Naïve Bayes and Hidden Markov Models

CS 759/859 Natural Language Processing Lecture 12

Samuel Carton, University of New Hampshire

Last lecture

Key idea: Probabilistic language modeling

Concepts

- Conditional probability
- Chain rule
- N-gram models
- Uses of language models
    - Generation
    - Evaluation
- Perplexity

Unigram model

Basic idea: model the text as the individual words occurring independently

- Parametrized by corpus token frequencies

$$P(w^{(1)} \dots w^{(N)}) = \prod_{i=1}^{N} P(w^{(i)})$$

What's the problem with this?

Bigram model

Basic idea: model text as words being dependent on **only** the prior word

- Parameterized by token co-occurrence frequencies

$$P(w^{(1)} \dots w^{(N)}) = \prod_{i=1}^{N} P(w^{(i)} \mid w^{(i-1)})$$

A bigram model is a type of **Markov Chain**

Bayes Rule

Conditional probability

When two variables may be dependent, then their joint probability is expressed as follows:

$$P(X,Y) \; = \; P(Y)P(X|Y) \; = \; P(X)P(Y|X)$$

If they happen to be independent, then $P(X|Y) \; = \; P(X)$ and $P(Y|X) \; = \; P(Y)$, so

$$P(X,Y) \; = \; P(Y)P(X) \; = \; P(X)P(Y)$$

Bayes Rule

It follows from

$$P(X,Y) = P(Y)P(X|Y) = P(X)P(Y|X)$$

that

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)}$$

Examples

$$P(Hypothesis\ |Observed\ event) = \frac{P(Observed\ event\ |\ Hypothesis)P(Hypothesis)}{P(Observed\ event)}$$

$$P(Lung\ cancer|Cough) = \frac{P(Cough|Lung\ cancer)P(Lung\ cancer)}{P(Cough)}$$

$$P(Aliens|Lights\ in\ the\ sky) = \frac{P(Lights\ in\ the\ sky|Aliens)P(Aliens)}{P(Lights\ in\ the\ sky)}$$

Relative probabilities

Often we only care about the relative probability of two possible outcomes, rather than their true probability:

$$P(Lung\ cancer|Cough) \text{ vs. } P(COVID|Cough)$$

$$\frac{P(Cough|Lung\ cancer)P(Lung\ cancer)}{\cancel{P(Cough)}} \text{ vs. } \frac{P(Cough|COVID)P(COVID)}{\cancel{P(Cough)}}$$

Because we only care about the relative value, we can ignore the denominator

$P(Cough|Lung\ cancer) \approx P(Cough|COVID) \approx 1.0$

~50 million COVID cases in 2022, ~300k new lung cancer cases in 2023

So P(COVID) = .15, and P(Lung cancer) = 0.001

So $P(COVID|Cough)$ is **150 times** higher than $P(Lung\ cancer|Cough)$

https://www.cancer.org/cancer/lung-cancer/about/key-statistics.html

https://covid.cdc.gov/covid-data-tracker/#trends_totalcases_select_00

Base rate fallacy

A lot of fallacious thinking comes from ignoring the **base rates** P(X) and P(Y) in $\frac{P(Y|X)P(X)}{P(Y)}$

$$P(Hypothesis \mid Rare\ event) = \frac{P(Rare\ event \mid Hypothesis)P(Hypothesis)}{P(Rare\ event)}$$

Example: aliens

$$P(Aliens|Lights\ in\ the\ sky) = \frac{P(Lights\ in\ the\ sky|Aliens){\color{red}P(Aliens)}}{\color{red}P(Lights\ in\ the\ sky)}$$

Relative probabilities:

$$\frac{P(Lights\ in\ the\ sky|Aliens){\color{red}P(Aliens)}}{\cancel{P(Lights\ in\ the\ sky)}} \text{ vs. } \frac{P(Lights\ in\ the\ sky|Military\ test){\color{red}P(Military\ test)}}{\cancel{P(Lights\ in\ the\ sky)}}$$

Problem:

- P(Hypothesis) is often lower than you think
- P(Rare event) is often higher than you think

https://en.wikipedia.org/wiki/Base_rate_fallacy
https://en.wikipedia.org/wiki/List_of_cognitive_biases

Naïve Bayes

Application to text

Classification:

$$P(Class \mid Words)$$

$$P(Class\ 0 \mid Words) \text{ vs. } P(Class\ 1 \mid Words)$$

$$\frac{P(Words \mid Class\ 0)P(Class\ 0)}{P(Words)} \text{ vs. } \frac{P(Words \mid Class\ 1)P(Class\ 1)}{P(Words)}$$

We can ignore P(Words), but how do we calculate:

- $P(Words \mid Class\ 0)$
- $P(Class\ 0)$
- $P(Words \mid Class\ 1)$
- $P(Class\ 1)$

Application to text

$$P(Class\ 0) = \frac{\#\ Class\ 0}{\#\ Class\ 0 + \#\ Class\ 1}$$

- And likewise for class 1

$P(Words \mid Class\ 0)$

- Build an n-gram model of all texts for which class is Class 0
- Use this model to estimate $P(Words \mid Class\ 0)$
- And likewise for Class 1

Naïve Bayes

Basic idea: apply Bayes rule to find relative likelihoods of $P(Class\ 0 \mid Words)$ vs. $P(Class\ 1 \mid Words)$, using **unigram model** for $P(Words \mid Class\ C)$

