

All of machine learning is based on a single idea:

1. Write out a probability model *last lecture*
2. Fit the model from data *today*

This is behind

- A-level statistics formulae
- our climate model
- ChatGPT training

§1

3. Use the fitted model

- report its parameters (e.g. estimated rate of temperature increase, for our climate model)
- run it to get a single sampled output

What might this
be useful for?

> If I ask you the same question twice, will you give the same answer?

- Not necessarily. My outputs are sampled, so the same question can get differently worded answers, and sometimes different substance on open or borderline questions.

What tends to stay stable: factual answers with a clear right answer, code that has an obvious correct implementation, and anything where the context strongly constrains the response.

What tends to vary: phrasing, structure, examples, which of several valid approaches I pick, and judgment calls on ambiguous questions. On genuinely uncertain topics I may even reach different conclusions across runs.

Also, within one conversation a repeated question is not independent. My second answer sees the first one, so I will usually stay consistent with it, or explicitly revise it if you push back. That consistency comes from context, not from determinism.

Jump to bottom (ctrl+End) ↓

> █

Fable 5.1 | this turn 30.3k cached - 9k new - 288 out

▶▶ auto mode on (shift+tab to cycle) · ← for agents

> If I ask you the same question twice, will you give the same answer?

- Not necessarily. My output is sampled, so the same question can get a differently worded answer, and occasionally a substantively different one.

A few things shape how much it varies:

- Stable facts stay stable. If you ask what 17 times 23 is, you'll get 391 every time. Well-established facts and clear-cut reasoning converge.
- Wording varies even when substance doesn't. Explanations, examples, and structure can shift between runs.
- Genuinely uncertain or judgment-based questions vary most. Where there's no single right answer, different samples can land on different recommendations.
- Context matters. Asking the same question twice in one conversation isn't a clean repeat. I see the first answer, and I may rephrase, elaborate, or ask why you're asking again. In a fresh conversation, I have no memory of the first answer at all.

Jump to bottom (ctrl+End) ↓

>

Fable 5.1 | this turn 30.3k cached - 9k new - 332 out

▶▶ auto mode on (shift+tab to cycle) · ← for agents

What is a probability model?

- A piece of code that gives a different answer each time you run it
- A random variable

An LLM *is* a probability model.

- When we prompt x , we can think of its reply as a random variable Y whose distribution depends on x .
- Whenever we use it, we're taking a single sample of Y .

What's it a model *of*?

Fitting = tuning its 60B parameters to fit training data.

§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)

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:

$$\begin{aligned} \text{lik} &= \mathbb{P}(X=x) \quad \text{where } x=6 \\ &= \binom{n}{x} p^x (1-p)^{n-x} \quad \text{where } n=10 \end{aligned}$$



Parameter that maximizes it:

$$\begin{aligned} \frac{d}{dp} \text{lik} &= \binom{n}{x} \left[x p^{x-1} (1-p)^{n-x} - p^x (n-x) (1-p)^{n-x-1} \right] \\ \frac{d}{dp} \text{lik} = 0 &\Rightarrow \hat{p} = \frac{x}{n} \end{aligned}$$

There are standard numerical random variables that you should know:

DISCRETE RANDOM VARIABLES

Binomial $X \sim \text{Bin}(n, p)$	$\mathbb{P}(X = x) = \binom{n}{x} p^x (1 - p)^{n-x}$ $x \in \{0, 1, \dots, n\}$	For count data, e.g. number of heads in n coin tosses
Poisson $X \sim \text{Pois}(\lambda)$	$\mathbb{P}(X = x) = \frac{\lambda^x e^{-\lambda}}{x!}$ $x \in \{0, 1, \dots\}$	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 $X \sim U[a, b]$	$\text{pdf}(x) = \frac{1}{b - a}$ $x \in [a, b]$	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:

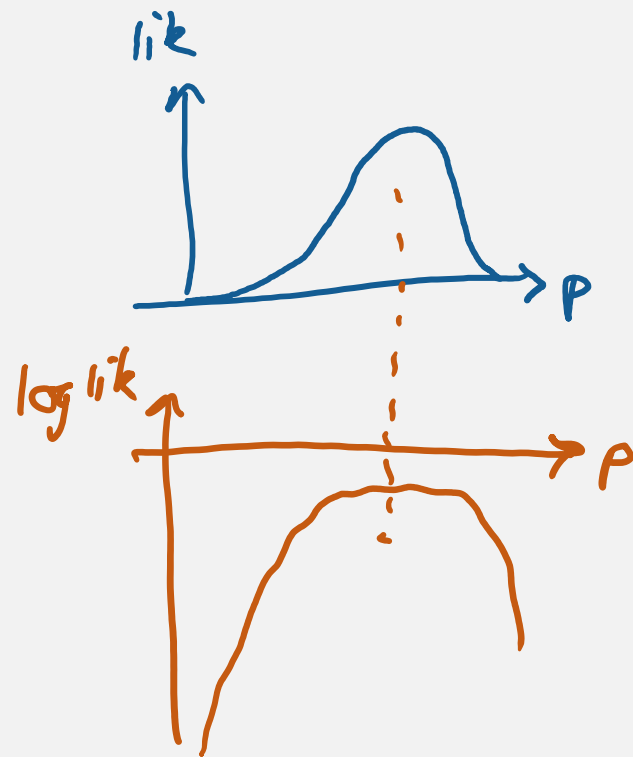
$$\text{lik} = P(X=x) = \binom{n}{x} p^x (1-p)^{n-x}$$

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

Parameter that maximizes it:

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

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



Exercise 1.3.6 (Handling boundaries)

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

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

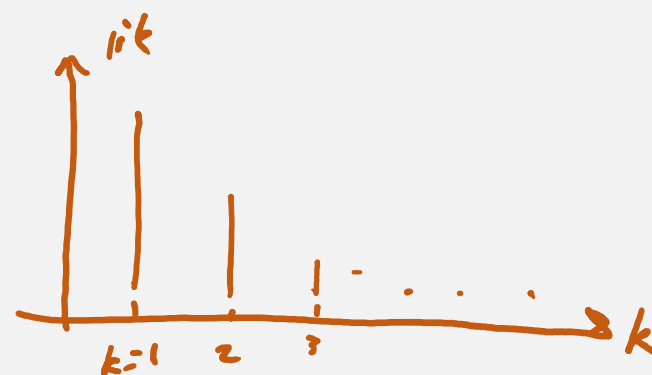


SANITY CHECK

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

lik = $\mathbb{P}(X=x)$ where $X = \text{random throw}$
 $x = 10$

$$= \frac{1}{k}$$



What's the mle?

$$\hat{k} = 1.$$

This is silly!

What went wrong: we got the likelihood wrong.

For details of how to deal with boundary conditions see lecture notes, exercise 1.3-6.

Example sheet 1

Learning with probability models

Data Science—DJW—2026/2027

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

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

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

Question 2. Give pseudocode for fitting a Poisson distribution to a dataset.
[Optional.] If you want to use the `PoissonModel` class, you need to implement the `fit` method.

In practice it'd be daft to implement a maximum likelihood estimator for the Poisson distribution. But it's good to get some practice with the problem where you what the

Question 3. Given a dataset $[x_1, \dots, x_n]$, show that the maximum likelihood estimator for λ is $\hat{\lambda} = n^{-1} \sum_{i=1}^n x_i$.

Question 4 (A/B testing). Suppose you have two systems, A and B. It has asked you to compare the performance of system A and (y₁, ..., y_n)

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)

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)

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)

The model typically has unknown parameters.

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

Let $X \sim N(\mu, \sigma^2)$. What's the
formula for $\mathbb{P}(X = x)$?

$\mathbb{P}(X=x) = 0$ for any x ,
since X is a cts r.v.

The likelihood is

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

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

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

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

Exercise 1.3.2 (Exponential sample)

Let the dataset be a list of real numbers, $x_1, \dots, x_n$, 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:

$$\text{lik}(\text{data}_{x_1, \dots, x_n}) = \prod_{i=1}^n \text{lik}(x_i) = \prod_{i=1}^n \lambda e^{-\lambda x_i} = \lambda^n e^{-\lambda \sum_i x_i}$$

$$\begin{aligned} \log \text{lik} &\approx \sum_{i=1}^n \log \text{lik}(x_i) \\ &= n \log \lambda - \lambda \sum_i x_i \end{aligned}$$

