Advanced Machine Learning
Lecture 9: Recurrent neural networks

Sandjai Bhulai
Vrije Universiteit Amsterdam

s.bhulai@vu.nl
06 October 2023

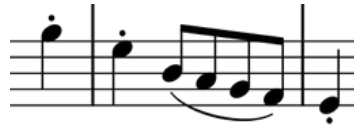
Recurrent neural networks

- Speech recognition

→ [illegible]

- Music generation

∅ →



- Sentiment classification

[illegible] →

- DNA sequence analysis

[illegible] → [illegible]

- Machine translation

[illegible] → [illegible]

- Video activity recognition

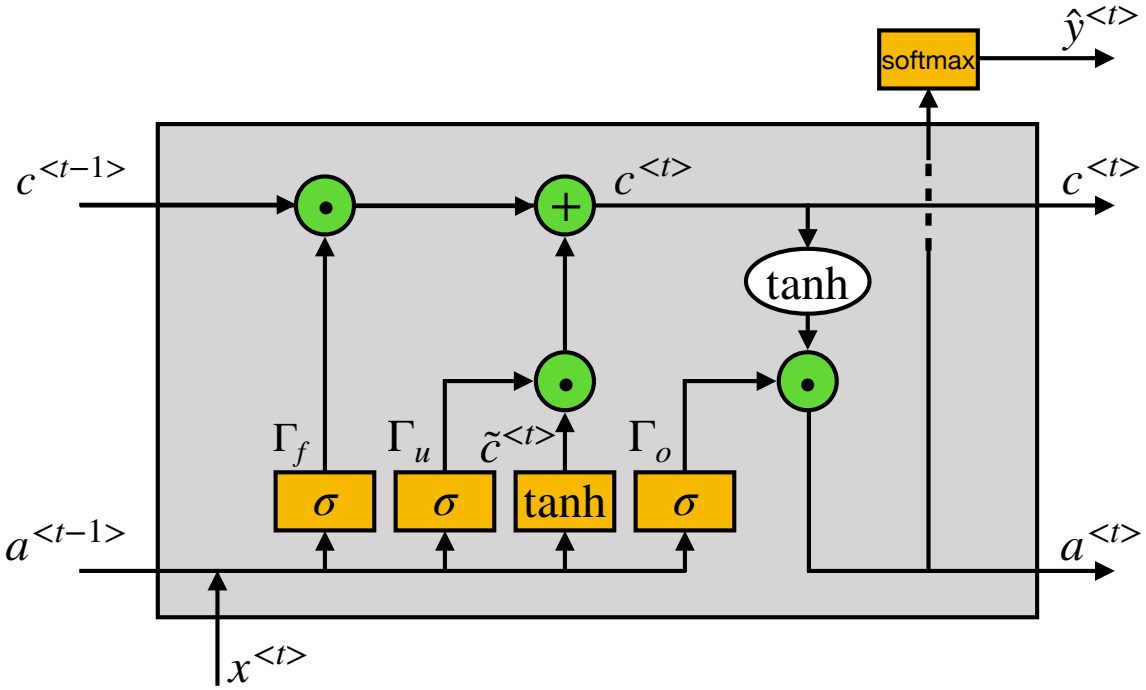
→ [illegible]

- Name entity recognition

[illegible] → [illegible]

VU

Long Short Term Memory (LSTM) unit



$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u)$$

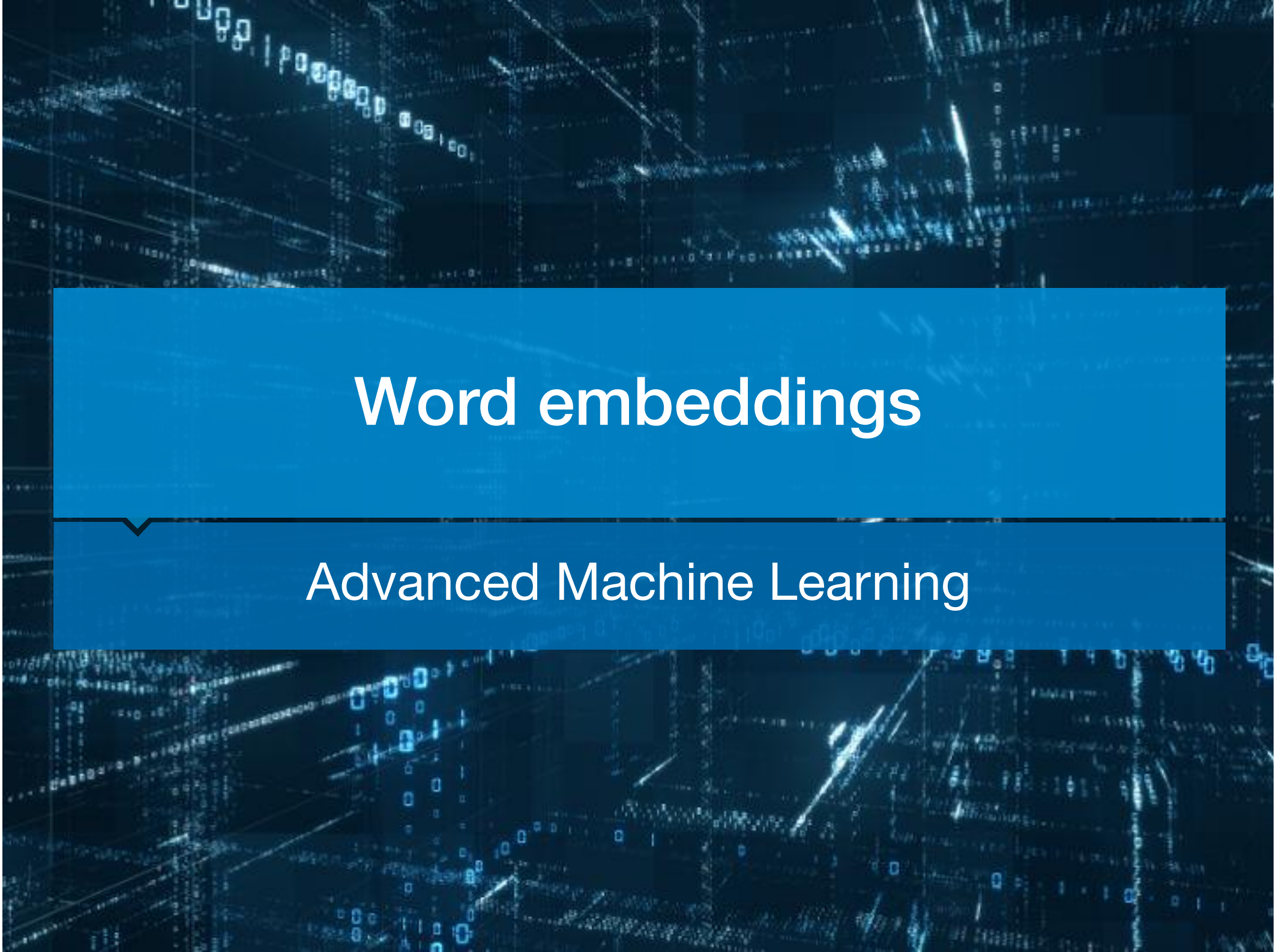
$$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f)$$

