COMP30027 Machine Learning
Structured Classification

Semester 1, 2019
Tim Baldwin & Karin Verspoor

THE UNIVERSITY OF MELBOURNE

— version: 128, date: May 21, 2019 —

Lecture Outline

Structured Classification

To date, we have always considered each instance independently, but in many tasks, there is "structure" between instances, e.g.:

- sequential structure (e.g. time series analysis, speech recognition, genomic data)
- hierarchical structure (e.g. classifying web pages within a web site)
- graph structure (e.g. deriving an "influence matrix" for a social network)

This calls for **structured classification** models which are able to capture the interaction between instances

Source(s): Blunsom [2007]

Markov Chains

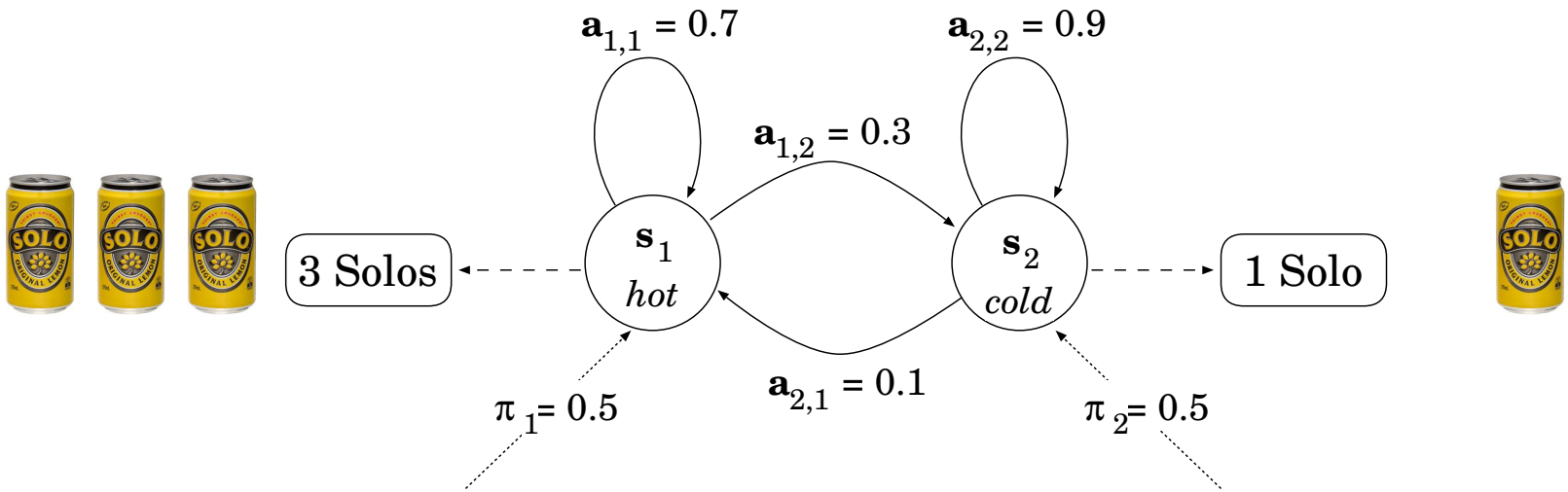
Markov chains are finite state automaton (FSA) of the form $\mu = (A, \Pi)$ over a set $S = \{s_i\}$ of N states and $O = \{o_k\}$ of N outputs, where:

$A = \{a_{ij}\}$ transition probability matrix; $\forall i : \sum_j a_{ij} = 1$
$\Pi = \{\pi_i\}$ the initial state distribution; $\sum_i \pi_i = 1$

Markov chains encode the assumption that:

$$P(q_i|q_1 ... q_{i-1}) = P(p_i|q_{i-1})$$

Example Markov Chain: Wannabe Solo Man



Example Calculation based on Wannabe Solo Man

What is the probability of observing 3-Solos, 3-Solos, 1-Solo?

Example Calculation based on Wannabe Solo Man

What is the probability of observing 3-Solos, 3-Solos, 1-Solo?

$$\begin{aligned} P(3,3,1) &= 0.5 \times 0.7 \times 0.3 \\ &= 0.105 \end{aligned}$$

Lecture Outline

Hidden Markov Models

But what if there are different possibilities attached to each state, rather than a unique state?

$\Rightarrow$ *introduce the notion of "observations" vs. "hidden states"*

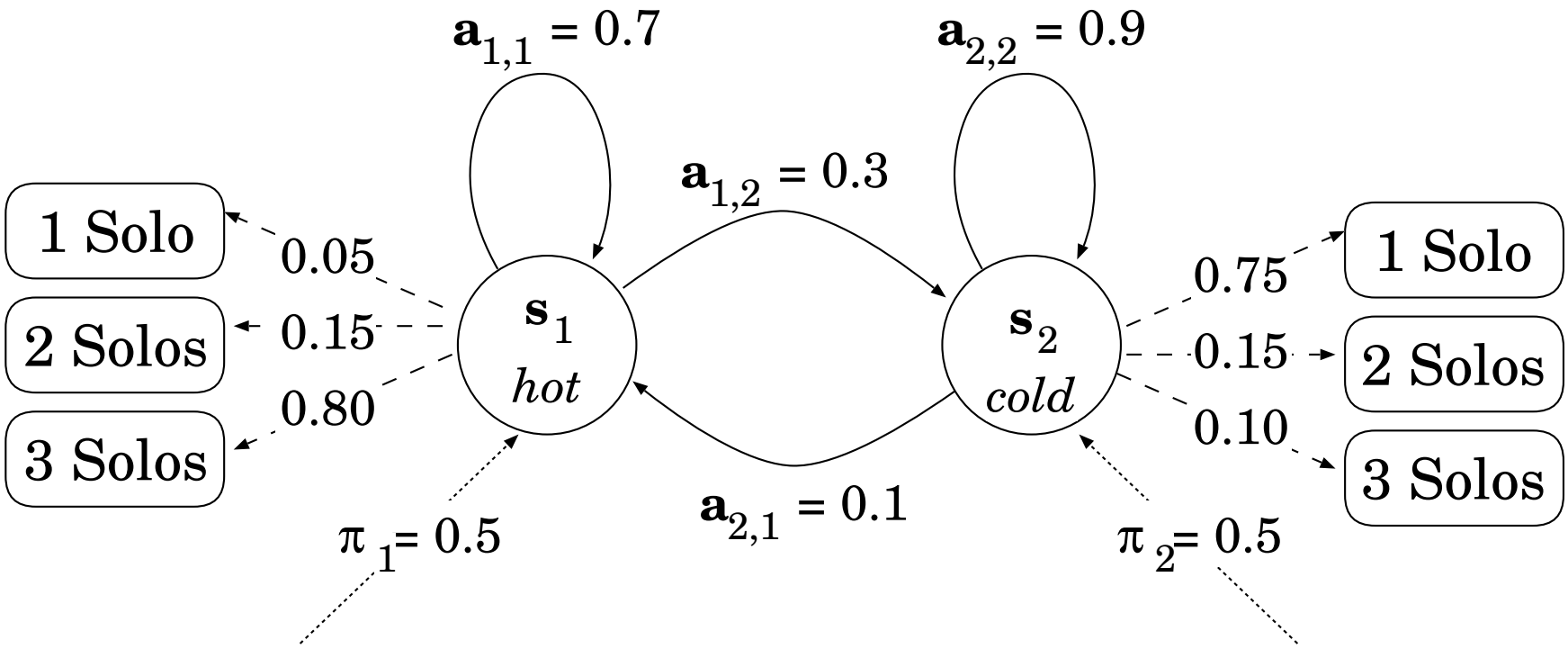
Hidden Markov models (HMMs) take the form $\mu = (A, B, \Pi)$:

$A = \{a_{ij}\}$	transition probability matrix; $\forall i : \sum_j a_{ij} = 1$
$B = \{b_i(o_k)\}$	output probability matrix; $\forall i : \sum_k b_i(o_k) = 1$
$\Pi = \{\pi_i\}$	the initial state distribution; $\sum_i \pi_i = 1$

HMMs make the additional independence assumption:

$$P(o_i|q_1, ..., q_i, o_1, ..., o_{i-1}) = P(o_i|q_i)$$

Example HMM: Wannabe Solo Man with Something to Hide



Fundamental Problems Associated with HMM

Evaluation: Given an HMM μ and observation sequence O, determine the likelihood $P(O|\mu)$

Decoding: Given an HMM μ and observation sequence O, determine the most probable hidden state sequence Q

Learning: Given an observation sequence O and the set of states in an HMM, learn the HMM parameters A, B and Π

Source(s): Rabiner [1989]

Evaluation based on Wannabe Solo Man with Something to Hide

What is the probability of observing 3-Solos, 3-Solos, 1-Solo?

Easy to calculate if we know that the associated days were hot, hot, cold ... ($O(T)$)

Harder to calculate if we don't know the "hidden state" sequence ... ($O(TN^T)$)

Evaluation

Probability of the state sequence Q:

$$P(Q|\mu) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \ldots a_{q_{T-1} q_T}$$

Probability of observation sequence O for state sequence Q:

$$P(O|Q, \mu) = \prod_{t=1}^{T} P(o_t|q_t, \mu)$$

Probability of a given observation sequence O:

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

Source(s): Rabiner [1989]

The Forward Algorithm

Efficient computation of total probability (e.g. $P(O|\mu)$) through "dynamic programming"

Probability of the first t observations is the same for all possible $t+1$ length sequences

Define forward probability:

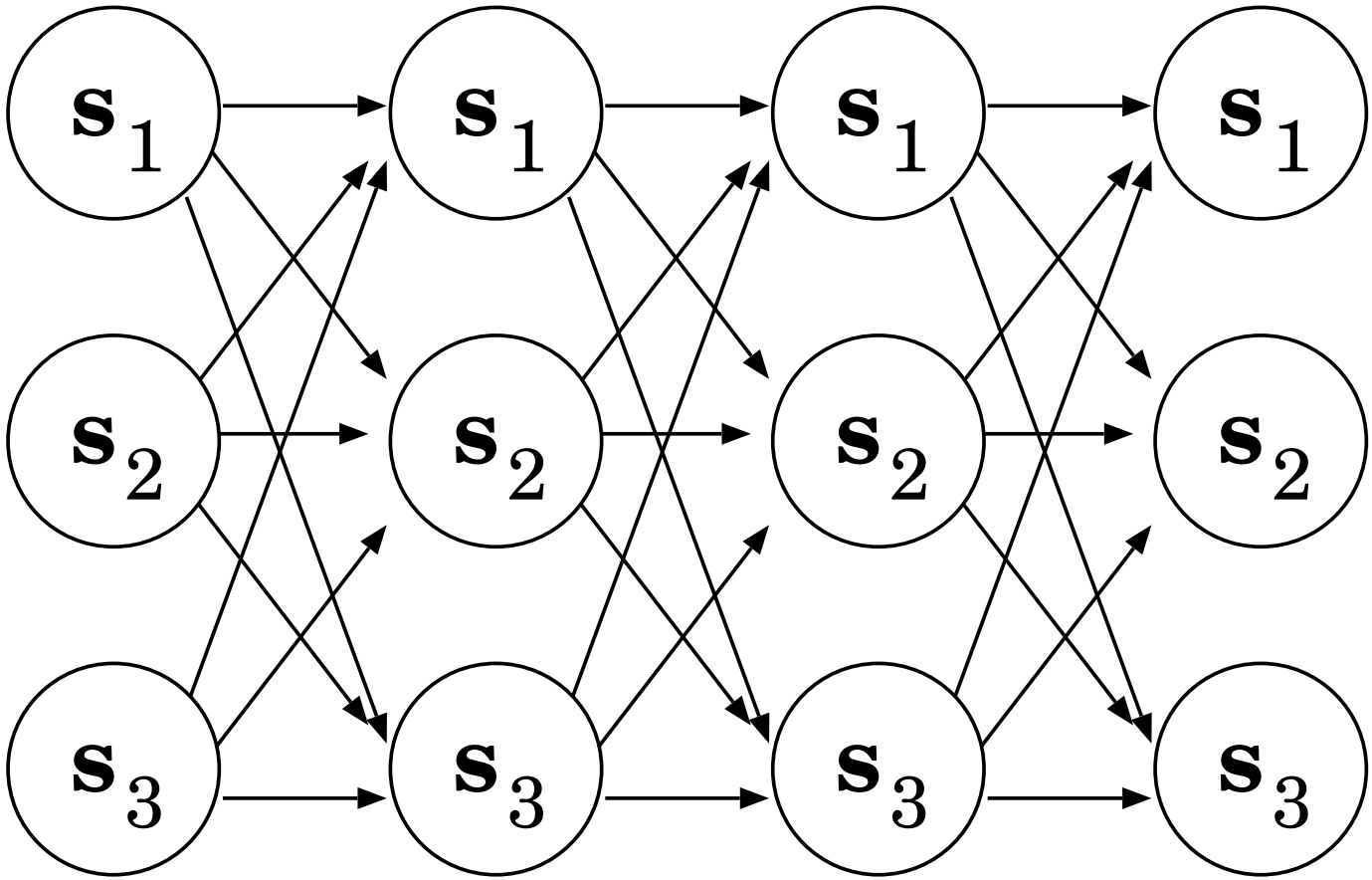
$$\alpha_t(i) = P(o_1 o_2 ... o_t, q_t = s_i|\mu)$$

i.e., the probability of the partial observation sequence, $o_1 o_2 ... o_t$, and state s_i at time t, given the model μ

By caching forward probabilities in a trellis we can avoid redundant calculations

The Backward Algorithm is just the reverse, i.e. start at T and work backwards through the trellis

