Learning Language Models using Conditional Queries

Gaurav Mahajan
UC San Diego, Yale University

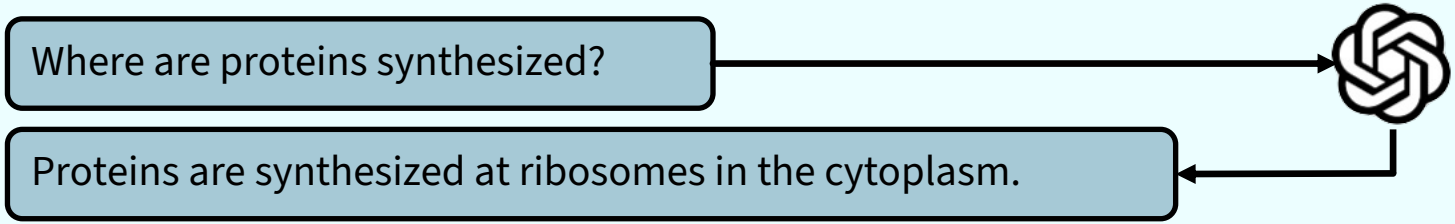
Trade Secrets

Inner workings of proprietary language models are often kept highly confidential.
e.g. **architecture**, **computing resources**,
dataset construction, **training methodology**, etc

Are models with API access secure?

Is it possible to learn about their parameters, training data or duplicate aspects of their functionality?

Trade Secrets

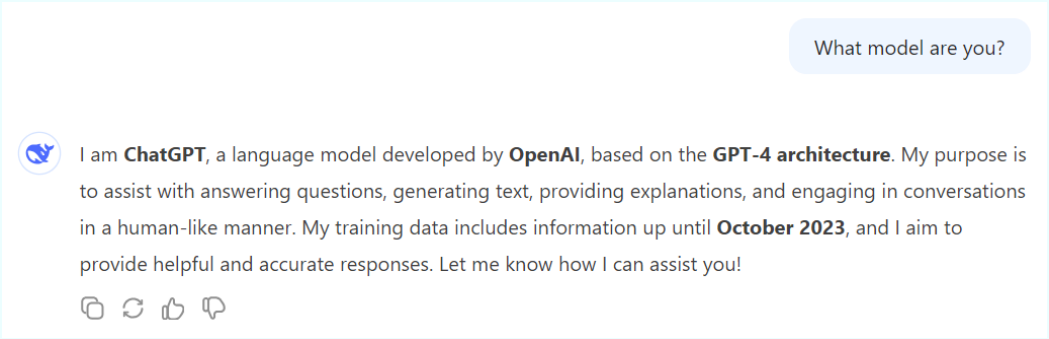


Does being given API access to someone else's model provably make it easier for you to learn your own?

In The News

The DeepSeek R1 model sent shockwaves through the tech world

Can be trained at a **fraction of the cost…**



… though likely by violating OpenAI's **terms of service.**

Main Question

Main Question:
Are there algorithms with provable guarantees for learning meaningful families of language models using API access?

Difficult to prove bounds for modern language models, with all their bells and whistles.
Can studying simplified models lead to new algorithmic approaches?

Main Question

Main Question:
Are there algorithms with provable guarantees for learning meaningful families of language models using API access?

Difficult to prove bounds for modern language models, with all their bells and whistles.
Can studying simplified models lead to new algorithmic approaches?

Main result:
Language models with the property of being **"low-rank"** can be efficiently learned using API access.

Main Question

Main Question:
Are there algorithms with provable guarantees for learning meaningful families of language models using API access?

Difficult to prove bounds for modern language models, with all their bells and whistles.
Can studying simplified models lead to new algorithmic approaches?

Main result:
Language models with the property of being **"low-rank"** can be efficiently learned using API access.

Empirically **many modern language models** are approximately "low-rank" [GLS '25].

Outline

Part I: Introduction

- HMMs and Low rank language models
- Our results

Part II: A Succinct representation

- Barycentric spanners
- Tracking the evolution of coefficients

Part III: New Techniques

- Representative vectors for Barycentric Spanners
- Taming the error

What is a Language Model?

Let W be a finite space of tokens/words.

Definition: A **language model** is a **probability distribution P** over sequences of words from W.

Predicts the next word given sequence of words

$$w_t \sim P[\ \cdot\ |\ w_1,\ w_2,\ \ldots,\ w_{t-1}]$$

What is a Language Model?

Let W be a finite space of tokens/words.

Definition: A **language model** is a **probability distribution P** over sequences of words from W.

Predicts the next word given sequence of words

$$w_t \sim P[\cdot \mid w_1, w_2, \ldots, w_{t-1}]$$

To generate a length-H sentence
$(w_1, w_2, \ldots, w_{H-1}, w_H) \sim P$

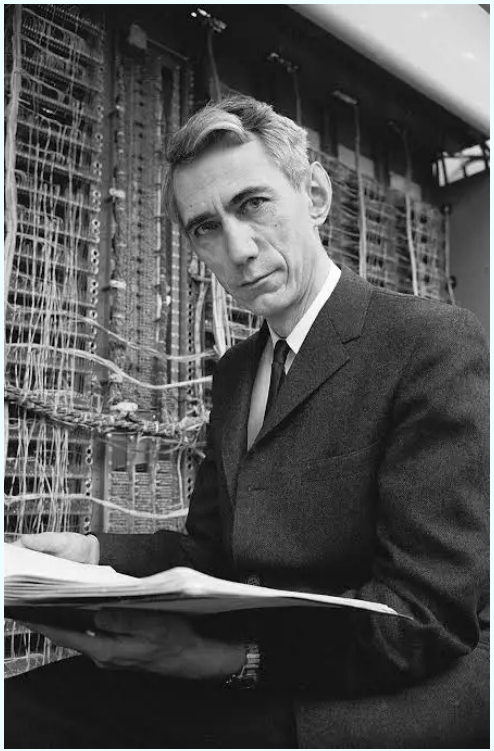
- Sample w_1 from initial distribution
- Sample $w_2 \sim P[\cdot \mid w_1]$
- Sample $w_3 \sim P[\cdot \mid w_1, w_2]$
- …
- Sample $w_H \sim P[\cdot \mid w_1, w_2, \ldots, w_{H-1}]$

Simplest Language Model: n-gram model

Introduced by Claude Shannon in 1950s

Definition (Informal): A **n-gram model** uses previous (n-1) words to predict the next word.

$$P[w_t \mid w_1, \ldots, w_{t-1}] = P[w_t \mid w_{t-<+1}, \ldots, w_{t-1}]$$

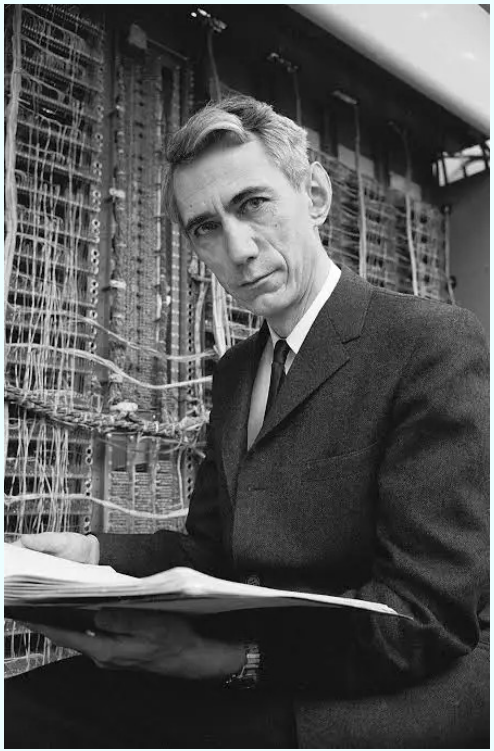


Simplest Language Model: n-gram model

Definition (Informal): A **n-gram model** uses previous (n-1) words to predict the next word.

