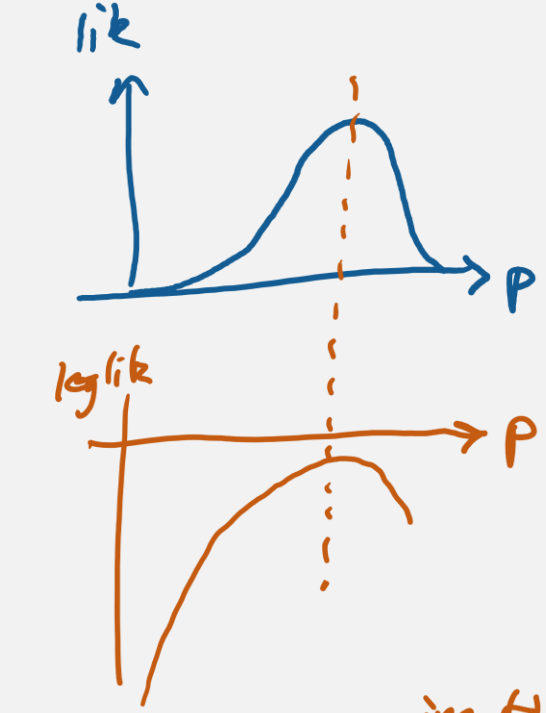
$$\log \text{lik} = \underbrace{\log \binom{n}{x}}_{\text{const}} + x \log p + (n-x) \log 1-p$$

[it depends on the data x but doesn't depend on p, so for the purposes finding the mle for p it's a constant]

Parameter that maximizes it:

$$\frac{d}{dp} \log \text{lik} = 0 + \frac{x}{p} - \frac{n-x}{1-p} = 0$$

$$\Rightarrow \hat{p} = \frac{x}{n}$$



log is an increasing function, so maximizing log lik(p) is the same as maximizing lik(p).

Exercise 1.3.6 (Handling boundaries)

We throw a k-sided dice, and get the answer x=10.
Estimate k, using the probability model

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

SANITY CHECK
Does our answer depend on the data? In the way we'd expect it to?

lik = $\mathbb{P}(X = x)$ where X is a random dice throw
$x = 10$ is the data

$= \frac{1}{k}$

maximized at $\hat{k} = 1$

But this is daft! How can it be that our estimate of the number of sides (k) doesn't depend on the value we saw (x)? And what even is a 1-sided dice?

Actually, our likelihood function is wrong, because it doesn't capture the constraint in the question, that $x \in \{1, \dots, k\}$ so $k \geq x$.

See lecture notes for a slick way to handle this.
See Ex. 1 q 3.
for a related mle question.

Example sheet 1

Learning with probability models
Data Science—DJW—2025/2026

Some of the questions ask for pseudocode. I suggest you implement your code, and test it using the online tester. There is a notebook with templates for answers and instructions for submission on the course materials webpage.

Question 1. Given a dataset $[x_1, \ldots, x_n]$, we wish to fit a Poisson distribution. This is a discrete random variable with a single parameter $\lambda > 0$, called the rate, and

$$\Pr(x\,;\,\lambda) = \frac{\lambda^x e^{-\lambda}}{x!} \quad \text{for } x \in \{0, 1, 2, \ldots\}.$$

Show that the maximum likelihood estimator for λ is $\hat{\lambda} = n^{-1} \sum_{i=1}^{n} x_i$.

Question 2. Give pseudocode to fit the model of question 1, using `scipy.optimize.fmin`. *[Optional.]* If you want to test your code using the online tester, fill in the answer template for `PoissonModel`.

In practice it
answer. But it's
problem where yo

Question 3. G
is unknown. Sho

Question 4 (A/
B. It has asked
from system A an
$\bar{x} = m^{-1} \sum_{i=1}^{m} x_i$
power of Machin

Suppose the x
$\delta, \sigma^2)$, and all the
estimators for th

Hints and comments

Question 1. This is a question about fitting parameters from a dataset, as in section 1.3. Formally it's unsupervised learning. There are more examples in section 1.7.

Question 2. See section 1.4. What parameter transform is needed here, to perform a maximization over the restricted domain $\lambda > 0$? How many maxima does the likelihood function have, and how might you choose a sensible starting point for the numerical optimization to make sure it finds a global maximum? Also, if you use numpy, watch out for which variables in your numpy code are vectors and which are scalars.

