IFT 7030,
Machine Learning for Signal Processing
Week 9: Sequence Models, from HMMs to LLMs

Cem Subakan





UNIVERSITÉ LAVAL



Mila

Admin

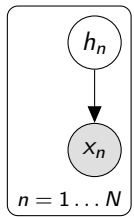
- Homework 2 is out. You can do it all on the notebook.
  - Devoir 2 est publié. Vous pouvez tout faire sur le notebook.
- If you have doubts on your projects contact me. Thanks for those of you who already did. If you do a good project it will help you in your job search / grad school search.
  - Si vous avez des doutes sur vos projets, c'est très important que vous me contactez. Merci pour ceux qui a déjà fait. Si vous faites un bon projet ça va vous aider à trouver un job, ou un grad student position...

Admin

- Homework 2 is out. You can do it all on the notebook.
  - Devoir 2 est publié. Vous pouvez tout faire sur le notebook.
- If you have doubts on your projects contact me. Thanks for those of you who already did. If you do a good project it will help you in your job search / grad school search.
  - Si vous avez des doutes sur vos projets, c'est très important que vous me contactez. Merci pour ceux qui a déjà fait. Si vous faites un bon projet ça va vous aider à trouver un job, ou un grad student position...
- I will share a sign-up sheet soon for project presentations.
  - Je partager un sign-up sheet bientot pour les presentations.
- This week: / Cette semaine: Time series!

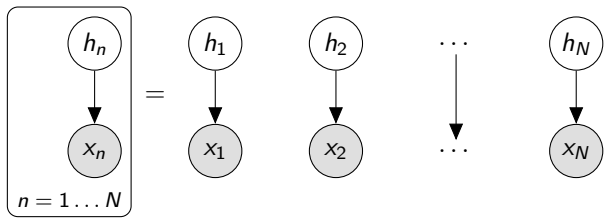
Our view of data so far (well, mostly)

- IID: (not the institute) Independent and Identically Distributed
- Remember last week? / Souvenez-vous de ça de la semaine dernière?



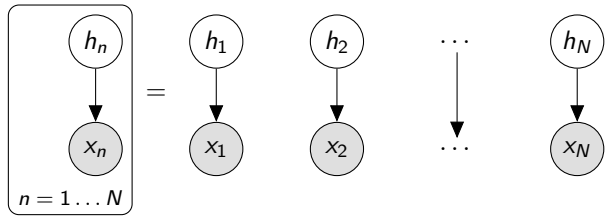
Our view of data so far (well, mostly)

- IID: (not the institute) Independent and Identically Distributed
- Remember last week? / Souvenez-vous de ça de la semaine dernière?

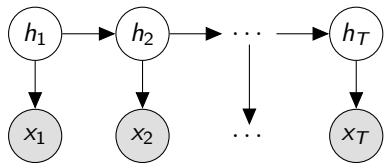


Our view of data so far (well, mostly)

- IID: (not the institute) Independent and Identically Distributed
- Remember last week? / Souvenez-vous de ça de la semaine dernière?

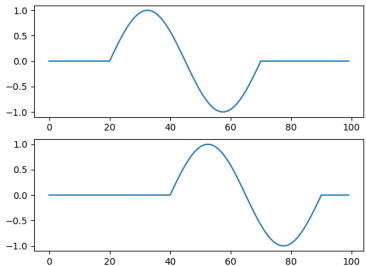


- Well, this week we will have this instead:
  - Bon, cette semain on aura ça:



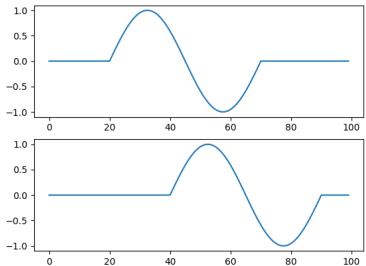
Motivation

- Are these signals similar? / Ces deux signals, sont-ils similaires?



Motivation

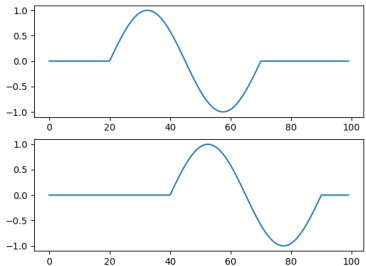
- Are these signals similar? / Ces deux signals, sont-ils similaires?



- If we calculate a distance in the 100d space they are distant.
  - Si on calcule une distance dans une espace de 100d, ils sont distants.

Motivation

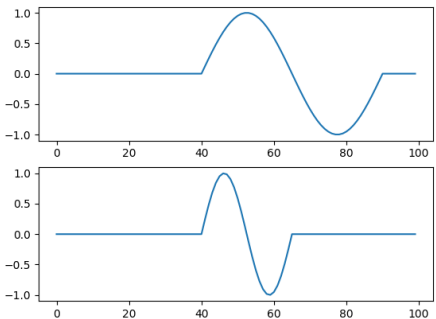
- Are these signals similar? / Ces deux signals, sont-ils similaires?



- If we calculate a distance in the 100d space they are distant.
  - Si on calcule une distance dans une espace de 100d, ils sont distants.
- What can we do?
  - Que peut-on faire?

How about these?

- Are these similar? / Sont-ils similaires?



- How about these? / Et ces deux?

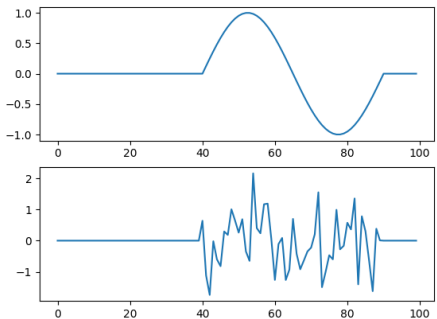


Table of Contents

Dynamic Time Warping

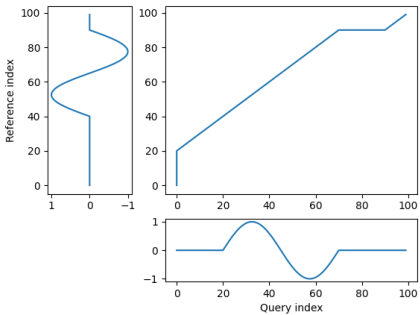
Hidden Markov Models
- Inference
- Learning
- Decoding
- HMM Applications
- HMM Variants

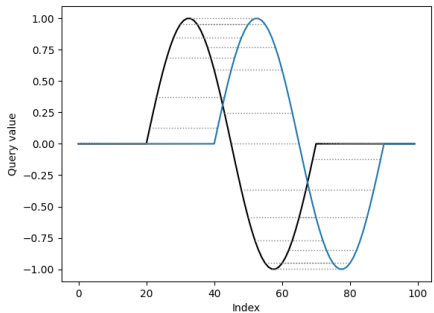
Autoregressive Models

Large Language Models
- Evaluating and Sampling
- Training an LLM

Dynamic Time Warping

- We can to find a 'warping function' between two signals to match them.
  - On peut trouver une fonction de 'warping' entre les deux signaux pour les matcher.





Dynamic Time Warping

- The problem formulation / La formulation du problème

$$\mathrm{DTW}(x, y) = \min_{\pi} \sqrt{\sum_{(i,j)\in\pi} d(x_i, y_i)}$$

  - where, π is a path defined over the indices. / où π est un path défini sur les indices.
- More formally/Plus formellement $\pi = [(i_0, j_0), (i_1, j_1), \ldots, (i_K, j_K)]$.
- Constraints: / Contraints:

$$\begin{aligned} \pi_0 &= (0, 0) \\ \pi_K &= (n-1, m-1) \\ i_{k-1} \leq i_k &\leq i_{k-1} + 1 \\ j_{k-1} \leq j_k &\leq j_{k-1} + 1 \end{aligned}$$

The dynamic programming solution

- The recursion / La recursion

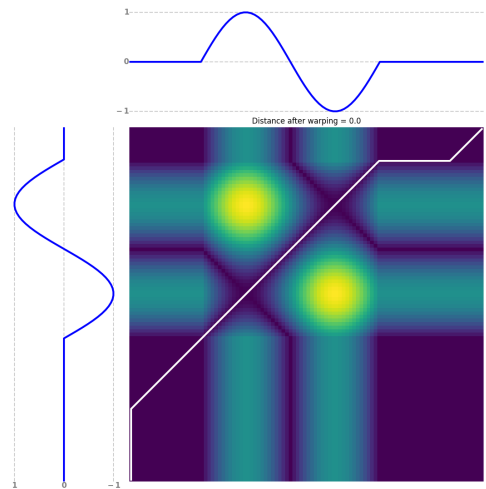
$$C_{i,j} = d(x_i, y_j) + \min(C_{i-1,j}, C_{i,j-1}, C_{i-1,j-1})$$

The dynamic programming solution

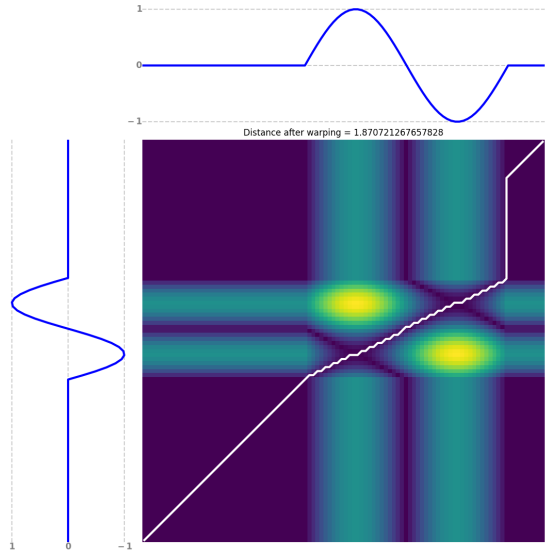
- The recursion / La recursion

$$C_{i,j} = d(x_i, y_j) + \min(C_{i-1,j}, C_{i,j-1}, C_{i-1,j-1})$$

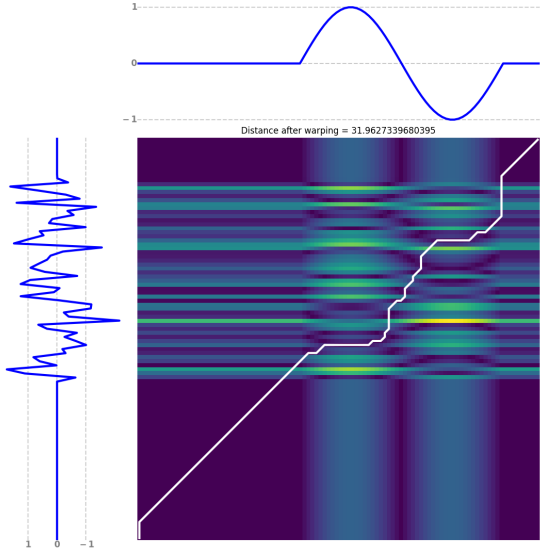
- Optimal path after backtracking / La trace optimale après backtracking



DTW examples

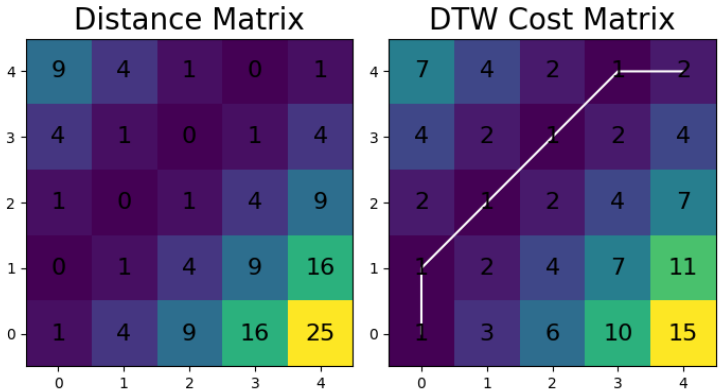


DTW examples



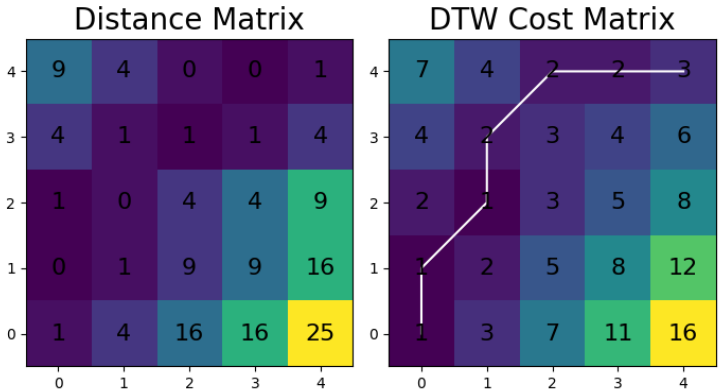
DTW on two arrays

- $x = [0, 1, 2, 3, 4]$, $y = [1, 2, 3, 4, 5]$.



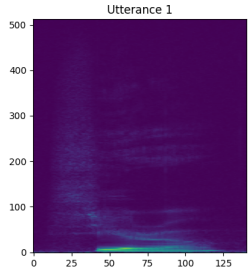
DTW on two arrays

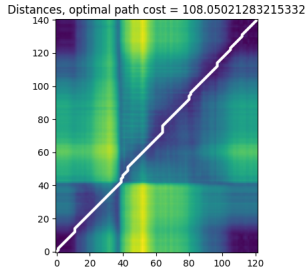
- $x = [0, 1, 2, 3, 4]$, $y = [1, 2, 4, 4, 5]$.

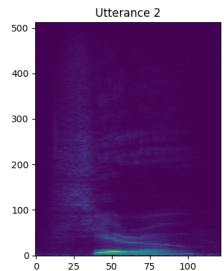


DTW on Audio

- DTW on same person saying 'zero' two different times.
  - DTW sur la meme personne qui dit 'zero' deux differentes fois.

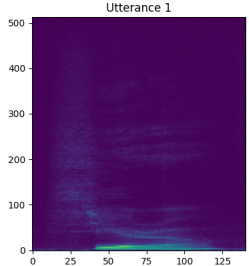


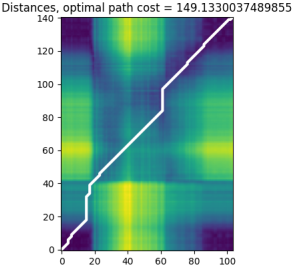


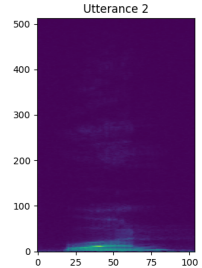


DTW on Audio

- DTW on same person saying 'zero' and 'one'
  - DTW sur la meme personne qui dit 'zero' et 'one'.
  - Do you notice something? / Vous remarquez quelque chose qui peut etre utile?