So if we consider words = $\{w_0, w_1, \ldots, wN\}$:

$$P(Class\ 0 \mid Words) \propto P(Class\ 0) \prod_{i=1}^{N} P(wi \mid Class\ 0)$$

$$P(Class\ 1 \mid Words) \propto P(Class\ 1) \prod_{i=1}^{N} P(wi \mid Class\ 1)$$

Naïve Bayes with Scikit-Learn

Code description

- Reading and preprocessing SST-2 dataset
- Training and evaluating a Scikit-Learn Naïve Bayes model on SST-2 data
- Inspecting model parameters for whole model & individual instances

Notebook headings

Read/preprocess SST-2 dataset

Read the SST-2 dataset

Preprocess and vectorize the data

Naive Bayes for classifying SST-2 data

Build and evaluate the model

Explaining the model

Explaining individual predictions

Interpreting log-probability differences

If :

$$\log(P(w_i|\ class\ 0))\ -\ \log(P(w_i|\ class\ 1))\ =\ 4.8$$

Then:

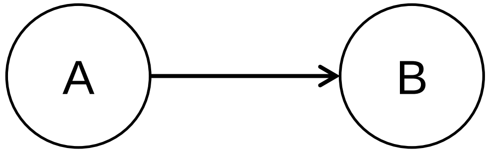
$$\frac{P(wi|class\ 0)}{P(wi|class\ 1)} = e^{4.8} = 2.718^{4.8} = 121.51$$

Meaning that w_i ("unfunny" in this case) is **121.51** times more likely to occur in class 0 than in class 1

Probabilistic graphical models

Bayesian networks

A collection of random variables linked together by conditional probability relationships



Implies that is dependent on/caused by A.

A Bayesian Network is a **directed acyclic graph** (DAG)

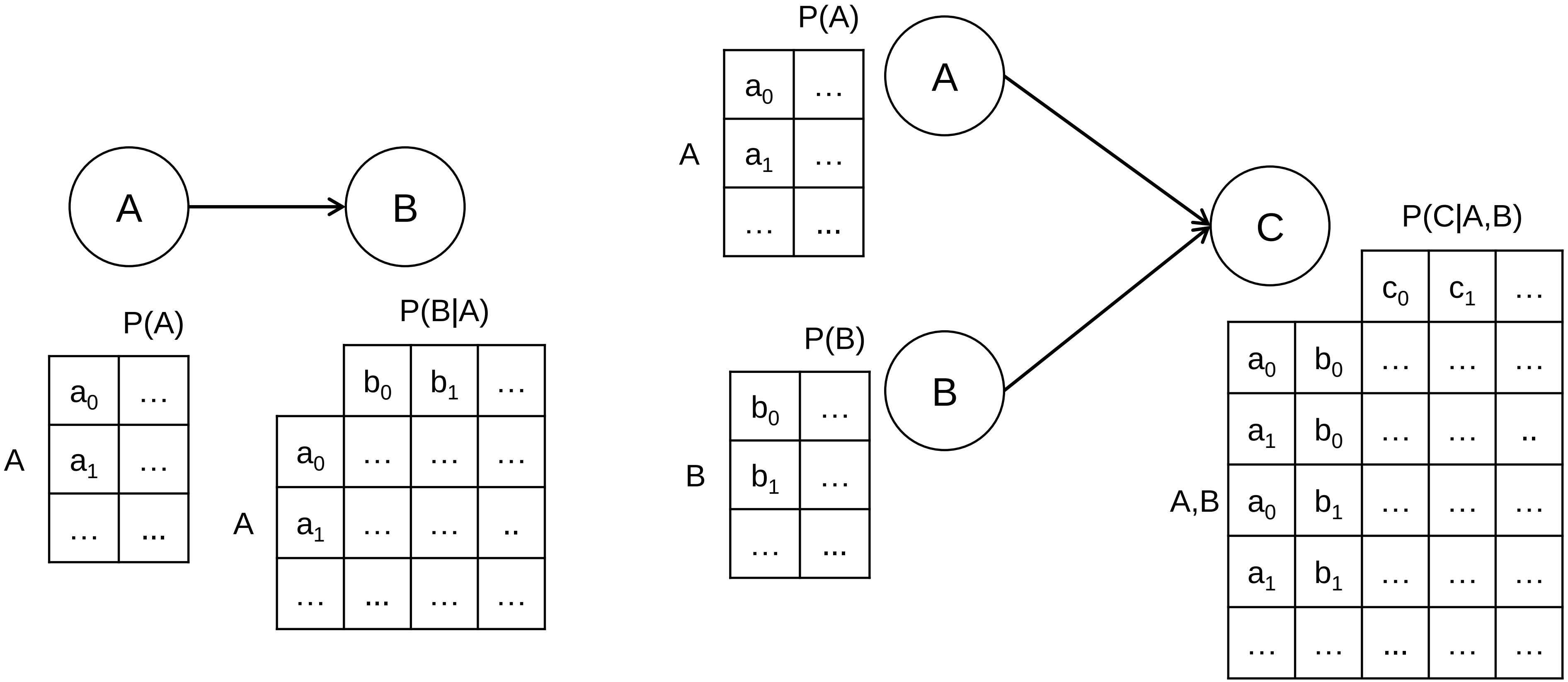
- Edges have a direction and imply causality
- No loops in the graph (No “A causes B causes C causes A” situations)

NLP examples:

- N-gram models
- Naïve Bayes model

Parameters of a Bayes Net

Each node in a bayes net is parameterized by its conditional probabilities relative to the nodes that connect to it



Three things we want to do

Learning: Given a dataset, what are the most likely parameters for our model?

