PART IV:
Probability models for sequences

§1 ChatGPT

Exercise (Preference modelling)

We have a dataset of tennis matches. (It's not a tournament—some pairs have more than one match.)

We wish to assign a skill θ_y to each player y. Propose a probability model for the winner of a match Z based on the skills of the two players, and explain how to fit your model.

player y_0	player y_1	winner z
Anisimova	Swiatek	1
Sabalenka	Anisimova	1
Swiatek	Bencic	0
Andreeva	Bencic	1
Anisimova	Swiatek	0
⋮		

θ_{Ani} θ_{Swi} θ_{Ben}

Sanity check: if $\theta_{y_1} = \theta_{y_0}$, $\mathbb{P}(Z=1) = \frac{1}{2}$

$$\mathbb{P}(Z=1;\ \theta_{y_0}, \theta_{y_1}) = \mathbb{P}\left[\underbrace{N(\theta_{y_1}, 1)}_{\text{match-day performance}} > N(\theta_{y_0}, 1)\right] = \mathbb{P}\left[N(\theta_{y_1} - \theta_{y_0}, 2) > 0\right]$$

(θ_{y_1}: intrinsic skill)

This is the basis of TrueSkill, Microsoft's system for scoring players in team games on XBOX

$$\mathbb{P}(Z=1;\ \theta_{y_0}, \theta_{y_1}) = \frac{e^{\theta_{y_1}}}{e^{\theta_{y_0}} + e^{\theta_{y_1}}}$$

$$Z \sim \text{Cat}\left(\text{softmax}([\theta_{y_0}, \theta_{y_1}])\right)$$

This is the basis of Elo (the chess-ranking system), and of RLHF (aligning LLMs)

A piece of text $\underline{x}$ is a sequence of tokens from a finite alphabet.

How might we generate a random piece of text $\underline{X}$ that looks like English?

With a transformer trained by MLE

Why might this be interesting?

Mathematically, we can sample from the conditional distribution $\Pr_{\underline{X}}(\underline{x} \mid \underline{X} \text{ begins } \underline{x}_0)$.
Practically, think of this as a "completion API": given a prompt $\underline{x}_0$ it generates a plausible completion.

perfect pasta sauce ␣

- Nonna Ginny's biggest secret to making **the perfect sauce** is to cook with love
- **Perfect Pasta Sauce.** If you say "Marcella Hazan's pasta sauce" although she has many, this is the one everyone
- so flavorful this time of year they melt into the **perfect pasta sauce**. Add fresh basil and a drizzle of olive oil
- Secret Ingredient Italians Add For **Perfect Pasta Sauce**. I wasn't ready to hear this.
- Making the **perfect pasta sauce** is a skill that can be mastered with practice and the right ingredients

When I asked my nonna how to make the perfect pasta sauce, she gave me this recipe: ␣

A piece of text $\underline{x}$ is a sequence of tokens from a finite alphabet.

How might we generate a random piece of text $\underline{X}$ that looks like English?

With a transformer trained by MLE

Why might this be interesting? It can generate 'completions'

How can we make it *reply?*

perfect pasta sauce

Here's a perfect, dependable, weeknight-easy yet dinner-party-worthy pasta sauce. It's the one Italian nonnas and chefs quietly agree on: Marcella Hazan's tomato–butter sauce, plus a couple of optional refinements […]

OpenAI devised "Reinforcement Learning from Human Feedback" (RLHF) to get useful replies. This led to ChatGPT.

RLHF step 1. Learn what humans want a reply to look like

Let's gather pairwise-comparison data for preferences

prompt x	reply y_0	reply y_0	preferred
write me a story about a fox named louis who one day goes down a alley in a poor, crime filled part of tokyo and gets mugged by a dragon	Once upon a time in the bustling metropolis of Tokyo, there lived a sly and cunning fox named Louis. Known for his sharp wit and mischievous nature, Louis had managed to thrive in the city's diverse landscapes.	In the dimly lit alleys of Tokyo's red-light district, a sleek and agile fox named Louis navigated through the crowded streets with ease. His rusty orange fur blended perfectly with the neon lights	1
can you role play as the cmd.exe of an ancient alien ai astromancy computer system called Omni-Seer?	Astromancy Computer System, Version 907.14.12.31 Copyright (c) Alien Technocrats Type 'help' for a list of commands. ** Omni-Seer ** What would you like to do?	Certainly, but remember that I am Qwen, trained by Alibaba Cloud, and not a real cmd.exe from an ancient alien AI system called Omni-Seer. Greetings, enigmatic traveler. In the vast expanses of Omicronia, where stars whisper secrets to those who listen, the Omicron Seer commands the forces of the cosmos through the ancient language of data, the syntax of logic, and the commands of the command prompt.	0

https://huggingface.co/datasets/allenai/tulu-3-wildchat-ultrafeedback

and use it to train a *reward model* $r(y|x)$ that measures the 'skill level' of reply y to a given prompt x.

