§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

i.e. estimate the parameters using Maximum Likelihood Estimation (MLE)

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

typically with 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

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

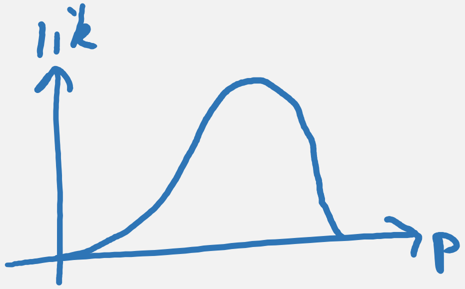
where p is the probability of heads. Estimate p.

Likelihood of the observed data:

$$\begin{aligned} lik &= P(X = x) \\ &= \binom{n}{x} p^x (1-p)^{n-x} \end{aligned}$$

Parameter that maximizes it:

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



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

There are standard numerical random variables that you should know:

DISCRETE RANDOM VARIABLES

Binomial $X \sim \text{Bin}(n, p)$	$\mathbb{P}(X=x) = \binom{n}{x} p^x (1-p)^{n-x}$ $x \in \{0,1,\dots,n\}$	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

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

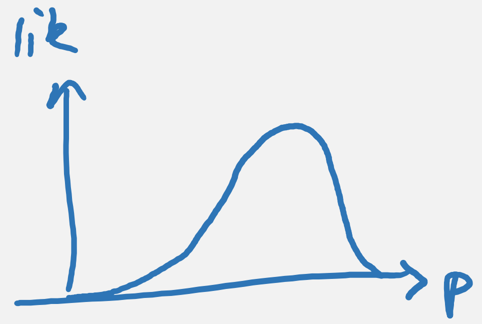
$$X \sim \mathrm{Binom}(n, p)$$

where p is the probability of heads. Estimate p.

Log likelihood of the observed data:

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

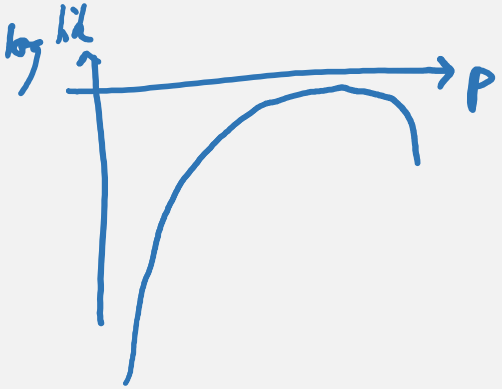
$$\log \text{lik} = \log \binom{n}{x} + x \log p + (n-x) \log (1-p)$$



Parameter that maximizes it:

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

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



Exercise 1.3.6 (Handling boundaries)

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

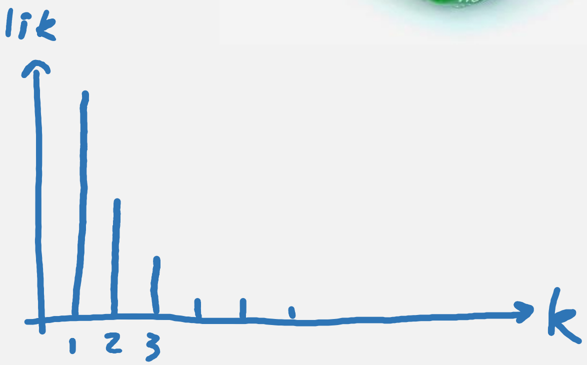
$$\mathbb{P}(\text{throw } x) = \frac{1}{k}, \qquad x \in \{1, \dots, k\}$$

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

$$\cancel{\text{lik} = \mathbb{P}(\text{throw } x) = \frac{1}{k}}$$

$\hat{k} = 1$

SILLY



$$\text{lik} = \mathbb{P}(\text{throw } x) = \begin{cases} \frac{1}{k} & \text{if } x \le k \\ 0 & \text{if } x > k \end{cases} = \frac{1}{k} 1_{x \le k}$$

$$= \frac{1}{k} 1_{k \ge x}$$

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

INDICATOR FUNCTIONS

The indicator function 1_A is simply

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

§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

typically with unknown parameters

i.e. estimate the parameters using Maximum Likelihood Estimation (MLE)

This is behind

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

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

$$\text{lik}(\text{data}) = \text{lik}(x_1) \times \text{lik}(x_2) \times \cdots \times \text{lik}(x_n)$$

It depends on the parameters.