Question 3. This is another question about maximum likelihood estimation on datasets, also known as unsupervised learning, like question 1. You will also need to use the indicator function trick, from section 1.3 exercise 1.3.6.

§1.3 Maximum likelihood estimation

All of machine learning is based on a single idea:

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

This is behind
- A-level statistics formulae
- our climate model
- ChatGPT training

i.e. estimate its parameters using Maximum Likelihood Estimation (mle)

§1

The model typically has unknown parameters.

The likelihood is the probability of seeing the data that we actually saw.

It depends on the parameters.

Let's simply pick the parameters that maximize the likelihood!

§1.3 Maximum likelihood estimation

All of machine learning is based on a single idea:

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

This is behind
- A-level statistics formulae
- our climate model
- ChatGPT training

i.e. estimate its parameters using Maximum Likelihood Estimation (mle)

§1

The model typically has unknown parameters.

The likelihood is the probability of seeing the data that we actually saw.

If the data consists of many datapoints $[x_1, \dots, x_n]$ and our model says they're independent, then

$$\text{lik}(\text{data}) = \prod_{i=1}^{n} \text{lik}(x_i)$$

§1.3 Maximum likelihood estimation

All of machine learning is based on a single idea:

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

This is behind
- A-level statistics formulae
- our climate model
- ChatGPT training

i.e. estimate its parameters using Maximum Likelihood Estimation (mle)

§1

The model typically has unknown parameters.

The likelihood is the probability of seeing the data that we actually saw.

$\mathbb{P}(\text{data})$ if our model is a discrete rand.var.

pdf(data) if our model is a continuous rand.var.

If $X \sim N(\mu, \sigma^2)$ then $\mathbb{P}(X = x) = 0$ for any x.
This is why we have to use pdf rather than prob.

If the data consists of many datapoints $[x_1, \ldots, x_n]$ and our model says they're independent, then

$$\text{lik}(\text{data}) = \prod_{i=1}^{n} \text{lik}(x_i)$$

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.

WARNING
Watch out copy-paste-itis. Your likelihood must describe the data in the question!

Log likelihood of the observed data:

$$\text{lik}\begin{pmatrix}\text{data}\\ x_1 \ldots x_n\end{pmatrix} = \text{lik}(x_1) \times \cdots \times \text{lik}(x_n)$$

$$= (\lambda e^{-\lambda x_1}) \times \cdots \times (\lambda e^{-\lambda x_n})$$

$$= \lambda^n e^{-\lambda \sum x_i}$$

$$\log \text{lik} = n \log \lambda - \lambda \sum x_i$$

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:

$$\frac{d}{d\lambda} \log \text{lik} = \frac{n}{\lambda} - \sum_i x_i = 0 \Rightarrow \hat{\lambda} = \frac{n}{\sum x_i}$$

Exercise 1.3.8

Consider a dataset consisting of two collections of real numbers $x_1, \dots, x_n$ and $y_1, \dots, y_n$. Model the first collection as $\text{Normal}(\mu, \sigma^2)$ and the second as $\text{Normal}(\nu, \sigma^2)$, where μ, ν, σ are all unknown. Estimate σ.

Log likelihood of the observed data:

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

call this e_1 / call this e_2

$$= -\frac{2n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}(e_1+e_2)$$

SANITY CHECK
Does my likelihood use all the parameters and all the data?

Parameter that maximizes it:

$$\frac{d}{d\sigma} = 0 \Rightarrow -\frac{2n\sigma}{\sigma^2} + \frac{1}{\sigma^3}(e_1+e_2) = 0 \Rightarrow \hat{\sigma} = \sqrt{\frac{1}{2n}(e_1+e_2)}$$

depends on ν

depends on μ so this formula is useless as an estimator.

SANITY CHECK
Does my answer depend only on the data, or also on unknown parameters?

