Statistical Natural Language Processing

Sequence learning

Çağrı Çöltekin

University of Tübingen
Seminar für Sprachwissenschaft

Summer Semester 2021

version: d7c7191 @2021-06-23

Some (typical) machine learning applications

	x (input)	y (output)
Spam detection	document	spam or not
Sentiment analysis	product review	sentiment
Medical diagnosis	patient data	diagnosis
Credit scoring	financial history	loan decision

The cases (input–output) pairs are assumed to be *independent and identically distributed* (i.i.d.).

Structured prediction

In many applications, the i.i.d. assumption is wrong

	x (input)	y (output)
POS tagging	word sequence	POS sequence
Parsing	word sequence	parse tree
OCR	image (array of pixels)	sequences of letters
Gene prediction	genome	genes

Structured prediction

In many applications, the i.i.d. assumption is wrong

	$\boldsymbol{x}$ (input)	$\boldsymbol{y}$ (output)
POS tagging	word sequence	POS sequence
Parsing	word sequence	parse tree
OCR	image (array of pixels)	sequences of letters
Gene prediction	genome	genes

Structured/sequence learning is prevalent in NLP.

In this lecture ...

- Hidden Markov models (HMMs)
- A short note on graphical probabilistic models
- Alternatives to HMMs (briefly): HMEM / CRF

... and soon

- Recurrent neural networks

Recap: chain rule

We rewrite the relation between the joint and the conditional probability as

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

We can also write the same quantity as,

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

In general, for any number of random variables, we can write

$$P(X_1, X_2, \ldots, X_n) = P(X_1 \mid X_2, \ldots, X_n)P(X_2, \ldots, X_n)$$

Recap: (conditional) independence

If two variables X and Y are independent,

$$P(X \mid Y) = P(X) \quad \text{and} \quad P(X, Y) = P(X)P(Y)$$

If two variables X and Y are independent given another variable Z,

$$P(X, Y \mid Z) = P(X \mid Z)P(Y \mid Z)$$

An example: probability of a sentence

$$P(\text{It's a beautiful day}) = ?$$

- We cannot just count all occurrences of the sentence, and divide it to the total number of sentences in English

An example: probability of a sentence

$$P(\text{It's a beautiful day}) = ?$$

- We cannot just count all occurrences of the sentence, and divide it to the total number of sentences in English
- But we can calculate its probability based on the probabilities of the words. Using chain rule

$$\begin{aligned} P(\text{It's a beautiful day}) &= P(\text{day} \mid \text{It's a beautiful})P(\text{It's a beautiful}) \\ &= P(\text{day} \mid \text{It's a beautiful})P(\text{beautiful} \mid \text{It's a})P(\text{It's a}) \\ &= P(\text{day} \mid \text{It's a beautiful})P(\text{beautiful} \mid \text{It's a})P(\text{a} \mid \text{It's})P(\text{It's}) \end{aligned}$$

An example: probability of a sentence

$$P(\text{It's a beautiful day}) = ?$$

- We cannot just count all occurrences of the sentence, and divide it to the total number of sentences in English
- But we can calculate its probability based on the probabilities of the words. Using chain rule

$$\begin{aligned}
P(\text{It's a beautiful day}) &= P(\text{day} \mid \text{It's a beautiful})P(\text{It's a beautiful}) \\
&= P(\text{day} \mid \text{It's a beautiful})P(\text{beautiful} \mid \text{It's a})P(\text{It's a}) \\
&= P(\text{day} \mid \text{It's a beautiful})P(\text{beautiful} \mid \text{It's a})P(\text{a} \mid \text{It's})P(\text{It's})
\end{aligned}$$

- Did we solve the problem?