WARNING

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

CONTINUOUS RANDOM VARIABLES (real-valued)

Exponential

$X \sim \text{Exp}(\lambda)$

`np.random.exponential(scale=1/λ)`

pdf(x) = $\lambda e^{-\lambda x}$
 $x > 0$

Parameter that maximizes it:

$$\hat{\lambda} = \dots$$

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:

$$\log \text{lik}(\mu, \nu, \sigma) = -n \log(2\pi\sigma^2) - \frac{1}{2\sigma^2} \left(\sum_i \overbrace{(x_i - \mu)^2}^{e_1} + \sum_i \overbrace{(y_i - \nu)^2}^{e_2} \right)$$

SANITY CHECK

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

~~assuming that all readings are independent.~~
modelling all readings as independent.

Parameter that maximizes it:

$$\frac{d}{d\sigma} \log \text{lik} = -\frac{n 2\pi 2\sigma}{\sigma^2} + \frac{1}{2\sigma^3} (e_1 + e_2)$$

$$\Rightarrow \hat{\sigma} = \sqrt{\frac{1}{4\pi} (e_1 + e_2)}$$

← This is no use! e_1 depends on μ , e_2 depends on ν , μ and ν are unknown, so my $\hat{\sigma}$ is useless.

SANITY CHECK

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

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

$$\frac{\partial}{\partial \mu} \log \text{lik} = 0$$

$$\frac{\partial}{\partial \nu} \log \text{lik} = 0$$

$$\frac{\partial}{\partial \sigma} \log \text{lik} = 0.$$

— we have to solve these three equations simultaneously.

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 $\mu - \nu$.

When we fit (μ, ν, σ) we get $\hat{\mu} = \frac{1}{n} \sum_i x_i$ and $\hat{\nu} = \frac{1}{n} \sum_i y_i$ and $\hat{\sigma} = \text{messy}$

These are the most likely (μ, ν, σ) out of all possibilities.

^a So the most likely value of $\mu - \nu$ is $\hat{\mu} - \hat{\nu}$

since any other value would need different, and less likely, (μ, ν, σ) .

no more likely

For this specific model, we can look at the loglik and persuade ourselves that $(\hat{\mu}, \hat{\nu}, \hat{\sigma})$ are unique. For other probability models, there may be many solutions to the likelihood maximization.

WARNING

Remember there might be multiple maxima (all with the same likelihood)

THE PLUG-IN PRINCIPLE

To estimate a derived quantity, estimate all the parameters first and then plug them into the definition of that quantity.

§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)

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 $[x_1, \dots, x_n]$
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).

Watch out for ties.

§1.4 Numerical optimization

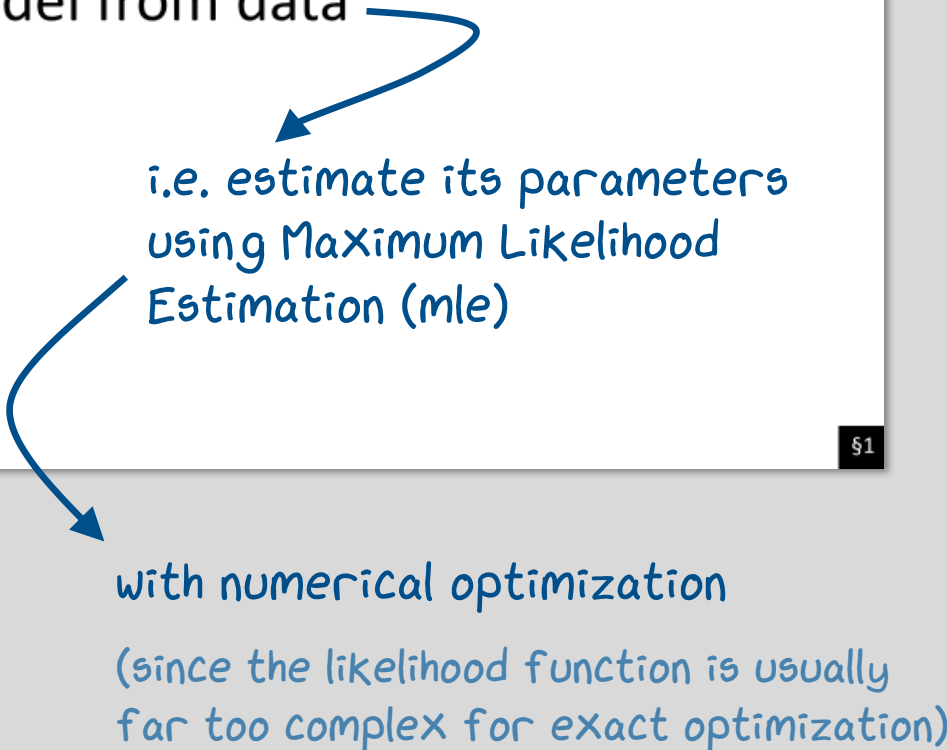
All of machine learning is based on a single idea:

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)



with numerical optimization

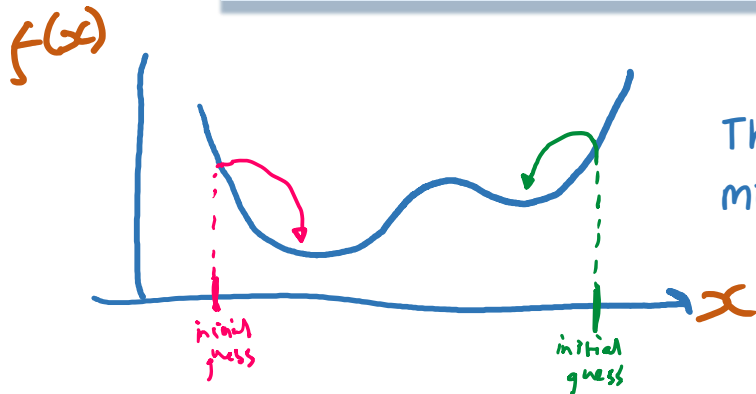
(since the likelihood function is usually
far too complex for exact optimization)

a local minimum

Numerical optimization with Python / scipy

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

```
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)`

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 $p_1 + p_2 + p_3 = 1$.

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 $p_i = \frac{e^{s_i}}{e^{s_1} + e^{s_2} + e^{s_3}}$

- Every (s_1, s_2, s_3) yields a valid (p_1, p_2, p_3) .
- Every valid (p_1, p_2, p_3) can be achieved by some $(s_1, s_2, s_3) \in \mathbb{R}^3$.

