

Statistical Natural Language Processing

Sequence learning

Çağrı Çöltekin

University of Tübingen
Seminar für Sprachwissenschaft

Summer Semester 2018

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/sequence learning is prevalent in NLP.

Sequence learning in NLP: examples

tokenization

The U.N. is the largest intergovernmental
BIIIOBIIIOBIIIOBIIIIIOBIIIIIIIIIIIIIIIIIIIO

Sequence learning in NLP: examples

named-entity recognition

UN Secretary-General Antonio Guterres
(ORG) (NONE) (PER) (PER)

plans to visit Ukraine
(NONE) (NONE) (NONE) (GEO)

Sequence learning in NLP: examples

part of speech tagging

Time flies like an arrow .
(NOUN) (VERB) (ADP) (DET) (NOUN) (PUNC)

- In all of the examples,
 - word/character–label pairs are not independent of each other
 - we want to get the best label sequence — not the best label for each word independently

In this lecture ...

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

... and later

- Recurrent neural networks

Markov chains

A *Markov chain* is a process where probability of an event depends only on the previous event(s).

A Markov chain is defined by,

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

$$A = \begin{bmatrix} a_{01} & a_{02} & \dots & a_{0n} \\ a_{11} & a_{12} & \dots & a_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \quad \text{where } a_{ij} \text{ is the probability of transition from state } i \text{ to state } j$$

Markov chains

calculating probabilities

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

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

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

- Sometimes this equality is just an assumption (as in n-gram models)
- In higher order chains, the dependence of history is extended, e.g., second-order Markov chain:

$$P(q_t | q_1, \dots, q_{t-1}) = P(q_t | q_{t-2}, q_{t-1})$$

A relevant example of a Markov Chain is n-gram language models (coming 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

An 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 data data, given the parameters

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

Hidden Markov models (HMM)

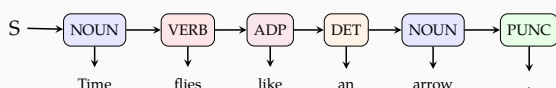
- HMMs are like Markov chains: probability of a state depends only a limited history of previous states

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

- 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 $\mathbf{q} = q_1, \dots, q_T$, and the corresponding observation sequence $\mathbf{o} = o_1, \dots, o_T$,

$$P(\mathbf{o}, \mathbf{q}) = p(q_1) \left[\prod_{t=2}^T P(q_t | q_{t-1}) \right] \prod_{t=1}^T P(o_t | q_t)$$

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 state $Q = \{q_1, \dots, q_n\}$
- The set of possible observations $V = \{v_1, \dots, v_m\}$
- A transition probability matrix

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \quad \begin{array}{l} a_{ij} \text{ is the probability of} \\ \text{transition from state } q_i \text{ to} \\ \text{state } q_j \end{array}$$

- Initial probability distribution $\pi = \{P(q_1), \dots, P(q_n)\}$

- Probability distributions of

$$\mathbf{B} = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \dots & b_{mn} \end{bmatrix} \quad \begin{array}{l} b_{ij} \text{ is the probability of} \\ \text{emitting output } o_i \text{ at state} \\ q_j \end{array}$$

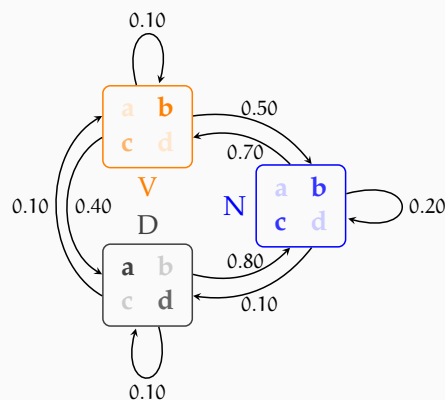
A simple example

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

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

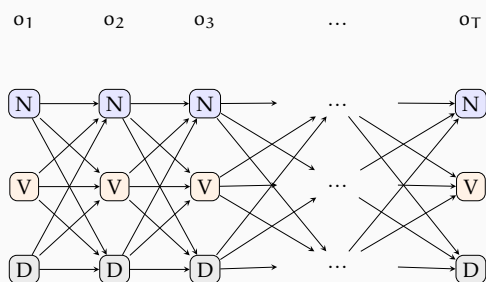
$$\pi = (0.3, 0.1, 0.6)$$

HMM transition diagram



Unfolding the states

HMM lattice (or trellis)