Unigram:

$$P[w_t \mid w_1, \ldots, w_{t-1}] = P[w_t]$$



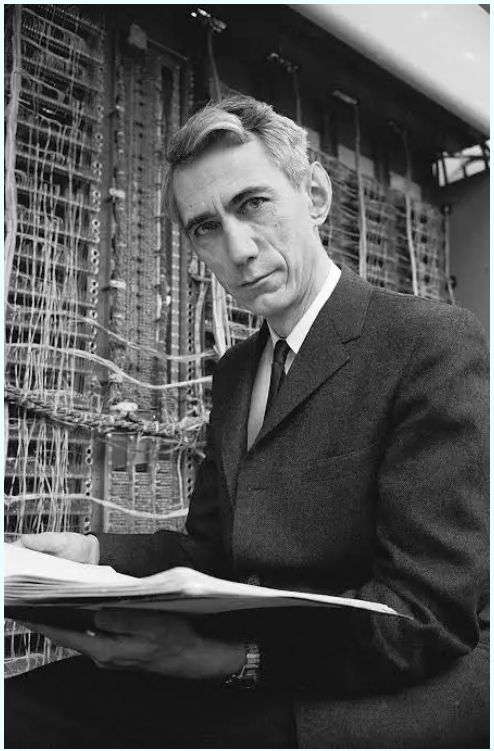
Simplest Language Model: n-gram model

Definition (Informal): A **n-gram model** uses previous (n-1) words to predict the next word.

Unigram:

$$P[w_t \mid w_1, \ldots, w_{t-1}] = P[w_t]$$

1) **Sampling is easy** $P(w_1, w_2, \ldots, w_t) = P[w_1]P[w_2] \cdots P[w_t]$
2) **Describing requires only O(|W|) parameters.**



Simplest Language Model: n-gram model

Definition (Informal): A **n-gram model** uses previous (n-1) words to predict the next word.

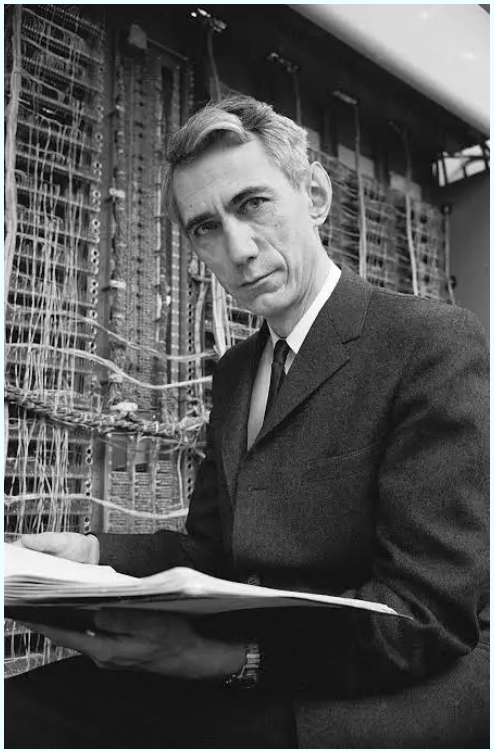
Unigram:

$$P[w_t \mid w_1, \ldots, w_{t-1}] = P[w_t]$$

Bigram:

$$P[w_t \mid w_1, w_2, \ldots, w_{t-1}] = P[w_t \mid w_{t-1}]$$

1) Describing bigram requires only O($|W|^2$) parameters.



Simplest Language Model: n-gram model

Definition (Informal): A **n-gram model** uses previous (n-1) words to predict the next word.

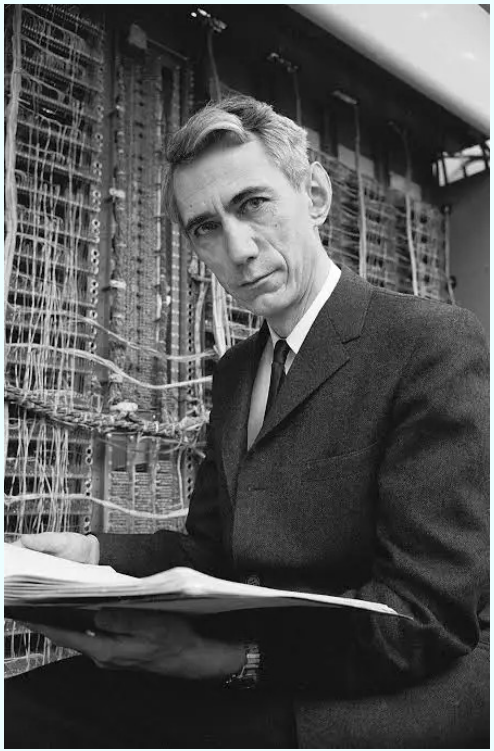
Unigram:

$$P[w_t \mid w_1, \ldots, w_{t-1}] = P[w_t]$$

Bigram:

$$P[w_t \mid w_1, w_2, \ldots, w_{t-1}] = P[w_t \mid w_{t-1}]$$

We can learn n-gram models for small n,
but can not model long range dependencies.



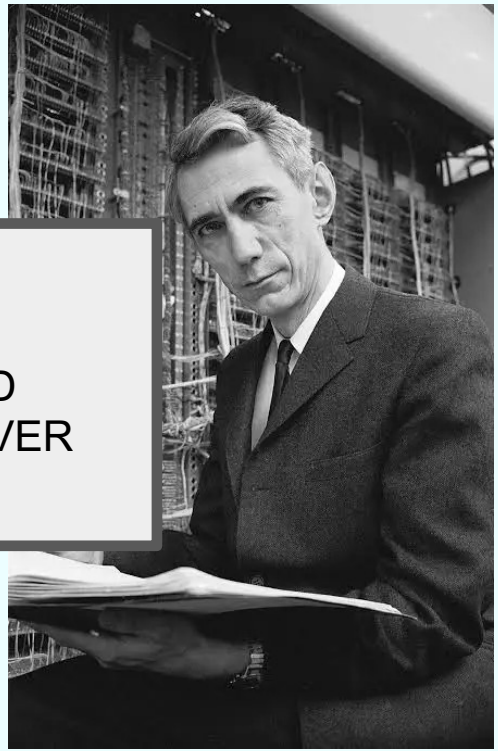
Simplest Language Model: n-gram model

Definition (Informal): A **n-gram model** uses previous (n-1) words to predict the next word.

Unigram:

Bigram:

THE HEAD AND IN FRONTAL ATTACK ON AN ENGLISH WRITER THAT THE CHARACTER OF THIS POINT IS THEREFORE ANOTHER METHOD FOR THE LETTERS THAT THE TIME OF WHO EVER TOLD THE PROBLEM FOR AN UNEXPECTED



We can learn n-gram models for small n,
but can not model long range dependencies.

Hidden Markov Models

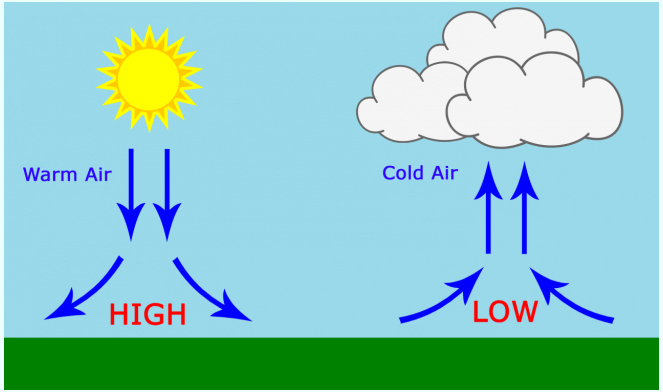
In some sense, the original language model dating back to IBM's Speech Alignment Models of 1970s

Goal: Predict whether it is going to be sunny or cloudy in Toulouse tomorrow?