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

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 $\mathrm{Exp}(\lambda)$ random variables, where λ is unknown. Estimate λ.

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

$\mathbb{P}(X = x_1) = 0$

$\mathrm{pdf}(x_1) = \lambda e^{-\lambda x_1}$

Log likelihood of the observed data:

$$\mathrm{lik}(\underset{x_1,\ldots x_n}{\mathrm{data}}) = \mathrm{lik}(x_1) \times \cdots \times \mathrm{lik}(x_n)$$

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

$$= \lambda^n e^{-\lambda \sum_{i=1}^{n} x_i}$$

$$\mathrm{loglik} = n \log \lambda - \lambda \sum_i x_i$$

CONTINUOUS RANDOM VARIABLES (real-valued)

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

Parameter that maximizes it:

$$\frac{d}{d\lambda} \mathrm{loglik} = \frac{n}{\lambda} - \sum_i x_i = 0 \quad \Rightarrow \quad \hat{\lambda} = \frac{n}{\sum_i x_i}$$

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.

Note: the question doesn't tell us the values of α, γ, σ, so we'll treat them as unknowns to be estimated.

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

solve these simultaneously.

§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

typically with unknown parameters

i.e. estimate the parameters using Maximum Likelihood Estimation (MLE)

This is behind

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

§1

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

$\mathbb{P}$(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 $[x_1, \cdots, x_n]$, and our model says they're independent,

$$\text{lik}(\text{data}) = \text{lik}(x_1) \times \text{lik}(x_2) \times \cdots \times \text{lik}(x_n)$$

It depends on the parameters.

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

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

Exercise 1.3.4 (Predictive models)

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

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

estimate γ, the annual rate of temperature change.

What would happen if we just solved one equation, for the parameter we're interested in?

$$\frac{d}{d\gamma} \log \mathrm{lik} = 0$$

We get the answer

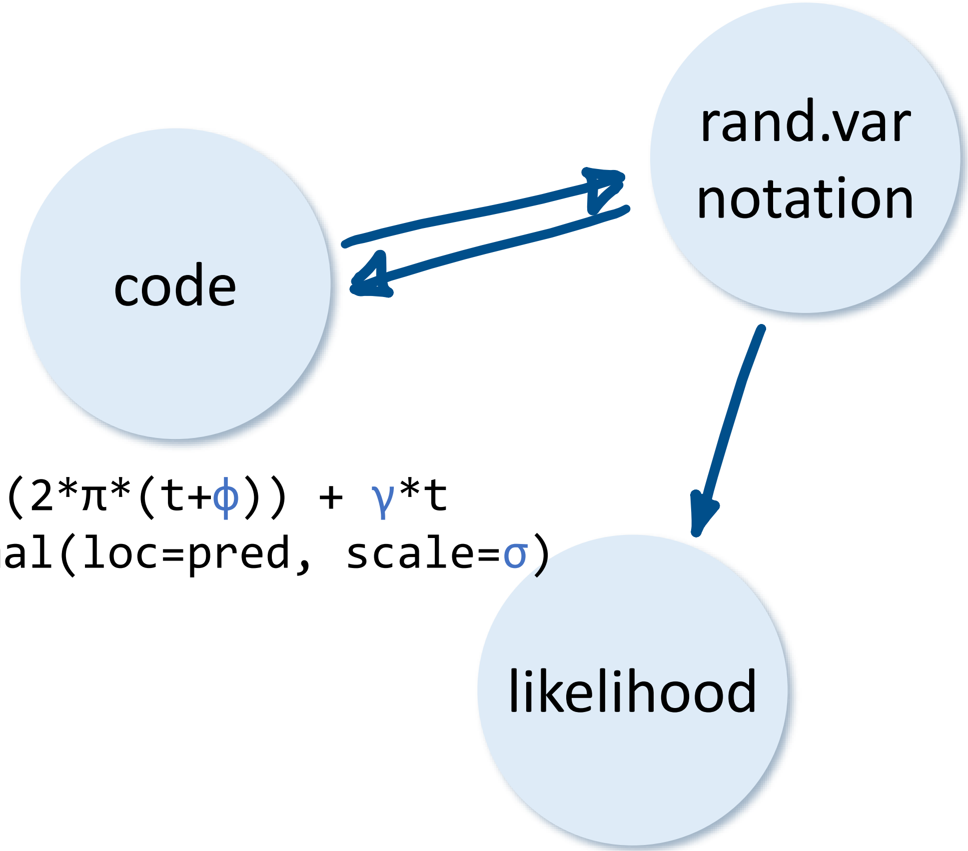
$$\hat{\gamma} = \frac{\Sigma_i t_i (y_i - \alpha)}{\Sigma_i t_i^2}$$