$$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o)$$

$$c^{<t>} = \Gamma_u \cdot \tilde{c}^{<t>} + \Gamma_f \cdot c^{<t-1>}$$

$$a^{<t>} = \Gamma_o \cdot \tanh(c^{<t>})$$

$$\hat{y}^{<t>} = g(W_y a^{<t>} + b_y)$$



Word embeddings

Advanced Machine Learning

Word representation

- Vocabulary: V = [a, aaron, …, zulu, <UNK>]

man	woman	king	queen	apple	orange
(5391)	(9853)	(4914)	(7157)	(456)	(6257)

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix} \quad \begin{pmatrix} 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix}$$

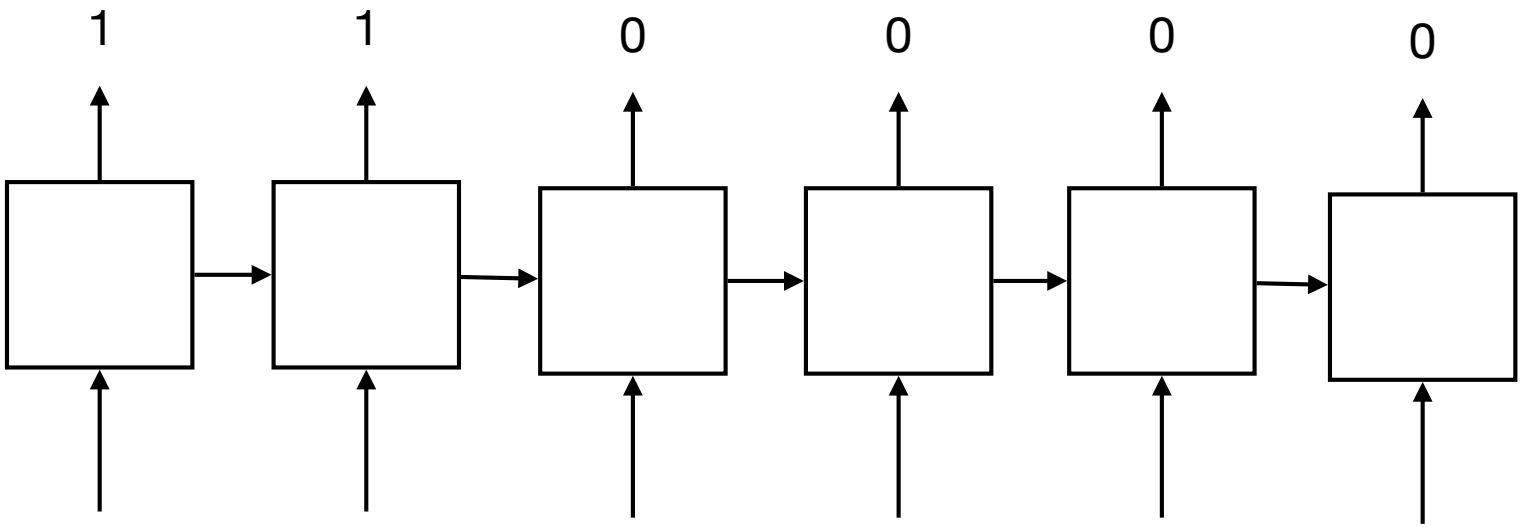
I want a glass of orange _____.

I want a glass of apple _____.

Word embedding

	man (5391)	woman (9853)	king (4914)	queen (7157)	apple (456)	orange (6257)
Gender	-1	1	-0.95	0.97	0.00	0.01
Royal	0.01	0.02	0.93	0.95	-0.01	0.00
Age	0.03	0.02	0.70	0.69	0.03	-0.02
Food	0.09	0.01	0.02	0.01	0.95	0.97