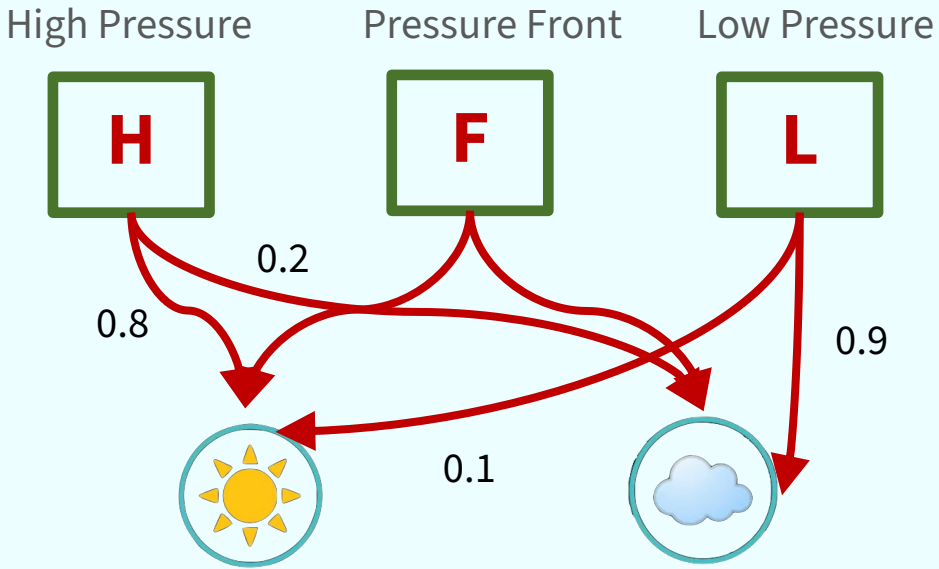
Sunny

Cloudy

Hidden Markov Models

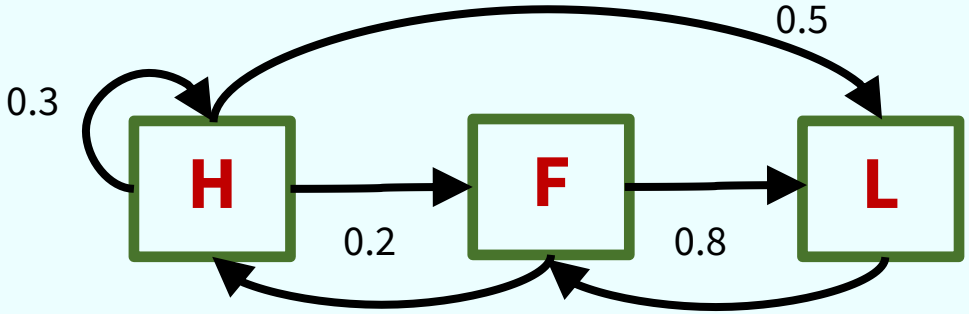


Each day's weather depends only on that day's pressure system.



Hidden Markov Models

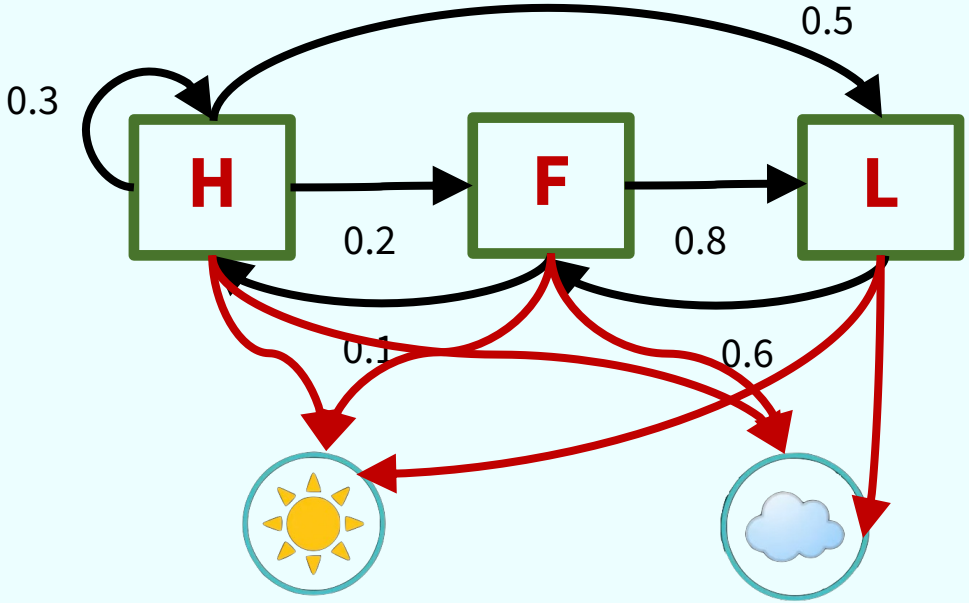
Markov assumption: **Tomorrow's pressure system only depends on today's pressure system.**



Hidden Markov Models

Putting it all together

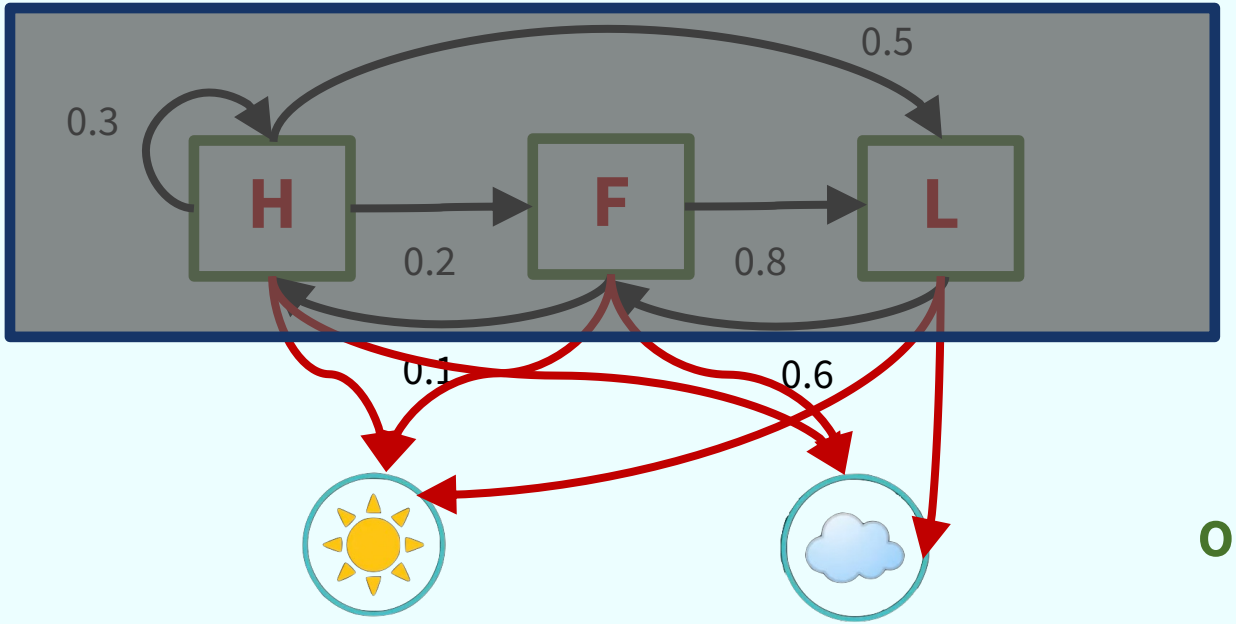
1. Tomorrow's pressure system **given** today's pressure system
2. Tomorrow's weather **given** tomorrow's pressure system



Hidden Markov Models

Issue:

1) We do not know the probabilities
2) We can not measure all the variables!!