When there are multiple unknowns, we have to find them all, even if we're only interested in one.

$$\left.\frac{\partial}{\partial\mu}\text{loglik} = 0,\ \frac{\partial}{\partial\nu}\text{loglik} = 0,\ \frac{\partial}{\partial\sigma}\text{loglik} = 0\right\} \leftarrow \text{solve these simultaneously.}$$

Exercise 1.3.8

Consider a dataset consisting of two collections of real numbers, $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$. Model the first collection as $\text{Normal}(\mu, \sigma^2)$ and the second as $\text{Normal}(\nu, \sigma^2)$, where μ, ν, σ are all unknown. Estimate $\mu - \nu$.

When we fit μ, ν, σ we get $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i$. $\hat{\nu} = \frac{1}{n}\sum_{i=1}^{n} y_i$. $\hat{\sigma}$ = messy.

So our estimate for $\mu - \nu$ is $\hat{\mu} - \hat{\nu} = \frac{1}{n}\sum x_i - \frac{1}{n}\sum y_i$.

THE PLUG-IN PRINCIPLE

If the unknown parameters are θ and I want to estimate $\phi = f(\theta)$, then the mle for ϕ is $f(\hat{\theta})$

There are standard numerical random variables that you should know:

DISCRETE RANDOM VARIABLES

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

CONTINUOUS RANDOM VARIABLES

Uniform $X \sim U[a, b]$	$\text{pdf}(x) = \frac{1}{b-a}$ $x \in [a, b]$	A uniformly-distributed floating point value
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
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

There are standard numerical random variables that you should know:

Useful properties of the Normal distribution:

- If we rescale a Normal, we get a Normal (like the Uniform)

$a + bN(0,1) \sim a + N(0, b^2) \sim N(a, b^2)$ for constants a and b

- If we add independent Normals, we get a Normal (like the Poisson, binomial)

$N(\mu, \sigma^2) + N(\nu, \rho^2) \sim N(\mu+\nu, \sigma^2+\rho^2)$

These two properties are known as "linearity of the Normal distribution".

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

§1.3 Maximum likelihood estimation

All of machine learning is based on a single idea:

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

This is behind
- A-level statistics formulae
- our climate model
- ChatGPT training

i.e. estimate its parameters using Maximum Likelihood Estimation (mle)

§1

The model typically has unknown parameters.

The likelihood is the probability of seeing the data that we actually saw.

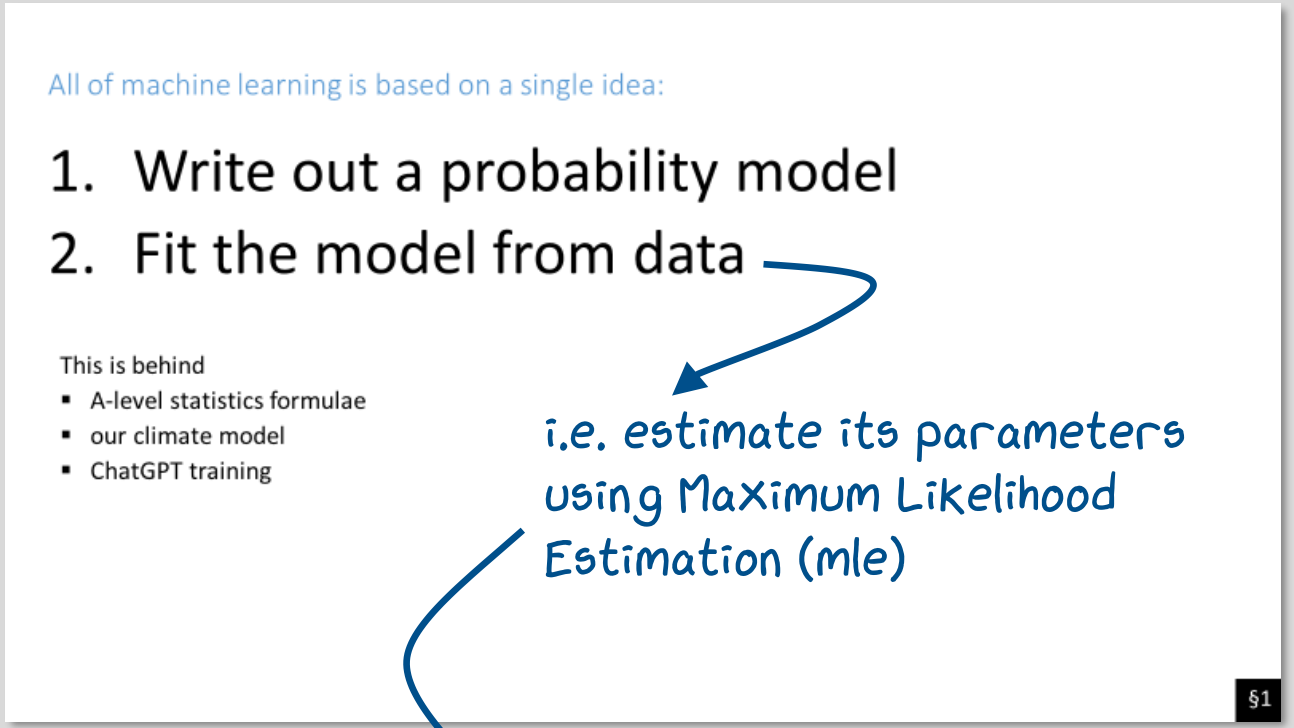
$\mathbb{P}(\text{data})$ if our model is a discrete rand.var.
$\text{pdf}(\text{data})$ if our model is a continuous rand.var.