Word embedding



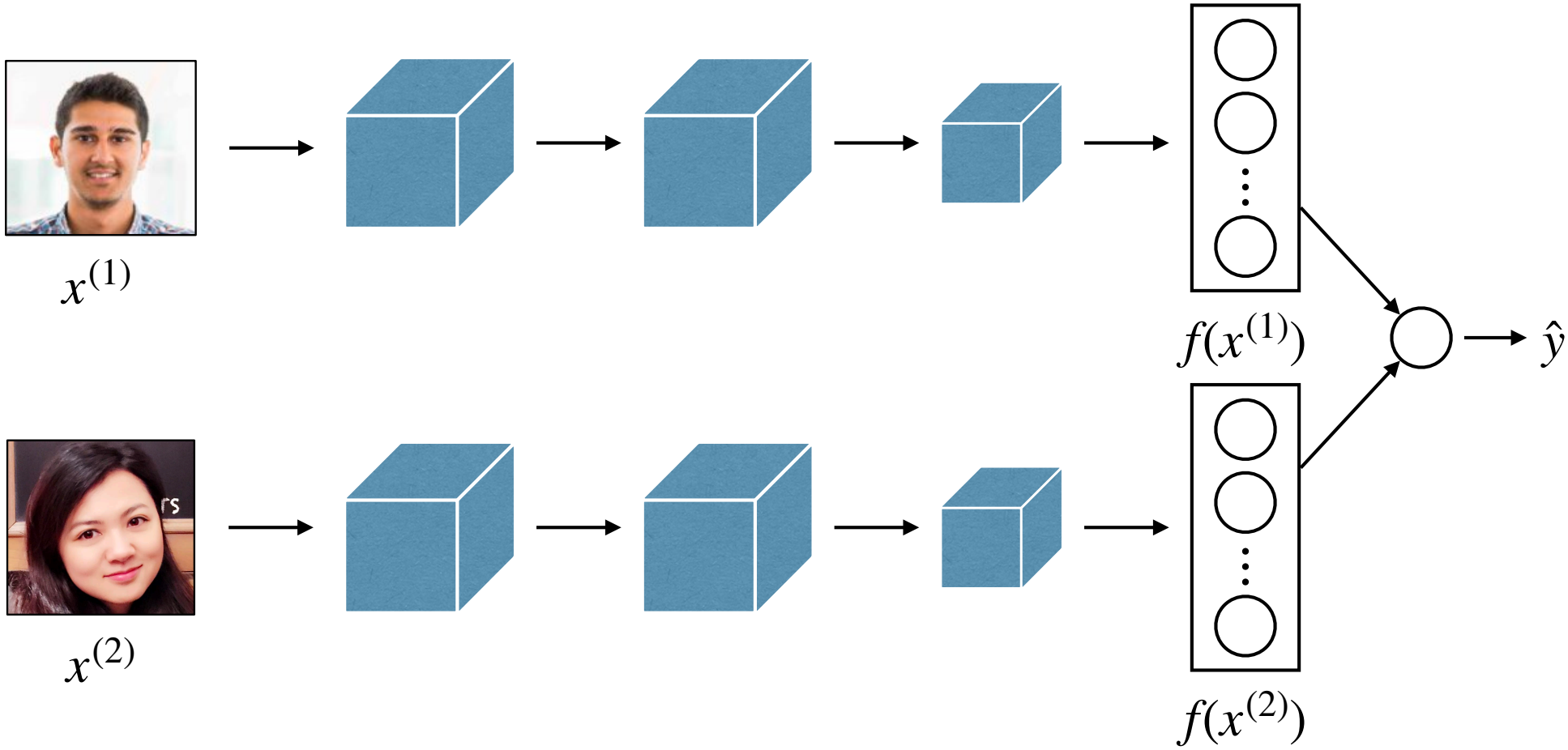
- Sally Johnson is an orange farmer
- Robert Lin is an apple farmer
- Jack Jones is a durian cultivator

Word embedding

- Learn word embeddings from large text corpus (1-100B words) or download pre-trained embedding online
- Transfer embedding to new task with smaller training set (say, 100K words)
- Optional: continue to finetune the word embeddings with new data

Word embedding

- Relation to face encoding



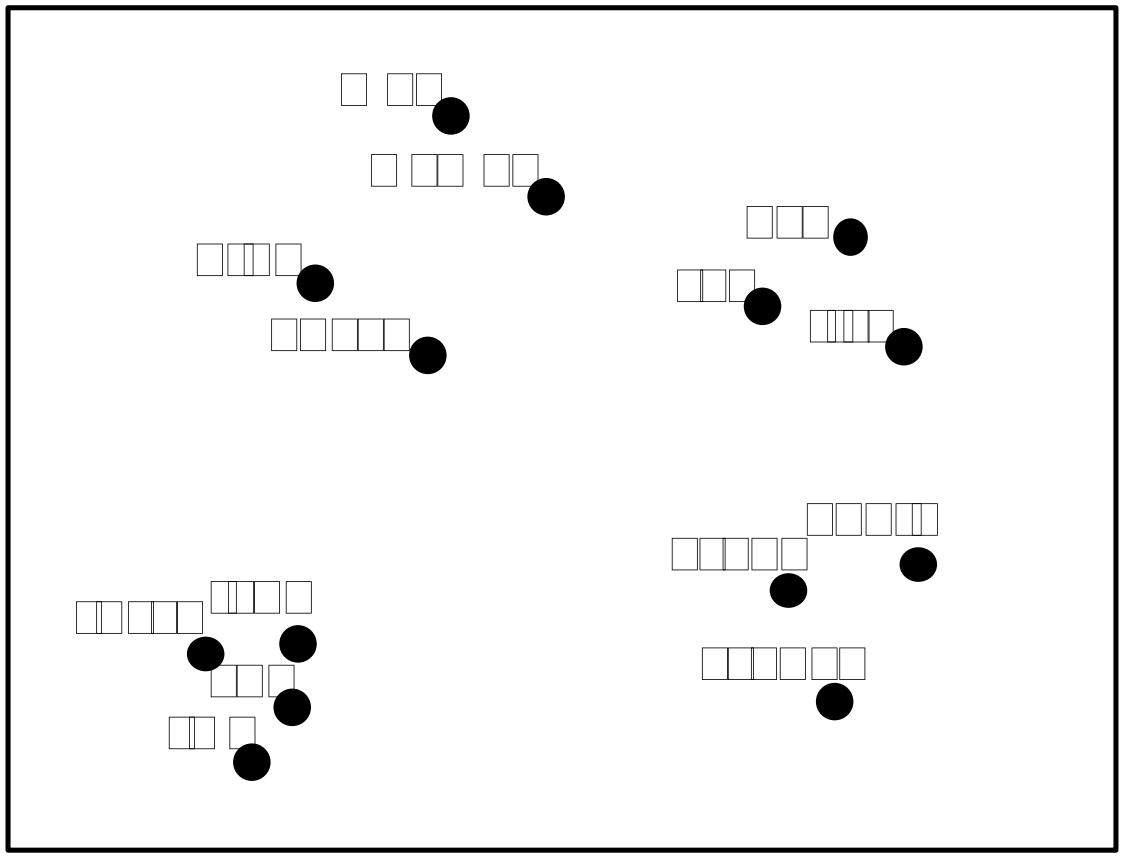
Word embedding

- Analogy to word embedding

	man (5391)	woman (9853)	king (4914)	queen (7157)	apple (456)	orange (6257)
Gender	-1	1	-0.95	0.97	0.00	0.01
Royal	0.01	0.02	0.93	0.95	-0.01	0.00
Age	0.03	0.02	0.70	0.69	0.03	-0.02
Food	0.09	0.01	0.02	0.01	0.95	0.97

- man → woman as king → ??