- I.e. which parameters would make the data most likely under this model
- But also smoothing, etc.

Inference: Given our (fully-parameterized) model and (optionally) the value of one or more nodes, what is the likelihood of a given set of values for the other nodes?

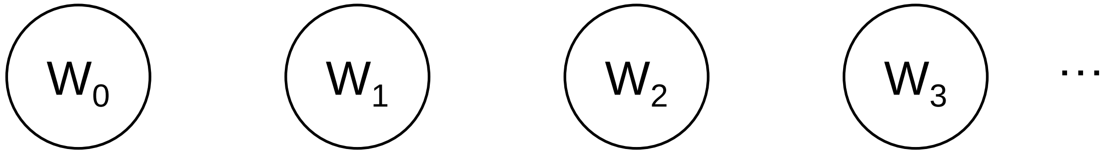
- E.g., what is the likelihood of a given sequence in a bigram model?
- E.g., what is the probability of class c_0 versus c_1 in a Naïve Bayes model, given the words in the text?

Generation: Given our model and (optionally) the values of some nodes, can we generate new values for the other nodes?

- Really just a special case of inference

Unigram model

A collection of freestanding nodes with the same CPT and no dependencies



W	P(W)
w_0	…
w_1	…
…	...

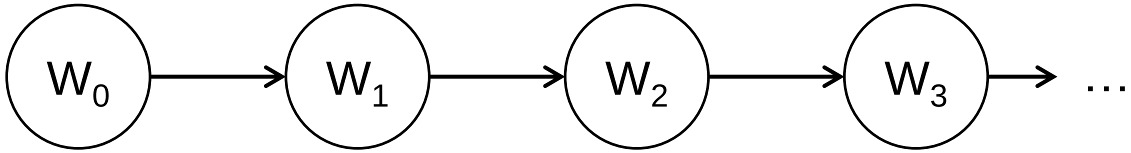
Learning: Count word frequencies within the corpus

Inference: Use product rule of independent probabilities

Generation: Generate each word separately according to CPT

Bigram model

A chain of dependent nodes, all using the same conditional probability table



$P(W_t|W_{t-1})$

		w_0	w_1	…
	w_0	…	…	…
W_{t-1}	w_1	…	…	..
	…	...	…	…

Learning: Count bigram frequencies within the corpus

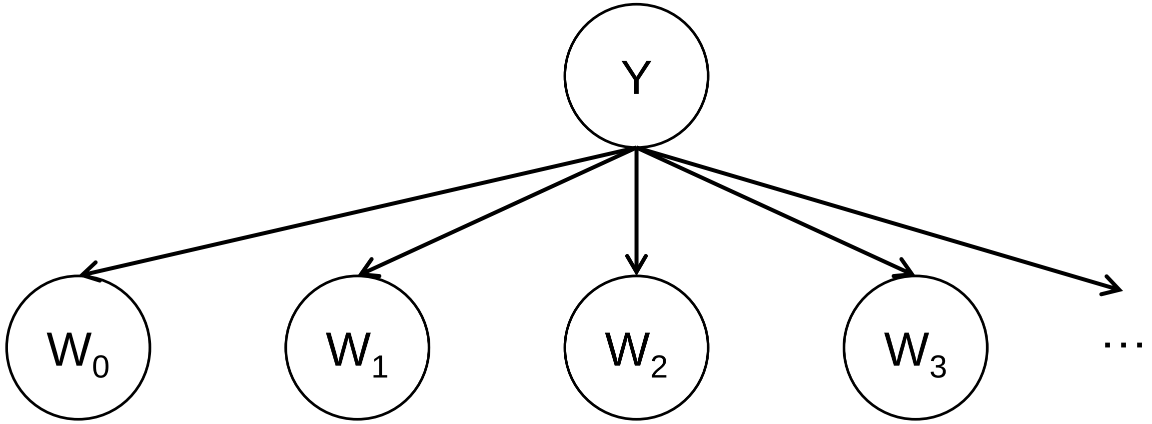
Inference: Use chain rule on consecutive bigrams

Generation: Generate words one at a time depending on preceding word

Can be interpreted as a **Markov Chain:** “a stochastic process describing a sequence of possible events in which the probability of each event depends only on the state attained in the previous event” https://en.wikipedia.org/wiki/Markov_chain

Naïve bayes model

A unigram model where the word probabilities depend (only) on the class



P(W|Y)

		w_0	w_1	…
	y_0	…	…	…
Y	y_1	…	…	..
	…	...	…	…

Learning: Count unigram frequencies for each class

Inference: Apply Bayes Rule to convert P(W|Y) into P(Y|W)P(Y)/P(PW)

Generation: Generate class, then generate words independently

- Not something you'd really do

Generative story

In NLP, the **generative story** of a corpus of texts is the probabilistic model we suppose was used to produce it.

Then we **learn** the most likely parameters for that model based on the corpus

Then we can perform **inference**, as well as **generate** new text.