If the data consists of many datapoints $[x_1, \ldots, x_n]$ and our model says they're independent, then

$$\text{lik}(\text{data}) = \prod_{i=1}^{n} \text{lik}(x_i)$$

When there are multiple unknown parameters, we must maximize over all of them simultaneously (even if we're only interested in one).

§1.4 Numerical optimization



i.e. estimate its parameters using Maximum Likelihood Estimation (mle)

with numerical optimization

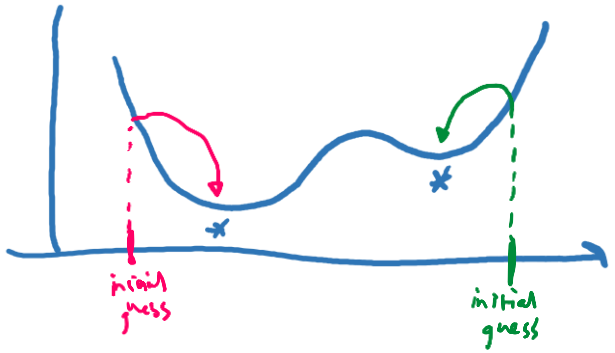
(since the likelihood function is usually far too complex for exact optimization)

a local minimum

Numerical optimization with Python / scipy

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

```
1  import scipy.optimize
2
3  def f(x):
4      return …
5
6  x₀ = […]  # initial guess
7  x̂ = scipy.optimize.fmin(f, x₀)
```