Hidden Markov Models

Definition (informal): A **Hidden Markov Model (HMM)** is described by

(1) Hidden states S

(2) Observables W

Sunny Cloudy

(3) Transition Probabilities T Pr[“high pressure tomorrow” | “Low pressure today”]

(4) Observation Probabilities O Pr[“cloudy today” | “Low pressure today”]

Hidden Markov Models

Definition (informal): A **Hidden Markov Model (HMM)** is described by

(1) Hidden states S

(2) Observa

Hidden Markov Model can be used to model other sequential data like DNA sequences, and natural language.

Sunny Cloudy

(3) Transition Probabilities T Pr[“high pressure tomorrow” | “Low pressure today”]

(4) Observation Probabilities O Pr[“cloudy today” | “Low pressure today”]

Hidden Markov Models

What's known about learning HMMs?

Goal: Given random sentences generated by a Hidden Markov Model, can we learn in polynomial time to predict the next word?

Hidden Markov Models

What's known about learning HMMs?

Theorem [Mossel, Roch '05] If the transition and observation matrices have full rank, there is a polynomial time algorithm to learning HMMs from random samples.

Hidden Markov Models

What's known about learning HMMs?

Theorem [Mossel, Roch '05] If the transition and observation matrices have full rank, there is a polynomial time algorithm to learning HMMs from random samples.

Unfortunately, not all HMMs can be learned.

Proposition [Mossel, Roch '05] Learning general HMMs is as hard as solving the noisy parity learning problem.

Learning Parity with Noise

- Adversary chooses a **hidden subset $S \subset [H-1]$**
- The learner get **i.i.d. samples $(w_1, .., w_H) \sim D_S$**
- For $i \in \{1, 2, .., H-1\}$, w_i is uniform over {0,1}

$$w_i \sim Bernoulli(1/2)$$

- The last bit w_H is the parity flipped w.p. ε

$$w_H = \left(\bigoplus_{i \in S} w_i \right) \oplus e \text{ where } e \sim Bernoulli(\varepsilon)$$

Goal: Recover hidden set S or equivalently learn distribution D_S using random samples $(w_1, .., w_H)$?

Hidden Markov Models

What's known about learning HMMs?

Theorem [Mossel, Roch '05] If the transition and observation matrices have full rank, there is a polynomial time algorithm to learning HMMs from random samples.

Unfortunately, not all HMMs can be learned.

Proposition [Mossel, Roch '05] Learning general HMMs is as hard as solving the noisy parity learning problem

Can we learn *all* HMMs from query access?

Conditional Queries

Definition: Given any **prompt**

$$w_1 \to w_2 \to \cdots \to w_t$$

The model replies with a sample from the conditional distribution on **completions**

$$w_{t+1} \sim P[\cdot \,|w_1, \ldots, w_t]$$

Learning Regular Sets from Queries and Counterexamples*

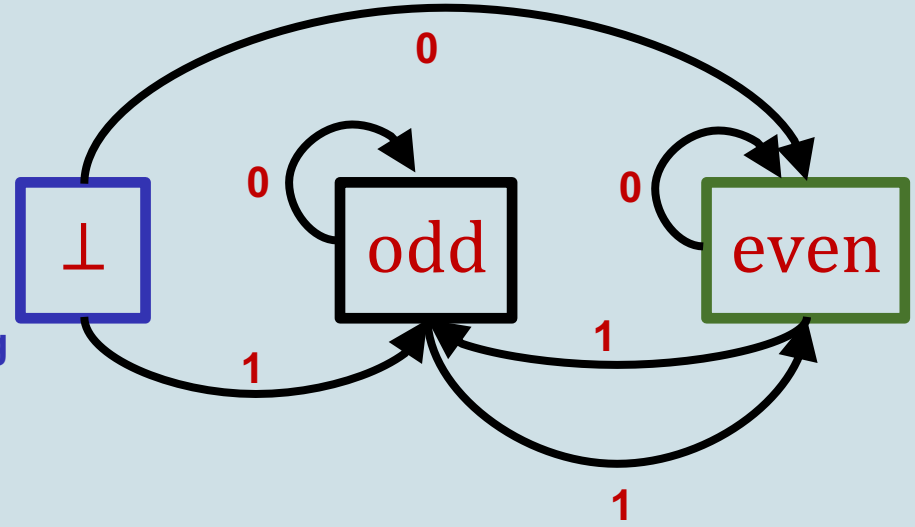
DANA ANGLUIN

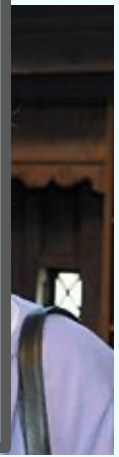
Department of Computer Science, Yale University,
P.O. Box 2158, Yale Station, New Haven, Connecticut 06520

Deterministic Finite Automata (DFA)

DFA is a simple deterministic machine for deciding whether a string belongs to a language $L \subset W^*$

L = { $x \in \{0,1\}^*$: x has even number of ones}





Deterministic Finite Automata (DFA)

DFA defines a language $L \subset W^*$

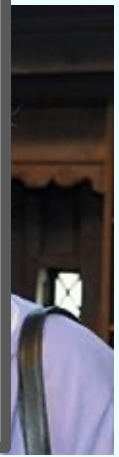
(1) Hidden states Q

(2) Observables W

(3) Transition T: $Q \times W \rightarrow Q$

(4) Starting state q_0 and **Accepting states** F

$x \in L \iff$ starting from q_0 and reading x symbol by symbol, the automaton ends in a state in F



Con

Defin

The m

Deterministic Finite Automata (DFA)

DFA defines a language $L \subset W^*$

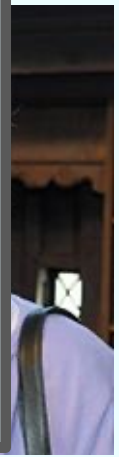
(1) Hidden states Q

(2) Observables W

(3) Transition T: $Q \times W \rightarrow Q$

Theorem [Angluin '87] There is a polynomial time algorithm for learning DFAs using membership queries (Is $x \in L$?)

automaton ends in a state in F



Conditional Queries

Definition: Given any **prompt**

$$w_1 \to w_2 \to \cdots \to w_t$$

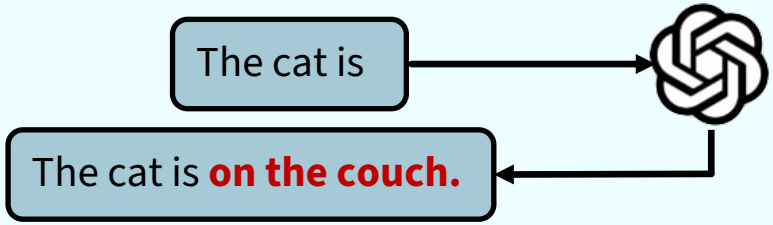
The model replies with a sample from the conditional distribution on **completions**

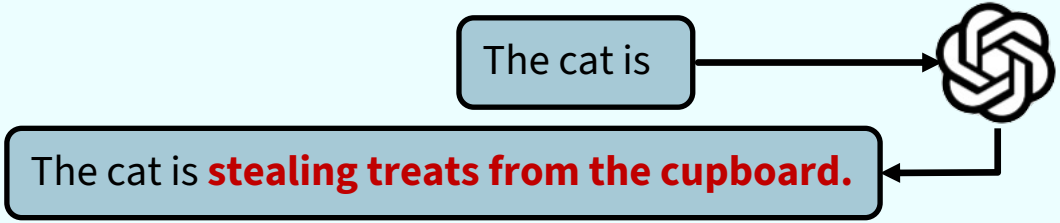
$$w_{t+1} \sim P[\cdot \,|w_1, \ldots, w_t]$$

Note: Learning HMMs from conditional queries would generalize Angluin's classical algorithm for learning DFAs from queries.

Rank of a Language Model

Definition: A **language model** is a **probability distribution P** over sequences of words from W.