Word embedding



$$e_{\text{man}} - e_{\text{woman}} \approx e_{\text{king}} - e_{?}$$

Word embedding

- Similarity function: $\text{sim}(u, v) = \dfrac{u^\top v}{\|u\|_2 \, \|v\|_2}$

- Examples:
  - Man:Woman as Boy:Girl
  - Ottawa:Canada as Nairobi:Kenya
  - Big:Bigger as Tall:Taller
  - Yen:Japan as Ruble:Russia

Learning word embeddings

- Vocabulary: V = [a, aaron, …, zulu, <UNK>]

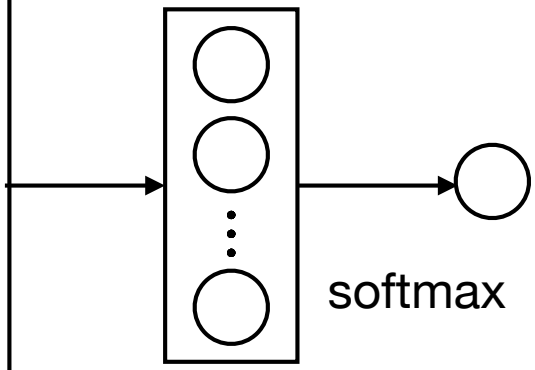
- Say that vocabulary has 10,000 words
- We want an embedding of 300 words
- How to learn an embedding matrix E ?
  size of matrix = $300 \times 10{,}000$

Neural language model

- I want a glass of orange _____.

4343 9665 1 3852 6163 6257

I	o_{4343}	$\rightarrow$	E	$\rightarrow$	e_{4343}
want	o_{9665}	$\rightarrow$	E	$\rightarrow$	e_{9665}
a	o_{1}	$\rightarrow$	E	$\rightarrow$	e_{1}
glass	o_{3852}	$\rightarrow$	E	$\rightarrow$	e_{3852}
of	o_{6163}	$\rightarrow$	E	$\rightarrow$	e_{6163}
orange	o_{6257}	$\rightarrow$	E	$\rightarrow$	e_{6257}



Neural language model

- I want a glass of orange juice to go along with my cereal.

- Other contexts:
  - 4 words on left and right
    “a glass of orange” … “to go along with”
  - Last 1 word
    “orange” …
  - Nearby 1 word (skip gram)
    “glass” …

Neural language model

- I want a glass of orange juice to go along with my cereal.

context	word
orange	juice
orange	glass
orange	my

- Vocabulary: 10,000 words
- Model: $O_c \rightarrow E \rightarrow e_c \rightarrow$ softmax $\rightarrow \hat{y}$
- Parameter associated with output t is θ_t

$$p(t \mid c) = \frac{e^{\theta_t^\top e_c}}{\sum_{j=1}^{10,000} e^{\theta_j^\top e_c}}$$

Word2vec - skip gram model

$$p(t \mid c) = \frac{e^{\theta_t^\top e_c}}{\sum_{j=1}^{10,000} e^{\theta_j^\top e_c}}$$

- Computationally expensive; you need complete vocabulary
  - > Hierarchical softmax ($\log V$)

- How to sample the context c?
  - > "the", "of", "a", "and", "to", …
  - > "orange", "apple", "durian", …

Word2vec - negative sampling

- I want a glass of orange juice to go along with my cereal.

context	word	target?
orange	juice	1
orange	king	0
orange	book	0
orange	the	0
orange	of	0

- Vocabulary: 10,000 words
- Model: $O_c \rightarrow E \rightarrow e_c \rightarrow$ 10,000 log.res. problems
- $\mathbb{P}(y = 1 \mid c, t) = \sigma(\theta_t^\top e_c)$

Word2vec - negative sampling

- I want a glass of orange juice to go along with my cereal.

context	word	target?
orange	juice	1
orange	king	0
orange	book	0
orange	the	0
orange	of	0

- How to select negative samples?
  - > Uniform
  - > Inverse frequency

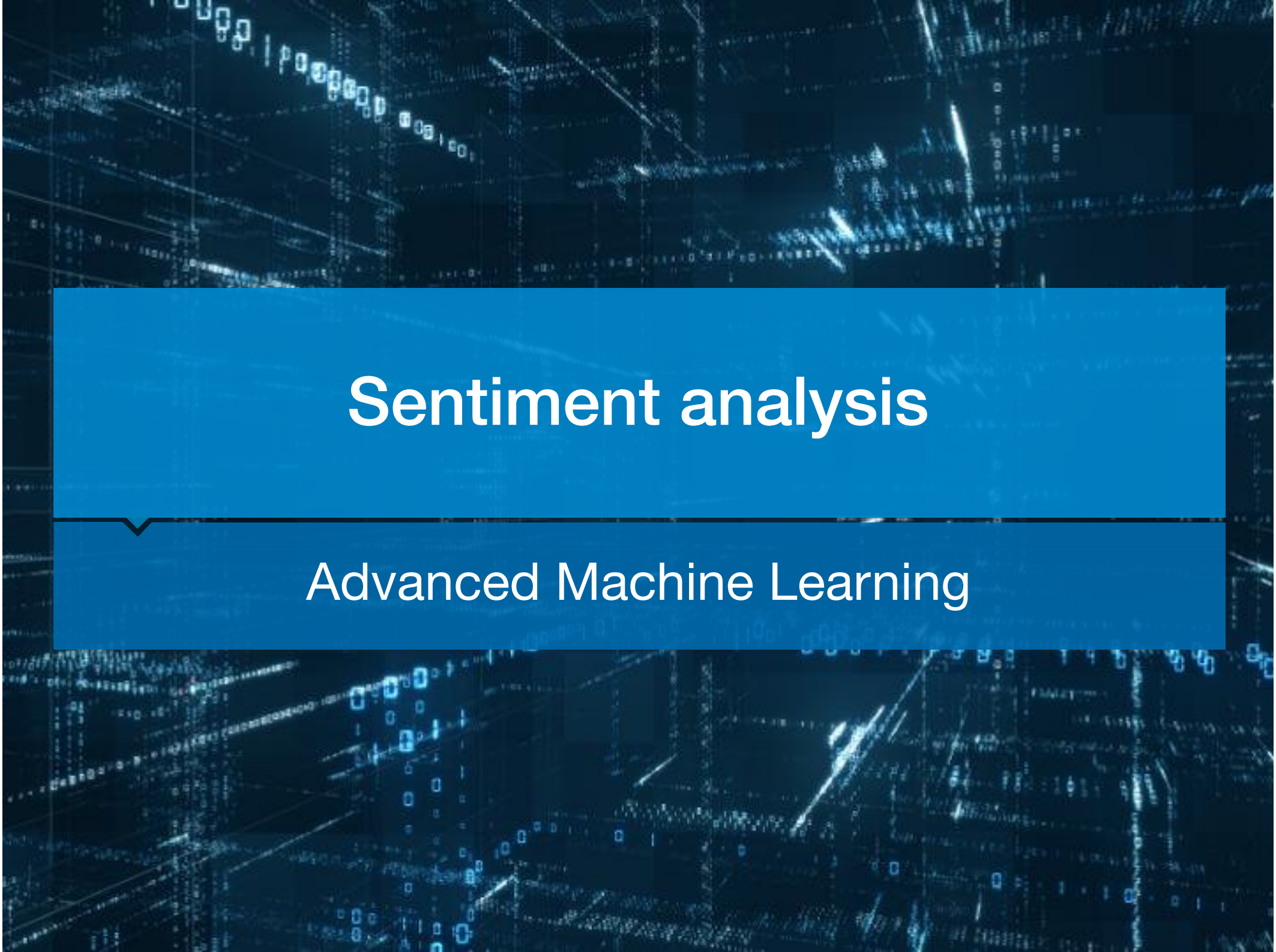
$$p(w_i) = \frac{f(w_i)^{\frac{3}{4}}}{\sum_{j=1}^{10,000} f(w_j)^{\frac{3}{4}}}$$