Markov chains

calculating probabilities

Given a sequence of events (or states), $q_1, q_2, \ldots q_t$,

- In a *first-order* Markov chain, the probability of an event q_t is

$$P(q_t | q_1, \ldots, q_{t-1}) = P(q_t | q_{t-1})$$

- In higher order chains, the dependence of history is extended, e.g., second-order Markov chain:

$$P(q_t | q_t, \ldots, q_{t-1}) = P(q_t | q_{t-2}, q_{t-1})$$

- The conditional independence properties simplify the probability distributions

Markov chains

definition

A Markov model is defined by,

- A set of states $Q = \{q_1, \ldots, q_n\}$
- A special start state q_0
- A transition probability matrix

$$\boldsymbol{A} = \begin{bmatrix} a_{01} & a_{02} & \ldots & a_{0n} \\ a_{11} & a_{12} & \ldots & a_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix}$$

where a_{ij} is the probability of transition from state i to state j

Back to sentence probability example

- With a first-order Markov assumption,

$$\begin{aligned} P(\text{It's a beautiful day}) &= P(\text{day} \mid \text{It's a beautiful}) P(\text{beautiful} \mid \text{It's a}) P(\text{a} \mid \text{It's}) P(\text{It's}) \\ &= P(\text{day} \mid \text{beautiful}) P(\text{beautiful} \mid \text{a}) P(\text{a} \mid \text{It's}) P(\text{It's} \mid \langle S \rangle) \end{aligned}$$

- Now the probabilities are easier to calculate
- The above approach is an example of *n-gram language models* that we will get back to very soon

Hidden/latent variables

- In many machine learning problems we want to account for unobserved/unobservable *latent* or *hidden* variables
- Some examples
  - 'personality' in many psychological data
  - 'topic' of a text
  - 'socio-economic class' of a speaker
- Latent variables make learning difficult: since we cannot observe them, how do we set the parameters?

Learning with hidden variables

(Another) informal/quick introduction to the EM algorithm

- The EM algorithm (or its variants) is used in many machine learning models with latent/hidden variables

1. Randomly initialize the parameters
2. Iterate until convergence:

E-step compute likelihood of the data, given the parameters

M-step re-estimate the parameters using the predictions based on the E-step

Hidden Markov models (HMM)

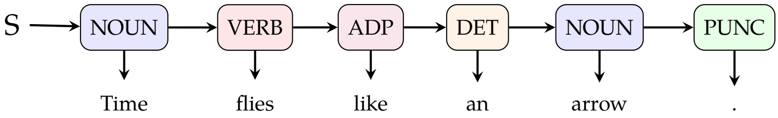
- HMMs are like Markov chains: probability of a state depends only a limited history of previous states

$$P(q_t|q_1, \ldots, q_{t-1}) = P(q_t|q_{t-1})$$

- Unlike Markov chains, state sequence is hidden, they are not the observations
- At every state q_t, an HMM *emits* an output, o_t, whose probability depends only on the associated hidden state
- Given a state sequence $\mathbf{q} = q_1, \ldots, q_T$, and the corresponding observation sequence $\mathbf{o} = o_1, \ldots, o_T$,

$$P(\mathbf{o}, \mathbf{q}) = p(q_1) \left[\prod_2^T P(q_t|q_{t-1})\right] \prod_1^T P(o_t|q_t)$$

Example: HMMs for POS tagging



- The tags are hidden
- Probability of a tag depends on the previous tag
- Probability of a word at a given state depends only on the current tag

HMMs: formal definition

An HMM is defined by

- A set of states $Q = \{q_1, \ldots, q_n\}$
- The set of possible observations $O = \{o_1, \ldots, o_m\}$
- A transition probability matrix