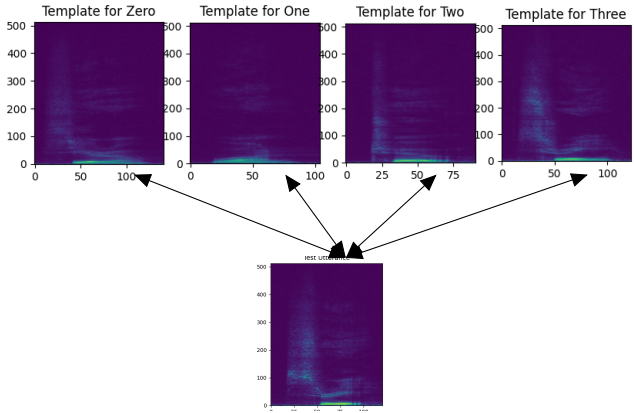




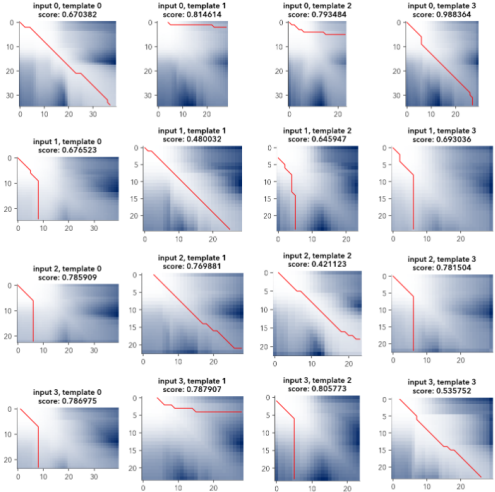


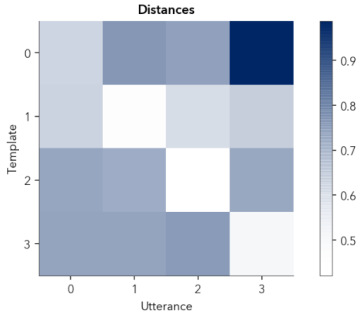
Using DTW for sequence classification

- An extremely basic speech classifier / Un classificateur vocale extrememement simple.
- We can store templates for each class and then assign to the one with smallest DTW cost.
  - On peut garder des templates pour chaque classe, et assigner le séquence test à la classe avec le cout DTW le plus petit.



DTW classification in action





Taken from UIUC MLSP class slides

To summarize DTW

- It's a distance over time-series, and having that is awesome.
  - C'est une distance sur des time-series, et c'est magnique d'en avoir un.
- You can train a neural net with it, do clustering (maybe? can you? what would be a problem?).
  - Vous pouvez entrainer un réseau neural avec ca, ou clustering (peut-etre vous pouvez?)

To summarize DTW

- It's a distance over time-series, and having that is awesome.
  - C'est une distance sur des time-series, et c'est magnique d'en avoir un.
- You can train a neural net with it, do clustering (maybe? can you? what would be a problem?).
  - Vous pouvez entrainer un réseau neural avec ca, ou clustering (peut-etre vous pouvez?)
- You can even dub movies! / Vous pouvez meme faire du doublage avec DTW.

Table of Contents

Dynamic Time Warping

Hidden Markov Models
- Inference
- Learning
- Decoding
- HMM Applications
- HMM Variants

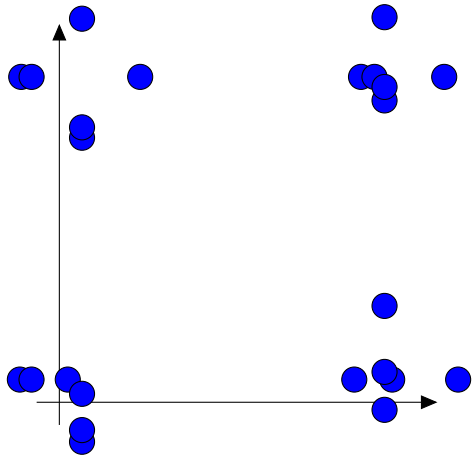
Autoregressive Models

Large Language Models
- Evaluating and Sampling
- Training an LLM

A model over time series

- DTW is nice and all, but it's just a distance, it's not a model.
  - DTW est bon mais ce n'est qu'une distance, n'est pas une modèle.
- Remember the duality between likelihood / and distances (e.g. Gaussian and Euclidean distance)

Let's remember GMMs



GMMs with time

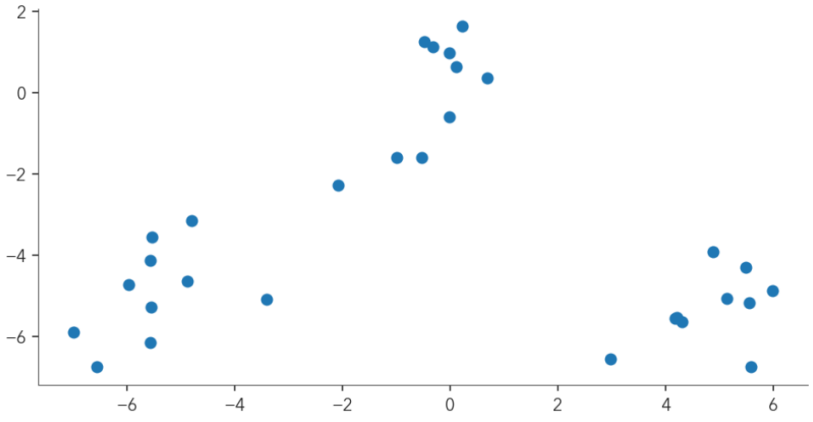


Image taken from UIUC MLSP class

GMMs with time

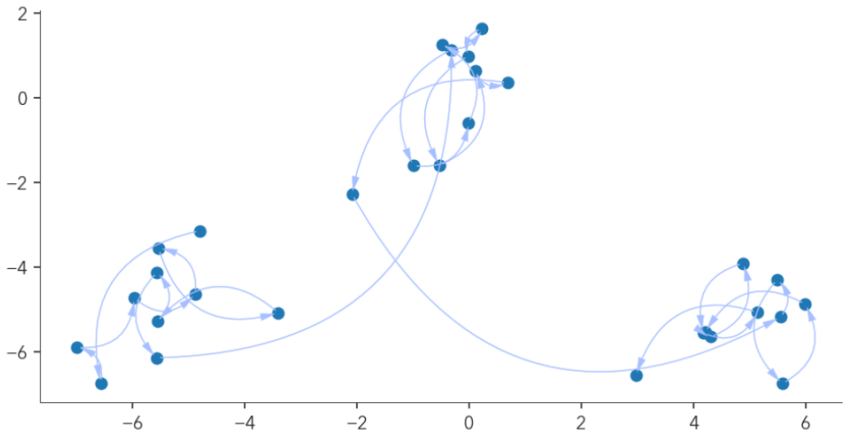
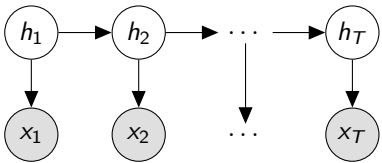


Image taken from UIUC MLSP class

Tired of IID models? HMMs

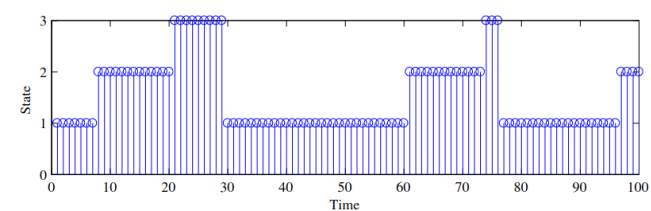
- Model / Modèle:

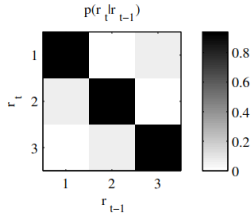


$$h_n|h_{n-1} \sim \text{Discrete}(A(:, h_{n-1}))$$
$$x_n|h_n \sim p(x_n|h_n, O)$$

- $h_n \in \{1, \ldots, K\}$, latent variables (embeddings). Difference from before is h_n and h_{n+1} are connected!
  - Différence avec avant est que les variables latent sont connéctés.
- $x_n \in \mathbb{R}^L$, observed data items / les données observées.
- The parameters $\theta = \{O, A\}$ / Les paramètres.
- $O \in \mathbb{R}^{L \times K}$, the emission matrix, $A \in \mathbb{R}^{K \times K}$, the transition matrix.
- Learning is conceptually all the same. Just that E-step is little different.
  - L'apprentissage est meme qu'avant. C'est juste que E-step est un peu différent.

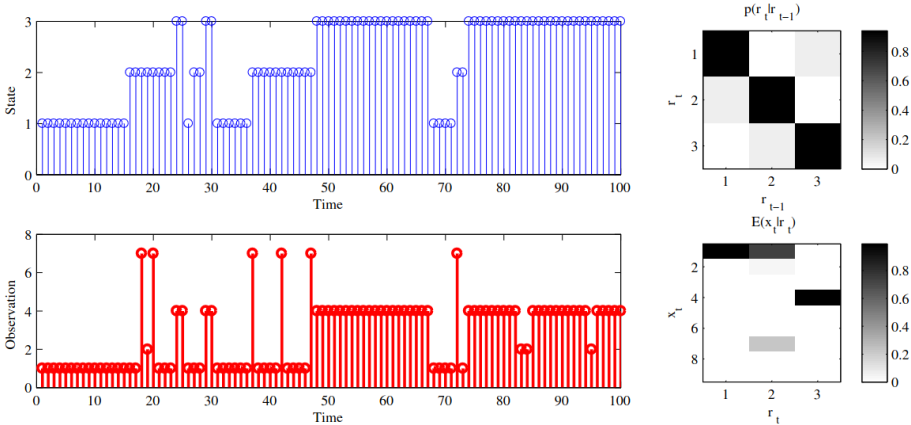
Continuous data generated from an HMM





In this case $p(x_t|r_t, O) = \mathcal{N}(x_n; \mu_{r_t}, \sigma^2 I)$.

Discrete data generated from an HMM



In this case $p(x_t|r_t, O) = \text{Discrete}(x_t; O[:, r_t])$.

Statistical Problems to solve with HMMs

- **Inference / Evaluation**
  - How do we calculate $p(x_{1:T}|\theta)$. / Comment est-ce qu'on calcule ce marginale?
- **Decoding**
  - What are the optimal state values (not probabilities) given $x_{1:T}$ and a learnt model. / Comment obtiens-t-on les valeurs des états optimales?
- **Learning**
  - Given a sequence $x_{1:T}$, how do we learn the optimal model parameters? / Étant donné une séquence $x_{1:T}$ comment est-ce qu'on peut apprendre les paramètres optimales du modèle?

Table of Contents

Dynamic Time Warping

Hidden Markov Models
- Inference
- Learning
- Decoding
- HMM Applications
- HMM Variants

Autoregressive Models

Large Language Models
- Evaluating and Sampling
- Training an LLM

Inference in HMMs

- The precise inference question / La question précise pour l'inférence.

$$p(x_{1:T}|\theta) = \sum_{h_{1:T}} p(x_{1:T}, h_{1:T}|\theta)$$

Inference in HMMs

- The precise inference question / La question précise pour l'inférence.

$$
\begin{aligned}
p(x_{1:T}|\theta) &= \sum_{h_{1:T}} p(x_{1:T}, h_{1:T}|\theta) \\
&= \sum_{h_{1:T}} \prod_{t=1}^{T} p(x_t|h_t) p(h_t|h_{t-1})
\end{aligned}
$$

Inference in HMMs

- The precise inference question / La question précise pour l'inférence.

$$
\begin{aligned}
p(x_{1:T}|\theta) &= \sum_{h_{1:T}} p(x_{1:T}, h_{1:T}|\theta) \\
&= \sum_{h_{1:T}} \prod_{t=1}^{T} p(x_t|h_t)p(h_t|h_{t-1}) \\
&= \sum_{h_T} \cdots \sum_{h_2} \sum_{h_1} p(x_T|h_T)p(h_T|h_{T-1}) \ldots p(x_2|h_2)p(h_2|h_1)p(h_1)
\end{aligned}
$$

Inference in HMMs

- The precise inference question / La question précise pour l'inférence.

$$
\begin{aligned}
p(x_{1:T}|\theta) &= \sum_{h_{1:T}} p(x_{1:T}, h_{1:T}|\theta) \\
&= \sum_{h_{1:T}} \prod_{t=1}^{T} p(x_t|h_t)p(h_t|h_{t-1}) \\
&= \sum_{h_T} \cdots \sum_{h_2} \sum_{h_1} p(x_T|h_T)p(h_T|h_{T-1}) \ldots p(x_2|h_2)p(h_2|h_1)p(h_1)
\end{aligned}
$$

- This is a huge sum! / C'est une operation immense.

Inference in HMMs

- The precise inference question / La question précise pour l'inférence.

$$
\begin{aligned}
p(x_{1:T}|\theta) &= \sum_{h_{1:T}} p(x_{1:T}, h_{1:T}|\theta) \\
&= \sum_{h_{1:T}} \prod_{t=1}^{T} p(x_t|h_t)p(h_t|h_{t-1}) \\
&= \sum_{h_T} \cdots \sum_{h_2} \sum_{h_1} p(x_T|h_T)p(h_T|h_{T-1}) \ldots p(x_2|h_2)p(h_2|h_1)p(h_1)
\end{aligned}
$$

- This is a huge sum! / C'est une operation immense.
- What can we do? / Qu'est-ce qu'on peut faire?

Inference in HMMs

- The forward inference: (The filtering density)
  - Inférence en avançant

$$\alpha(h_t) := p(x_{1:t}, h_t)$$

- The backward inference:
  - Inférence en réculant

$$\beta(h_t) := p(x_{t+1:T}|h_t)$$

The Dynamic Programming Solution (Again)

$$\alpha(h_t) = p(x_t|h_t) \sum_{h_{t-1}} p(h_t|h_{t-1})\, \underbrace{p(x_{t-1}|h_{t-1}) \ldots \underbrace{p(x_2|h_2) \sum_{h_1} p(h_2|h_1) p(x_1|h_1) \underbrace{p(h_1)}_{\alpha(h_1)}}_{\alpha(h_2)}}_{\alpha(h_{t-1})}$$

$$\beta(h_t) = \underbrace{\sum_{h_{t+1}} p(h_{t+1}|h_t) p(x_{t+1}|h_{t+1}) \ldots \underbrace{\sum_{h_T} p(h_T|h_{T-1}) p(x_T|h_T) \underbrace{1}_{\beta(h_T)}}_{\beta(h_{T-1})}}_{\beta(h_{t+1})}$$

The forward and backward recursions

- The forward recursion / La recurrence en avançant:

$$\alpha(h_t) = p(x_t|h_t) \sum_{h_{t-1}} p(h_t|h_{t-1})\ \alpha(h_{t-1})$$

- The backward recursion / La recurrence en reculant:

$$\beta(h_t) = \sum_{h_{t+1}} p(h_{t+1}|h_t) p(x_{t+1}|h_{t+1})\ \beta(h_{t+1})$$

Ok but what happened to the likelihood question?

- Note:

$$\alpha(h_T) = p(x_1, x_2, \ldots, x_T, h_T)$$

- So how do we get $p(x_{1:T})$?

Ok but what happened to the likelihood question?

- Note:

$$\alpha(h_T) = p(x_1, x_2, \ldots, x_T, h_T)$$

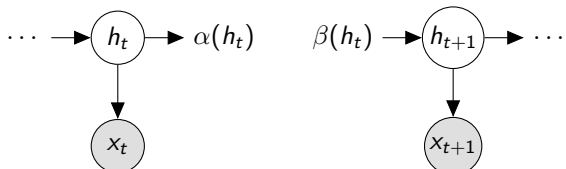
- So how do we get $p(x_{1:T})$?
- $p(x_{1:T}) = \sum_{h_T} p(x_{1:T}, h_T) = \sum_{h_T} \alpha(h_T)$.