```
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
```

Why do you think it's called softmax?

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

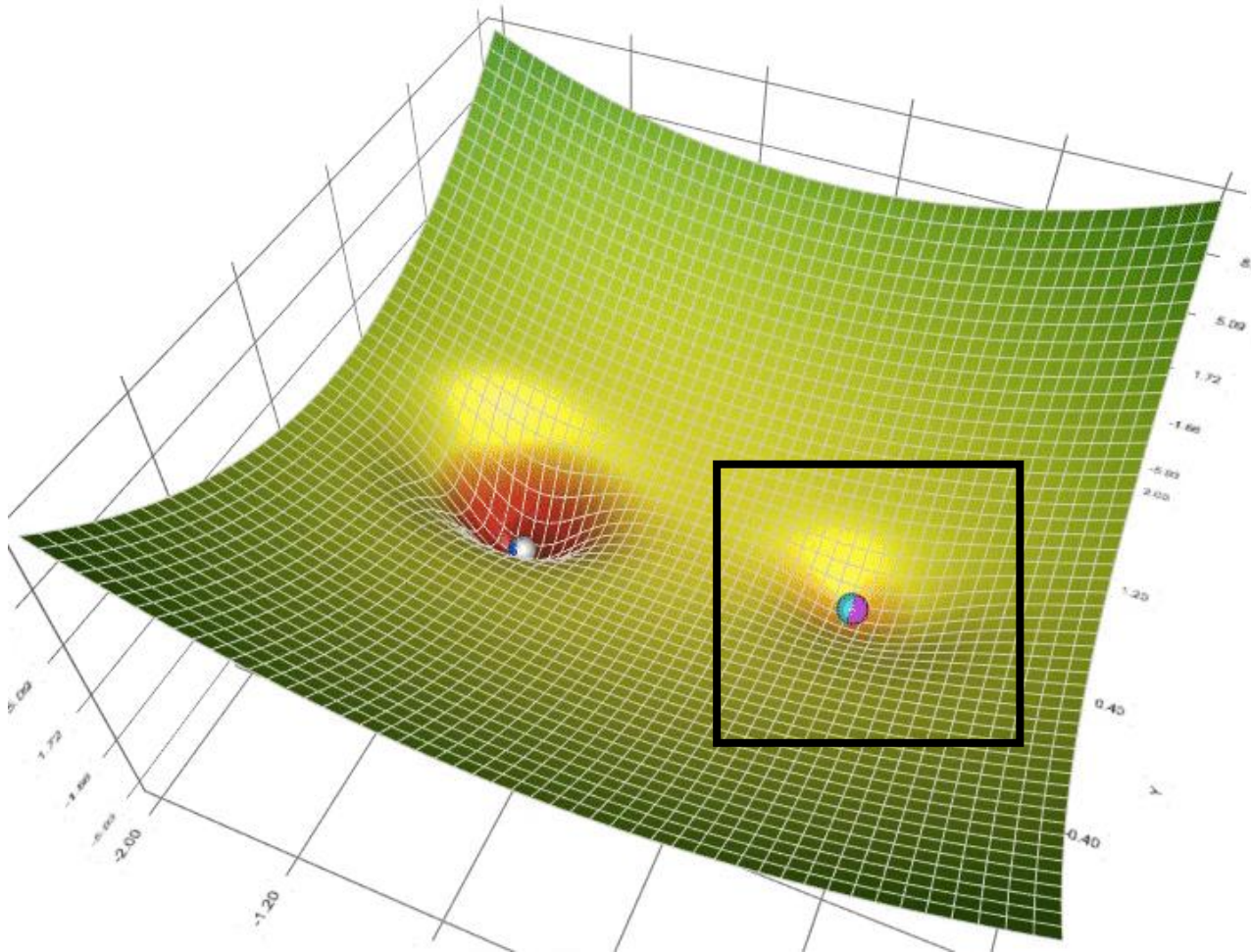
Optimization terminated successfully. Current function value: 1.02965. Iterations: 63.

Function evaluations: 120

array([0.19999474, 0.49999912, 0.30000614])

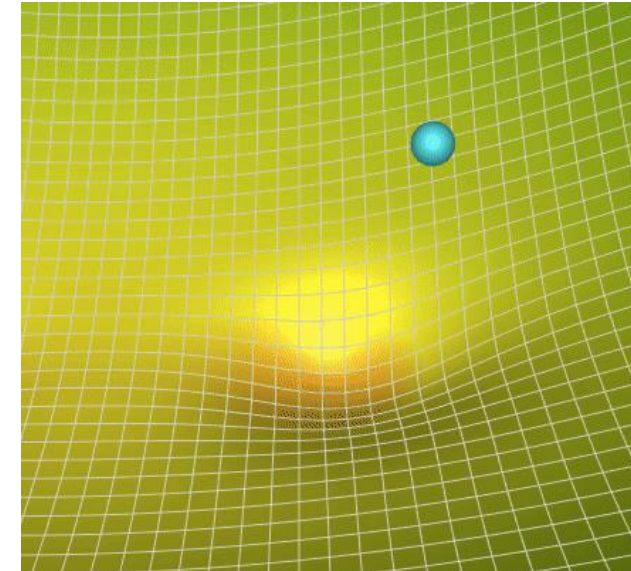
How does numerical optimization work?

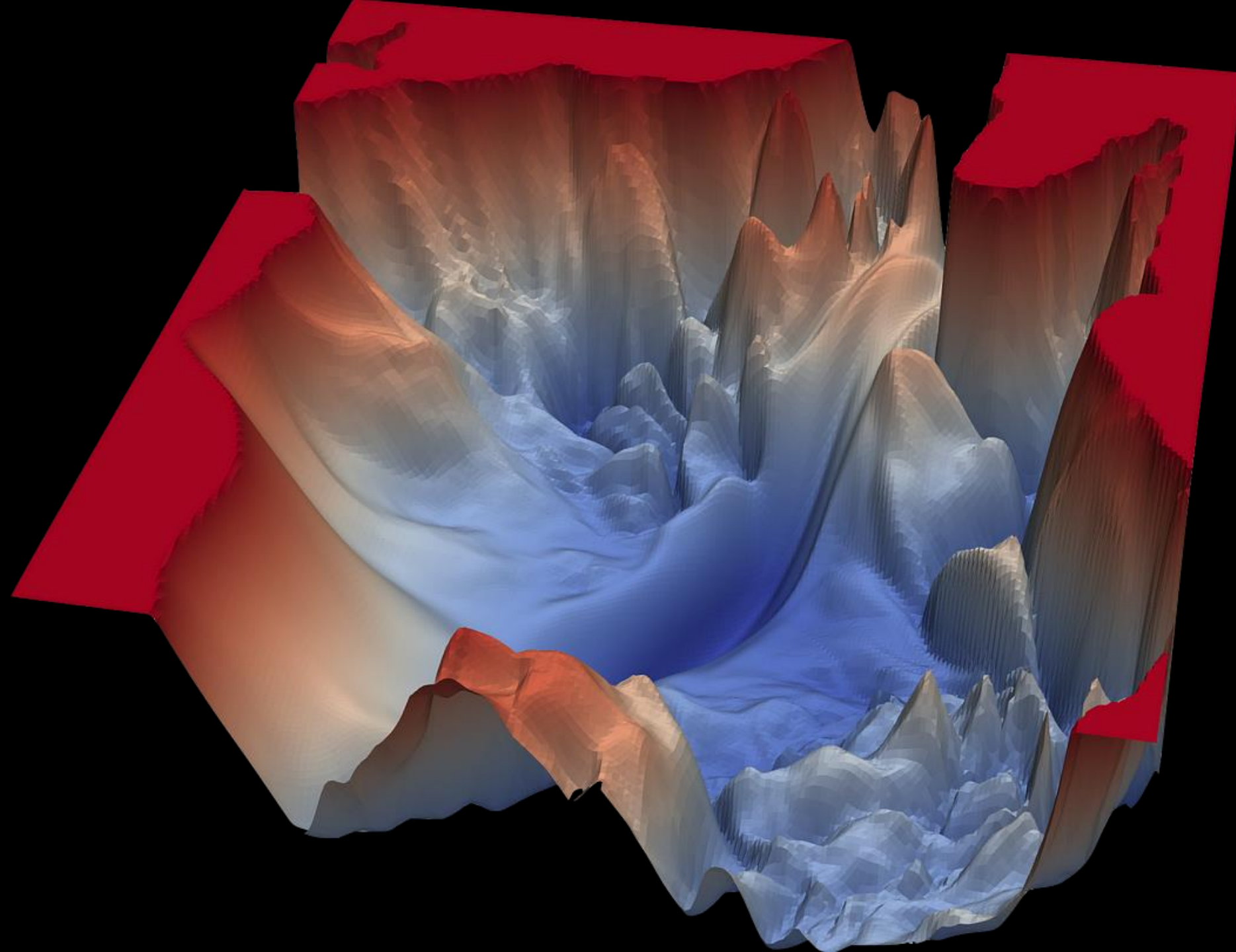
Animations by Lili Jiang, [Towards Data Science](#)



GRADIENT DESCENT

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





*Visualizing the Loss
Landscape of Neural
Nets*

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

[https://arxiv.org/abs/
1712.09913](https://arxiv.org/abs/1712.09913)

Department of Computer Science

Ask Gemini

cl.cam.ac.uk/teaching/2627/DataSci/materials.ht...

Lecture schedule

This is the planned lecture schedule. It will be updated as and when actual lectures deviate from schedule. Material marked * is non-examinable. **Slides** are uploaded the night before a lecture, and re-uploaded after the lecture with annotations made during the lecture.

Prerequisites

[Example sheet 0](#) and [solutions](#)
[NumPy tutorial](#)

§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. Neural networks as probability models (* non-examinable)
Lecture 5	2.1 Linear modelling
	2.2 Feature design
Lecture 6	2.3 Diagnosing a linear model
	2.5 The geometry of linear models
Lecture 7	2.6 Interpreting parameters
Lecture 8	2.4 Probabilistic linear modelling
	Discussion of climate dataset challenge (* non-examinable)
	Example sheet 1
	OPTIONAL ex1 practical exercises [ex1.ipynb] (for supervisions)
	OPTIONAL PyTorch introduction
	Datasets investigated: [climate.ipynb]

- The example sheet has (optional) coding exercises.

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.