New task: sequence tagging

Sequence tagging seeks to assign a label to every word in the text, not just for the whole text.

Example: Part-of-speech tagging

i	sentence	you	to	read	this	sentence	.
?	?	?	?	?	?	?	?

How could we approach this using what we've already learned?

New task: sequence tagging

Sequence tagging seeks to assign a label to every word in the text, not just for the whole text.

Example: Part-of-speech tagging

i	sentence	you	to	read	this	sentence	.
PP	V	PP	PREP	V	DET	NN	PUNCT

How could we approach this using what we've already learned?

New task: sequence tagging

Sequence tagging seeks to assign a label to every word in the text, not just for the whole text.

Example: Part-of-speech tagging

i	sentence	you	to	read	this	sentence	.
PP	V	PP	PREP	V	DET	NN	PUNCT

How could we approach this using what we've already learned?

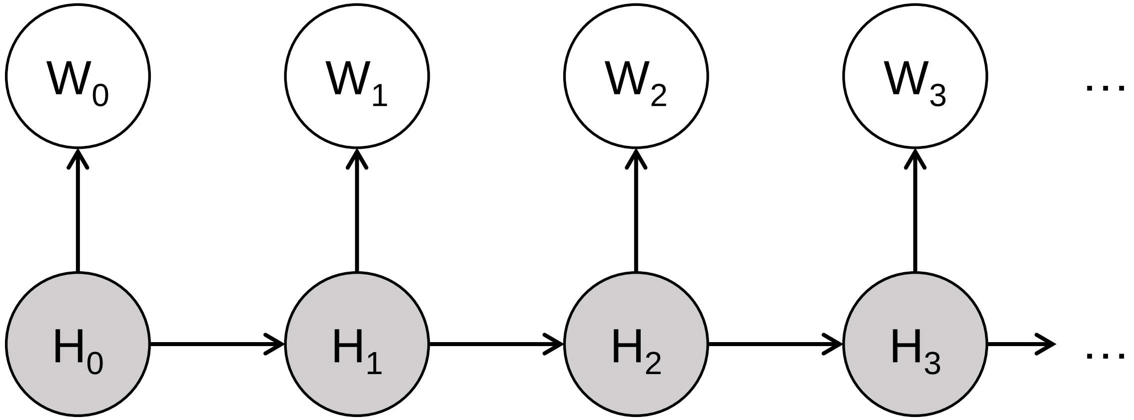
Answer: build a vector representation of each word, and classify each one separately.

- But, how would this account for what had been predicted for the previous word????

Hidden Markov Models

Hidden Markov Model

Generative story: there's a sequence of **latent, hidden states** that are connected together in a Markov Chain, and then words are generated dependent **only** on the state at time *t*.



