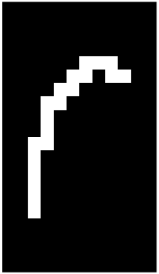
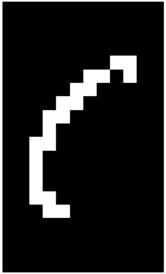
Sequence Modelling

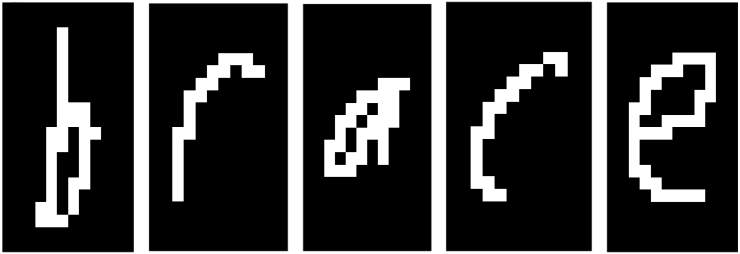
Mladen Kolar (mkolar@chicagobooth.edu)

Handwritten character recognition



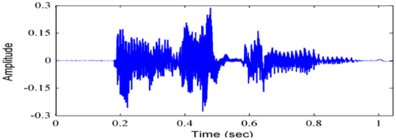


Structured prediction



Sequential data

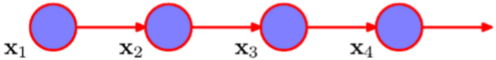
- time-series data (speach)
- characters in a sentence
- base pairs along a DNA strand



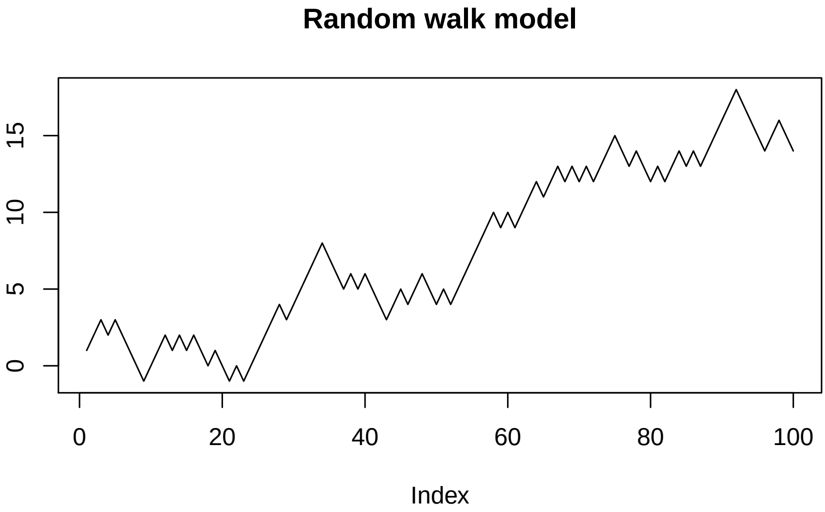
Markov Model

Joint distribution of n arbitrary random variables

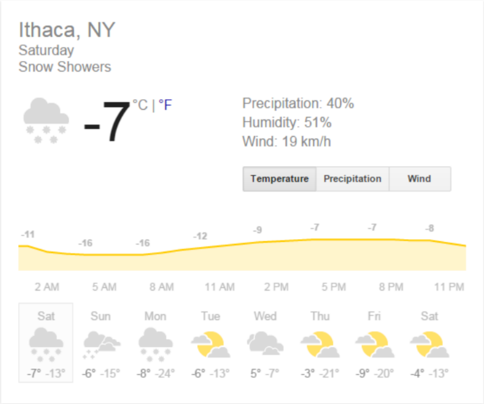
$$
\begin{aligned}
P(X_1, X_2, \ldots, X_n) &= P(X_1) \cdot P(X_2 \mid X_1) \cdot \ldots \cdot P(X_n \mid X_{n-1}) \\
&= P(X_1) \cdot \prod_{i=2}^{n} P(X_i \mid X_{i-1})
\end{aligned}
$$