Word2vec - global vectors (GloVe)

- I want a glass of orange juice to go along with my cereal.

- Context c
- Target t

- X_{ij} = # times word j appears in context of word i
- Usually, $x_{ij} = x_{ji}$
- Minimize $\sum_{i=1}^{10,000} \sum_{j=1}^{10,000} f(X_{ij})(e_i^\top e_j + b_i + b_j - \log X_{ij})^2$



Sentiment analysis

Advanced Machine Learning

Sentiment analysis

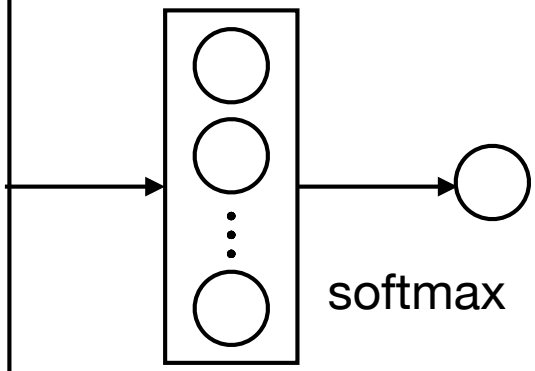
x	y
The dessert is excellent.	
Service was quite slow.	
Good for a quick meal, but nothing special.	
Completely lacking in good taste, good service, and good ambience.	

Sentiment analysis

- The dessert is excellent

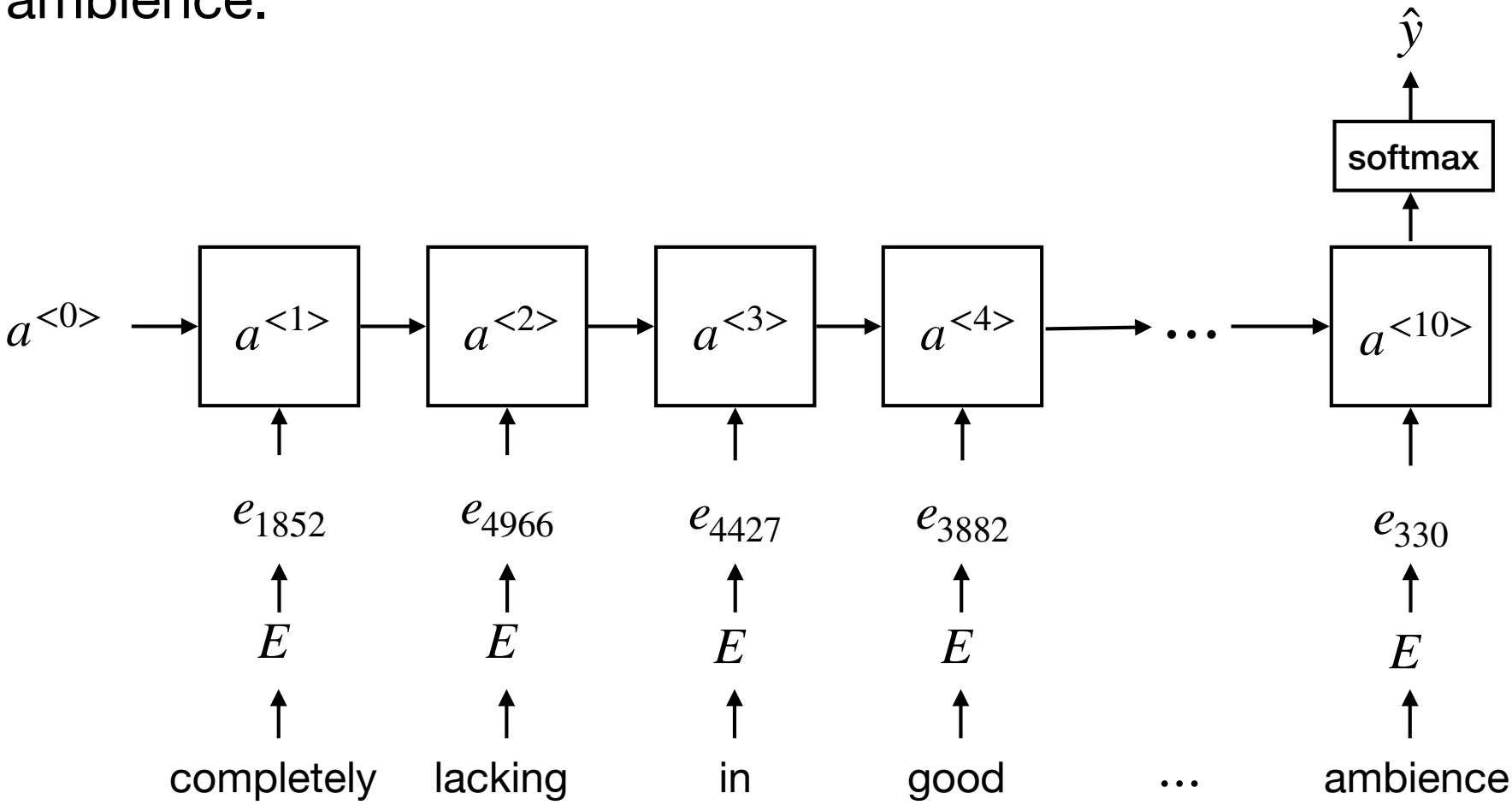
8928 2468 4694 3180

The	o_{8928}	$\rightarrow$	E	$\rightarrow$	e_{8928}
dessert	o_{2468}	$\rightarrow$	E	$\rightarrow$	e_{2468}
is	o_{4694}	$\rightarrow$	E	$\rightarrow$	e_{4694}
excellent	o_{3180}	$\rightarrow$	E	$\rightarrow$	e_{3180}



