Hidden Markov Model

course based on Jurafsky and Martin [2009, Chap.6]

UPPSALA
UNIVERSITET
Marie Dubremetz
marie.dubremetz@lingfil.uu.se

Uppsala, April 2016

Presentation Plan

1. **Why Markov Models ?**
   - Historical Introduction
   - Formal Introduction
   - Formal Definition Exercise
   - Problem Definition Exercise

2. **Your First Optimisation Algorithm : Viterbi**
   - Trellis Exercise
   - Decoding Application Exercise

Table of Contents

Table of Contents

A little history of Markov Models

So far you have learnt how to compute ngram probabilities.(MLE+smoothing)

However, applications, like PoS tagging, require to compute and combine even more probabilities over the ngram probabilities.

For that you need to learn the concept of Hidden Markov Models.

For your general culture here is an history of how we came up with the idea of Markov chain :

https://www.youtube.com/watch?v=o-jdJxXL_W4

Note

Hidden Markov Model is built on Markov Chain but is more complicated since the observations are not suficient to fully determine the current state.

A little history of Markov Models

So far you have learnt how to compute ngram probabilities.(MLE+smoothing)
However, applications, like PoS tagging, require to compute and combine even more probabilities over the ngram probabilities.
For that you need to learn the concept of Hidden Markov Models.
For your general culture here is an history of how we came up with the idea of Markov chain :
https://www.youtube.com/watch?v=o-jdJxXL_W4

Note

Hidden Markov Model is built on Markov Chain but is more complicated since the observations are not suficient to fully determine the current state.

A little history of Markov Models

So far you have learnt how to compute ngram probabilities.(MLE+smoothing)
However, applications, like PoS tagging, require to compute and combine even more probabilities over the ngram probabilities.
For that you need to learn the concept of Hidden Markov Models.
For your general culture here is an history of how we came up with the idea of Markov chain :
https://www.youtube.com/watch?v=o-jdJxXL_W4

Note

Hidden Markov Model is built on Markov Chain but is more complicated since the observations are not suficient to fully determine the current state.

A little history of Markov Models

So far you have learnt how to compute ngram probabilities.(MLE+smoothing)

However, applications, like PoS tagging, require to compute and combine even more probabilities over the ngram probabilities.

For that you need to learn the concept of Hidden Markov Models.

For your general culture here is an history of how we came up with the idea of Markov chain :

https://www.youtube.com/watch?v=o-jdJxXL_W4

Note

Hidden Markov Model is built on Markov Chain but is more complicated since the observations are not suficient to fully determine the current state.

A little history of Markov Models

To sum up and relate to our general problems : Markov has demonstrated that you can build a conceptual machine that represents the generation of a succession of events even if those events are themselves dependent of another succession of other events happening in the background.
Instead of using it on meteorological events, letters or pearls (see video) we will use it on PoS Tags and words.

Table of Contents

Hidden Markov Models

- Markov models are probabilistic sequence models used for problems such as:
  1. Speech recognition
  2. Spell checking
  3. Part-of-speech tagging
  4. Named entity recognition
- A Markov model runs through a sequence of states emitting observable signals
- If the state sequence cannot be determined from the observation sequence, the model is said to be hidden

Markov Assumptions

- State transitions are assumed to be independent of everything except the current state:

$$P(q_1, \ldots, q_n) = \prod_{i=1}^{n} P(q_i \mid q_{i-1})$$

- Signal emissions are assumed to be independent of everything except the current state:

$$P(q_1, \ldots, q_n, s_1, \ldots, s_n) = P(q_1, \ldots, q_n) \prod_{i=1}^{n} P(s_i \mid q_i)$$

- NB: subscripts on states and signals refer to sequence positions

More Formally

$Q = q_1 q_2 \dots q_N$	a set of **states**
$A = a_{01} a_{02} \dots a_{n1} \dots a_{nn}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \dots o_N$	a set of **observations**, each one drawn from a vocabulary $V = v_1, v_2, ..., v_V$.
$B = b_i(o_t)$	A set of **observation likelihoods:**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state i.
q_0, q_{end}	a special **start and end state** which are not associated with observation.