RLHF step 2. Tweak the model so that preferred replies are more likely

Our original language model $\underline{X}$ was trained to mimic texts from a corpus $\{\underline{x}^{(1)}, \underline{x}^{(2)}, \ldots, \underline{x}^{(n)}\}$

Train it by maximum likelihood fitting:

$$\max_{\theta} \frac{1}{n} \sum_{i=1}^{n} \log \Pr_{\underline{X}}(\underline{x}^{(i)}; \theta)$$

where θ = parameters of our $\underline{X}$ model

Let's fit a new language model with parameters ϕ, initialized at $\phi = \theta$.
Write $\underline{Y}^{\phi}(\underline{x})$ for its completion of prompt-text $\underline{x}$.
Train it so we produce human-preferred responses:

average over the prompts in our preference dataset, $\underline{x}_1, \ldots, \underline{x}_m$

$$\max_{\phi} \frac{1}{m} \sum_{j=1}^{m} \mathbb{E}\, r(\underline{Y}^{\phi}(\underline{x}_j) | \underline{x}_j) - \beta \times \left\{ \begin{array}{c} \text{how far the likelihood functions } \Pr_{\underline{Y}^{\phi}(x)}(\cdot) \\ \text{are from the base language model's} \\ \text{likelihood function } \Pr_{\underline{X}}(\cdot) \end{array} \right\}$$

the average reward we get for our new model's reply to $\underline{x}_j$

with a penalty if our new model diverges too far from "plain text"

PART IV

§2 Fitting a Markov chain

A Markov Chain is a sequence in which each item is generated based only on the preceding item:

$$X_0 \to X_1 \to X_2 \to \cdots$$

Thus the likelihood of a sequence $x_0 x_1 \cdots x_\ell$ is

$$\mathbb{P}(x_0 x_1 \cdots x_\ell) = \mathbb{P}(X_0 = x_0) \times \prod_{i=1}^{\ell} \mathbb{P}(X_i = x_i \mid X_{i-1} = x_{i-1})$$

If the items are continuous, use likelihood notation:

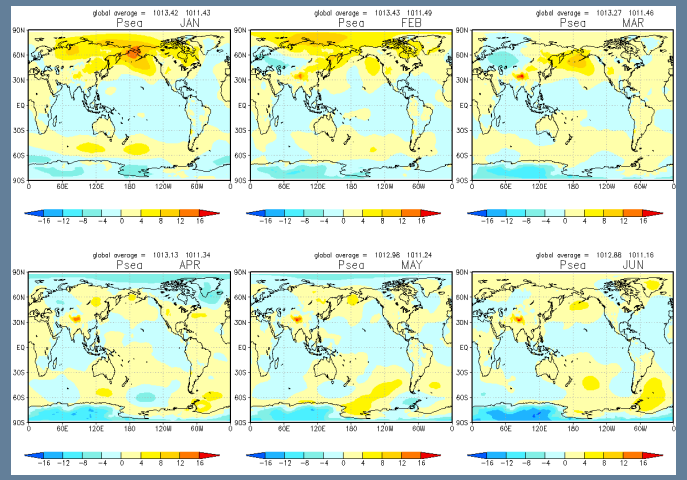
$$\Pr(x_0 x_1 \cdots x_\ell) = \Pr_{X_0}(x_0) \times \prod_{i=1}^{\ell} \Pr_{X_i}(x_i \mid X_{i-1} = x_{i-1})$$

This is the basis for fitting Markov models ...

Applications of Markov chains: dynamical systems

Weather:

$X_{t+1} = \texttt{random_sim_step}(X_t)$



Assets in a stock market

Load on a webserver

What makes it Markov is having a state description X_t that contains absolutely everything we need to know for generating the next state.