Why do we need the backward?

- $\alpha(h_t)$ are "forward messages". $\beta(h_t)$ are "backward messages". One forward pass and one backward pass is sufficient since, / Une passe en avançant et une passe en réculant suffissent parce que,

$$\begin{aligned} p(h_t|x_{1:T}) &\propto p(h_t, x_{1:T}) \\ &= p(h_t, x_{1:t}) p(x_{t+1:T}|h_t) \\ &= \alpha(h_t)\beta(h_t) \end{aligned}$$

- Traditionally (EE traditions), $\alpha_{1:T}$ is known as the filtering density. $\gamma_{1:T} := \alpha_{1:T} . * \beta_{1:T}$ is the smoothing density (la densité smoothing).



Forward Pass in Action

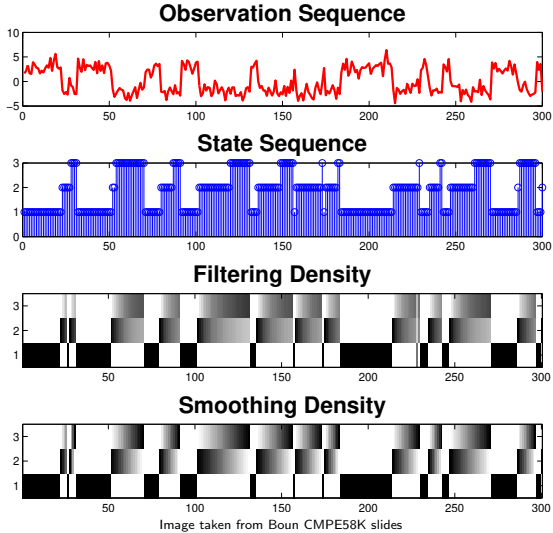


Image taken from Boun CMPE58K slides

Table of Contents

Dynamic Time Warping

Hidden Markov Models
- Inference
- Learning
- Decoding
- HMM Applications
- HMM Variants

Autoregressive Models

Large Language Models
- Evaluating and Sampling
- Training an LLM

The learning question is the same as GMMs

- The learning question / La problématique d'apprentissage

$$\widehat{\theta} = \arg\max_{\theta} p(x_{1:N}|\theta) = \arg\max_{\theta} \sum_{h_{1:N}} p(x_{1:N}, h_{1:N}|\theta)$$

The learning question is the same as GMMs

- The learning question / La problématique d'apprentissage

$$\widehat{\theta} = \arg\max_{\theta} p(x_{1:N}|\theta) = \arg\max_{\theta} \sum_{h_{1:N}} p(x_{1:N}, h_{1:N}|\theta)$$

- Write down/Écrit log-likelihood:

$$\log p(x_{1:N}|\theta) = \log \sum_{h_{1:N}} \frac{p(x_{1:N}, h_{1:N}|\theta)}{q(h_{1:N})} q(h_{1:N}) = \log \mathbb{E}_q \left[\frac{p(x_{1:N}, h_{1:N}|\theta)}{q(h_{1:N})} \right]$$

The learning question is the same as GMMs

- The learning question / La problématique d'apprentissage

$$\widehat{\theta} = \arg\max_{\theta} p(x_{1:N}|\theta) = \arg\max_{\theta} \sum_{h_{1:N}} p(x_{1:N}, h_{1:N}|\theta)$$

- Write down/Écrit log-likelihood:

$$\log p(x_{1:N}|\theta) = \log \sum_{h_{1:N}} \frac{p(x_{1:N}, h_{1:N}|\theta)}{q(h_{1:N})} q(h_{1:N}) = \log \mathbb{E}_q \left[\frac{p(x_{1:N}, h_{1:N}|\theta)}{q(h_{1:N})} \right]$$
$$\geq VLB := \mathbb{E}_q \left[\log \frac{p(x_{1:N}, h_{1:N}|\theta)}{q(h_{1:N})} \right] =^{+} \mathbb{E}_q \left[\log p(x_{1:N}, h_{1:N}|\theta) \right]$$

The learning question is the same as GMMs

- The learning question / La problématique d'apprentissage

$$\widehat{\theta} = \arg\max_{\theta} p(x_{1:N}|\theta) = \arg\max_{\theta} \sum_{h_{1:N}} p(x_{1:N}, h_{1:N}|\theta)$$

- Write down/Écrit log-likelihood:

$$\log p(x_{1:N}|\theta) = \log \sum_{h_{1:N}} \frac{p(x_{1:N}, h_{1:N}|\theta)}{q(h_{1:N})} q(h_{1:N}) = \log \mathbb{E}_q \left[\frac{p(x_{1:N}, h_{1:N}|\theta)}{q(h_{1:N})} \right]$$
$$\geq VLB := \mathbb{E}_q \left[\log \frac{p(x_{1:N}, h_{1:N}|\theta)}{q(h_{1:N})} \right] =^{+} \mathbb{E}_q \left[\log p(x_{1:N}, h_{1:N}|\theta) \right]$$

- Except the fact that the posterior distribution is $q(h_t|x_{1:T}) = p(h_t|x_{1:T})$. Not $p(h_t|x_t)$ (unlike the GMM case. / La différence du cas des GMMs est que maintenant le posterior ne se factorise pas sur temps.

EM Algorithm for HMMs

Randomly initialize θ.

while Not converged **do**

E-step:

Do a Forward and backward pass. Get all $\alpha(h_t)$ and $\beta(h_t)$.

M-step:

$$\widehat{\mu}_k = \frac{\sum_{t=1}^{T} \mathbb{E}_q[h_t{=}k] x_t}{\sum_{t=1}^{T} \mathbb{E}_q[h_t{=}k]}$$

$$\widehat{A}_{ij} = \frac{\sum_{t=1}^{T-1} \mathbb{E}_q[h_t{=}j, h_{t+1}{=}i]}{\sum_{t=1}^{T-1} \mathbb{E}_q[h_t{=}j]}$$

end while

- $\mathbb{E}_q[h_t] = \alpha(h_t)\beta(h_{t+1})/Z$
- $\mathbb{E}_q[h_t, h+1] = p(h_t, h_{t+1}|x_{1:T}) \propto p(x_{1:t}, h_t)p(h_{t+1}|h_t)p(x_{t+1}|h_{t+1})p(x_{t+2:T}|h_{t+1}) = \alpha(h_t)p(h_{t+1}|h_t)p(x_{t+1}|h_{t+1})\beta(h_{t+1}) = \alpha(h_t)AO[:, h_{t+1}]\beta(h_{t+1})$

Table of Contents

Dynamic Time Warping

Hidden Markov Models
- Inference
- Learning
- Decoding
- HMM Applications
- HMM Variants

Autoregressive Models

Large Language Models
- Evaluating and Sampling
- Training an LLM

Decoding in HMMs

- The precise inference question / La question précise pour l'inférence.

$$p(x_{1:T}, h^*_{1:T}|\theta) = \max_{h_{1:T}} p(x_{1:T}, h_{1:T}|\theta)$$

Decoding in HMMs

- The precise inference question / La question précise pour l'inférence.

$$
\begin{aligned}
p(x_{1:T}, h^*_{1:T}|\theta) &= \max_{h_{1:T}} p(x_{1:T}, h_{1:T}|\theta) \\
&= \max_{h_{1:T}} \prod_{t=1}^{T} p(x_t|h_t) p(h_t|h_{t-1})
\end{aligned}
$$

Decoding in HMMs

- The precise inference question / La question précise pour l'inférence.

$$
\begin{aligned}
p(x_{1:T}, h^*_{1:T}|\theta) &= \max_{h_{1:T}} p(x_{1:T}, h_{1:T}|\theta) \\
&= \max_{h_{1:T}} \prod_{t=1}^{T} p(x_t|h_t)p(h_t|h_{t-1}) \\
&= \max_{h_T} \ldots \max_{h_2} \max_{h_1} p(x_T|h_T)p(h_T|h_{T-1}) \ldots p(x_2|h_2)p(h_2|h_1)p(h_1)
\end{aligned}
$$

Decoding in HMMs

- The precise inference question / La question précise pour l'inférence.

$$
\begin{aligned}
p(x_{1:T}, h^*_{1:T}|\theta) &= \max_{h_{1:T}} p(x_{1:T}, h_{1:T}|\theta) \\
&= \max_{h_{1:T}} \prod_{t=1}^{T} p(x_t|h_t)p(h_t|h_{t-1}) \\
&= \max_{h_T} \ldots \max_{h_2} \max_{h_1} p(x_T|h_T)p(h_T|h_{T-1}) \ldots p(x_2|h_2)p(h_2|h_1)p(h_1)
\end{aligned}
$$

- This is a huge max! / C'est une operation immense.

Decoding in HMMs

- The precise inference question / La question précise pour l'inférence.

$$
\begin{aligned}
p(x_{1:T}, h^*_{1:T}|\theta) &= \max_{h_{1:T}} p(x_{1:T}, h_{1:T}|\theta) \\
&= \max_{h_{1:T}} \prod_{t=1}^{T} p(x_t|h_t)p(h_t|h_{t-1}) \\
&= \max_{h_T} \ldots \max_{h_2} \max_{h_1} p(x_T|h_T)p(h_T|h_{T-1}) \ldots p(x_2|h_2)p(h_2|h_1)p(h_1)
\end{aligned}
$$

- This is a huge max! / C'est une operation immense.
- What can we do? / Qu'est-ce qu'on peut faire?

The Dynamic Programming Solution (Again)

$$V(h_t) = p(x_t|h_t) \max_{h_{t-1}} p(h_t|h_{t-1})\, \underbrace{p(x_{t-1}|h_{t-1}) \ldots \underbrace{p(x_2|h_2) \max_{h_1} p(h_2|h_1) p(x_1|h_1) \underbrace{p(h_1)}_{V(h_1)}}_{V(h_2)}}_{V(h_{t-1})}$$

We run this recursion, and then backtrack to find the optimal path $h^*_{1:T}$. (The Viterbi algorithm) / On roule la recurrence et puis backtrack pour trouver la trace optimale $h^*_{1:T}$.

Table of Contents

Dynamic Time Warping

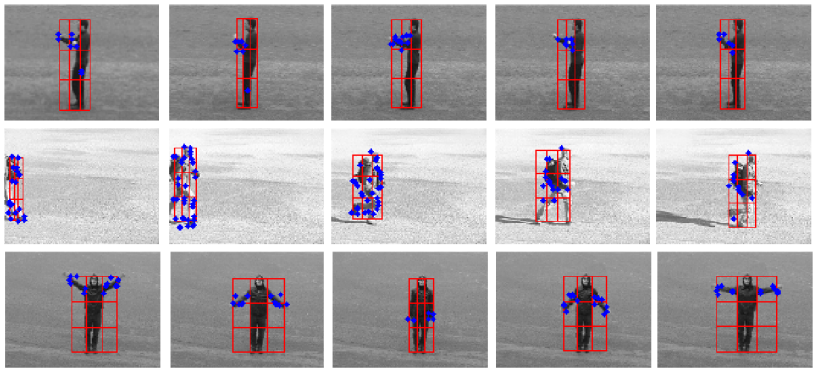
Hidden Markov Models
- Inference
- Learning
- Decoding
- HMM Applications
- HMM Variants

Autoregressive Models

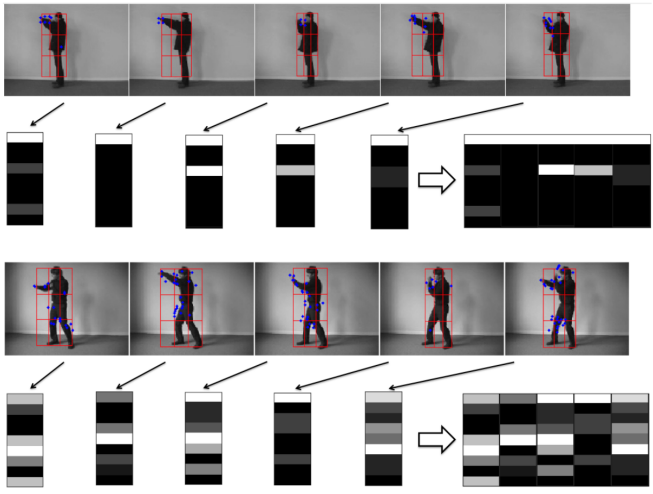
Large Language Models
- Evaluating and Sampling
- Training an LLM

An HMM Learning Application

- Human Action Recognition



Getting the sequences



HMMs for classification

- Train an HMM for each class / On entraine un HMM pour chaque classe.
- In test time we assign to the HMM that yields the max likelihood.

$$\widehat{c}_n = \arg\max_k p(x_n|\theta_k)$$

HMMs for classification

- Train an HMM for each class / On entraine un HMM pour chaque classe.
- In test time we assign to the HMM that yields the max likelihood.

$$\widehat{c}_n = \arg\max_k p(x_n|\theta_k)$$

- We saw this type of thing before, remember? / Vous vous souvenez de ça?

HMMs for classification

- Train an HMM for each class / On entraine un HMM pour chaque classe.
- In test time we assign to the HMM that yields the max likelihood.

$$\widehat{c}_n = \arg\max_k p(x_n|\theta_k)$$

- We saw this type of thing before, remember? / Vous vous souvenez de ça?
- Generative classification.. B=boxing, HC=Hand Clapping, HW=Hand Waving ...

EM, 70.1 %

	B	HC	HW	J	R	W
B	**32**	4	1	0	1	0
HC	1	**31**	6	0	1	0
HW	0	1	**29**	0	0	0
J	1	0	0	**17**	20	3
R	0	0	0	7	**10**	0
W	2	0	0	12	4	**33**

HMMs for speech recognition

- Each state should correspond to a semantically meaningful thing. (e.g. a phoneme) / Chaque état HMM devrait correspondre à quelque chose qui a une sense semantique.
- Someone saying 'one'. / Quelqu'un qui dit 'one'.

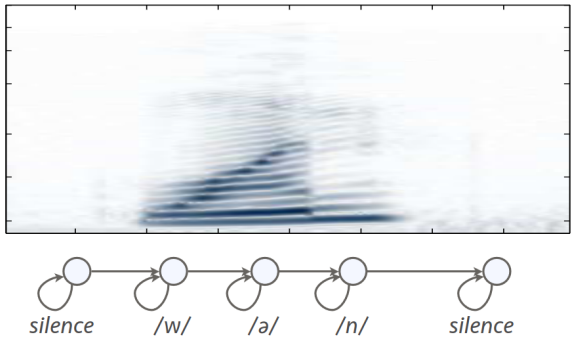


Image taken from UIUC MLSP course.