Table of Contents

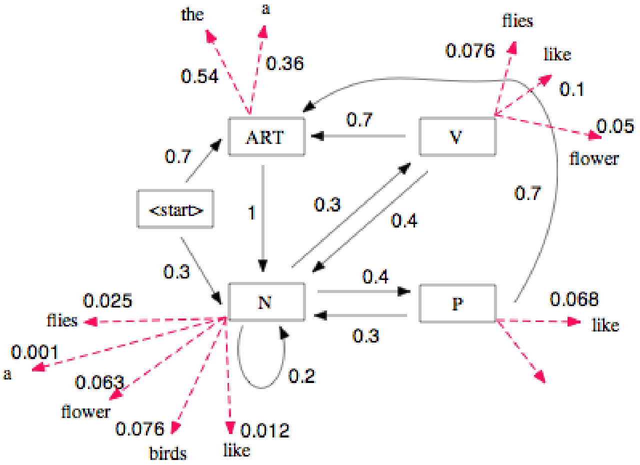
Exercise

The next slide describes a HMM
According to the formalism, color/circle what represents :

- A in red
- B in green
- V in dotted green
- Q in blue
- q_0 and q_{end} (if not missing) in dotted blue

$Q = q_1 q_2 \ldots q_N$	a set of **states**
$A = a_{01} a_{02} \ldots a_{n1} \ldots a_{nn}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \ldots o_N$	a set of **observations**, each one drawn from a vocabulary $V = v_1, v_2, ..., v_V$.
$B = b_i(o_t)$	A set of **observation likelihoods:**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state i.
q_0, q_{end}	a special **start and end state** which are not associated with observation.

A Simple First-Order HMM for Tagging



Conclusion on Exercise

Looks familiar ?

HMMs are like finite states transducers except :

- Transitions and emissions are decoupled
- The model first transitions to a state, then emits a symbol in that state
- Transitions and emissions are probabilistic

Table of Contents

Tasks on HMM

There is three kinds of problems when you deal with Hidden Markov Models :

Problem 1 (Computing Likelihood): Given an HMM $\lambda = (A,B)$ and an observation sequence O, determine the likelihood $P(O|\lambda)$.

Problem 2 (Decoding): Given an observation sequence O and an HMM $\lambda = (A,B)$, discover the best hidden state sequence Q.

Problem 3 (Learning): Given an observation sequence O and the set of states in the HMM, learn the HMM parameters A and B.

Write this down for the quiz !

QUIZ

Question

Can you find which sentence answers to which HMM Problem ?

1. The probability of this HMM to generate the sequence of observation O=[he,eats,cakes] is 0.34859
2. The transition probabilities are A=[0.6, 0.4...] and the observations likelyhood are B=[0.9,0.1...]
3. The most probable sequence of states that has generated O, is the state sequence Q=[Start, Noun, Verb, Adj, End]

- **a** Decoding
- **b** Likelihood
- **c** Learning

Table of Contents

Link to the slides on Viterbi :

The following slides (with a blue background) are extracted from this course :
http://courses.washington.edu/ling570/gina_fall11/slides/ling570_class12_viterbi.pdf
or http://stp.lingfil.uu.se/~marie/undervisning/textanalys16/levow.pdf (Slides designed by Fei Xia and Gina-Anne Levow used with their kind authorisation.)
We will learn formally what is the Viterbi algorithm. But don't hesitate to look at this link if you want more details about Viterbi implementation. Or if you would like to continue beyond what we will do in the next exercise.