Example

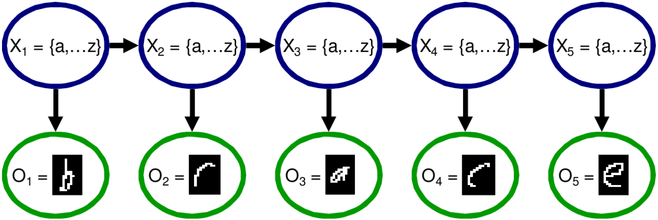


Example



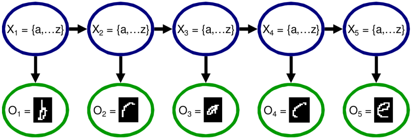
Ithaca, NY
Saturday
Snow Showers
-7 °C | °F
Precipitation: 40%
Humidity: 51%
Wind: 19 km/h
Temperature
Precipitation
Wind
-11
-16
-16
-12
-9
-7
-7
-8
2 AM
5 AM
8 AM
11 AM
2 PM
5 PM
8 PM
11 PM
Sat
Sun
Mon
Tue
Wed
Thu
Fri
Sat
-7° -13°
-6° -15°
-8° -24°
-6° -13°
5° -7°
-3° -21°
-9° -20°
-4° -13°

Understanding the HMM Semantics



$P(O_1 \mid X_1 = x_1)$ probability of an image given the letter is x_1
$P(X_2 = x_2 \mid X_1 = x_1)$ probability that letter x_2 will follow letter x_1
Decision about X_2 is influenced by all letters.

HMMs semantics: Details

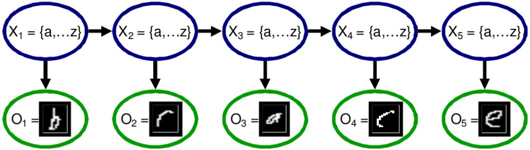


Need just 3 distributions:
$P(X_1)$ starting state distribution
$P(X_i \mid X_{i-1}) = P(X_j \mid X_{j-1}) \;\; \forall j$, transition model
$P(O_i \mid X_i) = P(O_j \mid X_j) \;\; \forall j$, observation model

Parameter sharing:

- more bias, need less data to train
- can deal with words of different length

HMMs semantics: Joint distribution

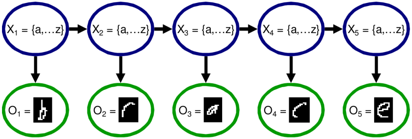


$$P(X_1)$$
$$P(X_i \mid X_{i-1})$$
$$P(O_i \mid X_i)$$

$$P(X_1, \ldots, X_n, O_1, \ldots, O_n) = P(X_1) \cdot P(O_1 \mid X_1) \cdot \prod_{i=2}^{n} P(X_i \mid X_{i-1}) \cdot P(O_i \mid X_i)$$

$$P(X_1, \ldots, X_n, \mid o_1, \ldots, o_n) \propto P(X_1) \cdot P(o_1 \mid X_1) \cdot \prod_{i=2}^{n} P(X_i \mid X_{i-1}) \cdot P(o_i \mid X_i)$$

Learning HMM from fully observable data

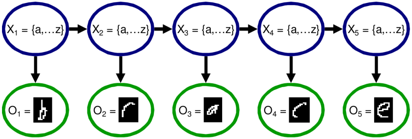


Have m data points

Each data point looks like:

- $X_1 = b$, $X_2 = r$, $X_3 = a$, $X_4 = c$, $X_5 = e$
- $O_1 =$ image of b, $O_2 =$ image of r, $O_3 =$ image of a, $O_4 =$ image of c, $O_5 =$ image of e

Learning HMM from fully observable data



Learn 3 distributions

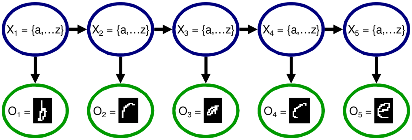
$$P(X_1 = a) = \frac{\mathrm{Count}(X_1 = a)}{m}$$

$$P(O_i = 54 \mid X_i = a) = \frac{\mathrm{Count}(\text{saw letter a and its observation was 54})}{\mathrm{Count}(\text{saw letter a})}$$

$$P(X_i = b \mid X_{i-1} = a) = \frac{\mathrm{Count}(\text{saw a letter b following an a})}{\mathrm{Count}(\text{saw an a followed by something})}$$

How many parameters do we have to learn?