$P(W_t|H_t)$

		w_0	w_1	…
	h_0	…	…	…
H_t	h_1	…	…	..
	…	...	…	…

Emission matrix

$P(H_t|H_{t-1})$

		h_0	h_1	…
	h_0	…	…	…
H_{t-1}	h_1	…	…	..
	…	...	…	…

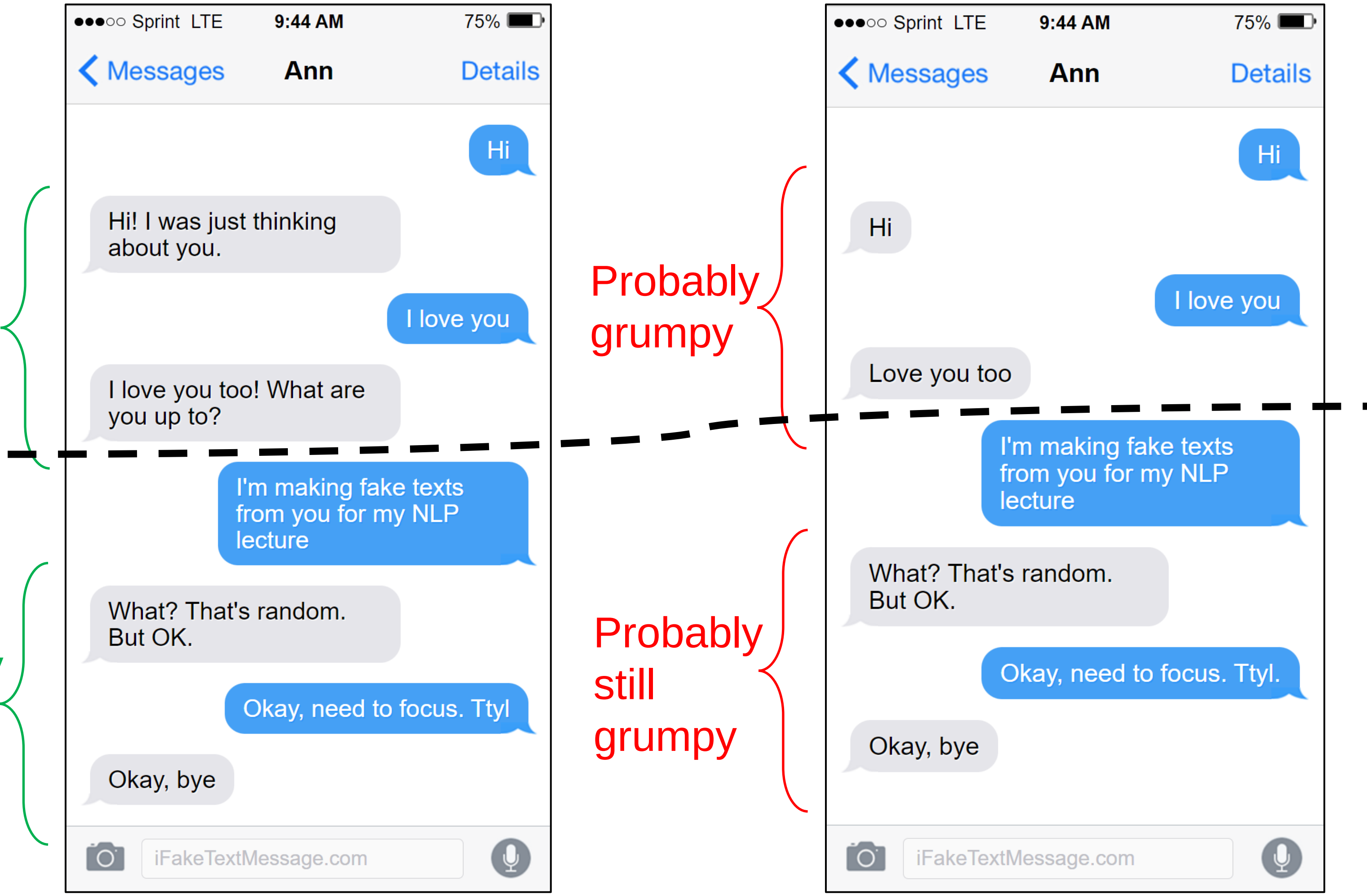
Transition matrix

Case study: Interpreting my wife's mood

A good example of a latent sequential process is texting. You have:

- **Observed output** in the form of the actual texts
- **Latent states** in the form of the emotions of your texting partner

…and your interpretation of the **same observed output** can depend on your **inference** of previous latent states.

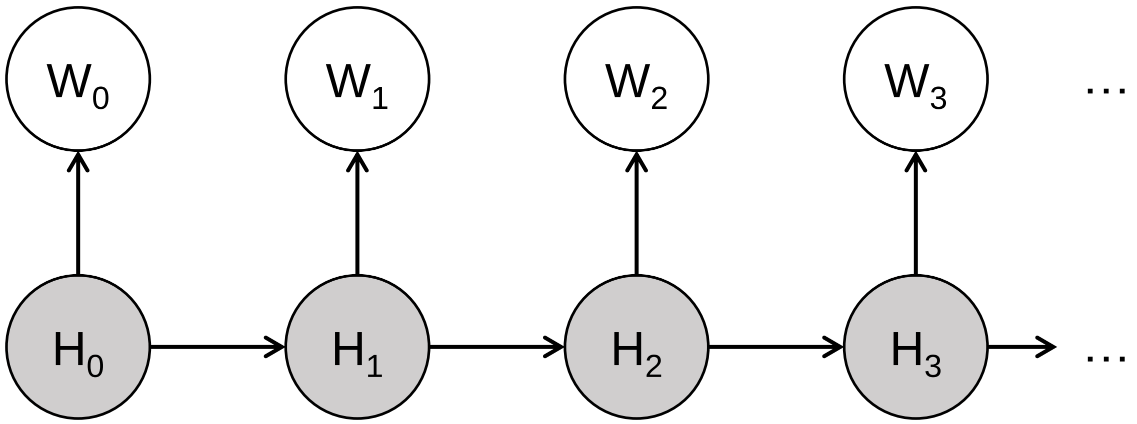


Case study: POS tagging

A popular application of HMMs in NLP is part-of-speech tagging

We imagine a generative story where parts-of-speech occur in a Markov chain, and then they emit words conditioned on their value.

i	sentence	you	to	read	this	sentence	.
PP	V	PP	PREP	V	DET	NN	PUNCT



Transition matrix

$P(H_t|H_{t-1})$

		h_0	h_1	…
	h_0	…	…	…
H_{t-1}	h_1	…	…	..
	…	...	…	…

Emission matrix

$P(W_t|H_t)$

		w_0	w_1	…
	h_0	…	…	…
H_t	h_1	…	…	..
	…	...	…	…

Things we want to do with HMMs

HMMs diverge from N-gram and Naïve Bayes in requiring complicated algorithms to do certain things with them

Also, multiple scenarios of each operation

Learning
- Labeled
- Unlabeled

Inference
- Likelihood
- Decoding

Generation
- Special case of inference

Speech and Language Processing (Jurafsky & Martin) is a definitive source: https://web.stanford.edu/~jurafsky/slp3/ (Appendix A)

Doing inference on HMMs

Inference: Given the values of some of the nodes, what are the most likely values of other nodes?

Two important inference problems:

- **Likelihood**: given the full model, what is the likelihood of a given output text?
  - **Forward algorithm**
- **Decoding**: given the full model and a particular output text, what was the most likely sequence of hidden states that generated it?
  - **Viterbi algorithm**

Generation is a special case of likelihood inference, because we can just generate new tokens according to their likelihood.

Likelihood

Likelihood: Given an HMM $\lambda = (A,B)$ and an observation sequence O, determine the likelihood $P(O|\lambda)$.

- A is transition matrix, B is emission matrix

POS example: Given the whole model of POS transitions and emission probabilities, what is the overall likelihood of a piece of text?