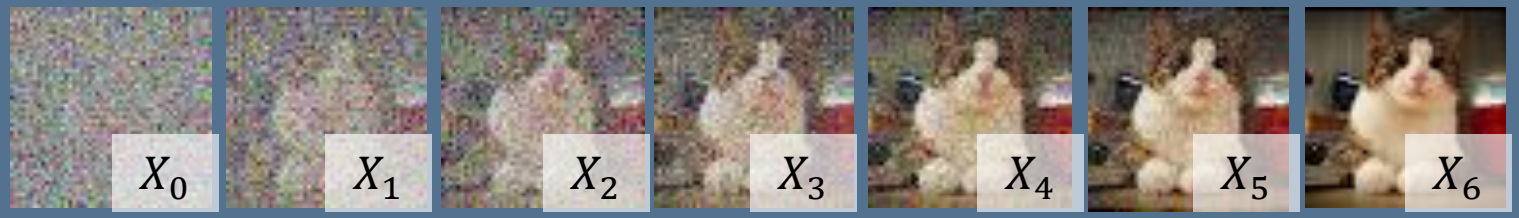
If we're not sure that our X_t captures everything, better train a more general model like a transformer.

Applications of Markov chains: Stable Diffusion image generators

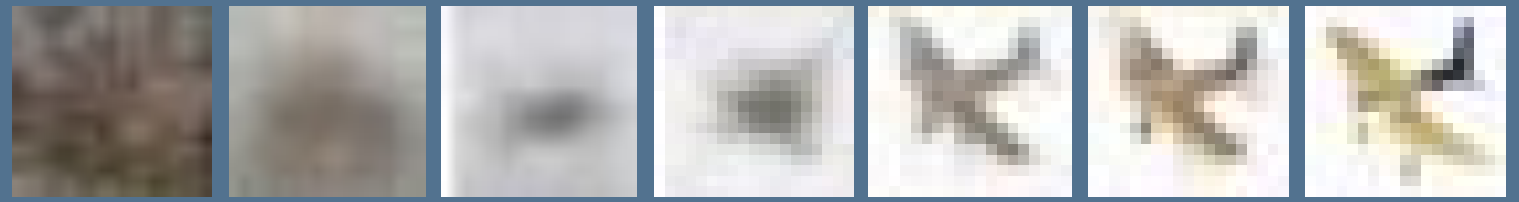
Given an image, create a sequence with progressively more and more noise, until we get pure noise.
Do this for many images, to create a training dataset of sequences.



Reverse the sequences. Train a Markov chain to learn the dynamics $(X_t | X_{t-1} = x)$.



If we apply these dynamics to a new pure-noise image, we will generate a novel image.



Example: fitting a Markov model

Let $[x_0, x_1, \ldots, x_\ell]$ be a time series which we believe is generated by

$$X_{i+1} = a + b\,X_i + N(0, \sigma^2).$$

Estimate a and b using maximum likelihood estimation.

[Also known as *red noise*. Klaus Hasselman, 2021 Nobel Prize for Physics, introduced such models to climate science.]

$$X_i = a + b\,X_{i-1} + N(0, \sigma^2)$$

$$(X_i \mid X_{i-1} = x_{i-1}) \sim N(a + b x_{i-1}, \sigma^2)$$

by linearity of Gaussian

$$\Pr(x_0 x_1 \cdots x_\ell) = \Pr_X(x_0) \prod_{i=1}^{\ell} \Pr(x_i \mid x_{i-1})$$

$$= \eta(x_0) \prod_{i=1}^{\ell} \frac{1}{\sqrt{2\pi\sigma^2}} e^{-(x_i - (a + b x_{i-1}))^2 / 2\sigma^2}$$