Possible inference tasks in an HMM



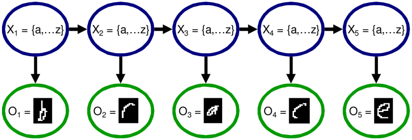
Evaluation

Given HMM parameters and observation sequence $\{o_i\}_{i=1}^{5}$ find the probability of observation sequence

$$P(o_1, \ldots, o_5)$$

Can be computed using *forward algorithm*.

Possible inference tasks in an HMM



Decoding

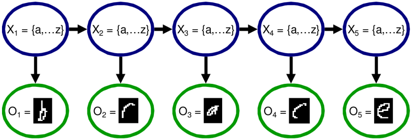
Marginal probability of a hidden variable

$$P(X_i = a \mid o_1, o_2, \ldots, o_n)$$

Can be computed using *forward-backward algorithm*.

- linear in the length of the sequence, because HMM is a tree

Possible inference tasks in an HMM



Viterbi decoding

Most likely trajectory for hidden vars

$$\max_{x_1,\ldots,x_n} P(X_1 = x_1, \ldots, X_n = x_n \mid o_1, \ldots, o_n)$$

- most likely word that generated images
- very similar to forward-backward algorithm

Not the same as decoding

Most likely state vs. Most likely trajectory

Most likely state at position i:

$$\arg\max_a P(X_i = a \mid o_1, o_2, \ldots, o_n)$$

Most likely assignment of state trajectory

$$\max_{x_1,\ldots,x_n} P(X_1 = x_1, \ldots, X_n = x_n \mid o_1, \ldots, o_n)$$

Solution not the same!

x	y	$P(x,y)$
0	0	0.35
0	1	0.05
1	0	0.3
1	1	0.3

Evaluation problem Details!!!

Given parameters of the model, find the find probability of observed sequence.

$$P(\{O_i\}_{i=1}^n) = \sum_k P(\{O_i\}_{i=1}^n, S_n = k) = \sum_k \alpha_n^k$$

Compute α_n^k recursively.
We use chain rule and Markov assumption:

$$\begin{aligned}
\alpha_n^k &= P(\{O_i\}_{i=1}^n, S_n = k) \\
&= P(O_n \mid S_n = k) \cdot \sum_l \alpha_{n-1}^l P(S_n = k \mid S_{n-1} = l)
\end{aligned}$$

Decoding most likely state Details!!!

Given parameters of the model and the observed sequence find the probability that hidden state at time t was k.

$$
\begin{aligned}
P(S_t = k\{O_i\}_{i=1}^{n}) &= P(O_1, \ldots, O_t, S_t = k, O_{t+1}, \ldots, O_n) \\
&= \underbrace{P(O_1, \ldots, O_t, S_t = k)}_{\alpha_t^k} \cdot \underbrace{P(O_{t+1}, \ldots, O_n \mid S_t = k)}_{\beta_t^k}
\end{aligned}
$$

Again, we compute β_t^k recursively.

$$
\begin{aligned}
\beta_t^k &= P(O_{t+1}, \ldots, O_n \mid S_t = k) \\
&= \sum_l P(S_{t+1} = l \mid S_t = k) \cdot P(O_{t+1} \mid S_{t+1} = l)\beta_{t+1}^l
\end{aligned}
$$

Computational complexity Details!!!

What is the running time for Forward, Backward, and Viterbi?

$O(K^2 \cdot T)$, which is linear linear in T, instead of exponential in T.

We have not talked about Viterbi algorithm, but it is similar to forward-backward algorithm.

Learning parameters when hidden states are not observed

Baum-Welch Algorithm

- this is essentially an EM algorithm

Where does this arise?

Summary

Useful for modeling sequential data with few parameters using discrete hidden states that satisfy Markov assumption.