HMM learning on speech

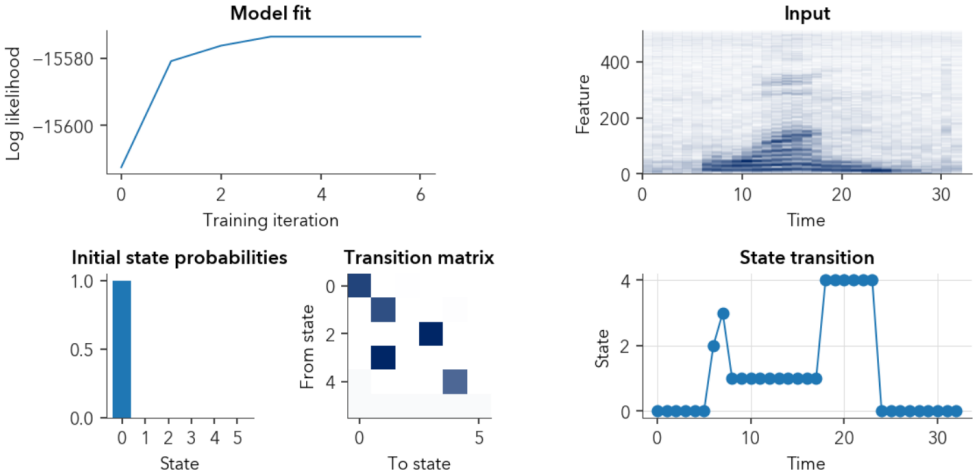


Image taken from UIUC MLSP course.

HMM learning on speech

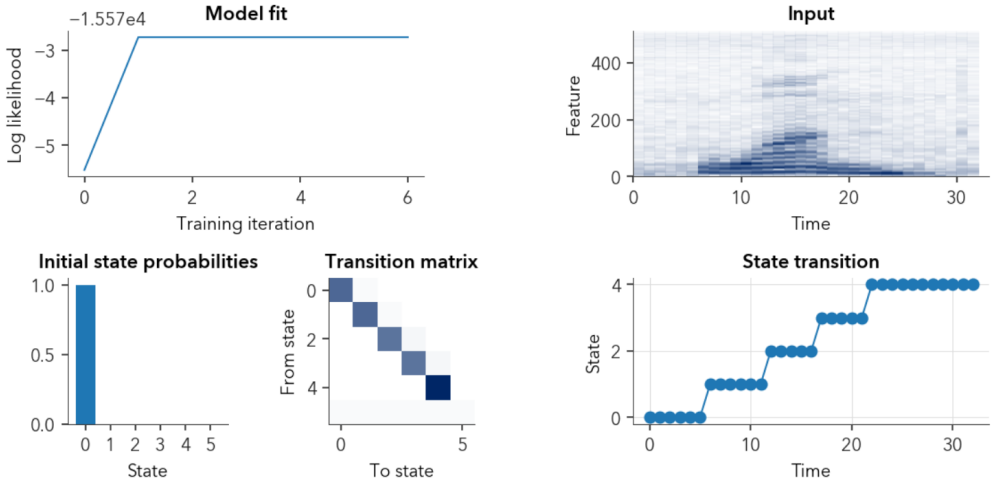


Image taken from UIUC MLSP course.

A big ASR system with HMMs

- A real life HMM speech recognizer in a nut shell / La sommaire d'un système ASR avec HMMs.

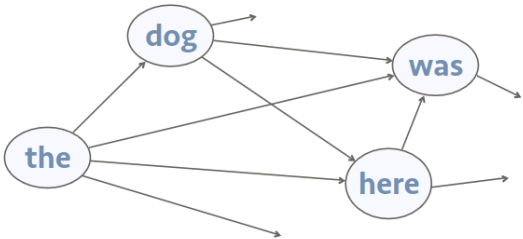


Image taken from UIUC MLSP course.

- You have an HMM for each word, then you connect the HMMs..

And don't say HMMs are outdated!

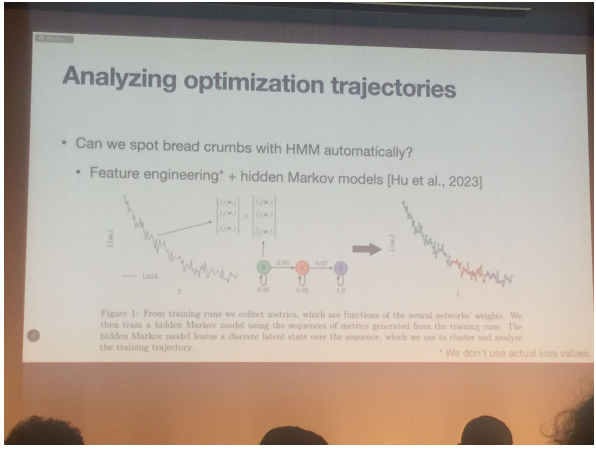


Image taken when I was at SANE 2023 last year

Using HMMs to analyze the training behavior of LLMs from 2023.

Table of Contents

Dynamic Time Warping

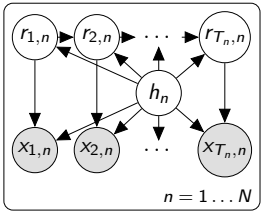
Hidden Markov Models
- Inference
- Learning
- Decoding
- HMM Applications
- HMM Variants

Autoregressive Models

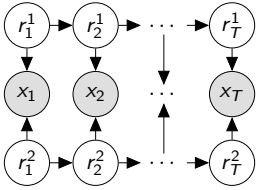
Large Language Models
- Evaluating and Sampling
- Training an LLM

More Advanced HMM Variants

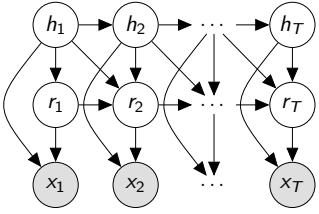
- Mixture of HMMs



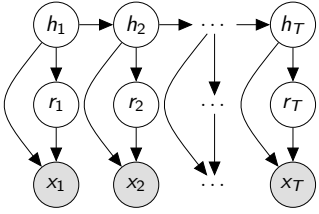
- Factorial HMM



- Switching HMMs

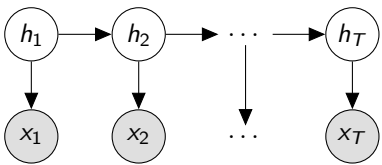


- HMM with Mixture observations



Super Similar but different: Linear Dynamical System

- Model:



$$h_n|h_{n-1} \sim \mathcal{N}(h_n; Ah_{n-1}, \Sigma_1)$$
$$x_n|h_n \sim \mathcal{N}(x_n; Oh_n, \Sigma_2)$$

- $h_n \in \mathbb{R}^K$, latent variables/variables latents.
- $x_n \in \mathbb{R}^L$, observed data items / données observées.
- $O \in \mathbb{R}^{L \times K}$, the emission matrix / la matrice d'emission, $A \in \mathbb{R}^{K \times K}$, the transition matrix / la matrice de transition.
- $\theta = \{O, A\}$ parameters / paramètres.
- The inference is done with a Kalman filter. / Le fameux filtre de Kalman est utilisé pour l'inférence.

Tired of directed graphs? MRFs

- The joint distribution is defined with clique "potentials".

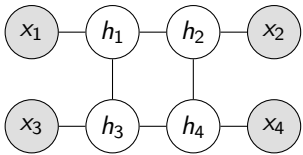
$$p(h_{1:K}, x_{1:J}|\theta) = \frac{1}{Z(\theta)} \prod_{C \in \mathcal{G}} \exp(\theta^T \phi(x_C, h_C))$$

Tired of directed graphs? MRFs

- The joint distribution is defined with clique "potentials".

$$p(h_{1:K}, x_{1:J}|\theta) = \frac{1}{Z(\theta)} \prod_{C \in \mathcal{G}} \exp(\theta^T \phi(x_C, h_C))$$

- Example: (An image segmentation model)



$$\begin{aligned}\phi(x_C, h_C) &= \phi_1(h_i, h_{\mathcal{N}(i)}) + \phi_2(x_i, h_i) \\ &= \theta_1 \mathbf{1}_{[h_i = h_{\mathcal{N}(i)}]} + \theta_2 \mathbf{1}_{[h_i \neq h_{\mathcal{N}(i)}]} \\ &\quad + \sum_{l,k} \theta_{3,i,k} \mathbf{1}_{[x_i = l][h_i = k]}\end{aligned}$$

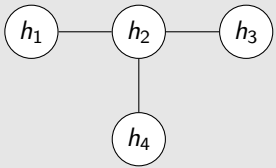
$$Z(\theta) = \int \prod_{C \in \mathcal{G}} \exp(\theta^T \phi(x_C, h_C)) dx_{1:J} dh_{1:K}$$

The notorious partition function!

How to do inference in general graphs?

- Forward-Backward algorithm is an instance of "Belief Propagation".

Example



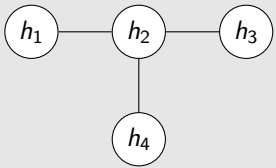
$$p(h_{1:4}) = \frac{1}{Z}\psi(h_1, h_2)\psi(h_2, h_4)\psi(h_2, h_3)$$

$$p(h_2) \propto \sum_{h_1, h_3, h_4} \psi(h_1, h_2)\psi(h_2, h_4)\psi(h_2, h_3)$$

$$= \underbrace{\left(\sum_{h_1} \psi(h_1, h_2)\right)}_{\mathbf{m}_{1\to 2}} \underbrace{\left(\sum_{h_4} \psi(h_2, h_4)\right)}_{\mathbf{m}_{4\to 2}} \underbrace{\left(\sum_{h_3} \psi(h_2, h_3)\right)}_{\mathbf{m}_{3\to 2}}$$

Example continued

Example



$$p(h_{1:4}) = \frac{1}{Z}\psi(h_1, h_2)\psi(h_2, h_4)\psi(h_2, h_3)$$

$$\begin{aligned} p(h_1) &\propto \sum_{h_2,h_3,h_4} \psi(h_1, h_2)\psi(h_2, h_4)\psi(h_2, h_3) \\ &= \sum_{h_2} \psi(h_1, h_2) \left(\sum_{h_4} \psi(h_2, h_4)\right) \left(\sum_{h_3} \psi(h_2, h_3)\right) \\ &= \sum_{h_2} \psi(h_1, h_2)\mathbf{m}_{4\to 2}(h_2)\mathbf{m}_{3\to 2}(h_2) \end{aligned}$$

BP, summarized

- Compute all messages for all possible (i, j) pairs with,

$$\mathbf{m}_{i \to j}(h_j) = \sum_{h_i} \psi(h_i, h_j) \overbrace{\prod_{l \in \mathcal{N}(i) \setminus j} \mathbf{m}_{l \to i}(h_i)}^{\text{Incoming Messages to node } i}$$

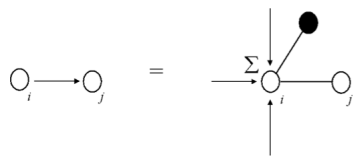


Figure is taken from Yedidia et al. 2001.

BP, summarized

- Compute all messages for all possible (i, j) pairs with,

$$\mathbf{m}_{i \to j}(h_j) = \sum_{h_i} \psi(h_i, h_j) \overbrace{\prod_{l \in \mathcal{N}(i) \setminus j} \mathbf{m}_{l \to i}(h_i)}^{\text{Incoming Messages to node } i}$$

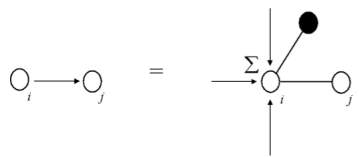


Figure is taken from Yedidia et al. 2001.

- The Belief for node i is $B(h_i) = p(h_i) = \prod_{j \in \mathcal{N}(i)} \mathbf{m}_{j \to i}(h_i)$.

BP, summarized

- Compute all messages for all possible (i, j) pairs with,

$$\mathbf{m}_{i \to j}(h_j) = \sum_{h_i} \psi(h_i, h_j) \overbrace{\prod_{l \in \mathcal{N}(i) \setminus j} \mathbf{m}_{l \to i}(h_i)}^{\text{Incoming Messages to node } i}$$

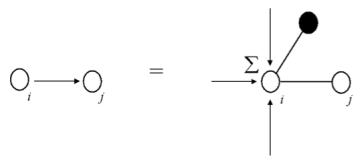


Figure is taken from Yedidia et al. 2001.

- The Belief for node i is $B(h_i) = p(h_i) = \prod_{j \in \mathcal{N}(i)} \mathbf{m}_{j \to i}(h_i)$.
- One pass from leaves to root and one pass from leaves to root, and we are done.

BP, summarized

- Compute all messages for all possible (i, j) pairs with,

$$\mathbf{m}_{i \to j}(h_j) = \sum_{h_i} \psi(h_i, h_j) \overbrace{\prod_{l \in \mathcal{N}(i) \setminus j} \mathbf{m}_{l \to i}(h_i)}^{\text{Incoming Messages to node } i}$$

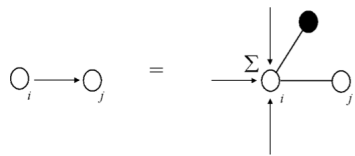


Figure is taken from Yedidia et al. 2001.

- The Belief for node i is $B(h_i) = p(h_i) = \prod_{j \in \mathcal{N}(i)} \mathbf{m}_{j \to i}(h_i)$.
- One pass from leaves to root and one pass from leaves to root, and we are done.
- BP converges to true beliefs in trees. What about general graphs?

Recap

- Dynamic Time Warping
  - It's a way to measure distances between sequences. / Way to measure distances between sequences
- HMMs
  - Probability distribution over sequences. Can be thought of generalization of DTW. / HMMs définissent une distribution de probabilité sur les séquences. Vous pouvez le voir comme étant une géneralization de DTW.
- HMM Applications
  - Speech Recognition, Human Action Recognition, Sequence Clustering, LLM State Transition Understanding, ...
- More Advanced HMMs
  - Mixture of HMMs, Swiching HMMs, HMMs with Mixture observations, Linear Dynamical Systems, MRFs, ...

Suggested reading

- The classic HMM Tutorial: `http://www.cs.ubc.ca/~murphyk/Bayes/rabiner.pdf`
- Bishop chapter 13.

Table of Contents

Dynamic Time Warping

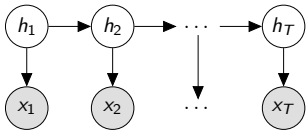
Hidden Markov Models
- Inference
- Learning
- Decoding
- HMM Applications
- HMM Variants

Autoregressive Models

Large Language Models
- Evaluating and Sampling
- Training an LLM

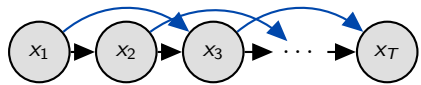
Two ways to remember the past

Hidden Markov Model



memory: a hidden state h_t (K values)

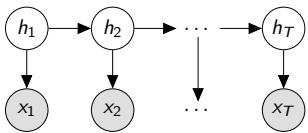
Autoregressive model



memory: the observed past $x_{t-1}, x_{t-2}, \ldots$

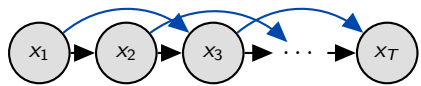
Two ways to remember the past

Hidden Markov Model



memory: a hidden state h_t (K values)

Autoregressive model

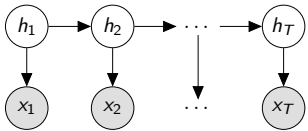


memory: the observed past $x_{t-1}, x_{t-2}, \ldots$

- The chain rule (week 2) is exact: $p(x_{1:T}) = \prod_t p(x_t|x_{<t})$. We only choose how much past to use, and the form of $p(x_t|\cdot)$: a Gaussian, a table, a network.
  - La règle de la chaîne est exacte: on choisit seulement le contexte et la forme de p.