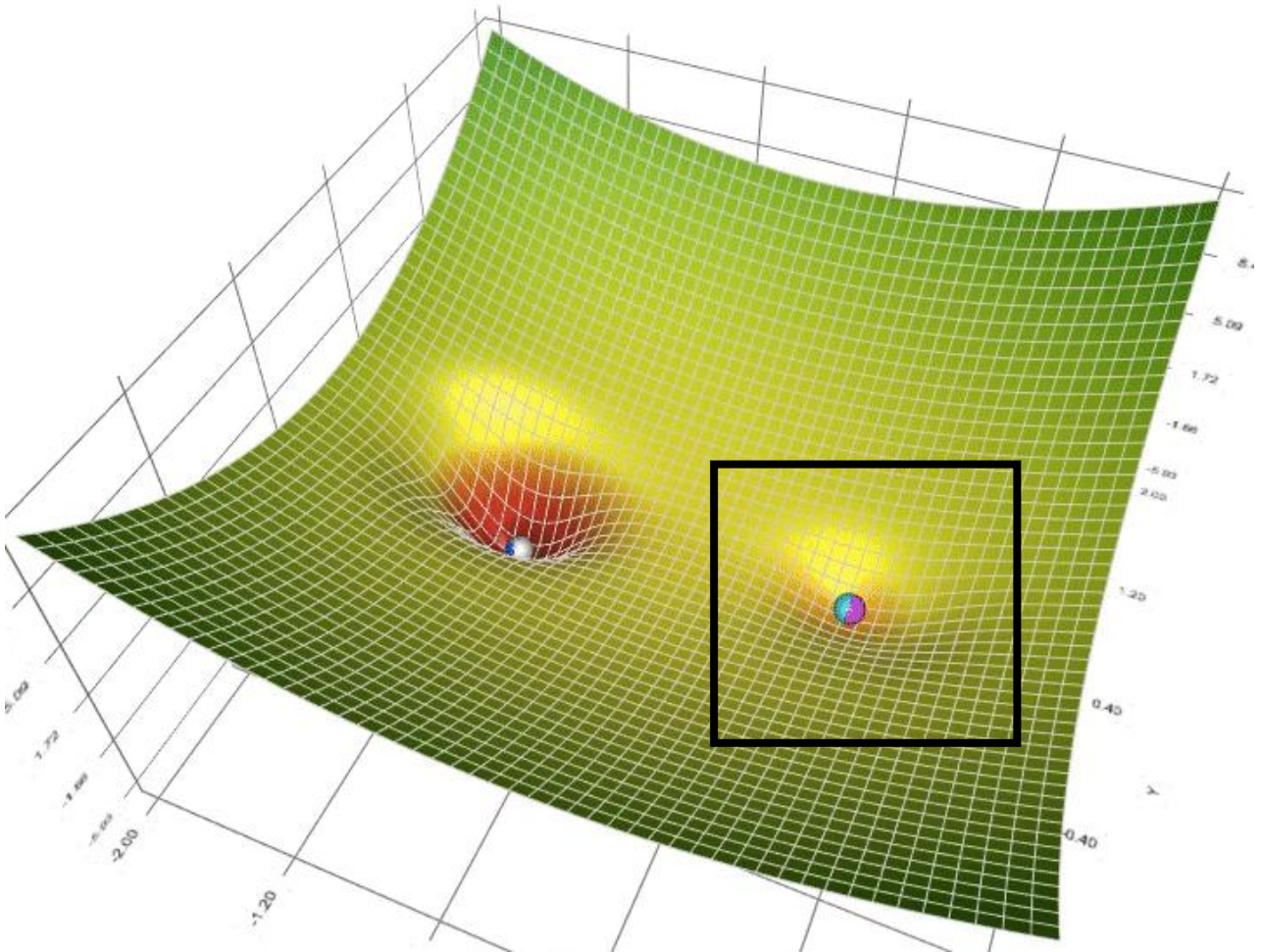


The initial guess will influence which local minimum the fmin ends up finding.

† There is no `scipy.optimize.fmax`. To maximize f, `scipy.optimize.fmin(lambda x: -f(x), x₀)`

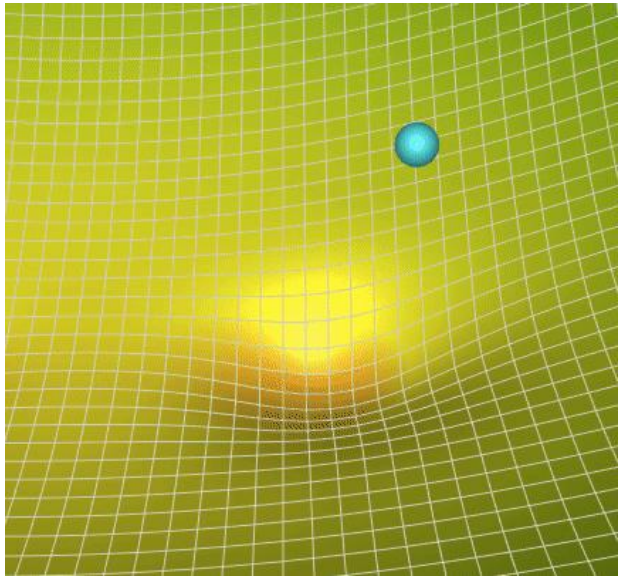
How does numerical optimization work?

Animations by Lili Jiang, [Towards Data Science]



GRADIENT DESCENT

Find the gradient of the function, and take a step in the direction of steepest descent



Exercise 1.4.2 (Constraints / 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 $p_1 + p_2 + p_3 = 1$.

fmin is not able to handle constraints.

Cunning trick: instead of maximizing over $p_1, p_2, p_3 \in (0,1)$ s.t. $p_1 + p_2 + p_3 = 1$

let's maximize over $(s_1, s_2, s_3) \in \mathbb{R}^3$, and set $p_i = \frac{e^{s_i}}{e^{s_1} + e^{s_2} + e^{s_3}}$.