Sentiment analysis

- "Completely lacking in good taste, good service, and good ambience."



Bias in word embeddings

- Bias in word embeddings

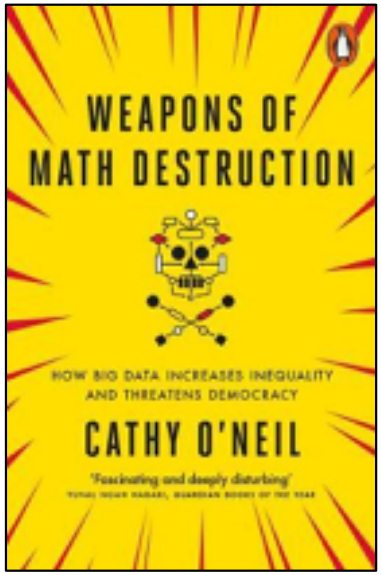
- Man:Woman as King:??
- Man:Computer_Programmer as Woman:??
- Father:Doctor as Mother:??

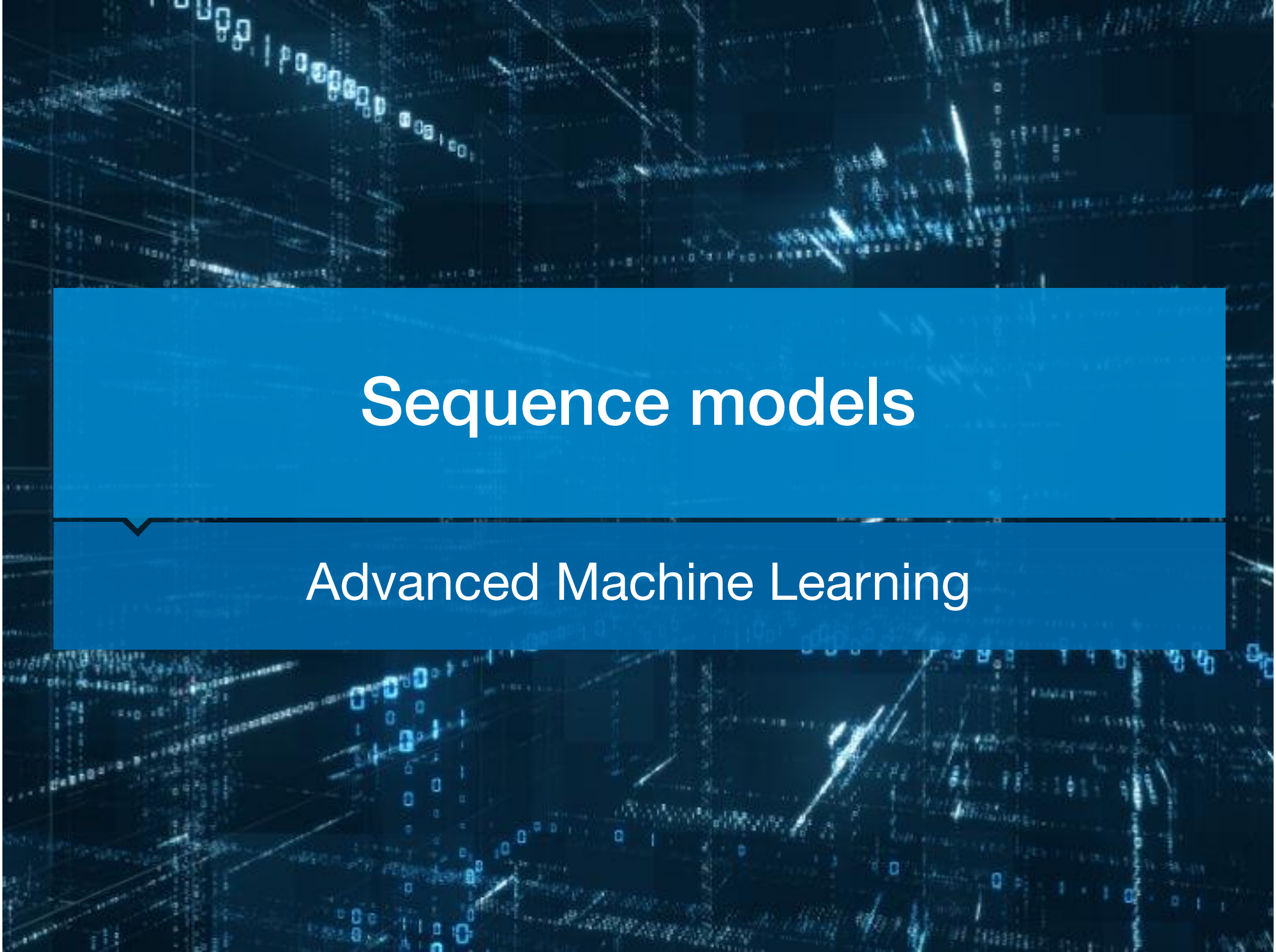
Bias in word embeddings

- Bias in word embeddings

- Man:Woman as King:Queen
- Man:Computer_Programmer as Woman:Homemaker
- Father:Doctor as Mother:Nurse

- Word embeddings can reflect gender, ethnicity age, sexual orientation, and other biases of the text used to train the model