Forward algorithm

- Dynamic programming algorithm
- $O(N^2T)$ time complexity
  - T is sequence length and N is number of possible states

Likelihood case study: POS tagging

POS-tagging example: Given the whole model of POS transitions and emission probabilities, what is the overall likelihood of a piece of text?

i	sentence	you	to	read	this	sentence	.
?	?	?	?	?	?	?	?

Overall likelihood of text

Transition matrix

$P(H_t|H_{t-1})$

		h_0	h_1	…
	h_0	…	…	…
H_{t-1}	h_1	…	…	..
	…	...	…	…

Emission matrix

$P(W_t|H_t)$

		w_0	w_1	…
	h_0	…	…	…
H_t	h_1	…	…	..
	…	...	…	…

Likelihood case study: POS tagging

Tricky because while there is a most likely sequence of POS tags…

i	sentence	you	to	read	this	sentence	.
PP	V	PP	PREP	V	DET	NN	PUNCT

Overall likelihood of text

Transition matrix

$P(H_t|H_{t-1})$

		h_0	h_1	…
	h_0	…	…	…
H_{t-1}	h_1	…	…	..
	…	...	…	…

Emission matrix

$P(W_t|H_t)$

		w_0	w_1	…
	h_0	…	…	…
H_t	h_1	…	…	..
	…	...	…	…

Likelihood case study: POS tagging

…other sequences are possible under the transition matrix, and could also have lead to the given text.

- So we need to account for **every** way the text could have been generated.

i	sentence	you	to	read	this	sentence	.
~~NN~~	~~JJ~~	~~NN~~	~~V~~	~~PREP~~	~~JJ~~	~~PREP~~	~~PREP~~

Overall likelihood of text

Transition matrix

$P(H_t|H_{t-1})$

		h_0	h_1	…
	h_0	…	…	…
H_{t-1}	h_1	…	…	..
	…	…	…	…

Emission matrix

$P(W_t|H_t)$

		w_0	w_1	…
	h_0	…	…	…
H_t	h_1	…	…	..
	…	…	…	…

Forward algorithm

For a particular state sequence Q, it is pretty easy to figure out the likelihood of a given output sequence O:

$$P(O|Q) \;=\; \prod_{i=1}^{T} P(o_i|q_i)$$

But: how do we calculate this for **every** possible state sequence?

Brute force: enumerate every possible state sequence, calculate probability of it (using chain rule), then use chain rule again to calculate P(O)

Key insight: instead of calculating the likelihood of every possible state sequence (which is exponentially large), we can just calculate, for each time step t, the total likelihood of having ended up at each state at that timestep

- And then we can just use the chain rule:

$$P(O) = \sum_{Q} P(O,Q) = \sum_{Q} P(O|Q)P(Q)$$

- Form of **dynamic programming**

Forward algorithm

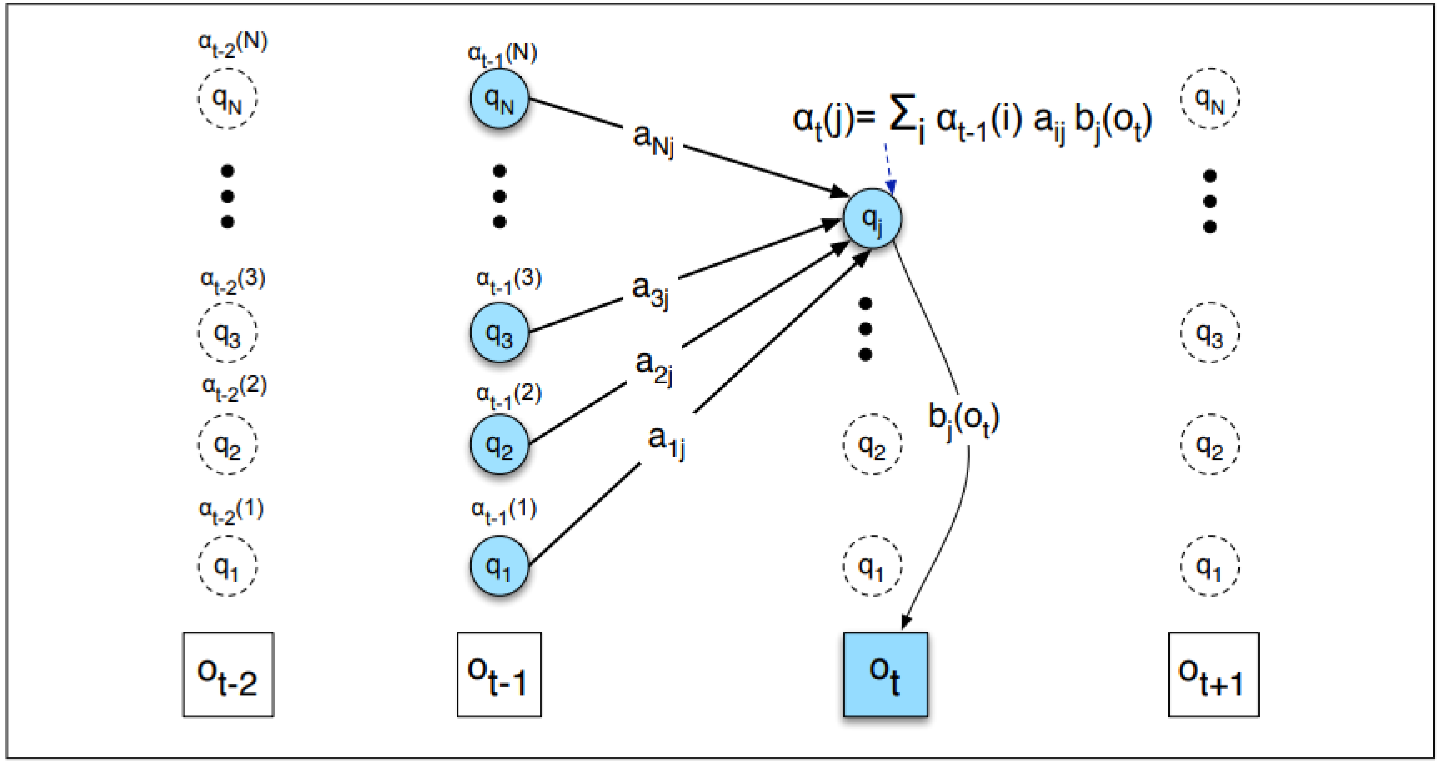


Figure A.6 Visualizing the computation of a single element $\alpha_t(i)$ in the trellis by summing all the previous values α_{t-1}, weighted by their transition probabilities a, and multiplying by the observation probability $b_i(o_t)$. For many applications of HMMs, many of the transition probabilities are 0, so not all previous states will contribute to the forward probability of the current state. Hidden states are in circles, observations in squares. Shaded nodes are included in the probability computation for $\alpha_t(i)$.

Forward algorithm

function FORWARD(*observations* of len T, *state-graph* of len N) **returns** *forward-prob*

create a probability matrix *forward[N,T]*
for each state s **from** 1 **to** N **do** ; initialization step
$$forward[s,1] \leftarrow \pi_s * b_s(o_1)$$
for each time step t **from** 2 **to** T **do** ; recursion step
for each state s **from** 1 **to** N **do**
$$forward[s,t] \leftarrow \sum_{s'=1}^{N} forward[s',t-1] * a_{s',s} * b_s(o_t)$$
$$forwardprob \leftarrow \sum_{s=1}^{N} forward[s,T]$$; termination step
return *forwardprob*

Figure A.7 The forward algorithm, where $forward[s,t]$ represents $\alpha_t(s)$.

Likelihood case study: POS tagging

POS-tagging example: Given the whole model of POS transitions and emission probabilities, what is the overall likelihood of a piece of text?

Transition matrix

$P(H_t|H_{t-1})$

		h_0	h_1	…
	h_0	…	…	…
H_{t-1}	h_1	…	…	..
	…	...	…	…

<S>

i	sentence	you	to	read	this	sentence	.
?	?	?	?	?	?	?	?

Step 1: Calculate distribution of first POS, given start-of-sequence

$\mathbf{P(H_0|<S>)}$

=

PP: 0.4

DET: 0.3

V: 0.1

…

Step 2: Calculate likelihood of first word being "i", given distribution of first POS

$\mathbf{P(W_0=\text{"i"} \mid H_0)}$

=

$P(W_0=\text{"i"} \mid H_0=PP)P(H_0=PP)$

$+P(W_0=\text{"i"} \mid H_0=DET)P(H_0=DET)$

$+P(W_0=\text{"i"} \mid H_0=V)P(H_0=V)$

$+…$

Step 3: Calculate distribution of second POS, given distribution of first POS

$\mathbf{P(H_1 \mid H_0)}$

=

$P(H_1=PP \mid H_0)$

$+P(H_1=DET \mid H_0)$

$+P(H_1=V \mid H_0)$

$+…$

…

Emission matrix

$P(W_t|H_t)$

		w_0	w_1	…
	h_0	…	…	…
H_t	h_1	…	…	..
	…	...	…	…

Decoding

Decoding: Given as input an HMM $\lambda = (A,B)$ and a sequence of observations $O = o_1, o_2, \ldots, o_T$, find the most probable sequence of states $Q = q_1 q_2 q_3 \ldots q_T$

POS example: Given a piece of text, what is the most likely sequence of parts of speech to have generated that text?

Viterbi algorithm

- Another dynamic programming algorithm

Decoding case study: POS tagging

POS-tagging example: Given a piece of text, what is the most likely sequence of parts of speech to have generated that text?

i	sentence	you	to	read	this	sentence	.
?	?	?	?	?	?	?	?

Transition matrix

$P(H_t|H_{t-1})$

		h_0	h_1	…
	h_0	…	…	…
H_{t-1}	h_1	…	…	..
	…	…	…	…

Emission matrix

$P(W_t|H_t)$

		w_0	w_1	…
	h_0	…	…	…
H_t	h_1	…	…	..
	…	…	…	…

Viterbi algorithm

Similar to Forward algorithm

Brute force: enumerate every possible sequence of states, calculate probability of each one (using chain rule), use the chain rule to calculate P(O) given each possible sequence, then use Bayes rule to find the most likely state sequence

Key idea: For each timestep t, for each state j, calculate and store the probability of having ended up in state j at time t… **after following the most likely state sequence prior to t**.