Three Problems for HMMs

- Likelihood:
  - Find the probability of an observation sequence given a model
    - Forward algorithm
- Decoding:
  - Find the most likely path through a model given an observed sequence
    - Viterbi algorithm
- Learning:
  - Find the most likely model (parameters) given an observed sequence
    - Supervised (MLE) or unsupervised Baum-Welch

Decoding

- Have complete model of ngram POS tagging
  - Need to compute

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n)$$

- Possible approach:

Decoding

- Have complete model of ngram POS tagging
  - Need to compute

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n)$$

- Possible approach:
  - Enumerate all paths through HMM, pick highest score
  - Good idea?

Decoding

- Have complete model of ngram POS tagging
  - Need to compute

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}}\, P(t_1^n | w_1^n)$$

- Possible approach:
  - Enumerate all paths through HMM, pick highest score
  - Good idea? No. Why?

Decoding

- Have complete model of ngram POS tagging
  - Need to compute

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n)$$

- Possible approach:
  - Enumerate all paths through HMM, pick highest score
  - Good idea? No. Why?
    - Computationally intractable
  - Dynamic programming can help!

Dynamic Programming

Definition

Dynamic Programming (also known as dynamic optimization) is a method for solving a complex problem by breaking it down into a collection of simpler subproblems, solving each of those subproblems just once, and storing their solutions - ideally, using a memory-based data structure. The next time the same subproblem occurs, instead of recomputing its solution, one simply looks up the previously computed solution, thereby saving computation time at the expense of a (hopefully) modest expenditure in storage space.

Example

- time flies like an arrow

Table of Contents

Trellis Definition

Definition

When working with HMM, one can use a visual representation tool called the **trellis** : it allows you to draw all transitions possible for a given sequence.

Trellis Exercise

Definition

In the next slides, you will see an HMM Model and an HMM trellis.

1. In the **HMM Model** circle in red the transition probabilities and in green the emission probabilities.
2. In the **trellis**, Write in green above each states the probability of emission of the word. (Note : DT=D ; if there is no emission probability defined write 0)
3. In the **trellis** again, draw in red with arrows the path possible between states. Do not draw path towards states that have a 0 emission probability.
4. In the **trellis** finally, write above your paths the corresponding transition values

If you feel confused just follow what your teacher do on the whiteboard.

HMM Model

\start_state
0

\transition
0 BOS 1.0
BOS N 0.5
BOS DT 0.4
BOS V 0.1
DT N 1.0
N N 0.2
N V 0.7
N P 0.1
V DT 0.4
V N 0.4
V P 0.1
V V 0.1
P DT 0.6
P N 0.4