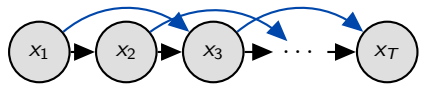
Two ways to remember the past

Hidden Markov Model



memory: a hidden state h_t (K values)

Autoregressive model



memory: the observed past $x_{t-1}, x_{t-2}, \ldots$

- The chain rule (week 2) is exact: $p(x_{1:T}) = \prod_t p(x_t|x_{<t})$. We only choose how much past to use, and the form of $p(x_t|\cdot)$: a Gaussian, a table, a network.
  - La règle de la chaîne est exacte: on choisit seulement le contexte et la forme de p.
- Exact log-likelihood $\sum_t \log p(x_t|x_{<t})$: no latent variable, no ELBO (week 8). Even the HMM fits: its forward pass computes $p(x_t|x_{<t})$.

Linear AR, now as a probabilistic model

- Model: $x_t = \sum_{k=1}^{K} a_k x_{t-k} + \varepsilon_t, \;\; \varepsilon_t \sim \mathcal{N}(0, \sigma^2)$. / Le modèle.

Linear AR, now as a probabilistic model

- Model: $x_t = \sum_{k=1}^{K} a_k x_{t-k} + \varepsilon_t, \quad \varepsilon_t \sim \mathcal{N}(0, \sigma^2)$. / Le modèle.
- Log-likelihood:

$$\log p(x_{1:T}) = -\frac{1}{2\sigma^2} \sum_t \left(x_t - \sum_{k=1}^{K} a_k x_{t-k} \right)^2 + \text{const}$$

  - Maximum likelihood = the least squares of week 4 ("LLM: Linear Language Model"). / Les moindres carrés de la semaine 4.

Linear AR, now as a probabilistic model

- Model: $x_t = \sum_{k=1}^{K} a_k x_{t-k} + \varepsilon_t, \;\; \varepsilon_t \sim \mathcal{N}(0, \sigma^2)$. / Le modèle.
- Log-likelihood:

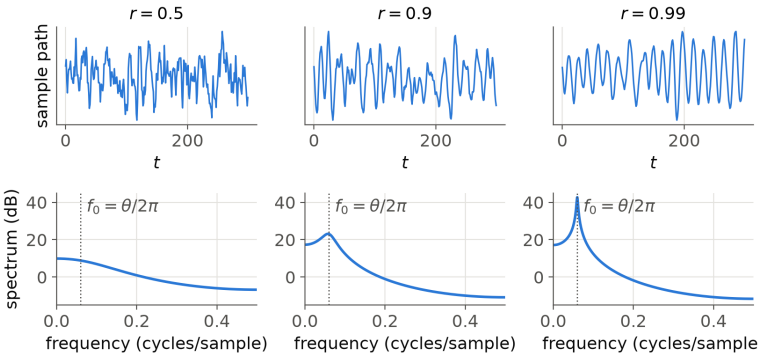
$$\log p(x_{1:T}) = -\frac{1}{2\sigma^2} \sum_t \left(x_t - \sum_{k=1}^{K} a_k x_{t-k} \right)^2 + \text{const}$$

  - Maximum likelihood = the least squares of week 4 ("LLM: Linear Language Model"). / Les moindres carrés de la semaine 4.
- New: we can **generate**. Draw ε_t, compute x_t, feed it back in. The same loop for every autoregressive model, LLMs included.
  - On peut générer: la même boucle pour tous les modèles autorégressifs.

An AR model is a filter

- $x_t - \sum_k a_k x_{t-k} = \varepsilon_t$: white noise through an all-pole (IIR) filter (week 3).
  - Du bruit blanc filtré par un filtre tout-pôles.

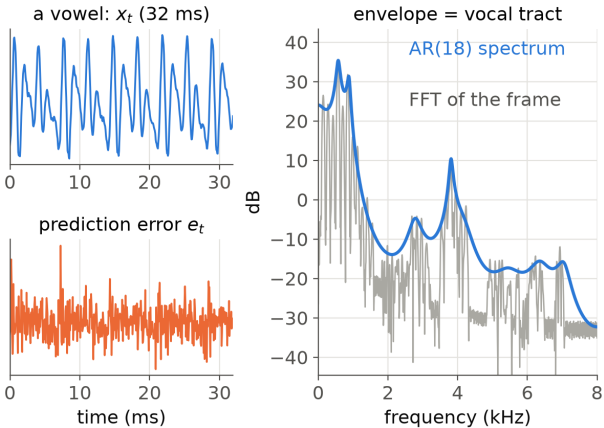
$$S(\omega) = \frac{\sigma^2}{\left|1 - \sum_k a_k e^{-j\omega k}\right|^2}$$



AR(2), poles $re^{\pm j\theta}$: $a_1 = 2r\cos\theta$, $a_2 = -r^2$. $r \to 1$: almost a sinusoid. $r \geq 1$: explodes (unstable).

Linear prediction of speech (LPC)

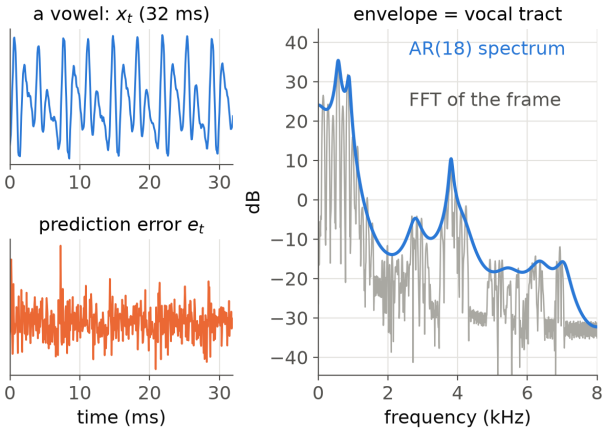
- Source-filter model: glottal pulses (or noise) through the vocal tract, an all-pole filter.
  - Source (glotte) + filtre (conduit vocal).



The LibriSpeech clip from week 3, $f_s = 16$ kHz, AR(18).

Linear prediction of speech (LPC)

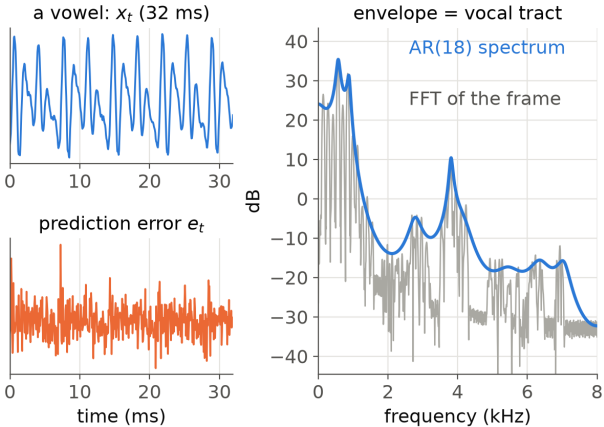
- Source-filter model: glottal pulses (or noise) through the vocal tract, an all-pole filter.
  - Source (glotte) + filtre (conduit vocal).
- Fit an AR(p) on each 20–30 ms frame, $p \approx f_s/1000 + 2$. The AR spectrum is the envelope (formants); the prediction error is the source.



The LibriSpeech clip from week 3, $f_s = 16$ kHz, AR(18).

Linear prediction of speech (LPC)

- Source-filter model: glottal pulses (or noise) through the vocal tract, an all-pole filter.
  - Source (glotte) + filtre (conduit vocal).
- Fit an AR(p) on each 20–30 ms frame, $p \approx f_s/1000 + 2$. The AR spectrum is the envelope (formants); the prediction error is the source.
- Phone codecs (CELP, e.g. AMR) send the filter and a compact excitation, not the waveform.



The LibriSpeech clip from week 3, $f_s = 16$ kHz, AR(18).

Discrete sequences: n-gram models

- x_t in a vocabulary $\mathcal{V}$ (characters, words), Markov of order $n-1$: $p(x_t|x_{<t}) \approx p(x_t|x_{t-n+1:t-1})$. A bigram is the HMM's A with nothing hidden.
  - Une chaîne de Markov sur les symboles: rien de caché.

Discrete sequences: n-gram models

- x_t in a vocabulary $\mathcal{V}$ (characters, words), Markov of order $n-1$:
  $p(x_t|x_{<t}) \approx p(x_t|x_{t-n+1:t-1})$. A bigram is the HMM's A with nothing hidden.
  - Une chaîne de Markov sur les symboles: rien de caché.
- Maximum likelihood = counting (week 2):
  $\widehat{p}(\mathtt{e} \mid \mathtt{capitain}) = N(\mathtt{capitaine}) \,/\, N(\mathtt{capitain}\cdot)$.

Discrete sequences: n-gram models

- x_t in a vocabulary $\mathcal{V}$ (characters, words), Markov of order $n-1$: $p(x_t|x_{<t}) \approx p(x_t|x_{t-n+1:t-1})$. A bigram is the HMM's A with nothing hidden.
  - Une chaîne de Markov sur les symboles: rien de caché.
- Maximum likelihood = counting (week 2): $\widehat{p}(\mathtt{e} \mid \mathtt{capitain}) = N(\mathtt{capitaine}) \,/\, N(\mathtt{capitain}\cdot)$.
- Trained on *Vingt mille lieues sous les mers* (Jules Verne, 783k characters), sampled from "Le capitaine Nemo":

$n = 2$: Le capitaine Nemouandenfainait, dene cougtét udinser, outispl, d'ause Le ", de los eramais anssepres t t s lennen les, vauri z e est s ...

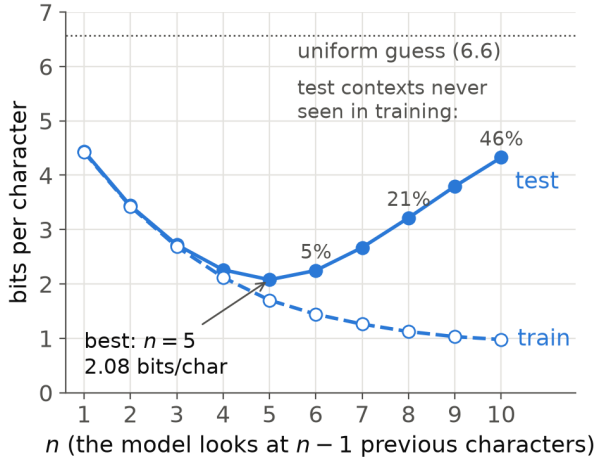
$n = 5$: Le capitaine Nemo y est se retourne, allâmes dont avoir épouvoire située du pieds, restre. Nul, nous racontre en régulier dessous attend. ...

$n = 8$: Le capitaine Nemo y entra. Nous ne renferme l'autre sur une colonne d'air qu'il marcha à raisonnement, il fallait pas été plus de mer ? – ...

$n = 8$: fluent, but pieces of the book glued together (up to 23 characters copied). / Des morceaux du livre recollés.

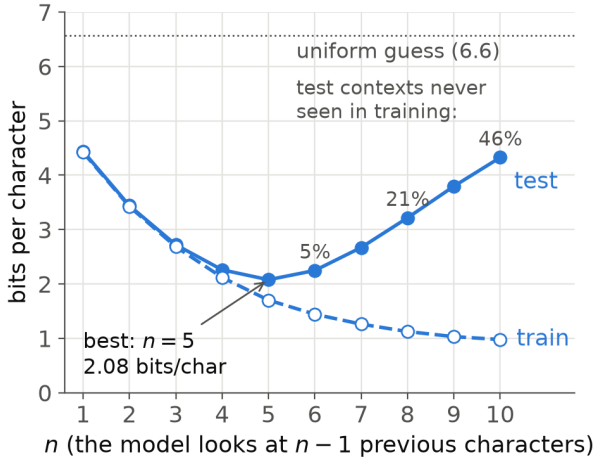
The curse of long contexts

- Score: **bits per character** on unseen text, $-\frac{1}{T}\sum_t \log_2 p(x_t|x_{<t})$. Uniform guess: $\log_2 95 = 6.6$.



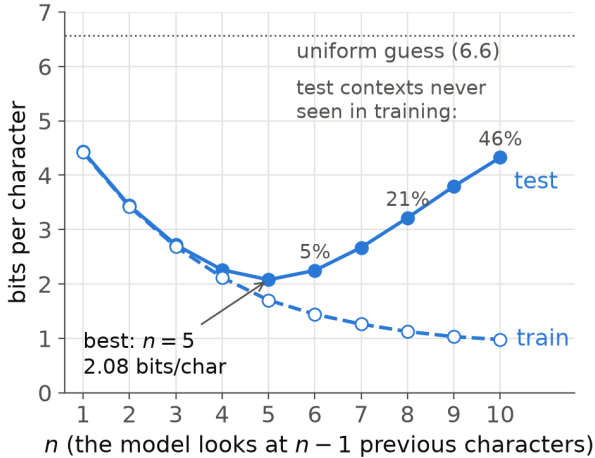
The curse of long contexts

- Score: **bits per character** on unseen text, $-\frac{1}{T}\sum_t \log_2 p(x_t|x_{<t})$. Uniform guess: $\log_2 95 = 6.6$.
- Train keeps improving, test gets worse after $n = 5$: overfitting.
  - Surapprentissage.



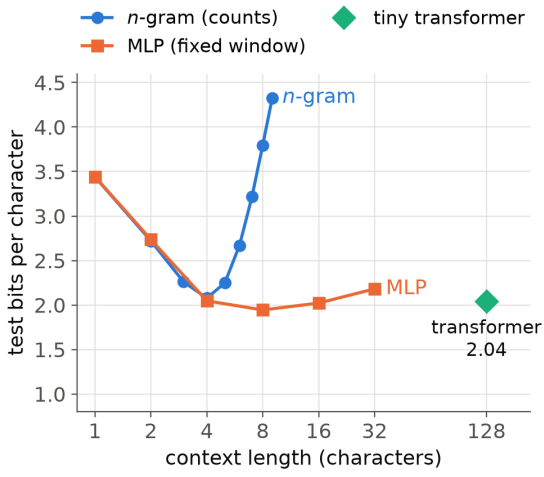
The curse of long contexts

- Score: **bits per character** on unseen text, $-\frac{1}{T}\sum_t \log_2 p(x_t|x_{<t})$. Uniform guess: $\log_2 95 = 6.6$.
- Train keeps improving, test gets worse after $n = 5$: overfitting.
  - Surapprentissage.
- Long test contexts were never seen. Smoothing (Dirichlet prior, week 2) only guesses; counts cannot share between similar contexts.



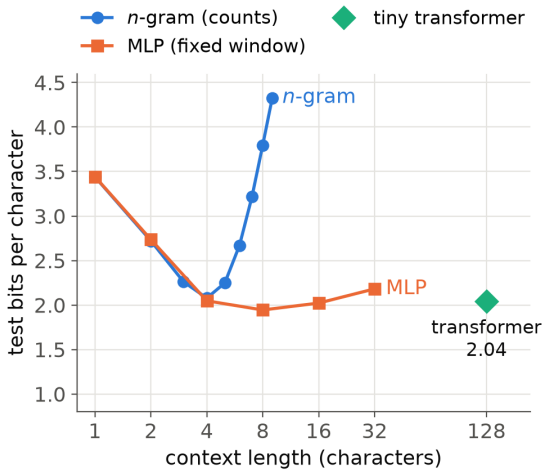
Neural autoregressive models

- Replace the table by a network: $p_\theta(x_t|x_{<t}) = \text{softmax}\big(f_\theta(x_{<t})\big)$. Similar contexts, similar predictions.
  - Remplacer la table par un réseau.