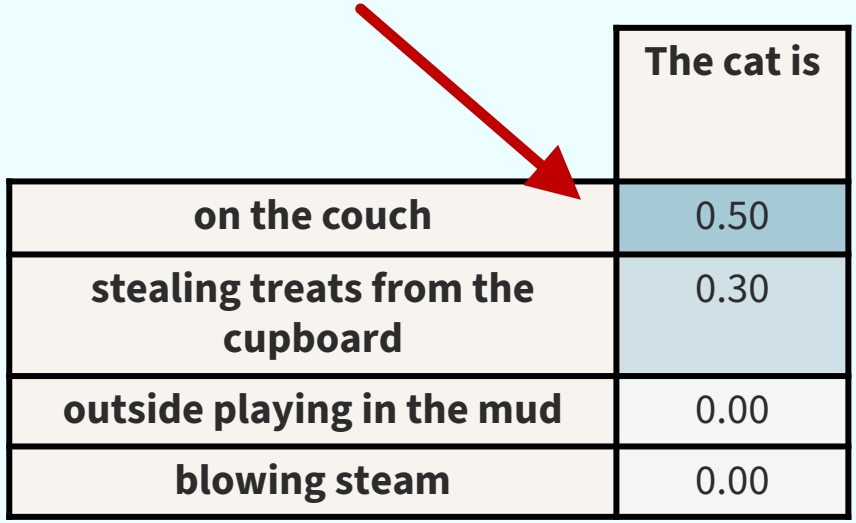


Prompt
The cat is

Completion	Probability
on the couch	0.50
stealing treats from the cupboard	0.30
outside playing in the mud	0.00
blowing steam	0.00

Rank of a Language Model

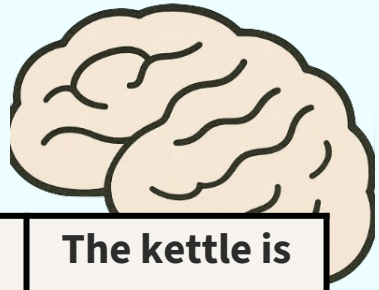
P[on the couch | The cat is]



	The cat is
on the couch	0.50
stealing treats from the cupboard	0.30
outside playing in the mud	0.00
blowing steam	0.00

Rank of a Language Model

P[on the couch | The dog is]



	The cat is	The dog is	The kettle is
on the couch	0.50	0.40	0.10
stealing treats from the cupboard	0.30	0.00	0.00
outside playing in the mud	0.00	0.50	0.00
blowing steam	0.00	0.00	0.70

This matrix characterizes how the language model behaves on prompts of length 3.

Rank of a Language Model

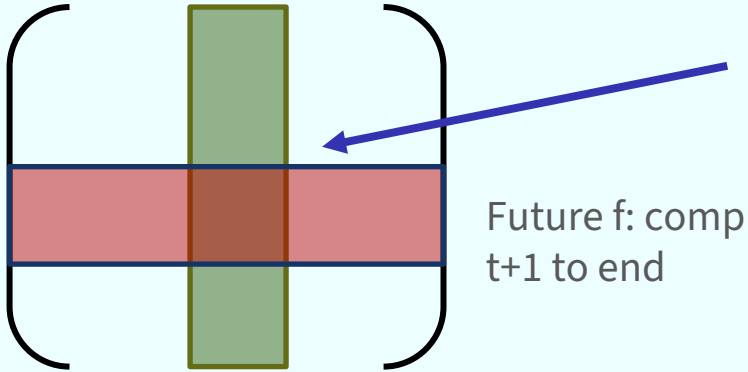
Given any language model P,
For prompt-length t, build a matrix M_t

History h : prompt of length t

Columns: **prompts of length t**:
histories $h \in W^t$

Rows: completions:
futures $f \in W^*$

$M_t =$



Future f: completion from t+1 to end

Rank of a Language Model

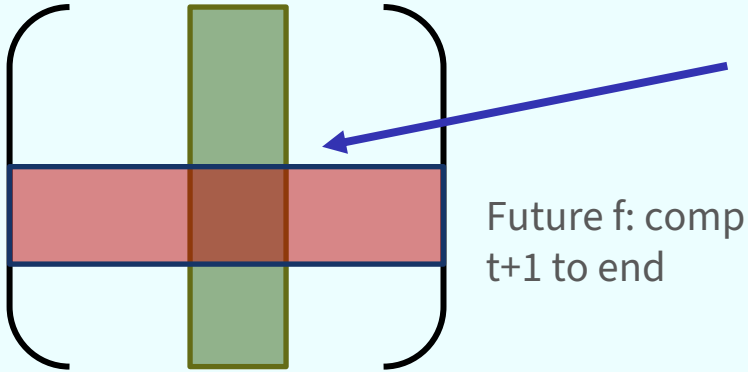
Given any language model P,
For prompt-length t, build a matrix M_t

History h : prompt of length t

Columns: **prompts of length t**:
histories $h \in W^t$

Rows: completions:
futures $f \in W^*$

$M_t =$



Future f: completion from t+1 to end

Matrices M_t characterize the language model P

Rank of a Language Model

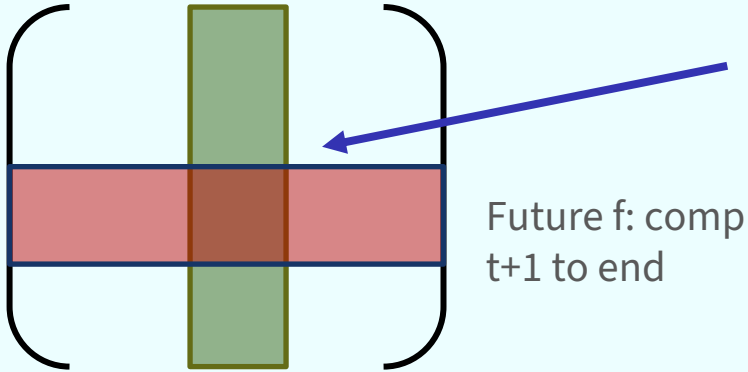
Given any language model P,
For prompt-length t, build a matrix M_t

History h : prompt of length t

Columns: **prompts of length t**:
histories $h \in W^t$

Rows: completions:
futures $f \in W^*$

$M_t =$

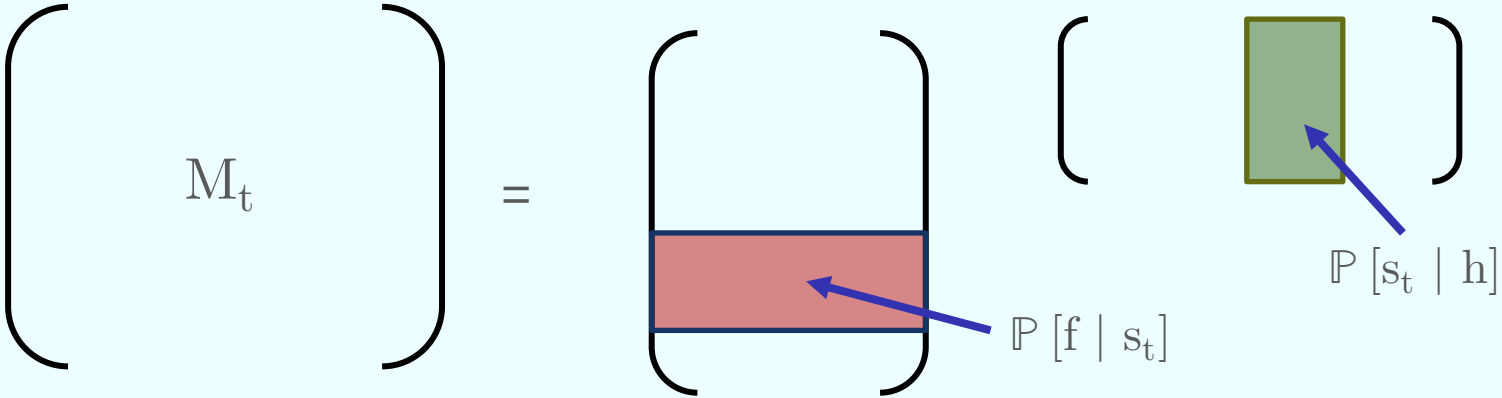


Definition: We say a language model P has rank r, if for every t, the underlying matrix M_t has rank at most r.