The Forward Algorithm: Trellis Traversal



The Forward Algorithm

Initialisation:

$$\alpha_1(i) = \pi_i b_i(o_1),\ 1 \leq i \leq N$$

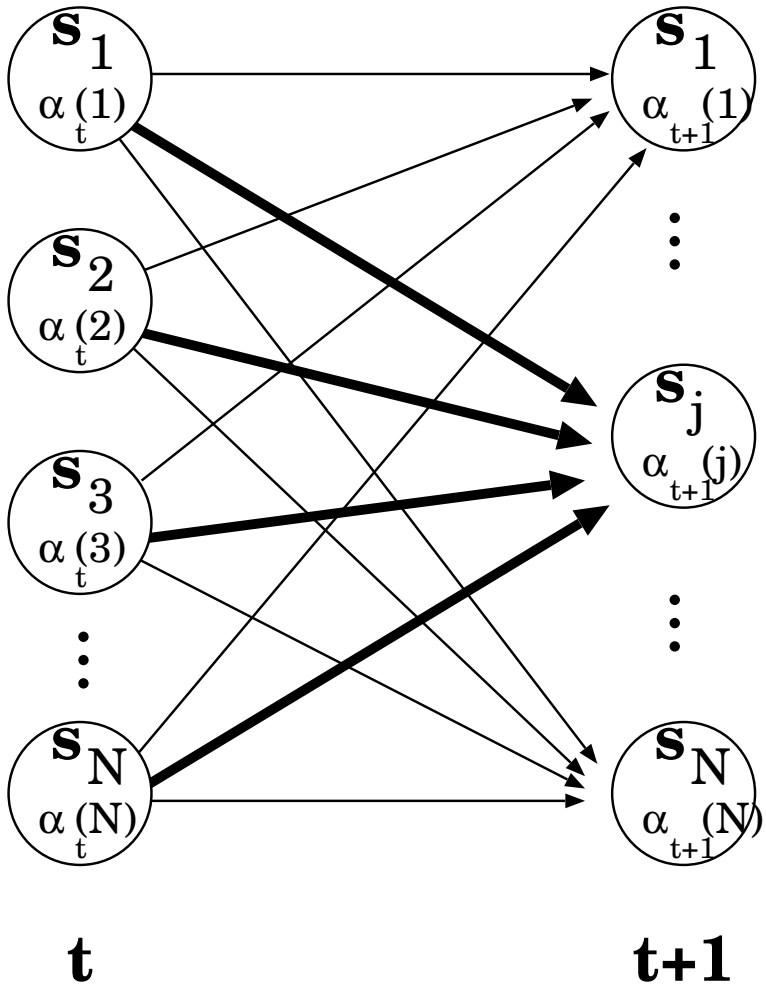
Induction:

$$\alpha_{t+1}(j) = \left(\sum_{i=1}^{N} \alpha_t(i) a_{ij} \right) b_j(o_{t+1}),\ 1 \leq t \leq T-1,\ 1 \leq j \leq N$$

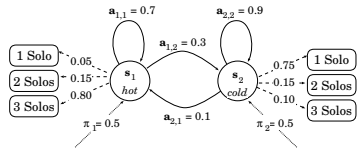
Termination:

$$P(O|\mu) = \sum_{i=1}^{N} \alpha_T(i)$$

The Forward Algorithm: Efficient Trellis Traversal



Returning to our Example ...



Initialisation/induction:

	t = 1	2	3
$\alpha_t(hot)$:	0.5×0.8 $= 0.4$	$0.4 \times 0.7 \times 0.8$ $+0.05 \times 0.1 \times 0.8$ $= 0.228$	$0.228 \times 0.7 \times 0.05$ $+0.0165 \times 0.1 \times 0.05$ $= 0.0080625$
$\alpha_t(cold)$:	0.5×0.1 $= 0.05$	$0.4 \times 0.3 \times 0.1$ $+0.05 \times 0.9 \times 0.1$ $= 0.0165$	$0.228 \times 0.3 \times 0.75$ $+0.0165 \times 0.9 \times 0.75$ $= 0.0624375$

Termination:

$$P(\text{3-Solos, 3-Solos, 1-Solo}|\mu) = 0.0080625 + 0.0624375 = 0.0705$$

Backward Algorithm I

Similar to forward algorithm is used for evaluation: finding $P(O|\mu)$, what is the probability of P(3-Solo, 3-Solo, 1-Solo)?

We define backward probability β, the probability of seeing the observations from time $t+1$ to the end, given that we are in state i at time t given the HMM μ:

$$\beta_t(i) = P(o_{t+1}, o_{t+1} \ldots o_T | q_t = i, \mu)$$

Backward Algorithm II

Initialisation:

$$\beta_T(i) = 1,\ 1 \leq i \leq N$$

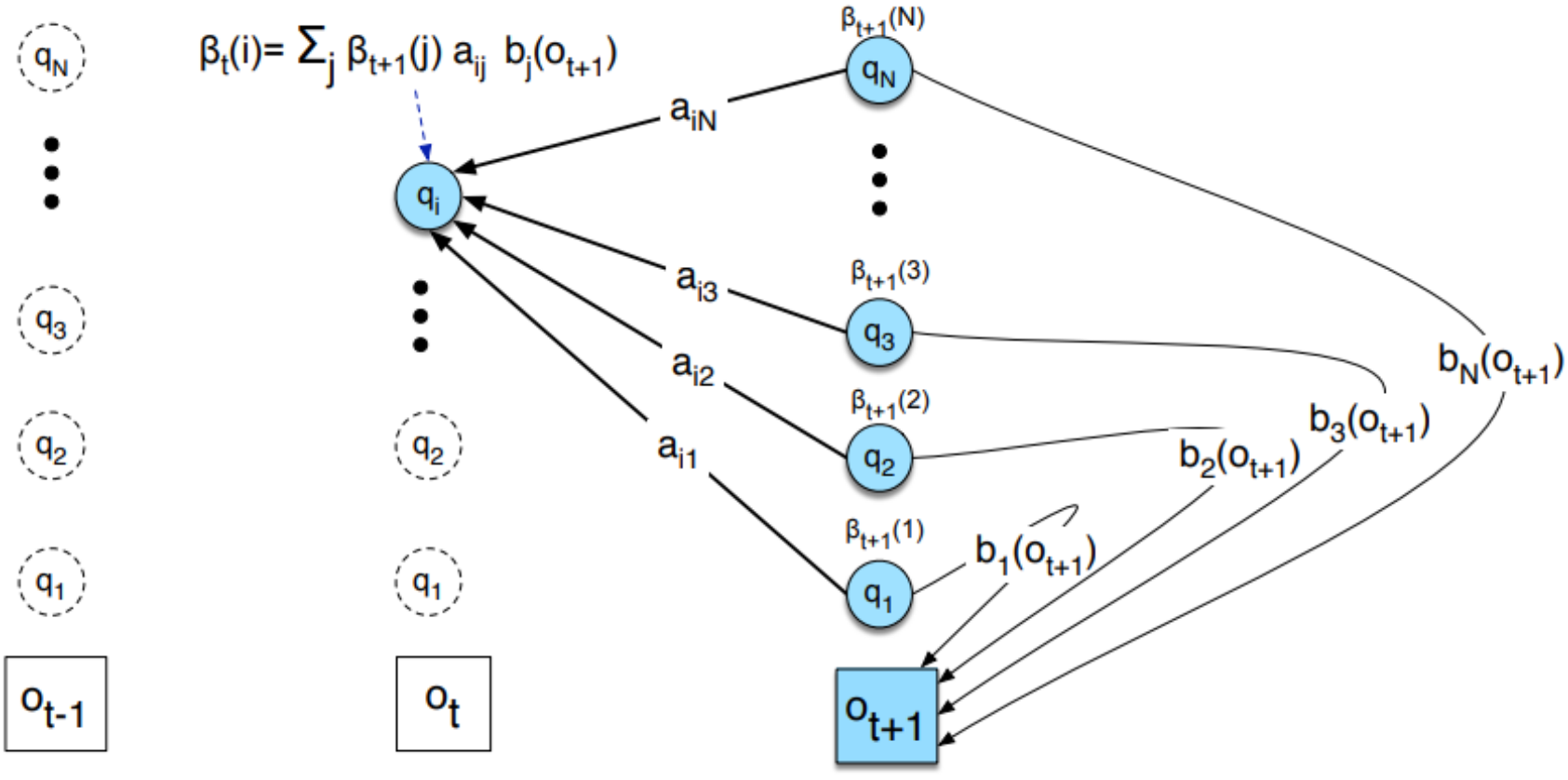
Induction:

$$\beta_t(i) = \left(\sum_{j=1}^{N} \beta_{t+1}(j) a_{ij} \right) b_j(o_{t+1}),\ 1 \leq t \leq T,\ 1 \leq j \leq N$$

Termination:

$$P(O|\mu) = \sum_{j=1}^{N} \pi_j b_j(o_1) \beta_1(j)$$

Backward Algorithm III



Jurafsky (2014)

Decoding based on Wannabe Solo Man with Something to Hide

Given the observation 3-Solos, 3-Solos, 1-Solo, what is the most probable weather sequence?

Decoding based on Wannabe Solo Man with Something to Hide

Given the observation 3-Solos, 3-Solos, 1-Solo, what is the most probable weather sequence?

Could enumerate all the hidden state sequences brute-force and sort ... $(O(TN^T + N^T \log N^T))$

Decoding based on Wannabe Solo Man with Something to Hide

Given the observation 3-Solos, 3-Solos, 1-Solo, what is the most probable weather sequence?

Could enumerate all the hidden state sequences brute-force and sort ... $(O(TN^T + N^T \log N^T))$

The Viterbi algorithm gives us a much more efficient method

Viterbi Algorithm: Preliminaries

Introduce notation for the maximum probability for a partial sequence along a single path:

$$\delta_t(i) = \max_{q_1 q_2 .. q_{t-1}} P(q_1 q_2 ... q_{t-1}, o_1 o_2 ... o_t, q_t = s_i | \mu)$$

Source(s): Rabiner [1989]

The Viterbi Algorithm I

Initialisation:

$$\begin{aligned} \delta_1(i) &= \pi_i b_i(o_1), \ 1 \leq i \leq N \\ \psi_1(i) &= 0 \end{aligned}$$

Induction:

$$\begin{aligned} \delta_t(j) &= \max_{1 \leq i \leq N} (\delta_{t-1}(i) a_{ij}) b_j(o_t), \ 2 \leq t \leq T, \ 1 \leq i \leq N \\ \psi_t(j) &= \arg\max_{1 \leq i \leq N} (\delta_{t-1}(i) a_{ij}), \ 2 \leq t \leq T, \ 1 \leq i \leq N \end{aligned}$$

The Viterbi Algorithm II

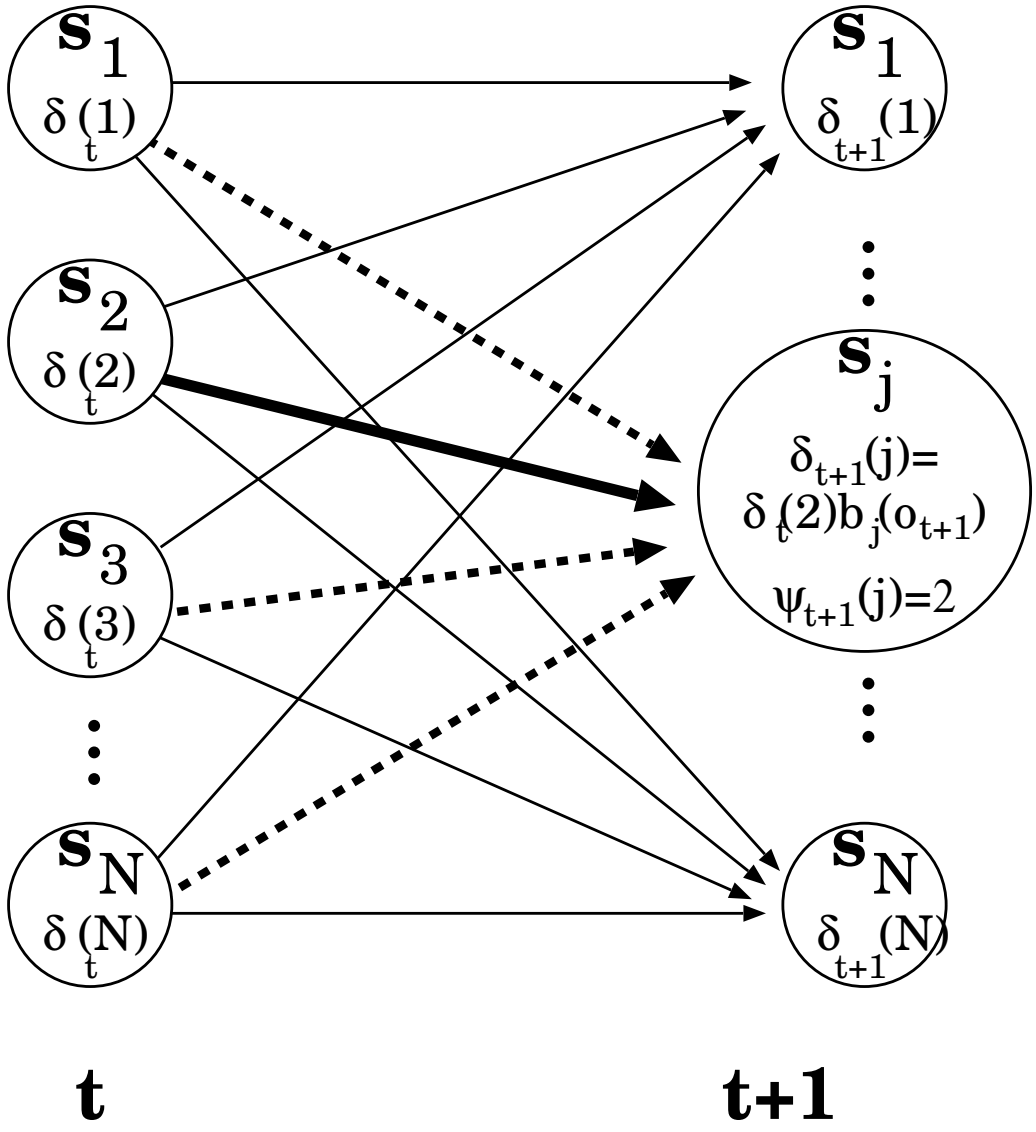
Termination:

$$\begin{aligned} P_{\text{best}} &= \max_{1 \le i \le N} \delta_T(i) \\ q_T &= \arg\max_{1 \le i \le N} \delta_T(i) \end{aligned}$$