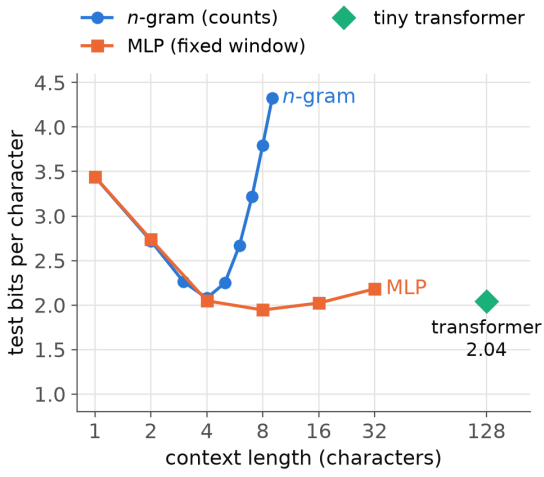
Neural autoregressive models

- Replace the table by a network: $p_\theta(x_t|x_{<t}) = \text{softmax}\big(f_\theta(x_{<t})\big)$. Similar contexts, similar predictions.
  - Remplacer la table par un réseau.
- Read the past with a window + MLP (Bengio et al., 2003), an RNN state, or a causal transformer (week 7).



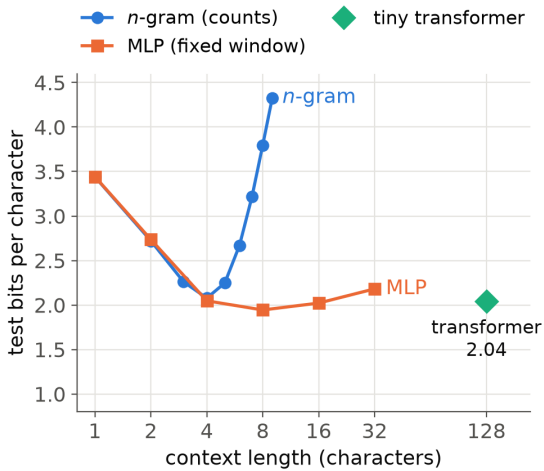
Neural autoregressive models

- Replace the table by a network: $p_\theta(x_t|x_{<t}) = \text{softmax}\big(f_\theta(x_{<t})\big)$. Similar contexts, similar predictions.
  - Remplacer la table par un réseau.
- Read the past with a window + MLP (Bengio et al., 2003), an RNN state, or a causal transformer (week 7).
- Train with the next-symbol cross-entropy $-\sum_t \log p_\theta(x_t|x_{<t})$ (week 6), on the true past (**teacher forcing**): all t in parallel.



Neural autoregressive models

- Replace the table by a network: $p_\theta(x_t|x_{<t}) = \text{softmax}\big(f_\theta(x_{<t})\big)$. Similar contexts, similar predictions.
  - Remplacer la table par un réseau.
- Read the past with a window + MLP (Bengio et al., 2003), an RNN state, or a causal transformer (week 7).
- Train with the next-symbol cross-entropy $-\sum_t \log p_\theta(x_t|x_{<t})$ (week 6), on the true past (**teacher forcing**): all t in parallel.
- MLP 1.94 bits/char ($K = 8$), best n-gram 2.08. Tiny transformer (0.83M parameters, 29 CPU-min) 2.04, still learning (train = test).

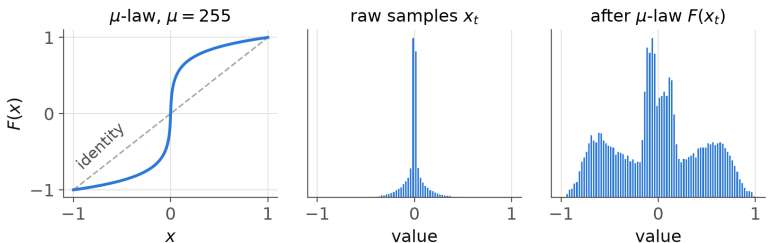


WaveNet: autoregressive on raw audio

- WaveNet (van den Oord et al., 2016): 16 000 samples/s, each predicted from the past with dilated causal convolutions (week 7).
  - Chaque échantillon prédit à partir des précédents.

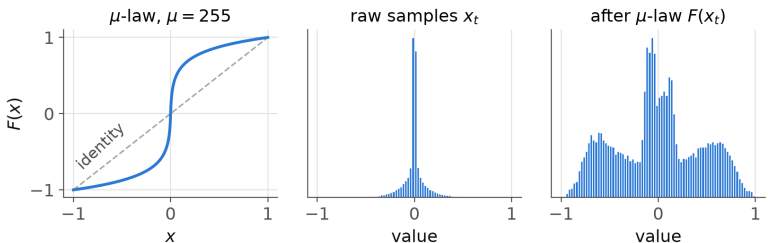
WaveNet: autoregressive on raw audio

- WaveNet (van den Oord et al., 2016): 16 000 samples/s, each predicted from the past with dilated causal convolutions (week 7).
  - Chaque échantillon prédit à partir des précédents.
- Not a Gaussian: μ-law to 256 levels, then a softmax, which can be multimodal (week 3 quantization).



WaveNet: autoregressive on raw audio

- WaveNet (van den Oord et al., 2016): 16 000 samples/s, each predicted from the past with dilated causal convolutions (week 7).
  - Chaque échantillon prédit à partir des précédents.
- Not a Gaussian: μ-law to 256 levels, then a softmax, which can be multimodal (week 3 quantization).



- Generation: 16 000 sequential network calls per second of audio. Slow! → GAN vocoders (HiFi-GAN, week 8).

Table of Contents

Dynamic Time Warping

Hidden Markov Models
- Inference
- Learning
- Decoding
- HMM Applications
- HMM Variants

Autoregressive Models

Large Language Models
- Evaluating and Sampling
- Training an LLM

Table of Contents

Dynamic Time Warping

Hidden Markov Models
- Inference
- Learning
- Decoding
- HMM Applications
- HMM Variants

Autoregressive Models

Large Language Models
- Evaluating and Sampling
- Training an LLM

How good is a language model? Perplexity

- Held-out cross-entropy, in bits per token (what we plotted for Verne):

$$H = -\frac{1}{T}\sum_{t=1}^{T} \log_2 p_\theta(x_t | x_{<t}), \qquad \textbf{perplexity} = 2^H$$

  - As unsure as a uniform choice among 2^H options. Verne, per character: uniform 95, n-gram 4.2, MLP 3.8, tiny transformer 4.1.

How good is a language model? Perplexity

- Held-out cross-entropy, in bits per token (what we plotted for Verne):

$$H = -\frac{1}{T}\sum_{t=1}^{T} \log_2 p_\theta(x_t | x_{<t}), \qquad \textbf{perplexity} = 2^H$$

  - As unsure as a uniform choice among 2^H options. Verne, per character: uniform 95, n-gram 4.2, MLP 3.8, tiny transformer 4.1.
- Maximum likelihood minimizes it, down to a floor (entropy, week 2): $H(p_{\text{data}}, p_\theta) = H(p_{\text{data}}) + \mathrm{KL}(p_{\text{data}} \,\|\, p_\theta) \geq H(p_{\text{data}})$.
  - L'entropie de la langue est un plancher.

How good is a language model? Perplexity

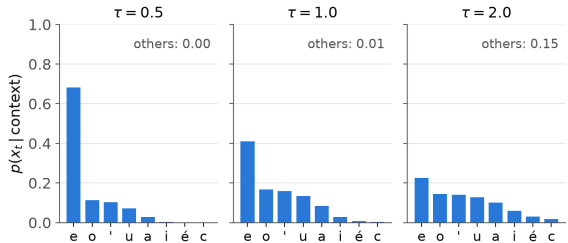
- Held-out cross-entropy, in bits per token (what we plotted for Verne):

$$H = -\frac{1}{T}\sum_{t=1}^{T} \log_2 p_\theta(x_t | x_{<t}), \qquad \textbf{perplexity} = 2^H$$

  - As unsure as a uniform choice among 2^H options. Verne, per character: uniform 95, n-gram 4.2, MLP 3.8, tiny transformer 4.1.
- Maximum likelihood minimizes it, down to a floor (entropy, week 2): $H(p_{\text{data}}, p_\theta) = H(p_{\text{data}}) + \mathrm{KL}(p_{\text{data}} \,\|\, p_\theta) \geq H(p_{\text{data}})$.
  - L'entropie de la langue est un plancher.
- Perplexity per token depends on the tokenizer: compare models in bits per character (or byte).

Sampling: temperature, top-k, top-p

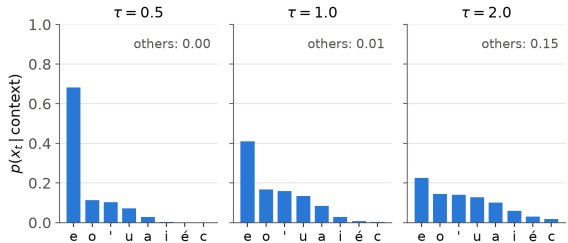
- Sample from $p_i \propto e^{z_i/\tau}$ (z: logits).
  $\tau \to 0$: greedy. $\tau = 1$: the model.
  $\tau > 1$: flatter.
  - La température aplatit ou accentue la distribution.



Our tiny transformer, next character after "Le capitaine Nemo s".

Sampling: temperature, top-k, top-p

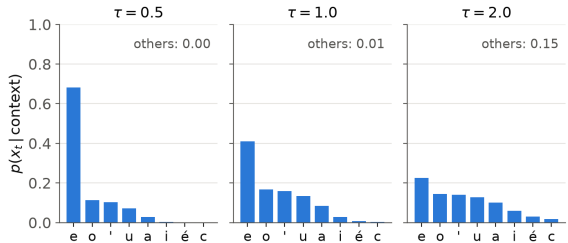
- Sample from $p_i \propto e^{z_i/\tau}$ (z: logits).
  $\tau \to 0$: greedy. $\tau = 1$: the model.
  $\tau > 1$: flatter.
  - La température aplatit ou accentue la distribution.
- Top-k: keep the k most likely. Top-p: keep the smallest set with probability $\geq p$ (Holtzman et al., 2020).



Our tiny transformer, next character after "Le capitaine Nemo s".

Sampling: temperature, top-k, top-p

- Sample from $p_i \propto e^{z_i/\tau}$ (z: logits).
  $\tau \to 0$: greedy. $\tau = 1$: the model.
  $\tau > 1$: flatter.
  - La température aplatit ou accentue la distribution.
- Top-k: keep the k most likely. Top-p: keep the smallest set with probability $\geq p$ (Holtzman et al., 2020).



Our tiny transformer, next character after "Le capitaine Nemo s".

$\tau = 0.3$: Le capitaine Nemo me reprit à la profondeur de l'Océan. Le capitaine Nemo me renferme de la mer de l'Amérique de l'Abraham-Lincoln de cent comme au ...

$\tau = 1.5$: Le capitaine Nemo oisaut. Jatr'aurayon fhéuillen, sa inoeur de sirvingt au2. Janlêmai le Nautilus, ni a milieu, ne pôle subiervemble êc z moçi'abats ...

Low τ: safe but repetitive. High τ: nonsense. / Faible τ: répétitif; fort τ: n'importe quoi.

Exposure bias, and why generation is slow

- Training (teacher forcing) sees the **true** past; generation sees **its own** samples, and errors compound (exposure bias).
  - À la génération, le modèle voit ses propres erreurs.

Exposure bias, and why generation is slow

- Training (teacher forcing) sees the **true** past; generation sees **its own** samples, and errors compound (exposure bias).
  - À la génération, le modèle voit ses propres erreurs.
- One forward pass per new token, one after the other. **KV cache**: only the new token goes through the network; keys and values of the past are stored ($2 \times \text{layers} \times d \times t$ numbers).
  - On garde les clés et valeurs du passé en mémoire.

Exposure bias, and why generation is slow

- Training (teacher forcing) sees the **true** past; generation sees **its own** samples, and errors compound (exposure bias).
  - À la génération, le modèle voit ses propres erreurs.
- One forward pass per new token, one after the other. **KV cache**: only the new token goes through the network; keys and values of the past are stored ($2 \times \text{layers} \times d \times t$ numbers).
  - On garde les clés et valeurs du passé en mémoire.
- Speculative decoding: a small model drafts k tokens, the big model checks all k in one parallel pass (Leviathan et al., 2023).

Table of Contents

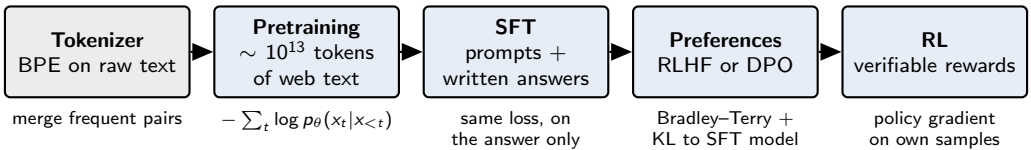
Dynamic Time Warping

Hidden Markov Models
- Inference
- Learning
- Decoding
- HMM Applications
- HMM Variants

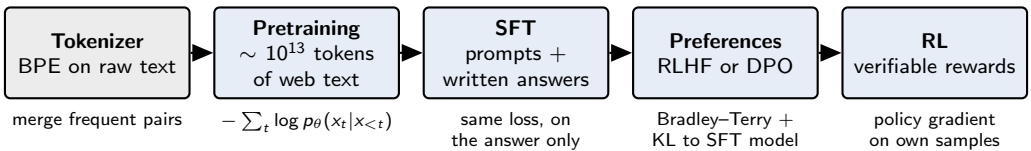
Autoregressive Models

Large Language Models
- Evaluating and Sampling
- Training an LLM

The recipe: one network, a sequence of losses



The recipe: one network, a sequence of losses



- The network is the causal transformer of week 7 all along. Each stage changes only the **data** and the **loss**, and every loss is one we know: cross-entropy, logistic regression, KL.
  - Le réseau ne change pas: seulement les données et la fonction de coût.

Tokenization: byte-pair encoding (BPE)

- Characters: sequences too long. Words: too many, unknown words. → **subwords**. / Des morceaux de mots.

Tokenization: byte-pair encoding (BPE)

- Characters: sequences too long. Words: too many, unknown words. → **subwords**. / Des morceaux de mots.
- BPE (Sennrich et al., 2016): start from single symbols (bytes in real LLMs), repeatedly **merge the most frequent adjacent pair**. On Verne, the first merges: `␣d` `es` `␣l` `en` `ai` `on` `␣p` `␣c` `␣s` `an` ...