This parameter transformation trick works because (i) every (s_1, s_2, s_3) yields a valid p, and (ii) every valid (p_1, p_2, p_3) can be achieved by some s.

```
1   def f(p):
2       p₁,p₂,p₃ = p
3       return 0.2*np.log(p₁) + 0.5*np.log(p₂) + 0.3*np.log(p₃)
4
5   def softmax(s):
6       p = np.exp(s)
7       return p / np.sum(p)
8
9   ŝ = scipy.optimize.fmin(lambda s: -f(softmax(s)), [0,0,0])
10  ŝ = softmax(ŝ)
```

initial guess $p_1 = p_2 = p_3 = \frac{1}{3}$

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

Department of Computer Scien

cl.cam.ac.uk/teaching/2526/DataSci/materials.html

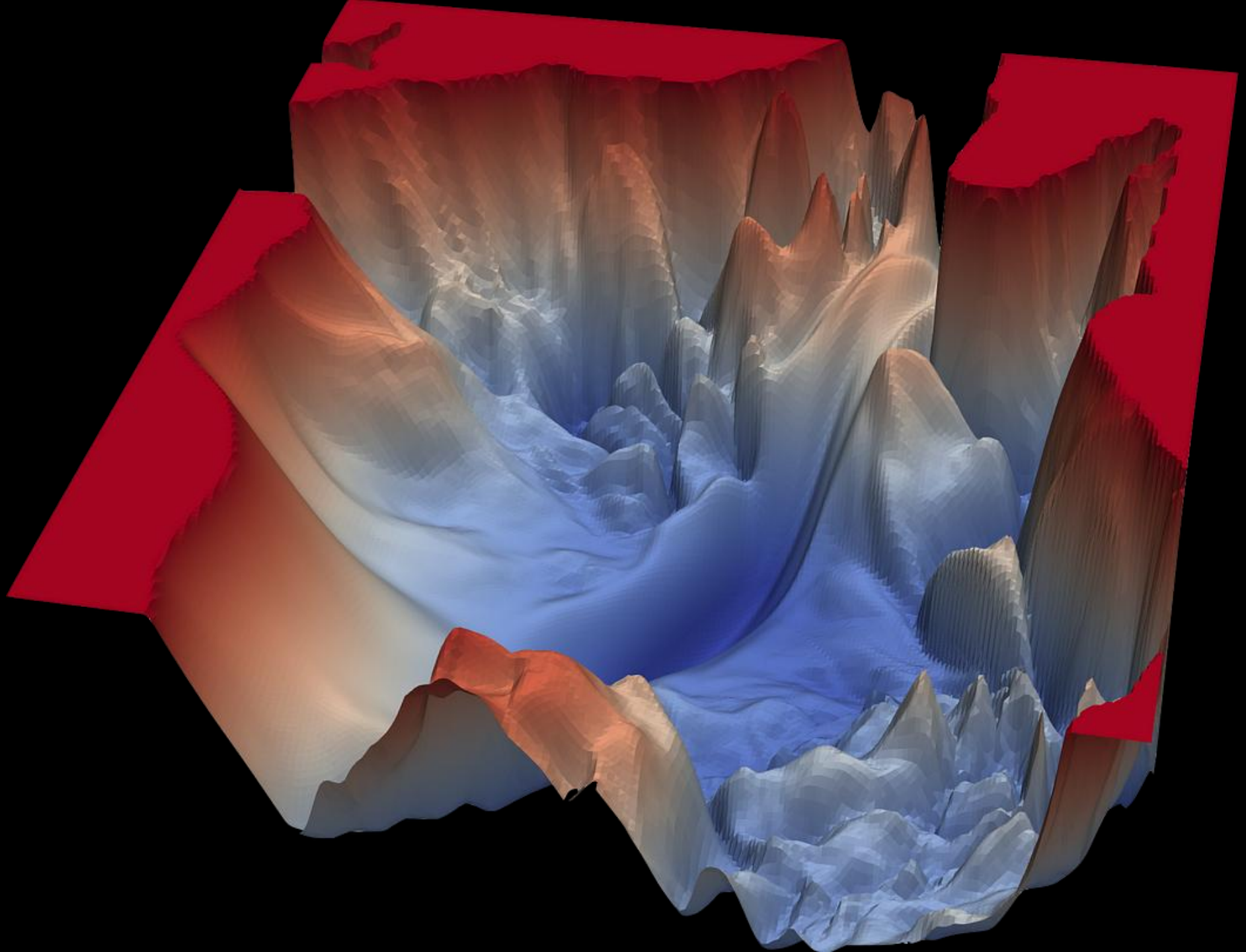
§1–§4. Learning with probability models

Lecture 1	1. Learning with probability models
[slides]	1.1 Specifying probability models
Lecture 2	1.2 Standard random variables
	1.3 Maximum likelihood estimation
	1.4 Numerical optimization with scipy
Lecture 3	1.5 Likelihood notation
	1.6 Types of model
	1.7 Supervised and unsupervised learning
Lecture 4	**Mock exam question 1**
	3.1, 3.2 Prediction accuracy *versus* probability modelling (* non-examinable)
	3.3 Neural networks (* non-examinable)
Lecture 5	2.1 Linear modelling