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix}$$

  a_{ij} is the probability of transition from state q_i to state q_j

- Initial probability distribution $\pi = \{P(q_1), \ldots, P(q_n)\}$
- Probability distributions of

$$\mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & \ldots & b_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \ldots & b_{mn} \end{bmatrix}$$

  b_{ij} is the probability of emitting output o_i at state q_j

A simple example

- Three states: N, V, D
- Four possible observations: a, b, c , d

$$
\mathbf{A} = \begin{array}{cc} & \begin{array}{ccc} \mathtt{N} & \mathtt{V} & \mathtt{D} \end{array} \\ & \left[\begin{array}{ccc} 0.2 & 0.7 & 0.1 \\ 0.5 & 0.1 & 0.4 \\ 0.8 & 0.1 & 0.1 \end{array}\right] \begin{array}{c} \mathtt{N} \\ \mathtt{V} \\ \mathtt{D} \end{array} \end{array}
\qquad
\mathbf{B} = \begin{array}{cc} & \begin{array}{ccc} \mathtt{N} & \mathtt{V} & \mathtt{D} \end{array} \\ & \left[\begin{array}{ccc} 0.1 & 0.1 & 0.5 \\ 0.4 & 0.5 & 0.1 \\ 0.4 & 0.3 & 0.1 \\ 0.1 & 0.1 & 0.3 \end{array}\right] \begin{array}{c} \mathtt{a} \\ \mathtt{b} \\ \mathtt{c} \\ \mathtt{d} \end{array} \end{array}
$$

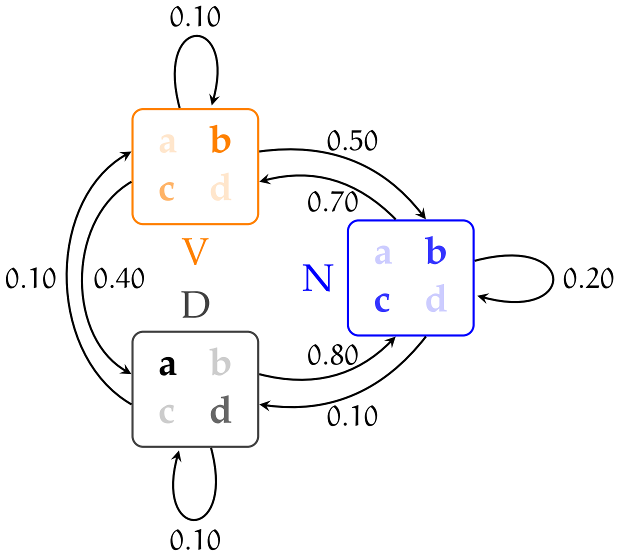
A simple example

- Three states: N, V, D
- Four possible observations: a, b, c , d

$$\mathbf{A} = \begin{array}{c c} \begin{array}{ccc} \mathtt{N} & \mathtt{V} & \mathtt{D} \end{array} & \\ \begin{bmatrix} 0.2 & 0.7 & 0.1 \\ 0.5 & 0.1 & 0.4 \\ 0.8 & 0.1 & 0.1 \end{bmatrix} & \begin{array}{c} \mathtt{N} \\ \mathtt{V} \\ \mathtt{D} \end{array} \end{array} \qquad \mathbf{B} = \begin{array}{c c} \begin{array}{ccc} \mathtt{N} & \mathtt{V} & \mathtt{D} \end{array} & \\ \begin{bmatrix} 0.1 & 0.1 & 0.5 \\ 0.4 & 0.5 & 0.1 \\ 0.4 & 0.3 & 0.1 \\ 0.1 & 0.1 & 0.3 \end{bmatrix} & \begin{array}{c} \mathtt{a} \\ \mathtt{b} \\ \mathtt{c} \\ \mathtt{d} \end{array} \end{array}$$

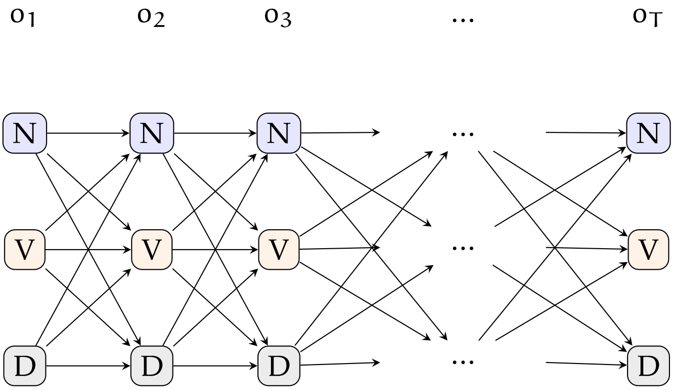
$$\boldsymbol{\pi} = (0.3, 0.1, 0.6)$$

HMM transition diagram



Unfolding the states

HMM lattice (or trellis)