- So, another **dynamic programming** approach

Takes max over previous possible states rather than sum (which is what Forward algorithm does)

Viterbi algorithm

function VITERBI(*observations* of len T,*state-graph* of len N) **returns** *best-path*, *path-prob*

create a path probability matrix *viterbi[N,T]*
for each state s **from** 1 **to** N **do** ; initialization step
    $viterbi[s,1] \leftarrow \pi_s \, * \, b_s(o_1)$
    $backpointer[s,1] \leftarrow 0$
for each time step t **from** 2 **to** T **do** ; recursion step
  for each state s **from** 1 **to** N **do**
    $viterbi[s,t] \leftarrow \max_{s'=1}^{N} \; viterbi[s',t-1] \, * \, a_{s',s} \, * \, b_s(o_t)$
    $backpointer[s,t] \leftarrow \operatorname{argmax}_{s'=1}^{N} \; viterbi[s',t-1] \, * \, a_{s',s} \, * \, b_s(o_t)$
$bestpathprob \leftarrow \max_{s=1}^{N} \; viterbi[s,T]$; termination step
$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} \; viterbi[s,T]$; termination step
bestpath ← the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

Figure A.9 Viterbi algorithm for finding optimal sequence of hidden states. Given an observation sequence and an HMM $\lambda = (A,B)$, the algorithm returns the state path through the HMM that assigns maximum likelihood to the observation sequence.

Decoding case study: POS tagging

POS-tagging example: Given the whole model of POS transitions and emission probabilities, what is the overall likelihood of a piece of text?

<S>

i	sentence	you	to	read	this	sentence	.
?	?	?	?	?	?	?	?

Step 1: Calculate distribution of first POS, given start-of-sequence and first word
$P(H_0|\text{<S>},W_0=\text{“i”})$

Step 2: Calculate distribution of second POS, given **MOST LIKELY** value of first POS, and current word
$P(H_1|\ H_0=PP,\ W_0=\text{“sentence”}\ )$

…

Transition matrix

$P(H_t|H_{t-1})$

		h_0	h_1	…
	h_0	…	…	…
H_{t-1}	h_1	…	…	..
	…	...	…	…

Emission matrix

$P(W_t|H_t)$

		w_0	w_1	…
	h_0	…	…	…
H_t	h_1	…	…	..
	…	...	…	…

Learning HMMs

Learning: Given some text data, what are the most likely parameters

Two possible scenarios:

- Supervised
  - Ground-truth examples of both words and hidden states present
  - **Frequency counting**
- Unsupervised
  - Ground-truth examples of words… but not hidden states?
  - **Forward-backward algorithm**

Supervised learning

Given a corpus of text data **and** associated hidden state values, find most likely parameters for transition matrix and emission matrix

POS example:

i	sentence	you	to	read	this	sentence	.
PP	V	PP	PREP	V	DET	NN	PUNCT

i	dug	the	well	very	well	.	
PP	V	DET	NN	ADV	ADV	PUNCT	

Easy—just count.

- Populate transition matrix by counting POS→POS pairs
- Populate emission matrix by counting POS→Word pairs

Unsupervised learning

Given a corpus of text and **no** associated ground-truth hidden state values, find most likely parameters for transition and emission matrix

POS example:

i	sentence	you	to	read	this	sentence	.
?	?	?	?	?	?	?	?

i	dug	the	well	very	well	.
?	?	?	?	?	?	?

Much harder. Depends on finding implicit patterns in the data where certain words seem to be generated by certain hidden states with a certain transition matrix…

- Ends up being a form of clustering
- Use the **Forward-Backward** algorithm

Forward-backward Algorithm

- Also known as the **Baum-Welch algorithm**
- Special case of the **Expectation-Maximization algorithm**
- Iterative, approximate algorithm (rather than dynamic programming)

Basic idea: start with an initial guess for A and B.

- **Expectation step**: Generate most likely state values given (A, B, and O).
  - Use Viterbi algorithm for this
- **Maximization step**: Calculate most likely parameter values for A and B given generated state values and O
  - Same as labeled learning
- Repeat those two steps until convergence

Why does this work??? Black magic.

Forward-backward algorithm

function FORWARD-BACKWARD(*observations* of len T, *output vocabulary* V, *hidden state set* Q) **returns** *HMM=(A,B)*

initialize A and B
iterate until convergence
E-step

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{\alpha_T(q_F)} \quad \forall\, t \text{ and } j$$

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij} b_j(o_{t+1})\beta_{t+1}(j)}{\alpha_T(q_F)} \quad \forall\, t,\ i, \text{ and } j$$

M-step

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{k=1}^{N} \xi_t(i,k)}$$

$$\hat{b}_j(v_k) = \frac{\sum_{t=1 \text{ s.t. } O_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

return A, B

Figure A.14 The forward-backward algorithm.

Concluding thoughts

Naïve bayes: application of Bayes Rule + unigram language modeling to classification

Bayes nets: General term for directed probabilistic graphical models (like Naïve Bayes, HMMs)

Hidden Markov models

- **Key idea**: **latent, hidden states** that are responsible for generating the text
  - HMMs most direct implementation of this idea
- Need fancy algorithms for learning and inference
- Not incredibly well-supported in Python
  - https://hmmlearn.readthedocs.io/en/latest/
- Largely superseded by RNNs these days
  - BUT… neurosymbolic and Bayesian neural networks are a hot research area: https://www.cs.toronto.edu/~duvenaud/distill_bayes_net/public/