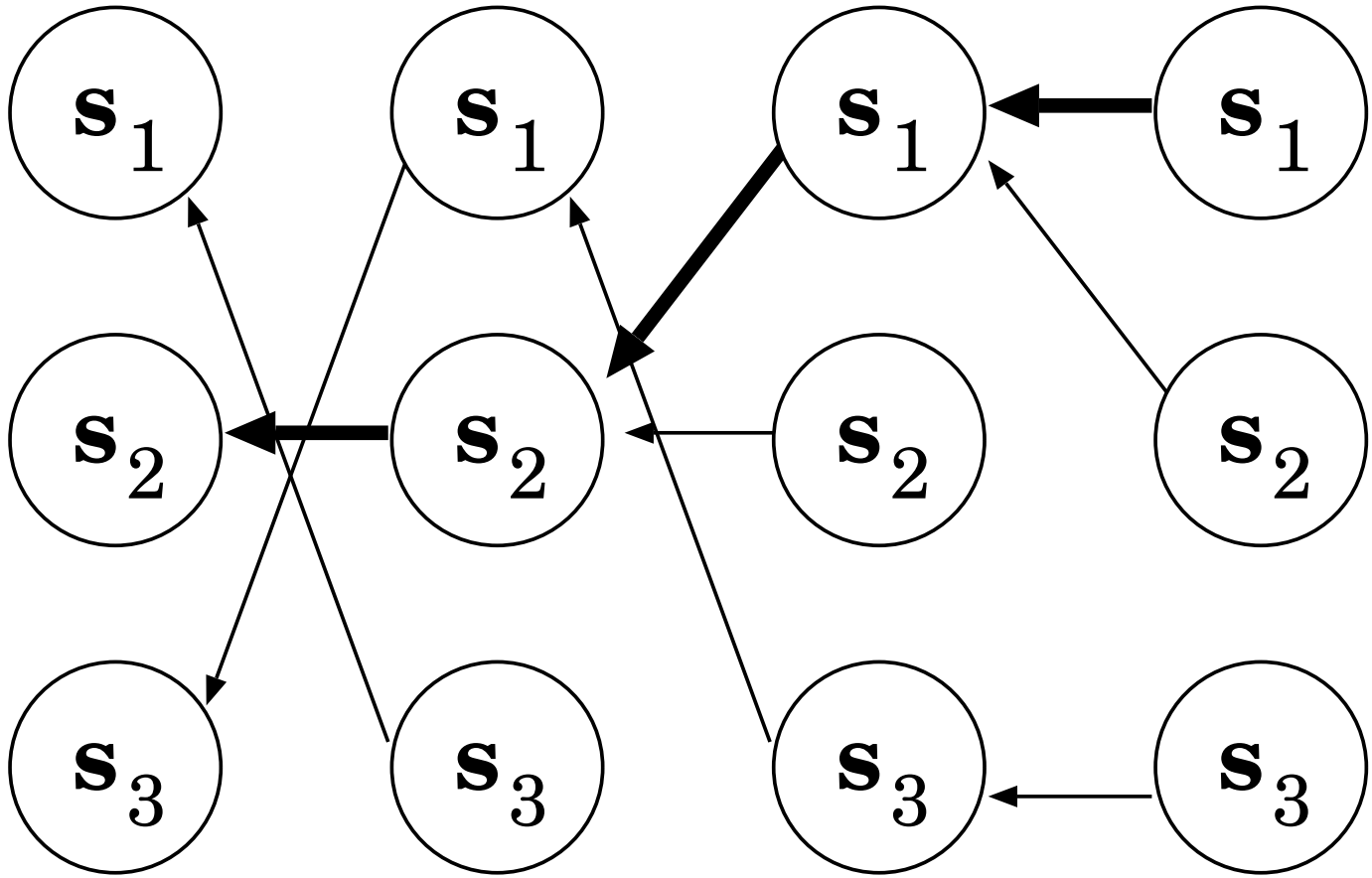
Backtrack to establish the best path:

$$q_t = \psi_{t+1}(q_{t+1}),\ t = T-1, T-2, ..., 1$$

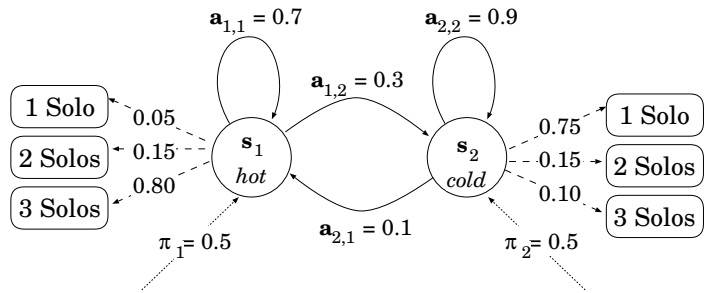
The Viterbi Algorithm: Induction



The Viterbi Algorithm: Backtrace



Returning again to our Example ... I



Initialisation/induction:

	t = 1	2	3
$\delta_t(hot)$:	0.5×0.8 $= 0.4$	$\max(0.4 \times 0.7 \times 0.8,$ $0.05 \times 0.1 \times 0.8)$ $= 0.224$	$\max(0.224 \times 0.7 \times 0.05,$ $0.012 \times 0.1 \times 0.05$ $= 0.00784$
$\psi_t(hot)$	0	$\leftarrow hot$	$\leftarrow hot$
$\delta_t(cold)$:	0.5×0.1 $= 0.05$	$\max(0.4 \times 0.3 \times 0.1,$ $0.05 \times 0.9 \times 0.1)$ $= 0.012$	$\max(0.224 \times 0.3 \times 0.75,$ $0.012 \times 0.9 \times 0.75)$ $= 0.0504$
$\psi_t(cold)$	0	$\nwarrow hot$	$\nwarrow hot$

Returning again to our Example ... II

Termination/backtracking:

$$\begin{aligned} P_{\text{best}} &= 0.0504 \\ q_T &= \textit{cold} \\ q_{T-1} &= \textit{hot} \\ q_{T-2} &= \textit{hot} \end{aligned}$$

$\rightarrow$ *the most probable sequence of hidden states which produces the observation sequence 3-Solos, 3-Solos, 1-Solo is hot, hot, cold*

Learning HMMs: The Supervised Case

Assume we have labelled data, it is possible to use simple MLE to learn the parameters of our model:

$$P(q_j|q_i) = \frac{freq(q_i, q_j)}{freq(q_i)} = a_{ij}$$

$$P(o_k|q_i) = \frac{freq(o_k, q_i)}{freq(q_i)} = b_i(o_k)$$

$$P(q_i|\mathrm{START}) = \frac{freq(\mathrm{START}, q_i)}{\sum_j freq(\mathrm{START}, q_j)} = \pi_i$$

Can also train models in an unsupervised fashion using Baum-Welch algorithm

Learning HMMs: The Unsupervised Case (EM)

- Goal: find transition, emission, and initial probabilities.
- Initialise the probabilities randomly (e.g. uniform)
- Use the initial HMM to tag unlabelled data (E-step)
- Use the tagged data as in supervised case to relearn HMM's probabilities.
- Iterate until convergence.