Useless — if involves α, whose value I don't know.

SANITY CHECK
Does our answer depend on unknown parameters?

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)
```

$\text{Temp}_i \sim N(\text{pred}_i, \sigma^2)$

where $\text{pred}_i = \alpha \sin(2\pi(t_i + \varphi))$
$+ c + \gamma t_i$

Exercise

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

The observed data is [temp_1,…,temp_n]. Find an expression for the log likelihood.

$\text{lik}(\text{data}) = \text{lik}(\text{temp}_1) \times \cdots \times \text{lik}(\text{temp}_n)$

$= \left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{-\underline{(\text{temp}_1 - \text{pred}_1)}^2/2\sigma^2} \right) \times \cdots$

Watch out for copy-paste-itis! We want the likelihood of seeing temp1, for the random variable $\text{Temp}_1 \sim N(\text{pred}_1, \sigma^2)$. Don't just paste in the formula from the random variable reference sheet,

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

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

There are 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

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

$a + \underline{b}\, N(0,1) \sim a + N(0, b^2) \sim N(a, b^2)$

for constants a and b

- If we add independent Normals, we get a Normal $\sim$

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

assuming the two Normals are independent.

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

§1.4 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

using maximum likelihood estimation with numerical optimization

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

Numerical optimization with Python / scipy

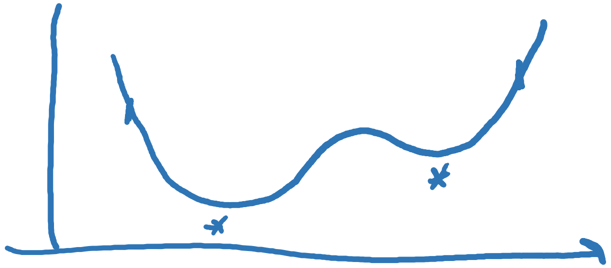
a local minimum

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

```
import scipy.optimize

def f(x):
      return …

x₀ = […]  # initial guess
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

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 $p_1 + p_2 + p_3 = 1.$

Cunning trick:

instead of finding max over (p_1, p_2, p_3) such that $p_1 + p_2 + p_3 = 1$,

we'll instead find max 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])
ŝ = softmax(ŝ)
```

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

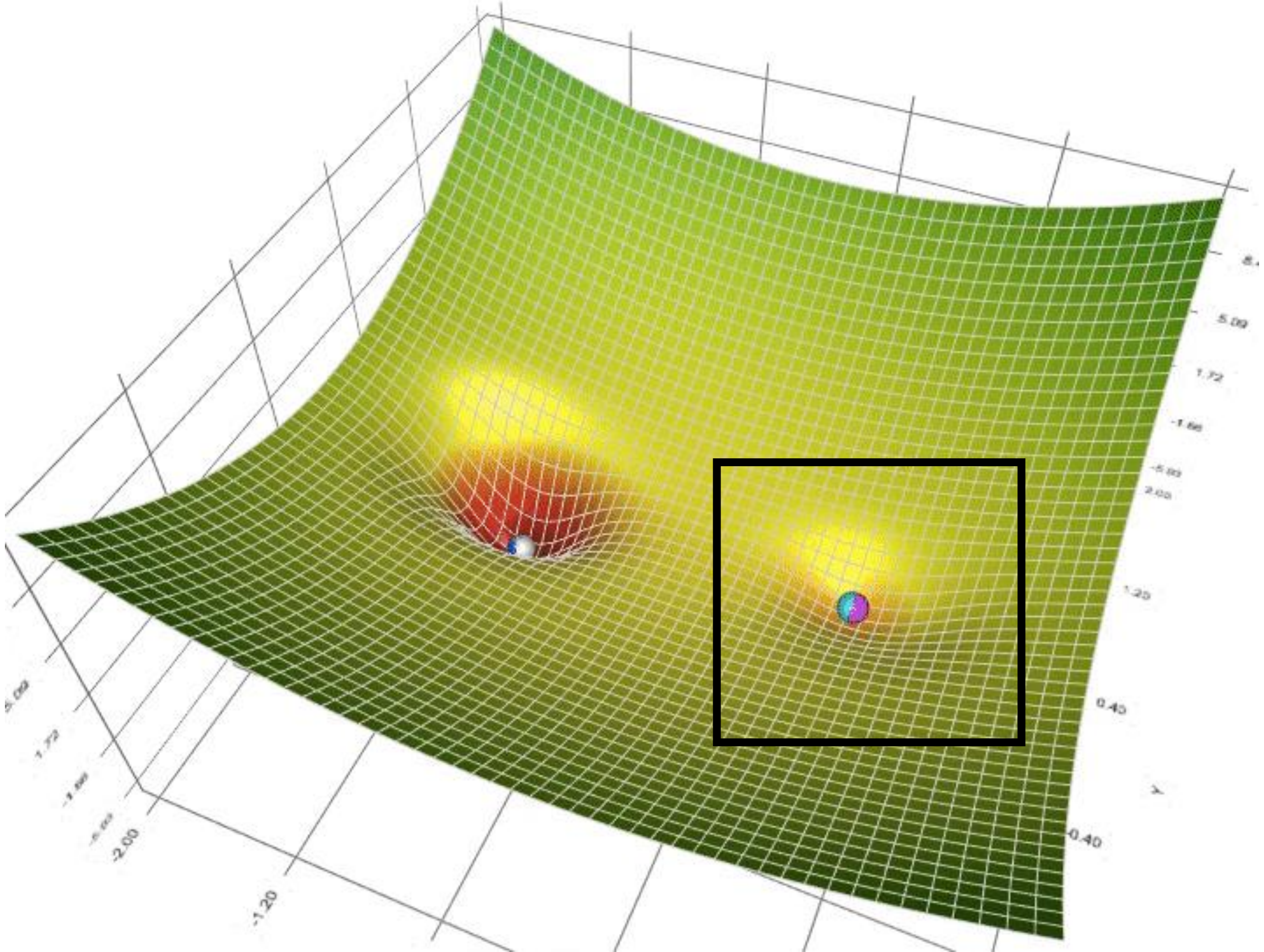
Exercise 1.4.1 (Positivity constraint)

Find the maximum over $\sigma > 0$ of