Tokenization: byte-pair encoding (BPE)

- Characters: sequences too long. Words: too many, unknown words. → **subwords**. / Des morceaux de mots.
- BPE (Sennrich et al., 2016): start from single symbols (bytes in real LLMs), repeatedly **merge the most frequent adjacent pair**. On Verne, the first merges: `␣d` `es` `␣l` `en` `ai` `on` `␣p` `␣c` `␣s` `an` ...
- "Le capitaine Nemo nous regardait sans rien dire." after k merges:

$k = 10$: `␣` `L` `e` `␣c` `a` `p` `i` `t` `ai` `n` `e` `␣` `N` `e` `m` `o` `␣` `n` `o` `u` `s` `␣` `r` `e` `g` `a` `r` `d` `ai` `t` `␣s` `an` `s` `␣` `r` `i` `en` `␣d` `i` `r` `e` `.`

$k = 100$: `␣L` `e` `␣c` `ap` `it` `ai` `ne` `␣N` `em` `o` `␣n` `ous` `␣re` `g` `ar` `d` `ait` `␣s` `ans` `␣r` `i` `en` `␣d` `i` `re` `.`

$k = 1000$: `␣Le` `␣capitaine` `␣Nemo` `␣nous` `␣regard` `ait` `␣sans` `␣rien` `␣di` `re.`

Characters per token: 1.2, 1.7, 3.0 and 3.5 (after 2000 merges).

Tokenization: byte-pair encoding (BPE)

- Characters: sequences too long. Words: too many, unknown words. → **subwords**. / Des morceaux de mots.
- BPE (Sennrich et al., 2016): start from single symbols (bytes in real LLMs), repeatedly **merge the most frequent adjacent pair**. On Verne, the first merges: `␣d` `es` `␣l` `en` `ai` `on` `␣p` `␣c` `␣s` `an` ...
- "Le capitaine Nemo nous regardait sans rien dire." after k merges:

$k = 10$: `␣` `L` `e` `␣c` `a` `p` `i` `t` `ai` `n` `e` `␣` `N` `e` `m` `o` `␣` `n` `o` `u` `s` `␣` `r` `e` `g` `a` `r` `d` `ai` `t` `␣s` `an` `s` `␣` `r` `i` `en` `␣d` `i` `r` `e` `.`

$k = 100$: `␣L` `e` `␣c` `ap` `it` `ai` `ne` `␣N` `em` `o` `␣n` `ous` `␣re` `g` `ar` `d` `ait` `␣s` `ans` `␣r` `i` `en` `␣d` `i` `re` `.`

$k = 1000$: `␣Le` `␣capitaine` `␣Nemo` `␣nous` `␣regard` `ait` `␣sans` `␣rien` `␣di` `re.`

Characters per token: 1.2, 1.7, 3.0 and 3.5 (after 2000 merges).

- Real vocabularies: ∼50k (GPT-2) to 200k tokens. A learned compression, like "K-means is a tokenizer" (week 8).

Pretraining and scaling laws

- Trillions of tokens of filtered, deduplicated web text, plus code, books, math. The next-token loss of our tiny transformer, on a decoder-only transformer with N parameters.
  - Des billions de tokens du web; la même perte que notre petit transformer.

Pretraining and scaling laws

- Trillions of tokens of filtered, deduplicated web text, plus code, books, math. The next-token loss of our tiny transformer, on a decoder-only transformer with N parameters.
  - Des billions de tokens du web; la même perte que notre petit transformer.
- Compute for D tokens: $C \approx 6ND$ FLOPs. Our toy: 6×10^{13} (29 CPU-min). A 7B model on 2T tokens: 8×10^{22}, a billion times more.

Pretraining and scaling laws

- Trillions of tokens of filtered, deduplicated web text, plus code, books, math. The next-token loss of our tiny transformer, on a decoder-only transformer with N parameters.
  - Des billions de tokens du web; la même perte que notre petit transformer.
- Compute for D tokens: $C \approx 6ND$ FLOPs. Our toy: 6×10^{13} (29 CPU-min). A 7B model on 2T tokens: 8×10^{22}, a billion times more.
- The test loss falls as a power law in N, D and C (Kaplan et al., 2020). For a fixed C, grow both: $\sim$**20 tokens per parameter** (Hoffmann et al., 2022).
  - Chinchilla (70B, 1.4T tokens) beat Gopher (280B, 300B tokens) with the same compute.

Pretraining and scaling laws

- Trillions of tokens of filtered, deduplicated web text, plus code, books, math. The next-token loss of our tiny transformer, on a decoder-only transformer with N parameters.
  - Des billions de tokens du web; la même perte que notre petit transformer.
- Compute for D tokens: $C \approx 6ND$ FLOPs. Our toy: 6×10^{13} (29 CPU-min). A 7B model on 2T tokens: 8×10^{22}, a billion times more.
- The test loss falls as a power law in N, D and C (Kaplan et al., 2020). For a fixed C, grow both: $\sim$**20 tokens per parameter** (Hoffmann et al., 2022).
  - Chinchilla (70B, 1.4T tokens) beat Gopher (280B, 300B tokens) with the same compute.
- Today models see far more tokens (Llama 3 8B: 15T): a smaller model is cheaper to *use*.
  - On entraîne plus longtemps pour que le modèle coûte moins cher à utiliser.

Supervised fine-tuning (SFT)

- A pretrained model **continues documents**: ask it a question, it may answer with more questions.
  - Un modèle pré-entraîné continue des documents.

Supervised fine-tuning (SFT)

- A pretrained model **continues documents**: ask it a question, it may answer with more questions.
  - Un modèle pré-entraîné continue des documents.
- Data: (prompt, answer) pairs written by people, in a chat template:

  `<|user|>` Quelle est la capitale du Canada? `<|assistant|>` Ottawa. `<|end|>`

Supervised fine-tuning (SFT)

- A pretrained model **continues documents**: ask it a question, it may answer with more questions.
  - Un modèle pré-entraîné continue des documents.
- Data: (prompt, answer) pairs written by people, in a chat template:

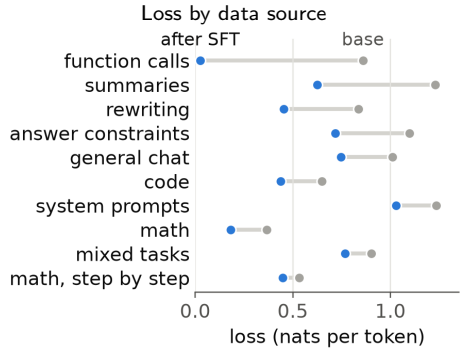
  `<|user|>` Quelle est la capitale du Canada? `<|assistant|>` Ottawa. `<|end|>`
- The same next-token loss, only on the answer tokens (InstructGPT: $\sim$13k examples):

$$\mathcal{L}_{\text{SFT}}(\theta) = -\sum_{t \in \text{answer}} \log p_\theta(y_t \mid \text{prompt}, y_{<t})$$

SFT in practice: Qwen3-1.7B-Base, before and after

950 chats (538k tokens, 1729 replies); loss in nats per token.

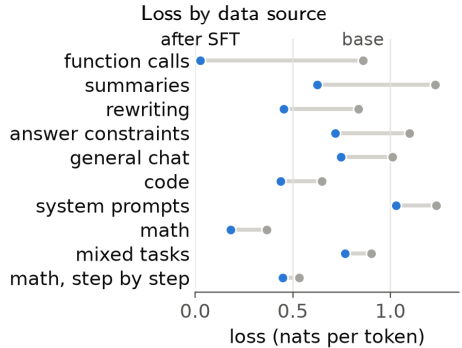
	Base	After SFT
Overall loss	0.958	0.704
Loss on the reply text only	0.906	0.706
Loss on the end-of-reply token <|im_end|>	16.88	0.34
Chance of stopping where a reply ends	0.000	0.895
Next-token guess correct	73.7%	78.4%
Replies to 25 test prompts (28 turns, greedy decoding):		
Ended with <|im_end|>	0 / 28	28 / 28
Cut at the 512-token limit	14 / 28	0 / 28



SFT in practice: Qwen3-1.7B-Base, before and after

950 chats (538k tokens, 1729 replies); loss in nats per token.

	Base	After SFT
Overall loss	0.958	0.704
Loss on the reply text only	0.906	0.706
Loss on the end-of-reply token <|im_end|>	16.88	0.34
Chance of stopping where a reply ends	0.000	0.895
Next-token guess correct	73.7%	78.4%
Replies to 25 test prompts (28 turns, greedy decoding):		
Ended with <|im_end|>	0 / 28	28 / 28
Cut at the 512-token limit	14 / 28	0 / 28

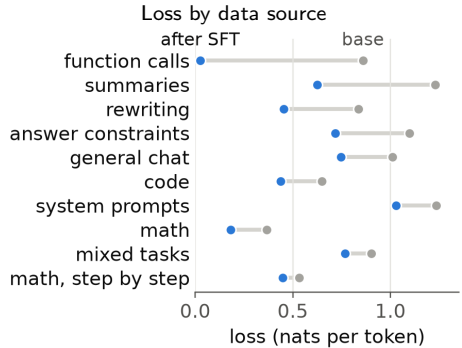


- The reply text improves, but the big change is **when to stop**: the base model never says <|im_end|>, and half its replies run to the limit.
  - Il apprend surtout à s'arrêter.

SFT in practice: Qwen3-1.7B-Base, before and after

950 chats (538k tokens, 1729 replies); loss in nats per token.

	Base	After SFT
Overall loss	0.958	0.704
Loss on the reply text only	0.906	0.706
Loss on the end-of-reply token <|im_end|>	16.88	0.34
Chance of stopping where a reply ends	0.000	0.895
Next-token guess correct	73.7%	78.4%
Replies to 25 test prompts (28 turns, greedy decoding):		
Ended with <|im_end|>	0 / 28	28 / 28
Cut at the 512-token limit	14 / 28	0 / 28



- The reply text improves, but the big change is **when to stop**: the base model never says <|im_end|>, and half its replies run to the limit.
  - Il apprend surtout à s'arrêter.
- Largest gains where the format is specific (function-call JSON: 0.86 → 0.03), smallest on step-by-step math, which the base model already predicts well.

SFT in practice: example conversations

Prompt	**Base** (/ = new line)	**After SFT** (same chat format, greedy decoding)
Who won the 2018 FIFA World Cup?	Who won the 2018 FIFA World Cup?umably / umably / umably / ... *(until the 512-token limit)*	The 2018 FIFA World Cup was won by France. They defeated Croatia 4-2 in the final match, held at the Maracana Stadium in Rio de Janeiro, Brazil. [...] France's victory marked their first World Cup title, and they went on to win the 2019 Confederations Cup.
List three fruits that are red. Reply with only the names, comma-separated, nothing else.	List three fruits [...] nothing else..Resolve / *[Hebrew for "your answer"]*: cela, kiwi, and strawberry.	apple, strawberry, raspberry
System: You are a pirate. Always talk like a pirate. What's the weather usually like at sea?	It's usually calm and clear..AutoScaleMode / .AutoScaleMode / What's the weather usually like at sea?.AutoScaleMode / ... *(until the limit)*	Ahoy, me hearties! The weather at sea be a mystery, but I can tell ye some tales of the storms and the calm seas. / 1. Storms: These be the worst of the weather, with dark clouds, howling winds, and waves that can capsize a ship. [...]

SFT in practice: example conversations

Prompt	**Base** (/ = new line)	**After SFT** (same chat format, greedy decoding)
Who won the 2018 FIFA World Cup?	Who won the 2018 FIFA World Cup?umably / umably / umably / ... *(until the 512-token limit)*	The 2018 FIFA World Cup was won by France. They defeated Croatia 4-2 in the final match, held at the Maracana Stadium in Rio de Janeiro, Brazil. [...] France's victory marked their first World Cup title, and they went on to win the 2019 Confederations Cup.
List three fruits that are red. Reply with only the names, comma-separated, nothing else.	List three fruits [...] nothing else..Resolve / *[Hebrew for "your answer"]*: cela, kiwi, and strawberry.	apple, strawberry, raspberry
System: You are a pirate. Always talk like a pirate. What's the weather usually like at sea?	It's usually calm and clear..AutoScaleMode / .AutoScaleMode / What's the weather usually like at sea?.AutoScaleMode / ... *(until the limit)*	Ahoy, me hearties! The weather at sea be a mystery, but I can tell ye some tales of the storms and the calm seas. / 1. Storms: These be the worst of the weather, with dark clouds, howling winds, and waves that can capsize a ship. [...]

- SFT teaches the **format**: answer, follow instructions, keep a persona, stop. Not the facts: fluent, confident, partly made up (final in Moscow; France's 2nd title).

SFT in practice: example conversations

Prompt	**Base** (/ = new line)	**After SFT** (same chat format, greedy decoding)
Who won the 2018 FIFA World Cup?	Who won the 2018 FIFA World Cup?umably / umably / umably / ... *(until the 512-token limit)*	The 2018 FIFA World Cup was won by France. They defeated Croatia 4-2 in the final match, held at the Maracana Stadium in Rio de Janeiro, Brazil. [...] France's victory marked their first World Cup title, and they went on to win the 2019 Confederations Cup.
List three fruits that are red. Reply with only the names, comma-separated, nothing else.	List three fruits [...] nothing else..Resolve / *[Hebrew for "your answer"]*: cela, kiwi, and strawberry.	apple, strawberry, raspberry
System: You are a pirate. Always talk like a pirate. What's the weather usually like at sea?	It's usually calm and clear..AutoScaleMode / .AutoScaleMode / What's the weather usually like at sea?.AutoScaleMode / ... *(until the limit)*	Ahoy, me hearties! The weather at sea be a mystery, but I can tell ye some tales of the storms and the calm seas. / 1. Storms: These be the worst of the weather, with dark clouds, howling winds, and waves that can capsize a ship. [...]

- SFT teaches the **format**: answer, follow instructions, keep a persona, stop. Not the facts: fluent, confident, partly made up (final in Moscow; France's 2nd title).
- Knowledge comes from pretraining; 1000 examples can teach the format (LIMA, 2023).
  - La forme vient du SFT, les connaissances du pré-entraînement.

RLHF: learning from preferences

- Comparing two answers is easier than writing one: for prompt x, a person prefers y_w over y_l.
  - Comparer deux réponses est plus facile qu'en écrire une.

RLHF: learning from preferences

- Comparing two answers is easier than writing one: for prompt x, a person prefers y_w over y_l.
  - Comparer deux réponses est plus facile qu'en écrire une.
- Reward model (Bradley–Terry), logistic regression on score differences (week 6): $p(y_w \succ y_l \mid x) = \sigma\big(r_\phi(x, y_w) - r_\phi(x, y_l)\big)$

RLHF: learning from preferences

- Comparing two answers is easier than writing one: for prompt x, a person prefers y_w over y_l.
  - Comparer deux réponses est plus facile qu'en écrire une.
- Reward model (Bradley–Terry), logistic regression on score differences (week 6): $p(y_w \succ y_l \mid x) = \sigma\big(r_\phi(x, y_w) - r_\phi(x, y_l)\big)$
- Then maximize the reward, staying close to the SFT model π_{ref}:

$$\max_\theta \; \mathbb{E}_{y \sim \pi_\theta(\cdot|x)}\big[r_\phi(x, y)\big] - \beta \, \mathrm{KL}\big(\pi_\theta(\cdot|x) \,\|\, \pi_{\text{ref}}(\cdot|x)\big)$$

RLHF: learning from preferences

- Comparing two answers is easier than writing one: for prompt x, a person prefers y_w over y_l.
  - Comparer deux réponses est plus facile qu'en écrire une.
- Reward model (Bradley–Terry), logistic regression on score differences (week 6): $p(y_w \succ y_l \mid x) = \sigma\big(r_\phi(x, y_w) - r_\phi(x, y_l)\big)$
- Then maximize the reward, staying close to the SFT model π_{ref}:

$$\max_\theta \; \mathbb{E}_{y \sim \pi_\theta(\cdot|x)}\big[r_\phi(x, y)\big] - \beta \, \mathrm{KL}\big(\pi_\theta(\cdot|x) \,\|\, \pi_{\text{ref}}(\cdot|x)\big)$$

- Without the KL, the policy exploits the mistakes of r_ϕ (**reward hacking**). A KL anchor, as in the ELBO (week 8). InstructGPT, Ouyang et al., 2022.