This EM algorithm is called Baum-Welch algorithm.

HMMs: Reflections

Highly efficient approach to structured classification, but limited representation of context (bigrams only)

As with NB, HMM tends to suffer from floating point underflow

- use logs for Viterbi Algorithm
- use scaling coefficients for Forward Algorithm

As with most generative models, it's hard to add ad hoc features

Lecture Outline

Other Structured Classifiers

Maximum Entropy Markov Models: logistic regression (= "maximum entropy") model where we also condition on the tag for the preceding instance:

$$\hat{c} = \arg\max_{T} \prod_{i} P(q_i | o_i, q_{i-1})$$

Unlike HMMs, it's possible to add extra features indiscriminately *as well as* capturing the (unidirectional) tag interactions

Conditional Random Fields: extension of logistic regression where we optimise over the full tag sequence

Source(s): Blunsom [2007], Lafferty et al. [2001]

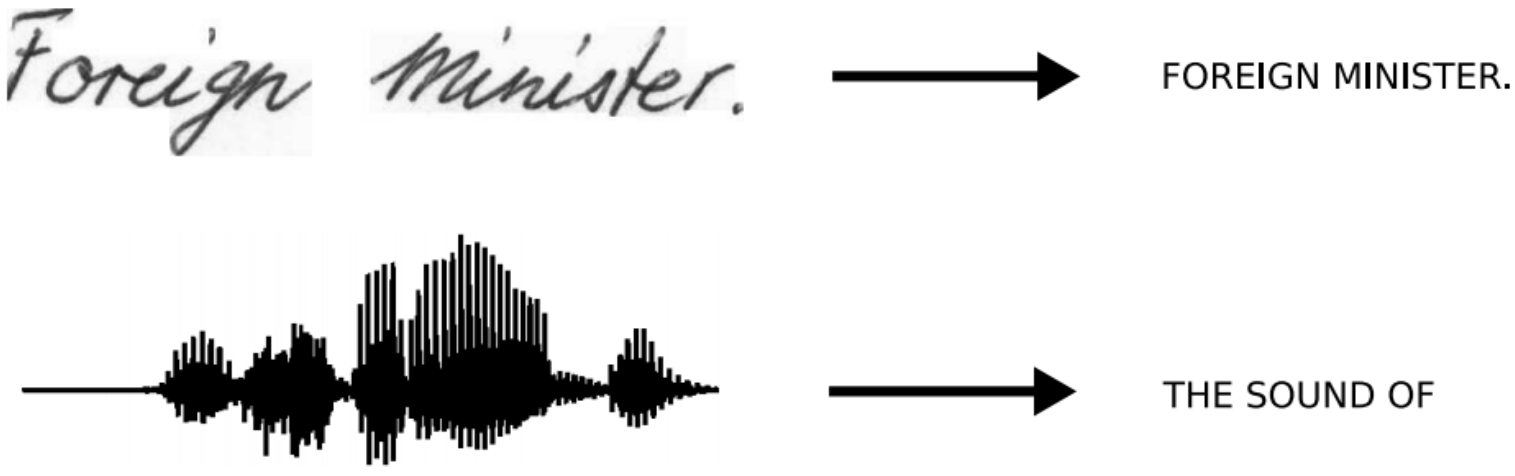
Applications of Sequence Labelling I

Part of Speech Tagging: He/**pronoun** went/**verb** home/**noun**.

Named Entity Recognition: Apple/**company_I** wants/**O** to/**O** release/**O** IPhone12/**product_B** in/**O** December/**date_B** 12,/**data_I** 2012/**date_I** before/**O** the/**O** end/**O** of/**O** year/**O**. BIO format: B: beginning of a named entity, I: inside a named entity, O: outside entity.

labels/hidden states: O, company_I, company_B, date_B, date_I, person_B, person_I, product_B, product_I, location_B, location_I, ...

Applications of Sequence Labelling II



Graves (2008)

Automatic Speech Recognition (ASR) and Optical Character Recognition (OCR)

Lecture Outline

Summary

What is structured classification?

How do we evaluate a HMM?

How do we decode a HMM?

How do you train an HMM given labelled training data?

What are limitations of HMMs, and what more sophisticated structured classification algorithms are there?