HMMs: three problems

Recognition/decoding
: Calculating probability of state sequence, given an observation sequence

$$P(\mathbf{q} \mid \mathbf{o}; M)$$

Evaluation
: Calculating likelihood of a given sequence

$$P(\mathbf{o} \mid M)$$

Learning
: Given observation sequences, a set of states, and (sometimes) corresponding state sequences, estimate the parameters $(\boldsymbol{\pi}, \mathbf{A}, \mathbf{B})$ of the HMM

Assigning probabilities to observation sequences

$$P(\mathbf{o} \mid M) = \sum_{\mathbf{q}} P(\mathbf{o}, \mathbf{q} \mid M)$$

- We need to sum over an exponential number of hidden state sequences
- The solution is using a dynamic programming algorithm
  - for each node of the trellis, store *forward probabilities*

$$\alpha_{t,i} = \sum_{j}^{N} \alpha_{t-1,j} P(q_i|q_j) P(o_i|q_i)$$

Assigning probabilities to observation sequences

the forward algorithm

- Start with calculating all forward probabilities for $t = 1$

$$\alpha_{1,i} = \pi_i P(o_1 | q_i) \quad \text{for } 1 \leqslant i \leqslant |Q|$$

  store the α values

- For $t > 1$,

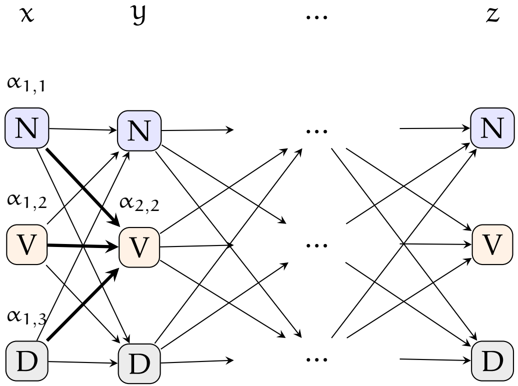
$$\alpha_{t,i} = \sum_{j=1}^{|Q|} \alpha_{t-1,j} P(q_i | q_j) P(o_t | q_i) \quad \text{for } 1 \leqslant i \leqslant |Q|, 2 \leqslant t \leqslant n$$

- Likelihood of the observation is the sum of the forward probabilities of the last step

$$P(\mathbf{o} | M) = \sum_{j=1}^{|Q|} \alpha_{n,j}$$

Forward algorithm

HMM lattice (or trellis)



$$\alpha_{1,1} = \pi_N b_{xN}$$

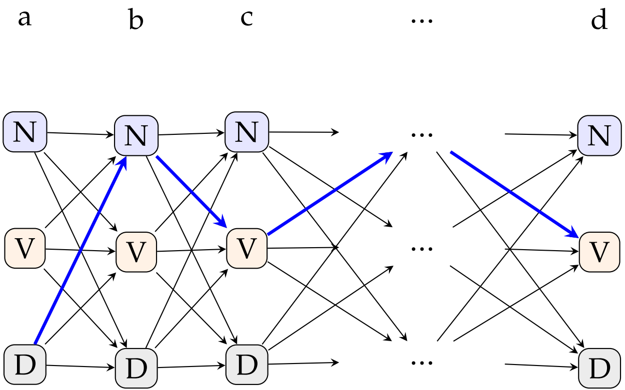
$$\alpha_{2,2} = \alpha_{1,1} a_{NV} b_{yV} + \alpha_{1,2} a_{VV} b_{yV} + \alpha_{1,3} a_{DV} b_{yV}$$