	2.2 Feature design
	2.3 Diagnosing a linear model
Lecture 6	2.5 The geometry of linear models
	2.6 Interpreting parameters
Lecture 7	2.4 Probabilistic linear modelling
	Code snippets: [fitting.ipynb] and [lm.ipynb]
	Example sheet 1
	OPTIONAL ex1 practical exercises [ex1.ipynb] (for supervisions)
	OPTIONAL PyTorch introduction and challenge
	OPTIONAL climate dataset challenge [climate.ipynb]
	Datasets investigated: [climate.ipynb], [stop-and-search.ipynb]

§5, §6, §8. Bayesian inference and Monte Carlo

Lecture 8	5.1 Bayes's rule for random variables
	5.2 Bayes's rule calculations
Lecture 9	6.1 Monte Carlo integration
	4.1 Measuring model fit (* non-examinable)

- CODE SNIPPETS from lectures
- CODING EXERCISES on example sheets



Visualizing the Loss Landscape of Neural Nets

Li, Xu, Taylor, Studer, Goldstein (2018)

https://arxiv.org/abs/1712.09913

Getting started with PyTorch

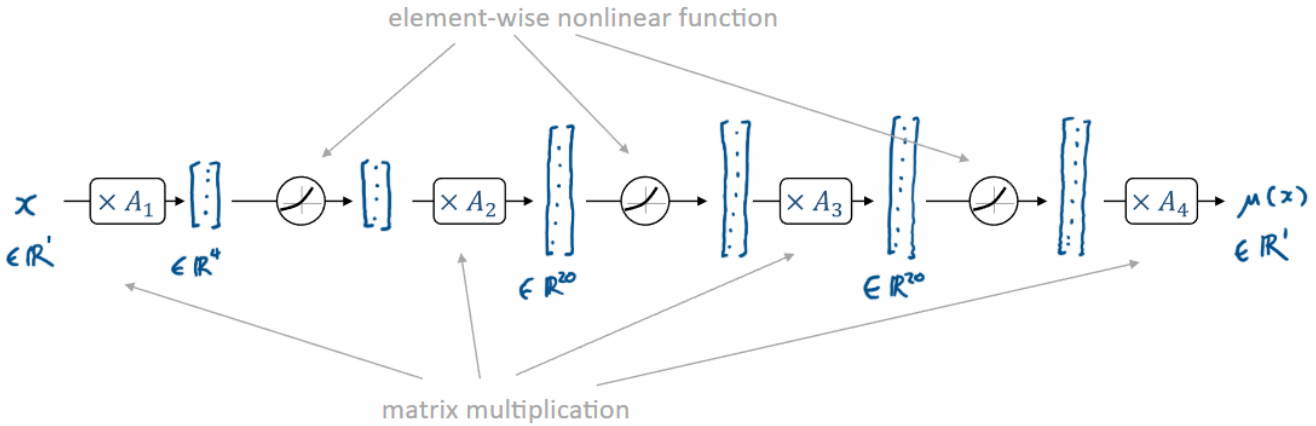
This is a brief introduction to PyTorch, designed to complement the IB Data Science course. It assumes you're familiar with the idea of maximum likelihood estimation. We'll use PyTorch to represent a probability model for regression, and fit it.