\emission

BOS <s> 1.0

N time 0.1

V time 0.1

N flies 0.1

V flies 0.2

V like 0.2

P like 0.1

DT an 0.3

N arrow 0.1

HMM Trellis

Adapted from F. Xia

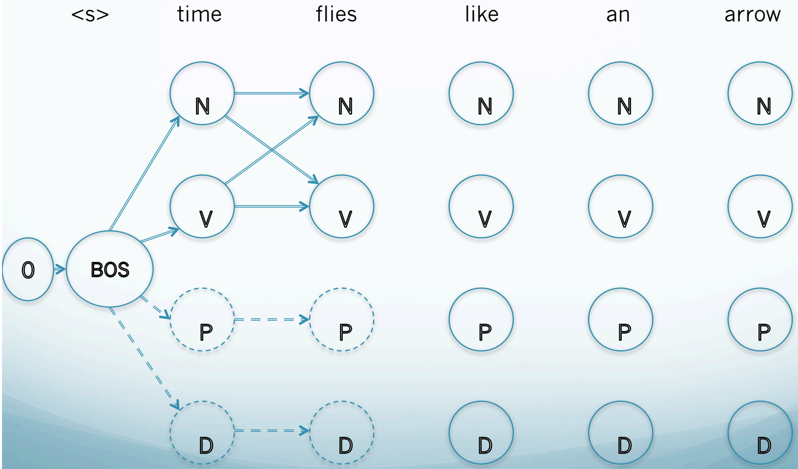
HMM Trellis

Adapted from F. Xia

HMM Trellis

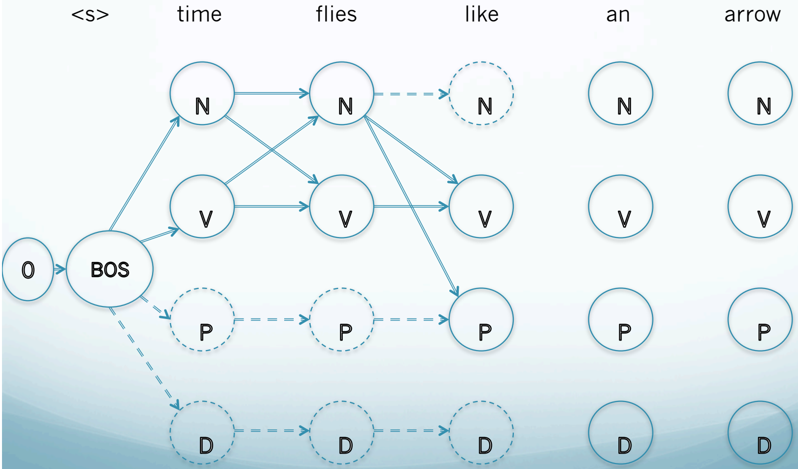
Adapted from F. Xia

HMM Trellis



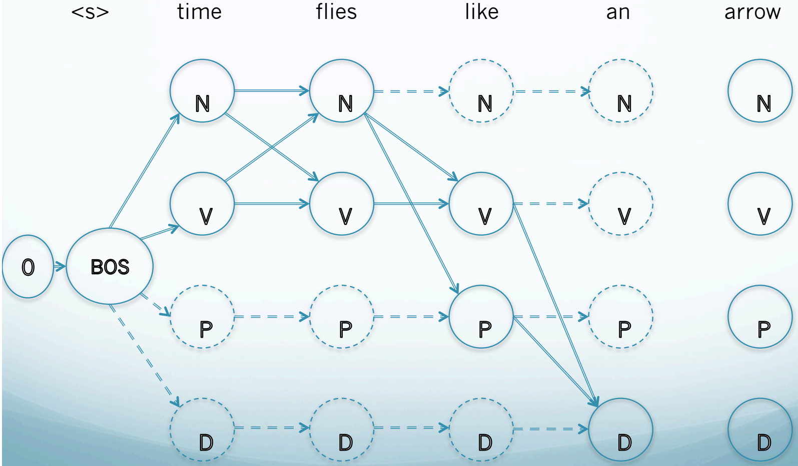
Adapted from F. Xia

HMM Trellis



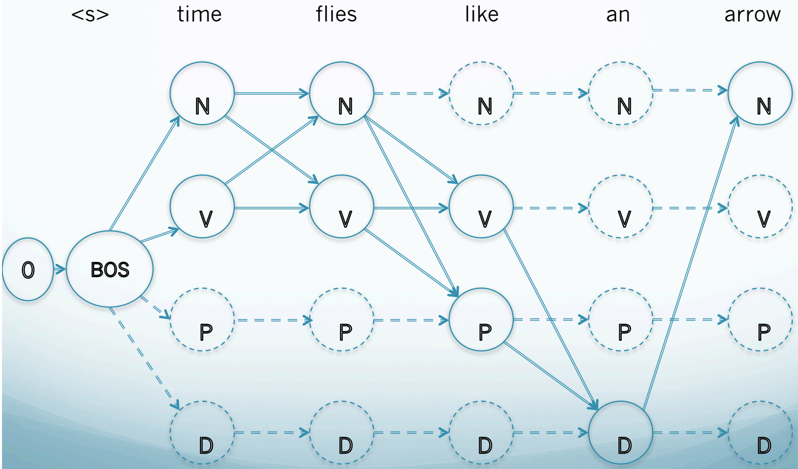
Adapted from F. Xia

HMM Trellis



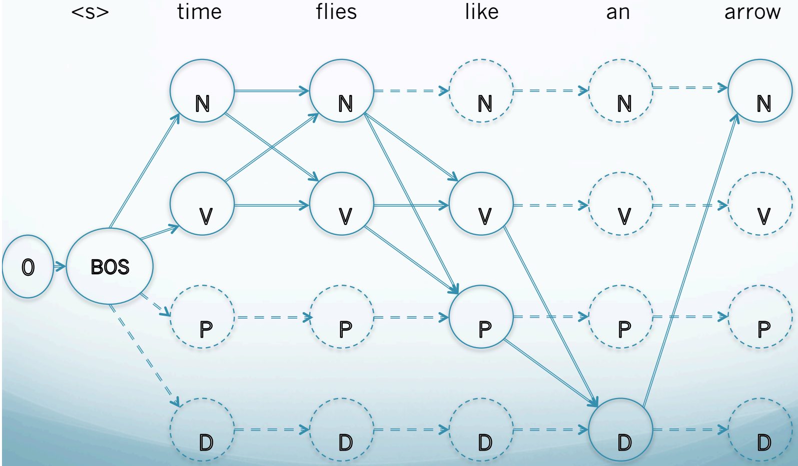
Adapted from F. Xia

HMM Trellis



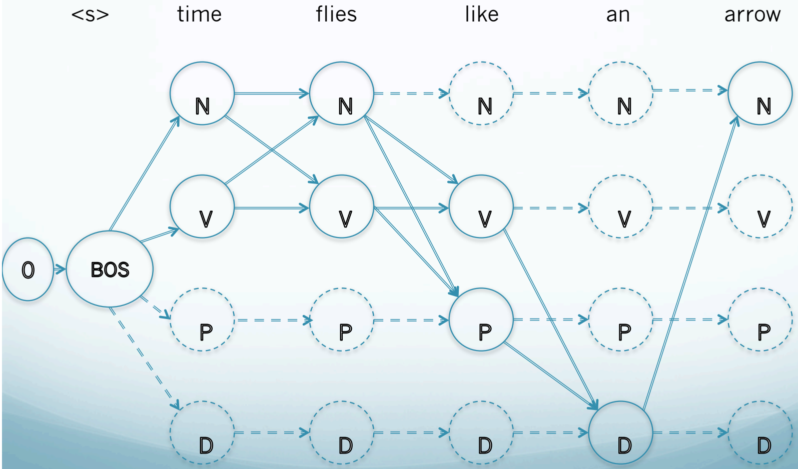
Adapted from F. Xia

HMM Trellis



Adapted from F. Xia

HMM Trellis



Adapted from F. Xia

Conclusion on Trellis Exercise

With a trellis you can keep track of all the sequence of tags possible, given a sentence, and write down the probabilities corresponding to the transitions and the emissions.

Decoding

- Find best hidden state sequence given observations and model
- Each cell represents:
  - Probability of being in state j after first t observations, passing through most probable sequence in model λ
  - Formally, $v_t(j) = \max_{q_0, q_1, q_2, \ldots, q_{t-1}} P(q_0, q_1, \ldots q_{t-1}, o_1, o_2, \ldots o_t, q_t = j \mid \lambda)$

Viterbi

- Initialization: $v_1(j) = a_{0j} b_j(o_1)$

  $bt_1(j) = 0$

Viterbi

- Initialization: $v_1(j) = a_{0j} b_j(o_1)$

  $bt_1(j) = 0$

- Recursion: $v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t), 1 \leq j \leq N, 1 < t \leq T$

  $bt_t(j) = \operatorname{argmax}_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t), 1 \leq j \leq N, 1 < t \leq T$

Viterbi

- Initialization: $v_1(j) = a_{0j} b_j(o_1)$

  $bt_1(j) = 0$

- Recursion: $v_t(j) = \max_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t), 1 \le j \le N, 1 < t \le T$