HMMs: three problems

Evaluation

Calculating likelihood of a given sequence

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

Recognition/decoding

Calculating probability of state sequence, given an observation sequence

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

Learning

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

Assigning probabilities to observation sequences

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

- 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*

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

Assigning probabilities to observation sequences

the forward algorithm

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

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

store the α values

- For $t > 1$,

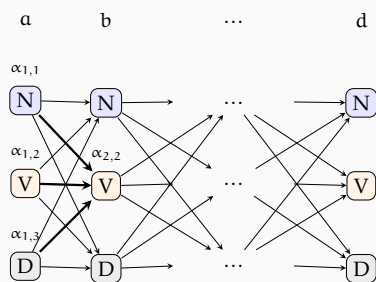
$$\alpha_{t,i} = \sum_{j=1}^N \alpha_{t-1,j} P(q_i | q_j) P(o_i | q_i) \quad \text{for } 1 \leq i \leq N, 2 \leq t \leq T$$

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

$$P(\mathbf{o} | M) = \sum_{i=1}^N \alpha_{i,T}$$

Forward algorithm

HMM lattice (or trellis)



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

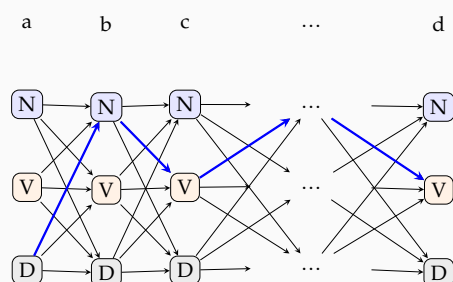
$$\alpha_{2,2} = \alpha_{1,1} a_{NV} b_{bV} + \alpha_{1,2} a_{VV} b_{bV} + \alpha_{1,3} a_{DV} b_{bV}$$

Determining best sequence of latent variables

Decoding

- We often want to know the hidden state sequence given an observation sequence, $P(\mathbf{q} | \mathbf{o}; M)$
 - 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 $\pi, \mathbf{A}, \mathbf{B}$
- 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 $\theta = (\pi, \mathbf{A}, \mathbf{B})$

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

- Unlike i.i.d. case, we cannot factorize the likelihood over all observations
- Instead we use 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., $p(\mathbf{q}|\mathbf{o})$ can be a normal distribution

Directed graphical models: a brief divergence

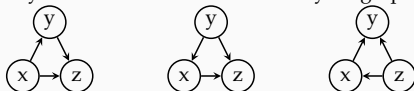
Bayesian networks

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

$$P(x, y, z) = P(x)P(y|x)P(z|x, y) = P(y)P(x|y)P(z|x, y) = P(z)P(x|z)P(y|x, z)$$

- 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

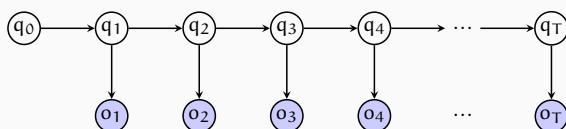


- A variable (node) depends only on its parents

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 called *Markov random fields*

HMM as a graphical model



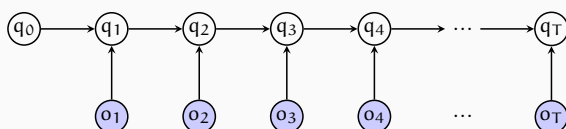
MaxEnt HMMs (MEMM)

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

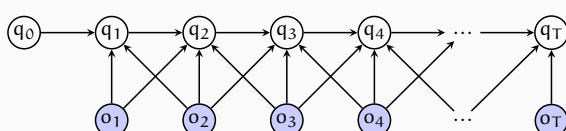
$$P(\mathbf{q}_t | \mathbf{q}_{t-1}, \mathbf{o}_t) = \frac{1}{Z} e^{\sum w_i f_i(\mathbf{o}_t, \mathbf{q}_t)}$$

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

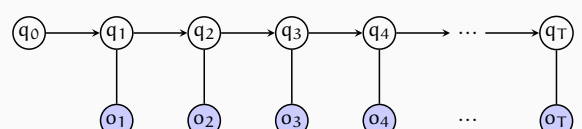
MEMMs as graphical models



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 $P(\mathbf{q} | \mathbf{o})$ directly

$$P(\mathbf{q} | \mathbf{o}) = \frac{1}{Z} \prod_t f(\mathbf{q}_{t-1}, \mathbf{q}_t) g(\mathbf{q}_t, \mathbf{o}_t)$$

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, $P(q)$, 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

Mon Unsupervised learning

Wed Practical session / discussion of Assignment 2 and (maybe) sample exam questions from Monday

Fri Neural networks