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

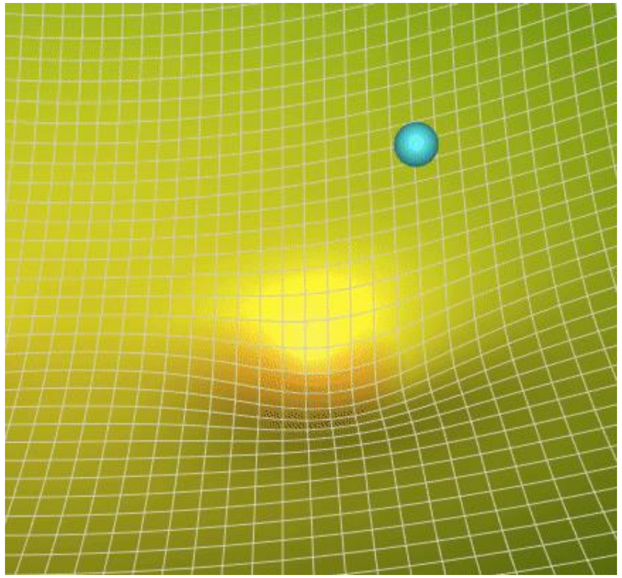
How does it work?

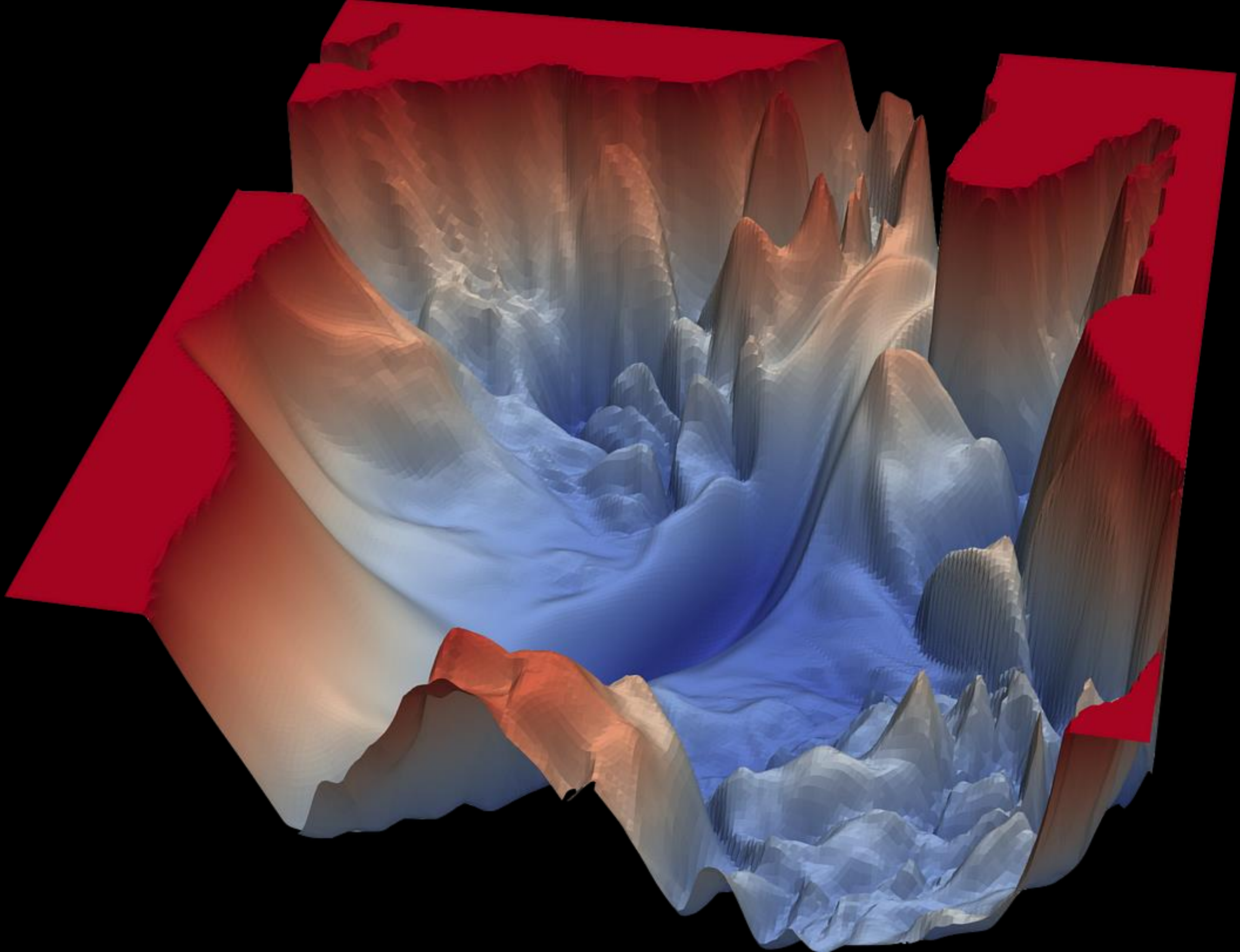
Animations by Lili Jiang, [Towards Data Science]



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

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.