  $bt_t(j) = \operatorname{argmax}_{i=1}^{N} v_{t-1}(i) a_{ij} b_j(o_t), 1 \le j \le N, 1 < t \le T$

- Termination: $P* = v_t(q_F) = \max_{i=1}^{N} v_T(i) a_{iF}$

  $q_T* = bt_T(q_F) = \operatorname{argmax}_{i=1}^{N} v_T(i) a_{iF}$

	1	2	3	4	5	
N	0					
V	0					
P	0					
D	0					
BOS	P(BOS|0)* P(<s>|BOS) =1.0					

	1	2	3	4	5	
N	0	[BOS,1]* P(N|BOS)* P(time|N) =				
V	0					
P	0					
D	0					
BOS	1.0 0					

	1	2	3	4	5	
N	0	[BOS,1]* P(N|BOS)* P(time|N) =				
V	0	[BOS,1]* P(V|BOS)* P(time|V) =				
P	0					
D	0					
BOS	1.0 0					

	1	2	3	4	5	
N	0	[BOS,1]* P(N|BOS)* P(time|N) =0.05				
V	0	[BOS,1]* P(V|BOS)* P(time|V) =0.01				
P	0	0				
D	0	0				
BOS	1.00	0				

	1	2	3	4	5	
N	0	0.05 BOS				
V	0	0.01 BOS				
P	0	0				
D	0	0				
BOS	1.0 0	0				

	1	2	3	4	5	
N	0	0.05 BOS	max(([N,2]*P(N|N),[V, 2]*P(N|V))*P(flies|N)=			
V	0	0.01 BOS	max(([V,2]*P(V|V),[N, 2]*P(V|N))*P(flies|V)=			
P	0	0				
D	0	0				
BOS	1.00	0				

Table of Contents

Exercise

Make the calculations of the 3rd column.

Conclusion on Exercise

After computing 3rd column what do we learn ?

At column 3 we know that we should not consider other sequences than the one starting by O,BOS,N,N or O,BOS,N,V.
For instance we don't need to compute O,BOS,V,N,N...

Viterbi Summary

- Create an array
  - With columns corresponding to inputs
  - Rows corresponding to possible states

Viterbi Summary

- Create an array
  - With columns corresponding to inputs
  - Rows corresponding to possible states
- Sweep through the array in one pass
  - Fill columns left-to-right based on transition/emission

Viterbi Summary

- Create an array
  - With columns corresponding to inputs
  - Rows corresponding to possible states
- Sweep through the array in one pass
  - Fill columns left-to-right based on transition/emission
- Dynamic programming key

Viterbi Summary

- Create an array
  - With columns corresponding to inputs
  - Rows corresponding to possible states
- Sweep through the array in one pass
  - Fill columns left-to-right based on transition/emission
- Dynamic programming key
  - Store maximum probability of path to each cell

Viterbi Summary

- Create an array
  - With columns corresponding to inputs
  - Rows corresponding to possible states
- Sweep through the array in one pass
  - Fill columns left-to-right based on transition/emission
- Dynamic programming key
  - Store maximum probability of path to each cell
  - Store backpointers to recover path

Conclusion

Overall Summary

- HMM relate a sequence of observation (e.g., *words*) to a sequence of hidden states (e.g., *tags*).
- The process of discovering the sequence of hidden states given a sequence of observation is called decoding
- HMM needs to be handled with **dynamic programming**
- Viterbi algorithm is an efficient way to perform decoding.

References

Daniel Jurafsky and James H Martin. *Speech and Language Processing : An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, volume 163 of *Prentice Hall Series in Artificial Intelligence*. Prentice Hall, 2009.

The blue slides :
http://courses.washington.edu/ling570/gina_fall11/slides/ling570_class12_viterbi.pdf
http://stp.lingfil.uu.se/~marie/undervisning/textanalys16/levow.pdf
(Slides designed by Fei Xia and Gina-Anne Levow used with their kind authorisation.)