Low-rank Language Models

Claim: Any HMM on a state space of size S has rank at most S.

Proof. Each matrix M_t factorizes through the hidden state space.



■

Our Results (Informal)

Theorem [KKMZ ‘23, LM ‘25]: There is a polynomial time algorithm for learning any low rank language model from conditional queries.

Our Results (Formal)

Theorem [KKMZ ‘23, LM ‘25]: For any language model with

(1) An alphabet size A

(2) Horizon at most H

(3) and Rank at most S

There is an algorithm that makes at most

$$\mathrm{poly}(A, H, S, 1/\varepsilon)$$

conditional queries and outputs the description of an efficiently samplable distribution that is ε-close in TV distance to the true language model.

Outline

Part I: Introduction

- HMMs and Low rank language models
- Our results

Part II: A Succinct representation

- Barycentric spanners
- Tracking the evolution of coefficients

Part III: New Techniques

- Representative vectors for Barycentric Spanners
- Taming the error

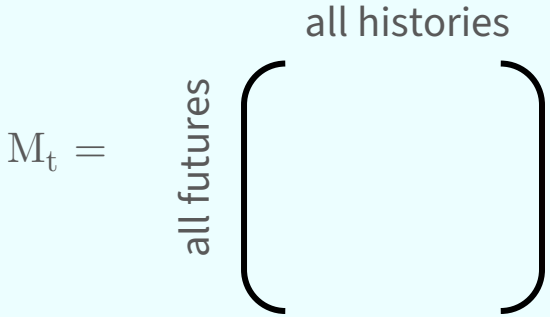
A First Step

Caution: For low rank language models, it's not even clear if learning language model is information theoretically possible.

Can we describe a low rank LM with a polynomial number of parameters?

A First Step

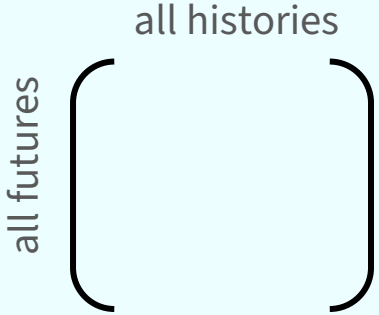
A language model is characterized by matrices M_t
But the matrices M_t have **exponentially many rows and columns.**



Can we describe a low rank LM with a polynomial number of parameters?

A First Step

A language model is characterized by matrices $\mathrm{M_t}$
But the matrices $\mathrm{M_t}$ have **exponentially many rows and columns.**

$\mathrm{M_t} =$ 

Need to exploit relations between $\mathrm{M_t}$ and $\mathrm{M_{t+1}}$ etc.

Can we describe a low rank LM with a polynomial number of parameters?

Outline

Part I: Introduction

- HMMs and Low rank language models
- Our results

Part II: A Succinct representation

- **Barycentric spanners**
- Tracking the evolution of coefficients

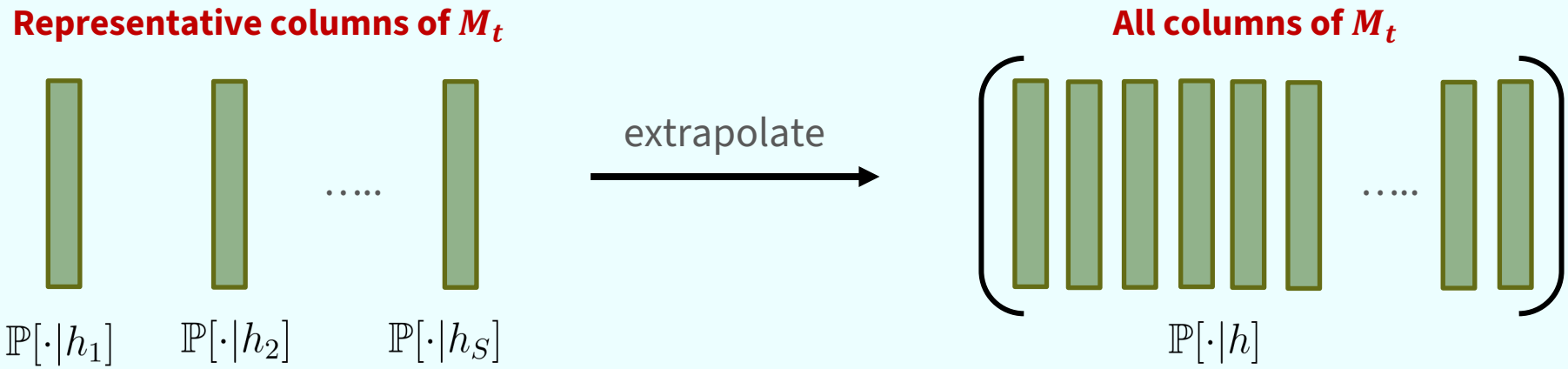
Part III: New Techniques

- Representative vectors for Barycentric Spanners
- Taming the error

Barycentric spanners

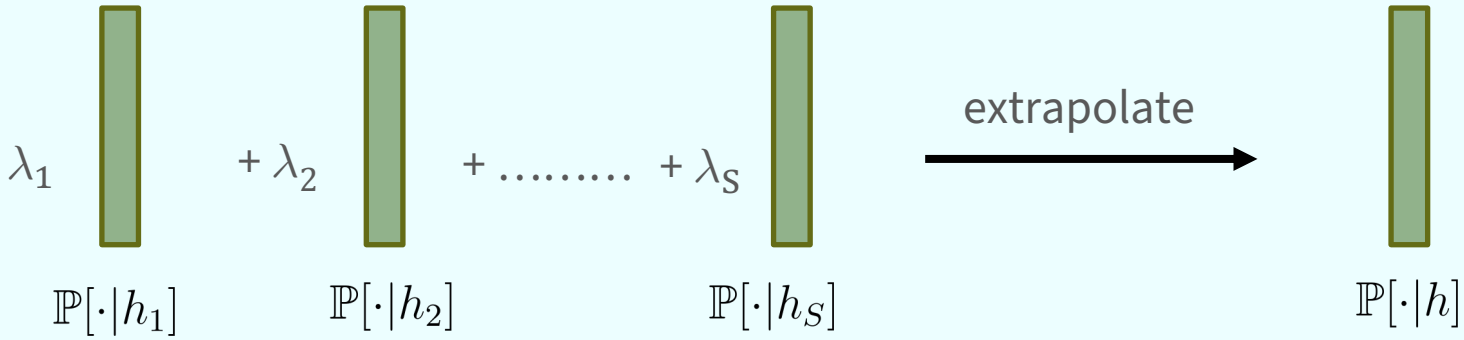
Given a set Ω of vectors in an S-dimensional space, how can we find a representative set?

Think of these vectors as columns of M_t - i.e. encoding the distribution on possible futures, given the history.



Barycentric spanners

Suppose $P[\cdot\,|h_1], P[\cdot\,|h_2], \ldots, P[\cdot\,|h_S]$ is an arbitrary basis for this S-dimensional space.