There are many other tutorials on PyTorch, including the tutorial in the official documentation. They typically present PyTorch as a software library, and go into much more depth on tensors and GPUs and so on. But they often don't give much guidance on how to use PyTorch for data science.

In this tutorial we'll work with the data behind the xkcd 2048 comic on curve-fitting. First, load the dataset:

STEP 7. Using a neural network

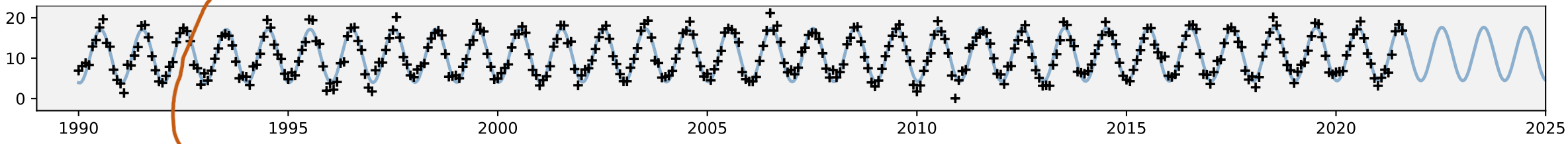
A neural network is just another function! We can swap out the $\mu(x) = a + bx + cx^2$ function and replace it by a neural network, i.e. a sequence of linear maps and nonlinear element-wise operations.



Exercise

The observed data is [temp$_1$,…,temp$_n$]. Estimate γ using the model

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



$\text{Temp}_i \sim N(\text{pred}_i, \sigma^2)$ where $\text{pred}_i = c + \alpha \sin(2\pi(t_i + \varphi)) + \gamma t_i$

by linearity of the Normal distribution: pred + $N(0,\sigma^2) \sim N(\text{pred},\sigma^2)$

$\text{lik(data)} = \text{lik}(\text{temp}_1) \times \dots \times \text{lik}(\text{temp}_n)$ assuming independence

since the likelihood of independent datapoints is a product

$$\text{log lik} = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum(\text{temp}_i - \text{pred}_i)^2.$$

by looking up the pdf, and avoiding copy-paste-itis

Exercise: maximize this numerically, over $c, \alpha, \gamma, \sigma, \varphi$

(Think about constraints: $\sigma > 0$)

(Think about initial guess)

$$Temp_i \sim N(pred_i, \sigma^2) \quad \text{where} \quad pred_i = c + \alpha \sin(2\pi(t_i + \phi)) + \gamma t_i$$

$$\log lik = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum(temp_i - pred_i)^2$$

```
url = 'https://www.cl.cam.ac.uk/teaching/current/DataSci/data/climate_202510.csv'
climate = pandas.read_csv(url)
climate['t'] = (climate.yyyy + (climate.mm-1)/12)
climate['temp'] = (climate.tmin + climate.tmax) / 2
df = climate.loc[(climate.station=='Cambridge') & (climate.yyyy>=1985)]


# Transform parameters: let σ = e^τ for τ ∈ ℝ

def loglik(θ,t,temp):
    (c,α,ϕ,γ,τ) = θ
    σ2 = np.exp(τ) ** 2
    pred = c + α * np.sin(2*π*(t+ϕ)) + γ*t
    n = len(t)
    return - n/2 * np.log(2*π*σ2) - 1/(2*σ2) * np.sum((temp - pred)**2)

init_guess = [12, 7, -0.2, 0, np.log(1.5)]
θhat = scipy.optimize.fmin(lambda p: -loglik(df.t,df.temp,θ), init_guess, maxiter=5000)

# View the fitted parameters, compared to initial guess
pandas.DataFrame({'par': ['c','α','ϕ','γ','τ','σ'],
                  'init_guess': list(init_guess) + [np.exp(init_guess[-1])],
                  'fitted': list(phat) + [np.exp(phat[-1])]})
```

The fitted value for c comes out to be −67.5, which is surprising compared to our guess of 12. But when we plot the fitted predictions, it looks right. Can you figure out why we got c = −67.5?