Determining best sequence of latent variables

Decoding

- We often want to know the hidden state sequence given an observation sequence, $P(\mathbf{q} \mid \mathbf{o}; M)$
  - For example, given a sequence of tokens, find the most likely POS tag sequence
- The problem (also the solution, the *Viterbi algorithm*) is very similar to the forward algorithm
- Two major differences
  - we store maximum likelihood leading to each node on the lattice
  - we also store backlinks, the previous state that leads to the maximum likelihood

HMM decoding problem



Learning the parameters of an HMM

supervised case

- We want to estimate $\boldsymbol{\pi}$, $\mathbf{A}$, $\mathbf{B}$
- If we have both the observation sequence $\mathbf{o}$ and the corresponding state sequence, MLE estimate is

$$\pi_i = \frac{C(q_0 \rightarrow q_i)}{\sum_k C(q_0 \rightarrow q_k)}$$
$$a_{ij} = \frac{C(q_i \rightarrow q_j)}{\sum_k C(q_i \rightarrow q_k)}$$
$$b_{ij} = \frac{C(q_i \rightarrow o_j)}{\sum_k C(q_i \rightarrow o_k)}$$

Learning the parameters of an HMM

- Given a training set with observation sequence(s) $\mathbf{o}$ and state sequence $\mathbf{q}$, we want to find $\theta = (\boldsymbol{\pi}, \mathbf{A}, \mathbf{B})$

$$\arg\max_{\theta} P(\mathbf{o} \mid \mathbf{q}, \theta)$$

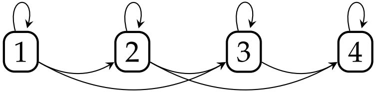
- Typically solved using EM
  1. Initialize θ
  2. Repeat until convergence

  E-step given θ, estimate the hidden state sequence

  M-step given the estimated hidden states, use 'expected counts' to update θ
- An efficient implementation of EM algorithm is called *Baum-Welch algorithm*, or *forward-backward algorithm*

HMM variations

- The HMMs we discussed so far are called *ergodic* HMMs: all a_{ij} are non-zero
- For some applications, it is common to use HMMs with additional restrictions
- A well known variant (Bakis HMM) allows only forward transitions



- The emission probabilities can also be continuous, e.g., $p(q|o)$ can be a normal distribution

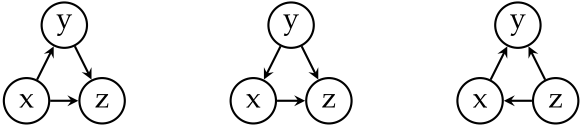
Directed graphical models: a brief divergence

Bayesian networks

- We saw earlier that joint distributions of multiple random variables can be factorized different ways

$$P(x, y, z) = P(x)P(y \mid x)P(z \mid x, y) = P(y)P(x \mid y)P(z \mid x, y) = P(z)P(x \mid z)P(y \mid x, z)$$

- *Graphical models* display this relations in graphs,
  - variables are denoted by nodes,
  - the dependence between the variables are indicated by edges
- Bayesian networks are directed acyclic graphs

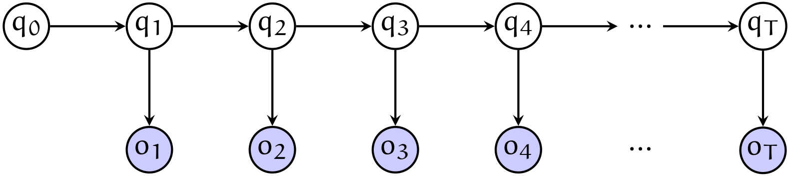


- A variable (node) depends only on its parents

Graphical models

- Graphical models define models involving multiple random variables
- It is generally more intuitive (compared to corresponding mathematical equations) to work with graphical models
- In a graphical model, by convention, the observed variables are shaded
- Graphs can also be undirected, which are also called *Markov random fields*