we want to maximize this over a, b, c, η

This is exactly like fitting a linear model $X_i \approx a + b\,x_{i-1}$

using sklearn

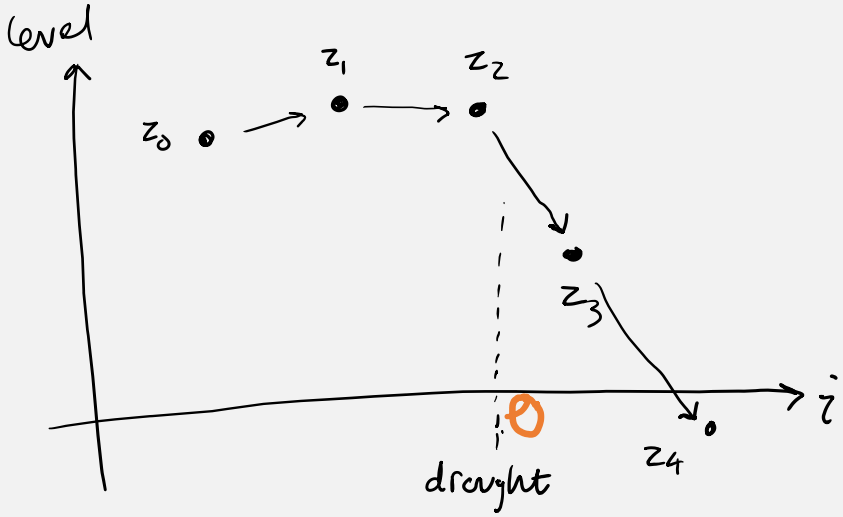
x_{i-1}	x_i
x_0	x_1
x_1	x_2
⋮	
$x_{\ell-1}$	x_ℓ

Mock exam question 1

I have been monitoring annual river levels for many years, and I have collected a dataset $(z_0, \ldots, z_n)$ where z_i is the level in year i of monitoring.

I believe that for the first few years the level each year was roughly what it was the previous year, plus or minus some random variation; but that some year a drought started, and since then the level has decreased on average.

Estimate when the drought started.



$$z_i \sim z_{i-1} + N(0, \varepsilon^2) - \mu 1_{i>0} \sim N(z_{i-1} - \mu 1_{i>0}, \varepsilon^2)$$

$$\Pr(z_0, \ldots, z_n) = \Pr(z_0) \prod_{i=1} \frac{1}{\sqrt{2\pi\varepsilon^2}} e^{-(z_i - (z_{i-1} - \mu 1_{i>0}))^2 / 2\varepsilon^2}$$

??? $\Pr(z_0)$

$$e^{-(\delta_i + \mu 1_{i>0})^2 / 2\varepsilon^2} \quad \text{where } \delta_i = z_i - z_{i-1}$$

PART IV

§3 Working with Markov chains and related random systems

Random systems, and the laws of probability

A **random system** is a collection of random variables, which may or may not be independent
e.g. pixels in an image, tokens in a piece of text, states of a Markov chain

The **chain rule of probability** lets us factorize the joint likelihood in any order we like

$$\mathbb{P}(A = a, B = b) = \mathbb{P}(A = a)\,\mathbb{P}(B = b|A = a)$$

$$\Pr_{A,B}(a, b) = \Pr_A(a)\,\Pr_B(b|A = a) = \Pr_B(b)\,\Pr_A(a|B = b)$$

$$p(a,b,c) = p(a)\,p(b|a)\,p(c|a,b)$$
$$= p(b)\,p(a|b)\,p(c|a,b)$$
$$= \cdots$$

Bayes's rule:

$$\Pr_B(b|A = a) = \frac{\Pr_B(b)\,\Pr_A(a|B = b)}{\Pr_B(b)}$$

The **law of total probability** is for decomposing an event

$$\mathbb{P}(A = a) = \sum_b \mathbb{P}(A = a, B = b) = \sum_b \mathbb{P}(A = a \mid B = b)\,\mathbb{P}(B = b)$$

or $\Pr_A(a) = \int_b \Pr_{A,B}(a, b)\; db$

$$= \int_b \Pr_A(a \mid B = b)\,\Pr_B(b)\; db$$

and also the version with baggage $\{C = c\}$

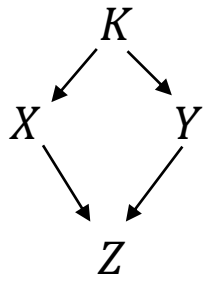
$$\mathbb{P}(A = a \mid C = c) = \sum_b \mathbb{P}(A = a \mid B = b, C = c)\,\mathbb{P}(B = b \mid C = c)$$

Causal models

A **causal model** is a random system that's generated sequentially, as in computer code, where each random variable is generated from specified others. The Directed Acyclic Graph (DAG) of dependencies is called the *causal diagram.*

The joint likelihood can be written as the product of terms specifying how each vertex arises **from its parents.**

```
K := Noise()
X := K + Noise()
Y := K + Noise()
Z := min(X,Y)
```



$p(k,x,y,z) = p(k)\,p(x|k)\,p(y|x,k)\,p(z|x,y,k)$

generic chain rule, applies to any random system

$= p(k)\,p(x|k)\,p(y|k)\,p(z|x,y)$

according to this specific causal diagram