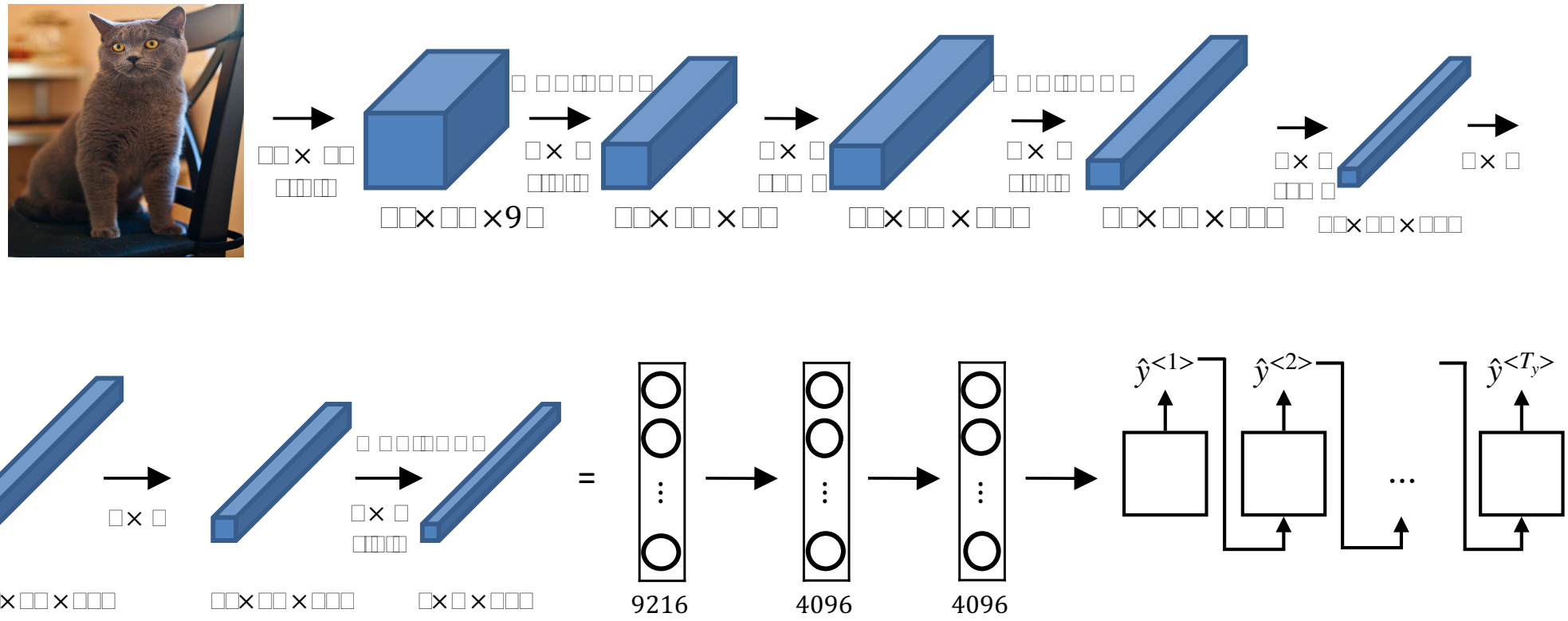




Sequence models

Advanced Machine Learning

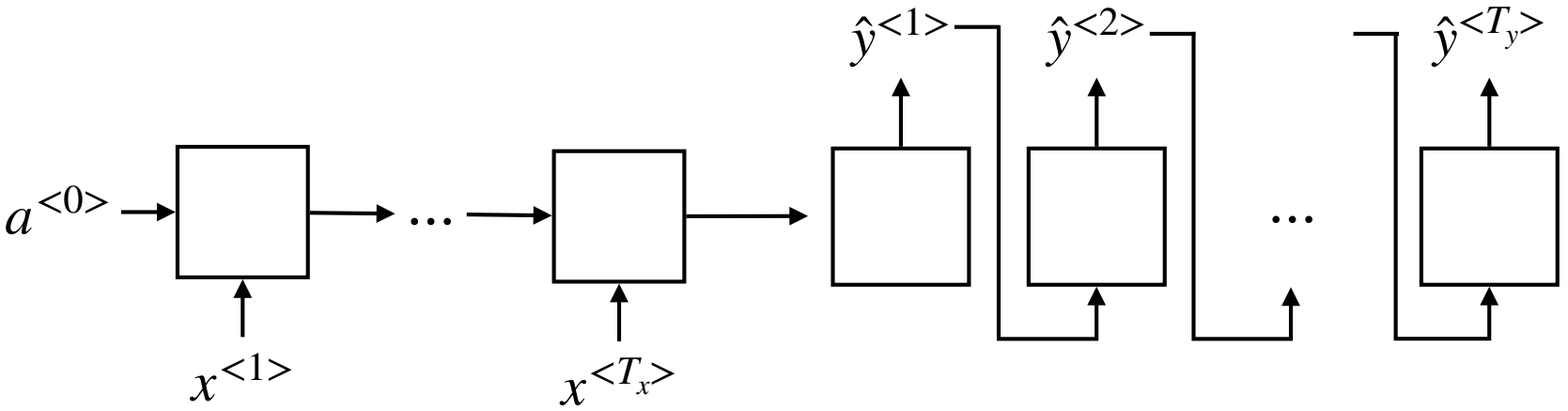
Image captioning



Machine translation

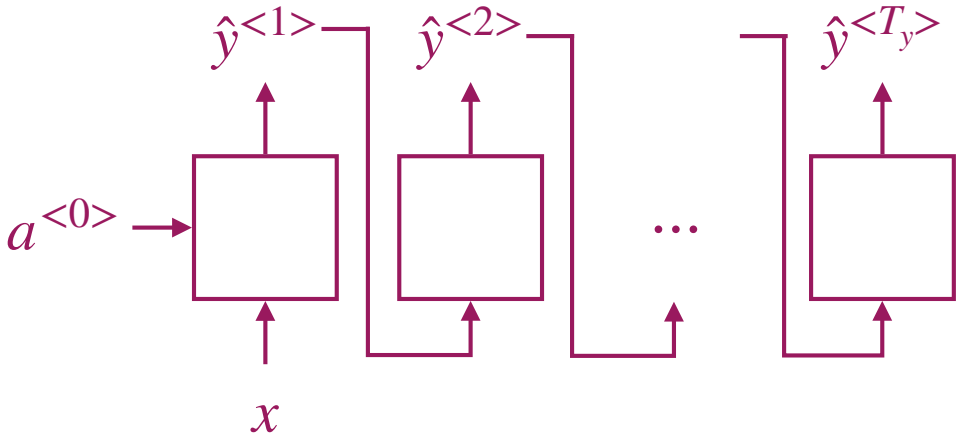
- Jane visite l'Afrique en Septembre.
  $x^{<1>}$ $x^{<2>}$ $x^{<3>}$ $x^{<4>}$ $x^{<5>}$