- Speech, OCR, finance

Representation

- initial prob, transition prob, emission prob
- Parameter sharing, only need to learn 3 distributions

Summary

Algorithms for inference and learning in HMMs

- Computing marginal likelihood of the observed sequence: forward algorithm
- Predicting a single hidden state: forward-backward
- Predicting an entire sequence of hidden states: viterbi
- Learning HMM parameters:
  - hidden states observed: simple counting
  - otherwise Baum-Welch algorithm (an instance of an EM algorithm)

Recurrent neural networks

Recurrent neural networks

RNNs are very powerful, because they combine two properties:

- Distributed hidden state that allows them to store a lot of information about the past efficiently.
- Non-linear dynamics that allows them to update their hidden state in complicated ways.

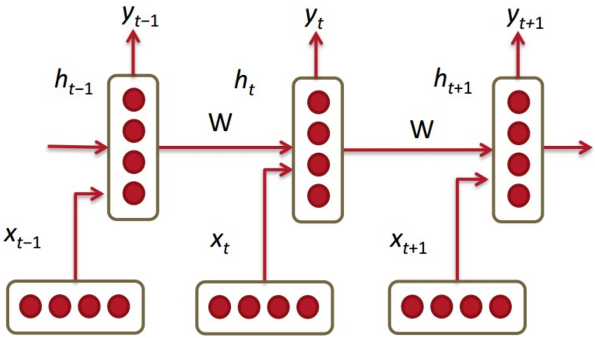
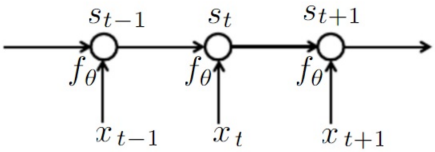


Figure: Richard Socher

Recurrent neural networks as Dynamic systems

$$s_t = f_\theta(s_{t-1}, x_t)$$



The state contains information about the whole past sequence.

$$s_t = g_t(x_t, x_{t-1}, x_{t-s}, \ldots, x_2, x_1)$$

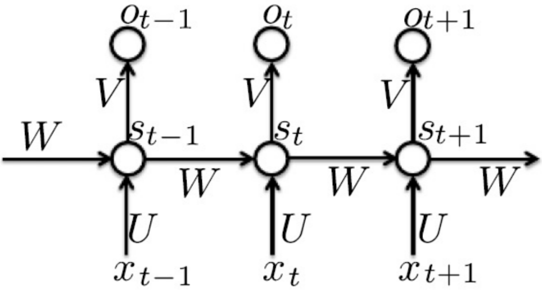
Parameter sharing

We can think of s_t as a summary of the past sequence of inputs up to t.

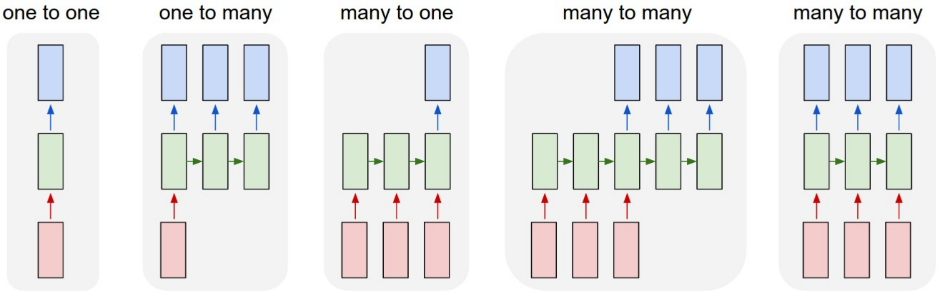
If we define a different function g_t for each possible sequence length, we would not get any generalization.

If the same parameters are used for any sequence length allowing much better generalization properties.

Recurrent Neural Networks



$$
\begin{aligned}
\mathbf{a}_t &= \mathbf{b} + W\mathbf{s}_{t-1} + U\mathbf{x}_t \\
\mathbf{s}_t &= \mathit{tanh}(\mathbf{a}_t) \\
\mathbf{o}_t &= \mathbf{c} + V\mathbf{s}_t \\
\mathbf{p}_t &= \mathit{softmax}(\mathbf{o}_t)
\end{aligned}
$$


one to one
one to many
many to one
many to many
many to many

Examples

Handwritting
https://www.cs.toronto.edu/~graves/handwriting.html

Image captioning
http://mscoco.org/explore/