Lecture Outline

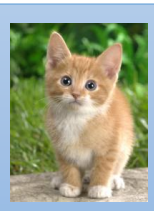
Transfer Learning

Transfer Learning

http://weebly110810.weebly.com/396403913129399.html

http://www.sucaitianxia.com/png/cartoon/200811/4261.html

Dog/Cat Classifier



cat

dog

Data ***not directly related to*** the task considered

elephant

tiger

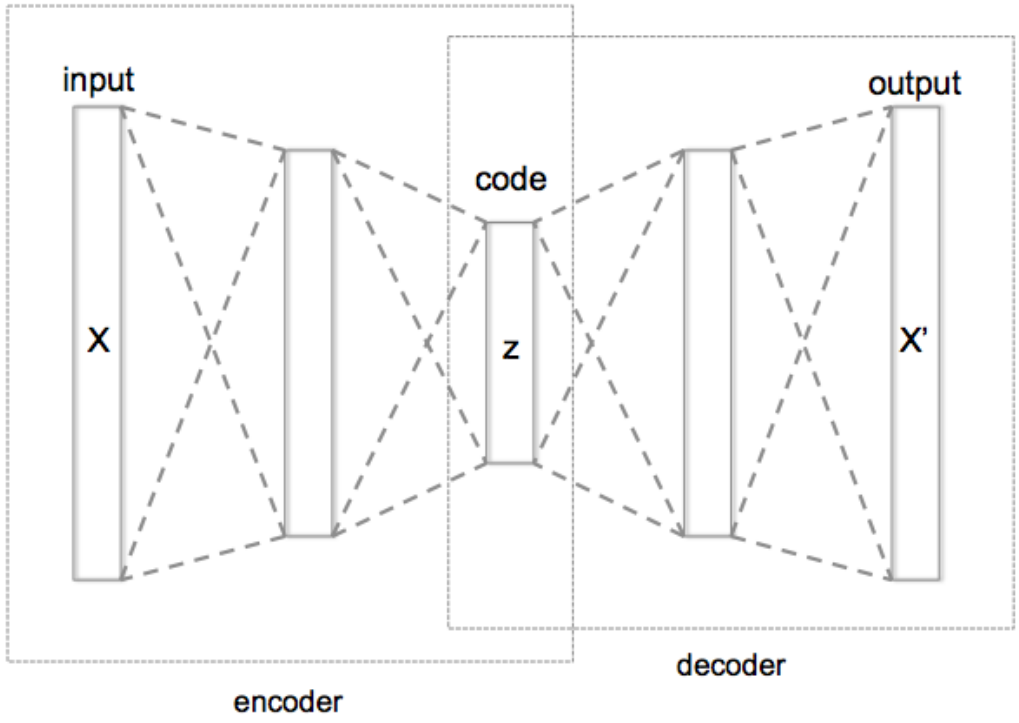
Similar domain, different tasks

dog

cat

Different domains, same task

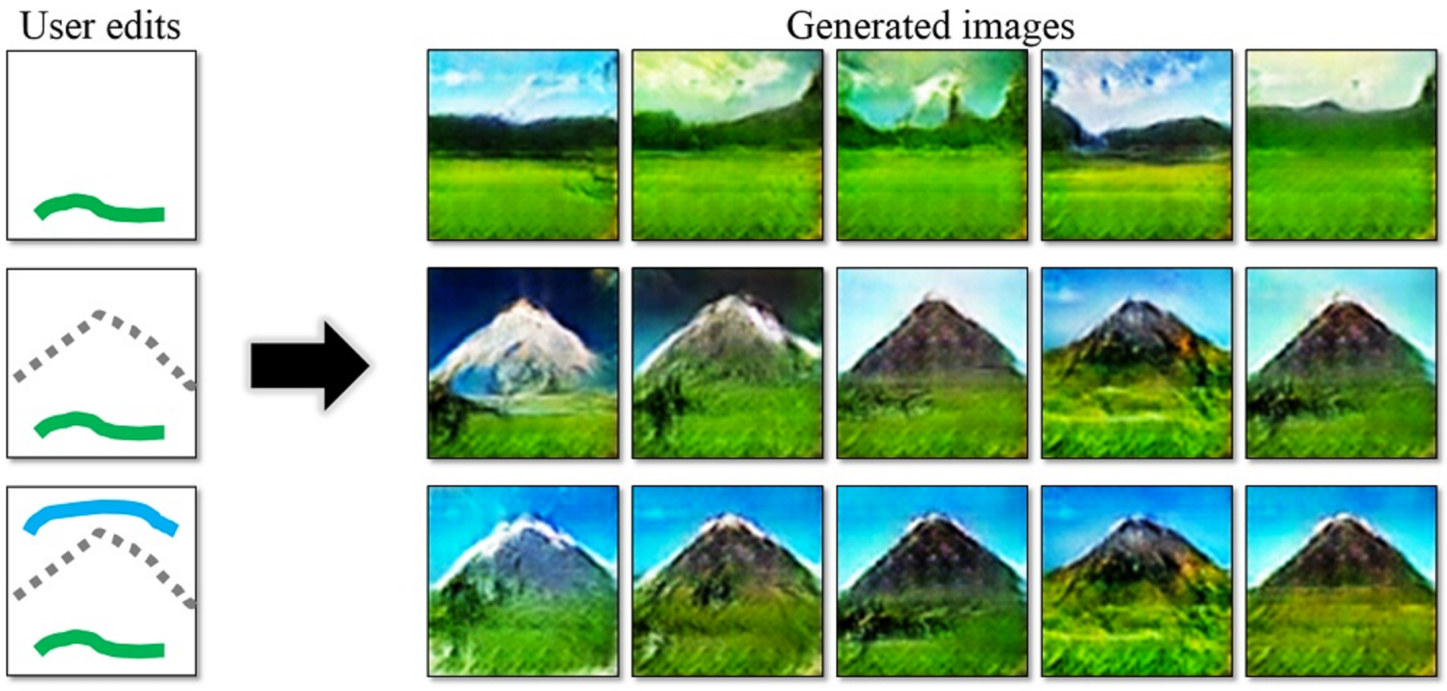
Autoencoders



Wikipedia

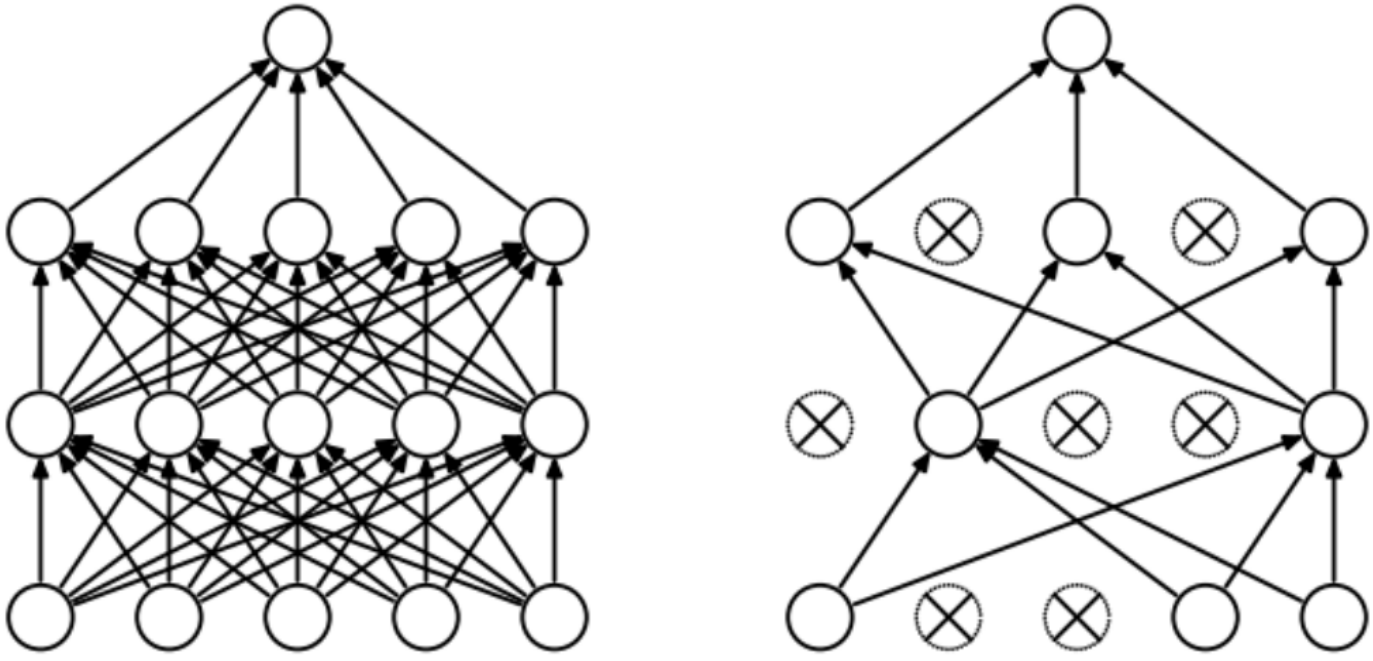
Generative Adversarial Networks (GANs)

Magic of GANs...