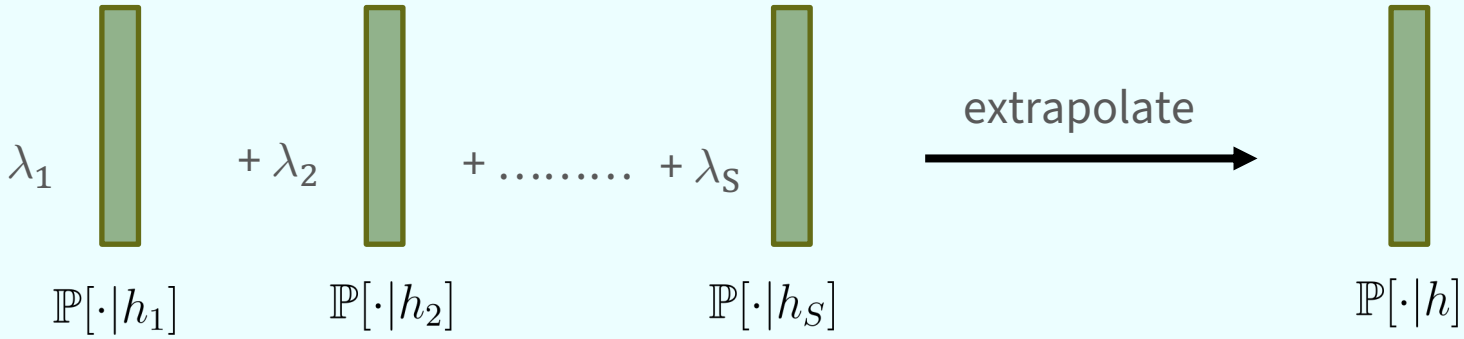


Barycentric spanners

Definition: Given a set Ω of vectors, we say that $x_1, x_2, \ldots, x_s$ is a C- approximate barycentric spanner, if for any x in Ω we can write

$$x = \lambda_1 x_1 + \lambda_2 x_2 + \ldots + \lambda_S x_S$$

with each $|\lambda_i| \leq C$



Barycentric spanners

Definition: Given a set Ω of vectors, we say that $x_1, x_2, \ldots, x_s$ is a C- approximate barycentric spanner, if for any x in Ω we can write

$$x = \lambda_1 x_1 + \lambda_2 x_2 + \ldots + \lambda_S x_S$$

with each $|\lambda_i| \leq C$

Do C-approximate barycentric spanners even exist?

Barycentric spanners

Definition: Given a set Ω of vectors, we say that $\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_s$ is a C- approximate barycentric spanner, if for any x in Ω we can write

$$x = \lambda_1 x_1 + \lambda_2 x_2 + \ldots + \lambda_S x_S$$

with each $|\lambda_i| \leq \mathrm{C}$

Proposition [Awerbuch, Kleinberg]: For any $\mathrm{C} \geq 1$, C −approximate barycentric spanner exist.

Barycentric spanners

Definition: Given a set Ω of vectors, we say that $\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_s$ is a C- approximate barycentric spanner, if for any x in Ω we can write

$$x = \lambda_1 x_1 + \lambda_2 x_2 + \ldots + \lambda_S x_S$$

with each $|\lambda_i| \leq \mathrm{C}$

Proposition [Awerbuch, Kleinberg]: For any $\mathrm{C} \geq 1$, C −approximate barycentric spanner exist.

Many applications in online learning and RL - can we use them to parameterize low rank LMs?

Outline

Part I: Introduction

- HMMs and Low rank language models
- Our results

Part II: A Succinct representation

- Barycentric spanners
- **Tracking the evolution of coefficients**

Part III: New Techniques

- Representative vectors for Barycentric Spanners
- Taming the error

Idealized Blueprint

Ignoring for now major statistical and algorithmic complications:

For each timestep t, we **compute a barycentric spanner** of the columns of M_{t}

While sampling a trajectory, **track how the coefficients evolve**.

Using barycentric spanners

Suppose we have computed a barycentric spanner for each timestep t
i.e. a representative set of histories

$$h_1^{(t)}, h_2^{(t)}, \dots, h_S^{(t)}$$

How do we use these barycentric spanners to sample from P[· | x]?

Using barycentric spanners

Suppose we have computed a barycentric spanner for each timestep t
i.e. a representative set of histories

$$\mathrm{h}_1^{(\mathrm{t})}, \mathrm{h}_2^{(\mathrm{t})}, \ldots, \mathrm{h}_{\mathrm{S}}^{(\mathrm{t})}$$

How do we use these barycentric spanners to sample from P[· | x]?

In principle for any history x, we can use the expression

$$\mathbb{P}[f \mid x] = \sum_i \lambda_i^{(t)}(x) \cdot \mathbb{P}[f \mid h_i^{(t)}]$$

to compute x's distribution on the futures.

Using barycentric spanners

Suppose we have computed a barycentric spanner for each timestep t
i.e. a representative set of histories

$$\mathrm{h}_1^{(\mathrm{t})}, \mathrm{h}_2^{(\mathrm{t})}, \ldots, \mathrm{h}_{\mathrm{S}}^{(\mathrm{t})}$$

How do we use these barycentric spanners to sample from P[· | x]?

In principle for any history x, we can use the expression

$$\mathbb{P}[f \mid x] = \sum_i \lambda_i^{(t)}(x) \cdot \mathbb{P}[f \mid h_i^{(t)}]$$

to compute x's distribution on the futures.

But how do we get these coefficients??

Tracking the coefficients

Main problem: Even if we know the coefficients $\lambda_{\mathrm{i}}^{(\mathrm{t})}(\mathrm{x})$ and we can sample the next token probabilities from the correct distribution $\mathbb{P}\left[\mathrm{o} \mid \mathrm{x}\right]$, **how do we get the new coefficients?**

$$\mathbb{P}[f \mid x \vee o] = \sum_i \lambda_i^{(t+1)}(x \vee o) \cdot \mathbb{P}[f \mid h_i^{(t+1)}]$$

Tracking the coefficients

Claim (informal): Can use Bayes rule to compute new coefficients

Tracking the coefficients

Claim (informal): Can use Bayes rule to compute new coefficients

First for any future f whose t+1st token is o we have

$$\mathbb{P}[f \mid x] = \mathbb{P}[f \mid x \vee o] \cdot \mathbb{P}[o \mid x]$$

Tracking the coefficients

Claim (informal): Can use Bayes rule to compute new coefficients

First for any future f whose t+1st token is o we have

$$\mathbb{P}[f \mid x] = \mathbb{P}[f \mid x \vee o] \cdot \mathbb{P}[o \mid x]$$

Returning to our earlier expression

$$\mathbb{P}[f \mid x] = \sum_i \lambda_i^{(t)}(x) \cdot \mathbb{P}[f \mid h_i^{(t)}]$$

Tracking the coefficients

Claim (informal): Can use Bayes rule to compute new coefficients

First for any future f whose t+1st token is o we have

$$\mathbb{P}[f \mid x] = \mathbb{P}[f \mid x \vee o] \cdot \mathbb{P}[o \mid x]$$

Returning to our earlier expression

$$\mathbb{P}[f \mid x \vee o] \cdot \mathbb{P}[o \mid x] = \sum_i \lambda_i^{(t)}(x) \cdot \mathbb{P}[f \mid h_i^{(t)} \vee o] \cdot \mathbb{P}[o \mid h_i^{(t)}]$$

Tracking the coefficients

Claim (informal): Can use Bayes rule to compute new coefficients

First for any future f whose t+1st token is o we have

$$\mathbb{P}[f \mid x] = \mathbb{P}[f \mid x \vee o] \cdot \mathbb{P}[o \mid x]$$

Returning to our earlier expression

$$\mathbb{P}[f \mid x \vee o] = \sum_i \lambda_i^{(t)}(x) \cdot \frac{\mathbb{P}[o \mid h_i^{(t)}]}{\mathbb{P}[o \mid x]} \cdot \mathbb{P}[f \mid h_i^{(t)} \vee o]$$

Tracking the coefficients

Claim (informal): Can use Bayes rule to compute new coefficients

First for any future f whose t+1st token is o we have

$$\mathbb{P}[f \mid x] = \mathbb{P}[f \mid x \vee o] \cdot \mathbb{P}[o \mid x]$$

Returning to our earlier expression

$$\mathbb{P}[f \mid x \vee o] = \sum_i \lambda_i^{(t)}(x) \cdot \frac{\mathbb{P}[o \mid h_i^{(t)}]}{\mathbb{P}[o \mid x]} \cdot \mathbb{P}[f \mid h_i^{(t)} \vee o]$$

Can do a change of basis to express these in terms of t+1st barycentric spanner

■

Idealized Blueprint

Ignoring for now major statistical and algorithmic complications:

> For each timestep t, we **compute a barycentric spanner** of the columns of M_{t}
>
> While sampling a trajectory, **track how the coefficients evolve**.