Differentiating through sampling: the policy gradient

- The answer y is **sampled** discrete tokens: no reparameterization trick (week 8). Use $\nabla\pi = \pi\,\nabla\log\pi$:

$$\nabla_\theta\,\mathbb{E}_{y\sim\pi_\theta}\big[r(x,y)\big] = \sum_y r(x,y)\,\nabla_\theta\pi_\theta(y|x) = \mathbb{E}_{y\sim\pi_\theta}\big[r(x,y)\,\nabla_\theta\log\pi_\theta(y|x)\big]$$

  - Pas d'astuce de reparamétrisation pour des tokens discrets.

Differentiating through sampling: the policy gradient

- The answer y is **sampled** discrete tokens: no reparameterization trick (week 8). Use $\nabla\pi = \pi\,\nabla\log\pi$:

$$\nabla_\theta\,\mathbb{E}_{y\sim\pi_\theta}\big[r(x,y)\big] = \sum_y r(x,y)\,\nabla_\theta\pi_\theta(y|x) = \mathbb{E}_{y\sim\pi_\theta}\big[r(x,y)\,\nabla_\theta\log\pi_\theta(y|x)\big]$$

  - Pas d'astuce de reparamétrisation pour des tokens discrets.
- With $\log\pi_\theta(y|x) = \sum_t \log\pi_\theta(y_t \mid x, y_{<t})$ (the chain rule again): a next-token loss on the model's **own samples**, weighted by their reward. Use $r - b$ with a baseline b: less variance.

Differentiating through sampling: the policy gradient

- The answer y is **sampled** discrete tokens: no reparameterization trick (week 8). Use $\nabla\pi = \pi\,\nabla \log \pi$:

$$\nabla_\theta\, \mathbb{E}_{y\sim\pi_\theta}\big[r(x,y)\big] = \sum_y r(x,y)\,\nabla_\theta \pi_\theta(y|x) = \mathbb{E}_{y\sim\pi_\theta}\big[r(x,y)\,\nabla_\theta \log \pi_\theta(y|x)\big]$$

  - Pas d'astuce de reparamétrisation pour des tokens discrets.
- With $\log \pi_\theta(y|x) = \sum_t \log \pi_\theta(y_t \mid x, y_{<t})$ (the chain rule again): a next-token loss on the model's **own samples**, weighted by their reward. Use $r - b$ with a baseline b: less variance.
- RLHF optimizes this with PPO (clipped updates). / Une perte sur ses propres échantillons, pondérée par la récompense.

DPO: preferences without RL

- The RLHF objective has a closed-form optimum: $\pi^*(y|x) = \frac{1}{Z(x)}\,\pi_{\text{ref}}(y|x)\,e^{r(x,y)/\beta}$, so $r(x,y) = \beta \log \dfrac{\pi^*(y|x)}{\pi_{\text{ref}}(y|x)} + \beta \log Z(x)$.

DPO: preferences without RL

- The RLHF objective has a closed-form optimum: $\pi^*(y|x) = \frac{1}{Z(x)}\,\pi_{\text{ref}}(y|x)\,e^{r(x,y)/\beta}$, so $r(x,y) = \beta \log \frac{\pi^*(y|x)}{\pi_{\text{ref}}(y|x)} + \beta \log Z(x)$.
- Put it into Bradley–Terry; $Z(x)$ cancels in the difference (Rafailov et al., 2023):

$$\mathcal{L}_{\text{DPO}}(\theta) = -\log \sigma \left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right)$$

DPO: preferences without RL

- The RLHF objective has a closed-form optimum: $\pi^*(y|x) = \frac{1}{Z(x)}\,\pi_{\text{ref}}(y|x)\,e^{r(x,y)/\beta}$, so $r(x,y) = \beta \log \frac{\pi^*(y|x)}{\pi_{\text{ref}}(y|x)} + \beta \log Z(x)$.
- Put it into Bradley–Terry; $Z(x)$ cancels in the difference (Rafailov et al., 2023):

$$\mathcal{L}_{\text{DPO}}(\theta) = -\log \sigma \left(\beta \log \frac{\pi_\theta(y_w|x)}{\pi_{\text{ref}}(y_w|x)} - \beta \log \frac{\pi_\theta(y_l|x)}{\pi_{\text{ref}}(y_l|x)} \right)$$

- A classification loss on preference pairs: no reward model, no sampling during training.
  - Ni modèle de récompense, ni échantillonnage.

RL with verifiable rewards: reasoning models

- For math and code the reward can be **checked**: right final answer, unit tests pass. No reward model to hack.
  - La récompense est vérifiable.

RL with verifiable rewards: reasoning models

- For math and code the reward can be **checked**: right final answer, unit tests pass. No reward model to hack.
  - La récompense est vérifiable.
- Sample G answers per prompt; the group mean is the baseline (GRPO, Shao et al., 2024): $A_i = \big(r_i - \text{mean}(r_{1:G})\big)/\text{std}(r_{1:G})$

RL with verifiable rewards: reasoning models

- For math and code the reward can be **checked**: right final answer, unit tests pass. No reward model to hack.
  - La récompense est vérifiable.
- Sample G answers per prompt; the group mean is the baseline (GRPO, Shao et al., 2024): $A_i = \big(r_i - \text{mean}(r_{1:G})\big)/\text{std}(r_{1:G})$
- With RL only, the model learned to write long chains of reasoning before answering (DeepSeek-R1-Zero, 2025).

RL with verifiable rewards: reasoning models

- For math and code the reward can be **checked**: right final answer, unit tests pass. No reward model to hack.
  - La récompense est vérifiable.
- Sample G answers per prompt; the group mean is the baseline (GRPO, Shao et al., 2024): $A_i = \big(r_i - \text{mean}(r_{1:G})\big)/\text{std}(r_{1:G})$
- With RL only, the model learned to write long chains of reasoning before answering (DeepSeek-R1-Zero, 2025).
- It trains on its own samples: a remedy for exposure bias. / Il s'entraîne sur ses propres échantillons.

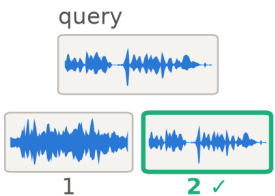
Our work: Audio-R1 (Jeong et al., 2026)

- What can be checked for audio? Labeled audio questions are costly, and a right answer does not prove the model listened.

Our work: Audio-R1 (Jeong et al., 2026)

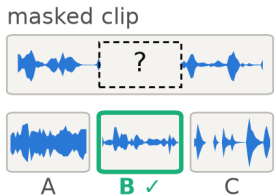
- What can be checked for audio? Labeled audio questions are costly, and a right answer does not prove the model listened.
- **Self-supervised tasks** (as in wav2vec 2.0, HuBERT): the answer y^* is known by construction. A free verifier.
  - La réponse est connue par construction: aucune annotation.

Identity detection



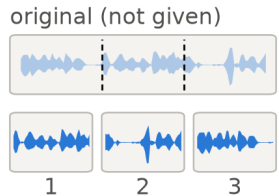
Same recording as the query?

Region inpainting



Which one was cut out of the gap?

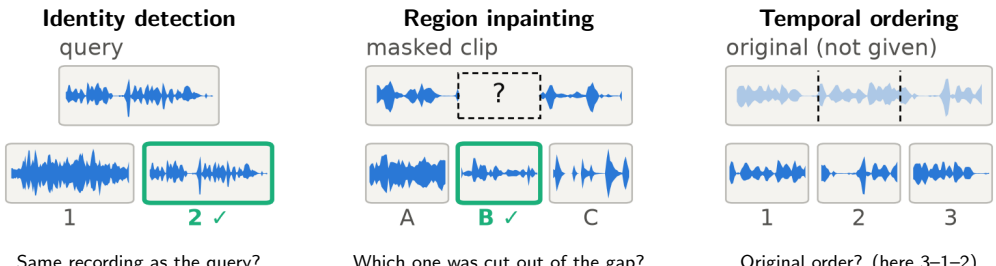
Temporal ordering



Original order? (here 3–1–2)

Our work: Audio-R1 (Jeong et al., 2026)

- What can be checked for audio? Labeled audio questions are costly, and a right answer does not prove the model listened.
- **Self-supervised tasks** (as in wav2vec 2.0, HuBERT): the answer y^* is known by construction. A free verifier.
  - La réponse est connue par construction: aucune annotation.



- Reward: 1 if the answer is right, +0.2 for the `<think>...<answer>` format. GRPO with 8 samples per prompt, KL to π_{ref}; 24k unlabeled recordings.

Audio-R1: what RL adds over SFT

- Four open audio LMs (Gemma 4, AF-Next-Think, Qwen2.5-Omni, Qwen3-Omni) on three benchmarks never trained on (MMAR, MMAU-mini, MMAU-Pro): $+1.0$ to $+4.4$ points in all 12 cases.
  - Qwen3-Omni + Audio-R1: 70.3 / 77.5 / 63.6, above Gemini 2.5 Flash (68.9 / 71.8 / 59.2).

Audio-R1: what RL adds over SFT

- Four open audio LMs (Gemma 4, AF-Next-Think, Qwen2.5-Omni, Qwen3-Omni) on three benchmarks never trained on (MMAR, MMAU-mini, MMAU-Pro): $+1.0$ to $+4.4$ points in all 12 cases.
  - Qwen3-Omni + Audio-R1: 70.3 / 77.5 / 63.6, above Gemini 2.5 Flash (68.9 / 71.8 / 59.2).
- Each task helps alone; the three together are best.

Audio-R1: what RL adds over SFT

- Four open audio LMs (Gemma 4, AF-Next-Think, Qwen2.5-Omni, Qwen3-Omni) on three benchmarks never trained on (MMAR, MMAU-mini, MMAU-Pro): +1.0 to +4.4 points in all 12 cases.
  - Qwen3-Omni + Audio-R1: 70.3 / 77.5 / 63.6, above Gemini 2.5 Flash (68.9 / 71.8 / 59.2).
- Each task helps alone; the three together are best.
- Same tasks, same answers, two losses (Qwen2.5-Omni, accuracy in %):

	Self-supervised tasks (trained on)	**Benchmarks** (never seen)
Base model	65.7	62.2
SFT: imitate the answer	85.7	63.7
Audio-R1: RL on its own reasoning	**87.0**	**66.2**

Audio-R1: what RL adds over SFT

- Four open audio LMs (Gemma 4, AF-Next-Think, Qwen2.5-Omni, Qwen3-Omni) on three benchmarks never trained on (MMAR, MMAU-mini, MMAU-Pro): +1.0 to +4.4 points in all 12 cases.
  - Qwen3-Omni + Audio-R1: 70.3 / 77.5 / 63.6, above Gemini 2.5 Flash (68.9 / 71.8 / 59.2).
- Each task helps alone; the three together are best.
- Same tasks, same answers, two losses (Qwen2.5-Omni, accuracy in %):

	Self-supervised tasks (trained on)	**Benchmarks** (never seen)
Base model	65.7	62.2
SFT: imitate the answer	85.7	63.7
Audio-R1: RL on its own reasoning	**87.0**	**66.2**

- Both learn the tasks; RL transfers much more (+4.0 vs +1.5). An LLM judge that hears the audio also rates Audio-R1's reasoning more complete, coherent and grounded than both; SFT's is mostly rated below the base model's.
  - Mêmes données, deux pertes: c'est l'apprentissage par renforcement qui généralise.

Training an LLM: summary

Stage	Data	Loss	What it learns
Tokenizer (BPE)	raw text	greedy pair merges	the units
Pretraining	$\sim 10^{13}$ web tokens	$-\sum_t \log p_\theta(x_t \mid x_{<t})$	language, facts
SFT	prompts + written answers	same, on the answer	the assistant format
RLHF / DPO	preference pairs	Bradley–Terry + KL	what people prefer
RL (verifiable)	checkable problems	policy gradient, group baseline	reasoning

Training an LLM: summary

Stage	Data	Loss	What it learns
Tokenizer (BPE)	raw text	greedy pair merges	the units
Pretraining	$\sim 10^{13}$ web tokens	$-\sum_t \log p_\theta(x_t \mid x_{<t})$	language, facts
SFT	prompts + written answers	same, on the answer	the assistant format
RLHF / DPO	preference pairs	Bradley–Terry + KL	what people prefer
RL (verifiable)	checkable problems	policy gradient, group baseline	reasoning

- Underneath all of it: the chain rule $p(x_{1:T}) = \prod_t p(x_t \mid x_{<t})$. / En dessous de tout: la règle de la chaîne.

Training an LLM: summary

Stage	Data	Loss	What it learns
Tokenizer (BPE)	raw text	greedy pair merges	the units
Pretraining	$\sim 10^{13}$ web tokens	$-\sum_t \log p_\theta(x_t \mid x_{<t})$	language, facts
SFT	prompts + written answers	same, on the answer	the assistant format
RLHF / DPO	preference pairs	Bradley–Terry + KL	what people prefer
RL (verifiable)	checkable problems	policy gradient, group baseline	reasoning

- Underneath all of it: the chain rule $p(x_{1:T}) = \prod_t p(x_t \mid x_{<t})$. / En dessous de tout: la règle de la chaîne.
- Train your own: nanoGPT (`https://github.com/karpathy/nanoGPT`) on a book, like our Verne model.

Suggested reading (autoregressive models and LLMs)

- J. Makhoul, Linear Prediction: A Tutorial Review, Proc. IEEE, 1975.
- Bengio et al., A Neural Probabilistic Language Model, JMLR 2003.
- van den Oord et al., WaveNet: A Generative Model for Raw Audio, arXiv:1609.03499, 2016.
- Holtzman et al., The Curious Case of Neural Text Degeneration, ICLR 2020.
- Sennrich, Haddow & Birch, Neural Machine Translation of Rare Words with Subword Units, ACL 2016.
- Kaplan et al., Scaling Laws for Neural Language Models, arXiv:2001.08361, 2020.
- Hoffmann et al., Training Compute-Optimal Large Language Models, NeurIPS 2022.
- Ouyang et al., Training Language Models to Follow Instructions with Human Feedback, NeurIPS 2022.
- Zhou et al., LIMA: Less Is More for Alignment, NeurIPS 2023.
- Rafailov et al., Direct Preference Optimization, NeurIPS 2023.
- DeepSeek-AI, DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning, 2025.
- Jeong, Ravanelli, Wang & Subakan, Audio-R1: Audio Reasoning through Reinforcement Learning on Self-Supervised Tasks, arXiv:2610.34609, 2026.

Next week

- Graph Signal Processing / Machine Learning