http://people.eecs.berkeley.edu/~junyanz/projects/gvm/

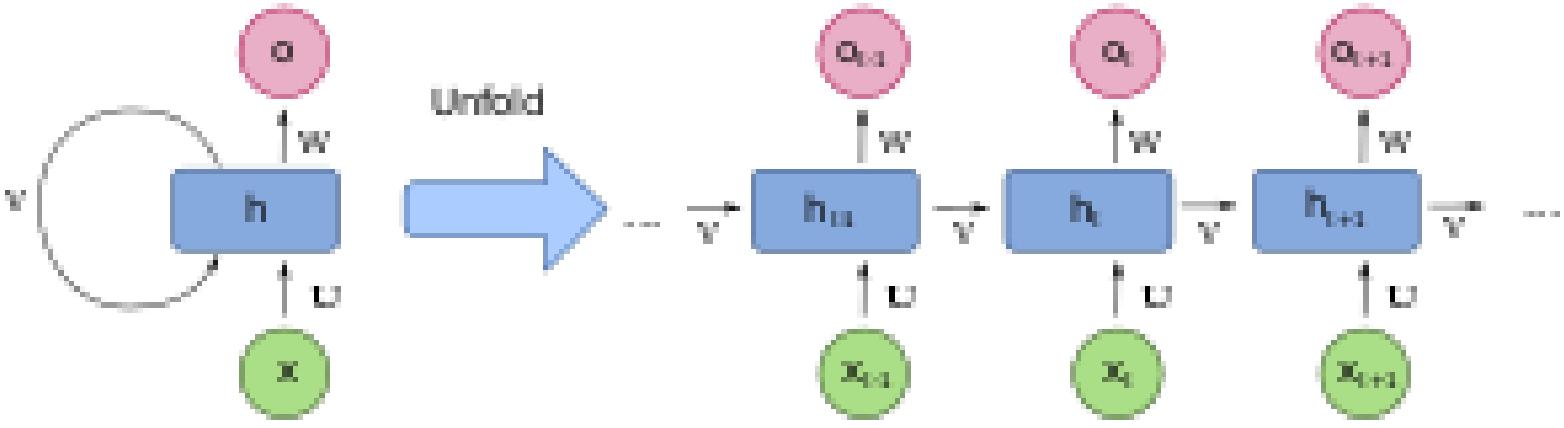
Dropout



(a) Standard Neural Net (b) After applying dropout.

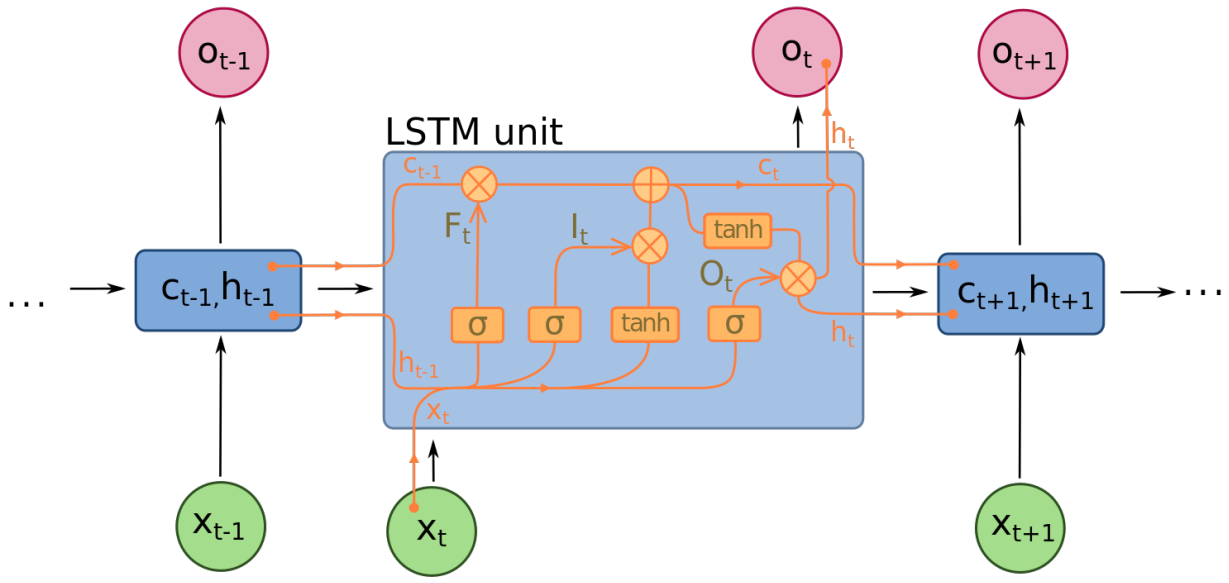
Srivastava (2014)

Recurrent Neural Networks (RNN)



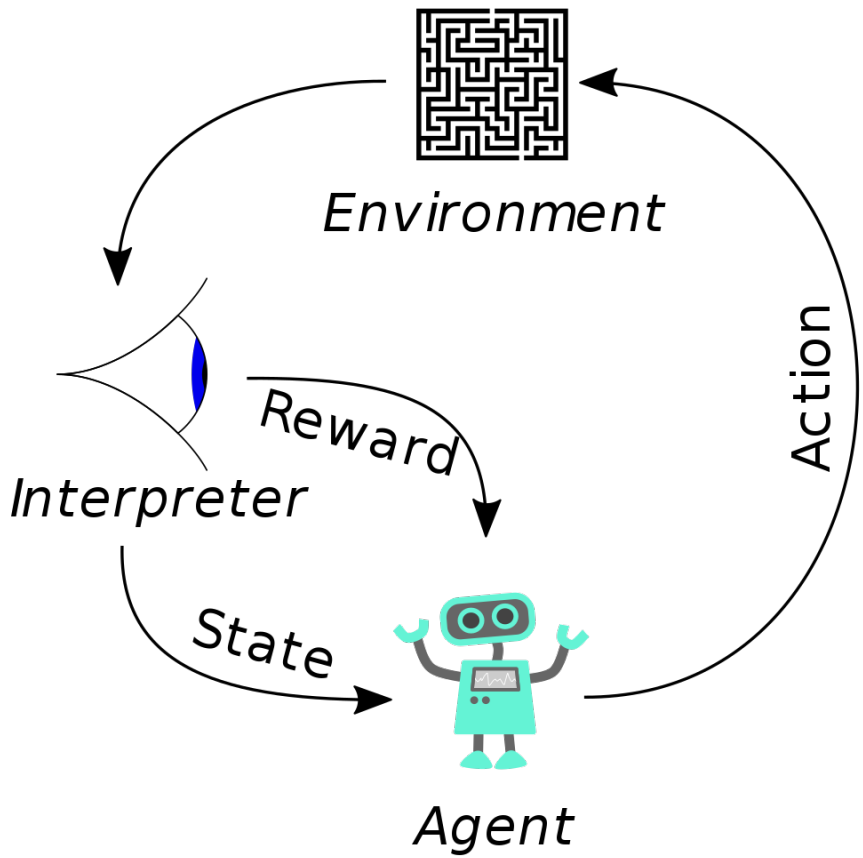
Wikipedia

Long Short-Term Memory Networks (LSTM)



Wikipedia

Reinforcement Learning (RL)



Wikipedia

References I

Philip Blunsom. *Structured Classification for Multilingual Natural Language Processing*. PhD thesis, University of Melbourne, 2007.

Dan Jurafsky and James H Martin. *Speech and language processing*, volume 3. Pearson London, 2014.

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289, Williamstown, USA, 2001.

Lawrence R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2), 1989.