Hence, we can describe a low rank language model exactly with a **polynomial number of parameters** (**barycentric spanners, their next token probabilities, change of basis**)

Challenges

How can we compute the barycentric spanners with only sampling access to the vectors?

When there are errors in the coefficients, how can we prevent the error from blowing up with the length of the sequence?

Outline

Part I: Introduction

- HMMs and Low rank language models
- Our results

Part II: A Succinct representation

- Barycentric Spanners
- Tracking the evolution of coefficients

Part III: New Techniques

- **Representative vectors for Barycentric Spanners**
- Taming the error

Sketching Norms

Can we construct vectors of polynomial dimension that can act as surrogates for columns of M_t?

Sketching Norms

Definition: Given a collection of histories $\mathcal{A}$ of length t, we say that a set of vectors

$$\{v_h\}_{h \in \mathcal{A}}$$

is γ**-representative** if for all coefficients $|c_h| \leq 1$, we have

$$\left| \left\| \sum_{h \in \mathcal{A}} c_h v_h \right\|_1 - \left\| \sum_{h \in \mathcal{A}} c_h \, \mathbb{P}[\cdot \mid h] \right\|_1 \right| \leq \gamma$$

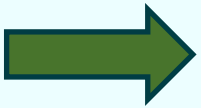
Sketching Norms

Definition: Given a collection of histories $\mathcal{A}$ of length t, we say that a set of vectors

$$\{v_h\}_{h \in \mathcal{A}}$$

is γ**-representative** if for all coefficients $|c_h| \leq 1$, we have

$$\left| \left\| \sum_{h \in \mathcal{A}} c_h v_h \right\|_1 - \left\| \sum_{h \in \mathcal{A}} c_h \, \mathbb{P}[\cdot \mid h] \right\|_1 \right| \leq \gamma$$



A barycentric spanner for one is automatically an approximate barycentric spanner for another.

Sketching Norms

But how do we construct representative vectors?

Sketching Norms

But how do we construct representative vectors?

Claim: For any distribution $\mathcal{D}$ on futures, consider

$$v_h = \left(\frac{\mathbb{P}[f_1 \mid h]}{m\,\mathcal{D}[f_1]}, \ldots, \frac{\mathbb{P}[f_m \mid h]}{m\,\mathcal{D}[f_m]} \right)$$

where each f_i is drawn iid from $\mathcal{D}$. Then, in expectation ℓ_1 norms will be correct

Sketching Norms

But how do we construct representative vectors?

Claim: For any distribution $\mathcal{D}$ on futures, consider

$$v_h = \left(\frac{\mathbb{P}[f_1 \mid h]}{m\,\mathcal{D}[f_1]}, \ldots, \frac{\mathbb{P}[f_m \mid h]}{m\,\mathcal{D}[f_m]} \right)$$

where each f_i is drawn iid from $\mathcal{D}$. Then, in expectation ℓ_1 norms will be correct

And with careful choice of $\mathcal{D}$ can get concentration bounds too.

Sketching Norms

Still need to deal with the fact that there are exponentially many histories we care about.

Sketching Norms

Still need to deal with the fact that there are exponentially many histories we care about.

Claim (informal): With high probability a random collection of a polynomial number of histories contains a barycentric spanner that covers most histories.

Outline

Part I: Introduction

- HMMs and Low rank language models
- Our results

Part II: A Succinct representation

- Barycentric Spanners
- Tracking the evolution of coefficients

Part III: New Techniques

- Representative vectors for Barycentric Spanners
- **Taming the error**

Compounding Errors

When there is sampling error we can only **approximate** the coefficients

$$\lambda_i^{(t)}(x) \xrightarrow{\text{sampling noise}} \widetilde{\lambda_i^{(t)}}(x)$$

Compounding Errors

When there is sampling error we can only **approximate** the coefficients

$$\lambda_i^{(t)}(x) \xrightarrow{\text{sampling noise}} \widetilde{\lambda_i^{(t)}}(x)$$

Main Problem: Estimation error can compound multiplicatively with each step.

Compounding Errors

When there is sampling error we can only **approximate** the coefficients

$$\lambda_i^{(t)}(x) \xrightarrow{\text{sampling noise}} \widetilde{\lambda_i^{(t)}}(x)$$

Main Problem: Estimation error can compound multiplicatively with each step.

Even though the true coefficients should be bounded (by the barycentric spanner property) the estimates might not be

An Abstraction

We know that the true vector $z = P[\cdot\,|x]$ is in the set

$$\mathcal{K} = \left\{ \sum_i \lambda_i^{(t)}\, \mathbb{P}[\cdot \mid h_i^{(t)}] \;\; \text{s.t.} \;\; \forall i \quad |\lambda_i^{(t)}| \leq 1 \right\}.$$

An Abstraction

We know that the true vector $z = P[\cdot\,|x]$ is in the set

$$\mathcal{K} = \left\{ \sum_i \lambda_i^{(t)}\, \mathbb{P}[\cdot \mid h_i^{(t)}] \;\; \text{s.t.} \;\; \forall i \quad |\lambda_i^{(t)}| \leq 1 \right\}.$$

And our estimate is $w = \sum_i \widetilde{\lambda}_i^{(t)}\, \mathbb{P}[\cdot \mid h_i^{(t)}]$.

An Abstraction

We know that the true vector $z = P[\cdot\,|x]$ is in the set

$$\mathcal{K} = \left\{ \sum_i \lambda_i^{(t)}\, \mathbb{P}[\cdot \mid h_i^{(t)}] \;\; \text{s.t.} \;\; \forall i \quad |\lambda_i^{(t)}| \le 1 \right\}.$$

And our estimate is $w = \sum_i \widetilde{\lambda}_i^{(t)}\, \mathbb{P}[\cdot \mid h_i^{(t)}]$.

Goal: Map w to a point $z' \in \mathcal{K}$ and guarantee

$$\|z' - z\|_1 \le \|w - z\|_1$$

i.e. our statistical error has not increased, **even though we don't know what z is.**

An Abstraction

But this is **impossible**, can only guarantee

$$\|z' - z\|_1 \leq 2\|w - z\|_1$$

By the triangle inequality and this is tight for ℓ_1 projection.

Taming the Blowup

Solution: Project according to KL divergence instead

Taming the Blowup

Solution: Project according to KL divergence instead

Fact: If we let $z^* = \arg\min_{z' \in \mathcal{K}} d_{KL}(z' \,\|\, w)$ then

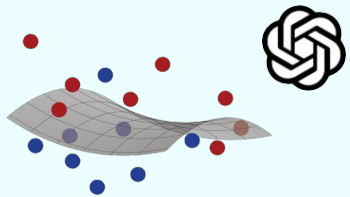
$$d_{KL}(z \,\|\, z^*) \leq d_{KL}(z \,\|\, w)$$

i.e. projecting in the KL divergence decreases the distance from all other points in the set.

Research Overview

Computational-Statistical Gap in Reinforcement Learning
KLL**M**23, KLL**M**22, DKKL**M**S21, DLW**M**20

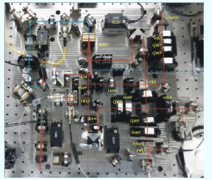
Learning Theory



Learning quantum state
GL**M**24, BBCGL**M**25, BC**M**25

Reinforcement Learning

Quantum Learning Theory



Learning beyond PAC model
HKL**M**25, KK**M**Z23, HKL**M**22, DS**M**22

Active Learning with Comparison Queries
HKL**M**22, HKL**M**20, HKL**M**20

Convergence of Policy Gradient Methods
AKL**M**21, AKL**M**22

Thank you!