HMM as a graphical model



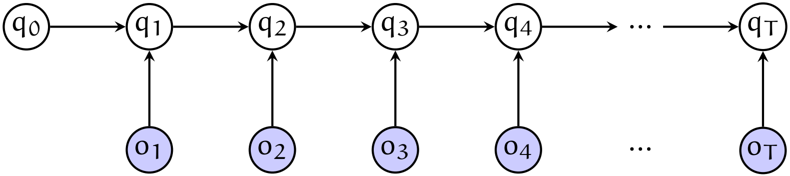
MaxEnt HMMs (MEMM)

- In HMMs, we model $P(\mathbf{q}, \mathbf{o}) = P(\mathbf{q})P(\mathbf{o} \mid \mathbf{q})$
- In many applications, we are only interested in $P(\mathbf{q} \mid \mathbf{o})$, which we can calculate using the Bayes theorem
- But we can also model $P(\mathbf{q} \mid \mathbf{o})$ directly using a *maximum entropy model*

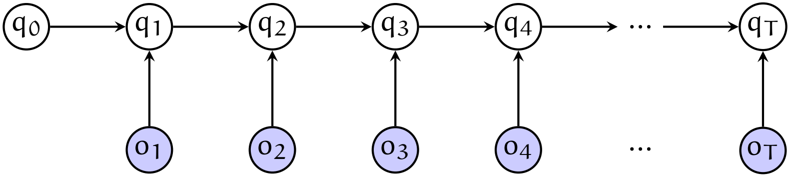
$$P(q_t \mid q_{t-1}, o_t) = \frac{1}{Z} e^{\sum w_i f_i(o_t, q_t)}$$

f_i are features – can be any useful feature
Z normalizes the probability distribution

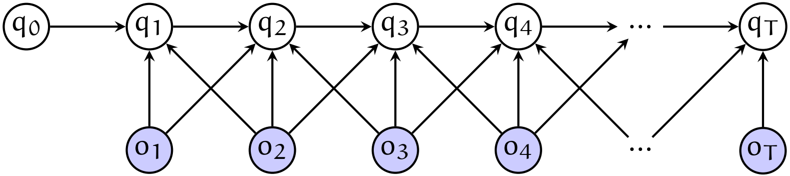
MEMMs as graphical models



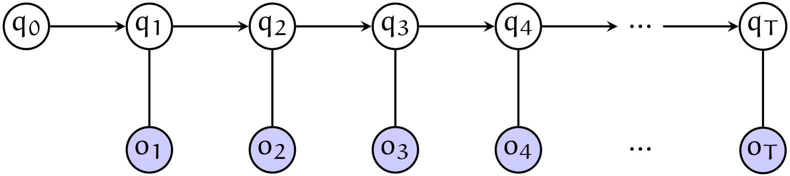
MEMMs as graphical models



We can also have other dependencies as features, for example



Conditional random fields



- A related model used in NLP is *conditional random field* (CRF)
- CRFs are *undirected models*
- CRFs also model $P(\mathbf{q} \mid \mathbf{o})$ directly

$$P(\mathbf{q} \mid \mathbf{o}) = \frac{1}{Z} \prod_t f(q_{t-1}, q_t) g(q_t, o_t)$$

Generative vs. discriminative models

- HMMs are *generative* models, they model the joint distribution
  - you can generate the output using HMMs
- MEMMs and CRFs are *discriminative* models they model the conditional probability directly
- It is easier to add arbitrary features on discriminative models
- In general: HMMs work well when the state sequence, $P(\mathbf{q})$, can be modeled well

Summary

- In many problems, e.g., POS tagging, i.i.d. assumption is wrong
- We need models that are aware of the effects of the sequence (or structure in general) in the data
- HMMs are generative sequence models:
  - Markov assumption between the hidden states (POS tags)
  - Observations (words) are conditioned on the state (tag)
- There are other sequence learning methods
  - Briefly mentioned: MEMM, CRF
  - Coming soon: recurrent neural networks

Next

- Recurrent and convolutional networks

blank

blank

blank

blank