- Jane is visiting Africa in September.
  $y^{<1>}$ $y^{<2>}$ $y^{<3>}$ $y^{<4>}$ $y^{<5>}$ $y^{<6>}$

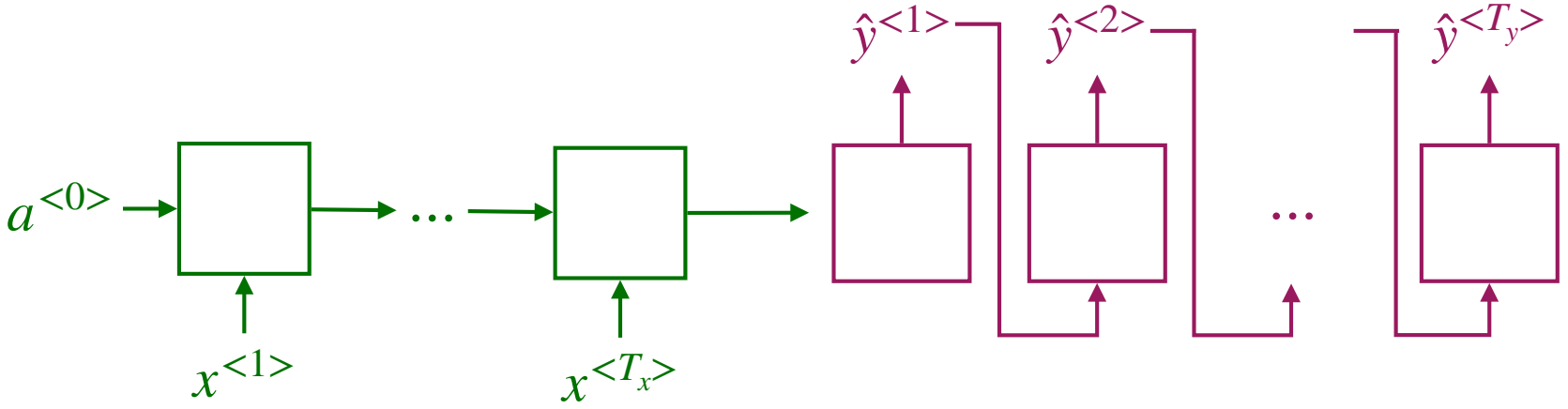


Machine translation

- Language model



- Machine translation



Machine translation

- Jane visite l'Afrique en Septembre.
  - > Jane is visiting Africa in September.
  - > Jane is going to be visiting Africa in September.
  - > In September, Jane will visit Africa.
  - > Her African friend welcomed Jane in September.

$$\mathbb{P}(y^{<1>}, \ldots, y^{<T_y>} \quad x)$$

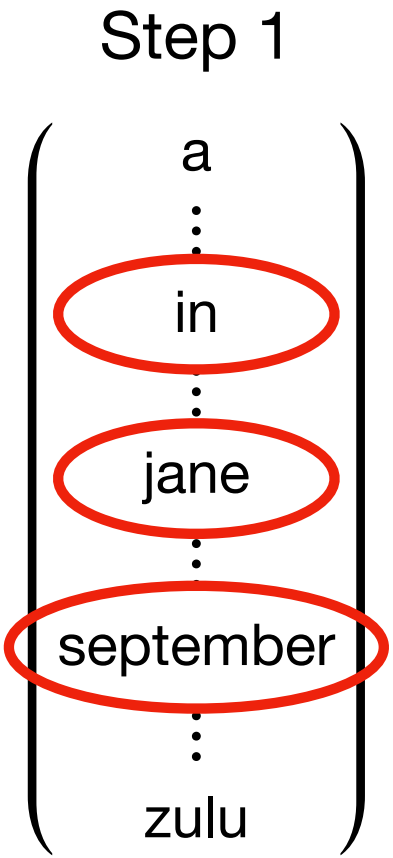
Machine translation

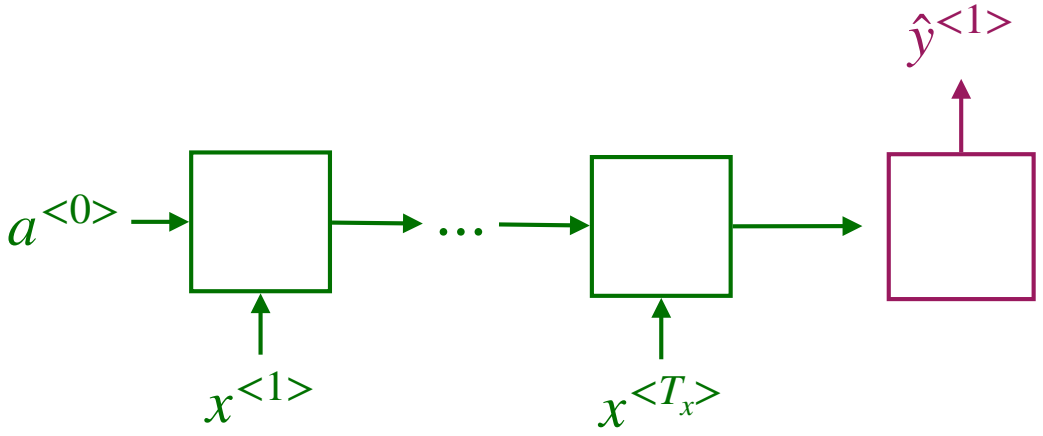
$$\arg \max_{y^{<1>},\ldots,y^{<T_y>}} \mathbb{P}(y^{<1>}, \ldots, y^{<T_y>} \mid x)$$

- Why not greedy search?
  - Jane is visiting Africa in September.
  - Jane is going to be visiting Africa in September.

$$\mathbb{P}(\text{Jane is going} \mid x) > \mathbb{P}(\text{Jane is visiting} \mid x)$$

Beam search

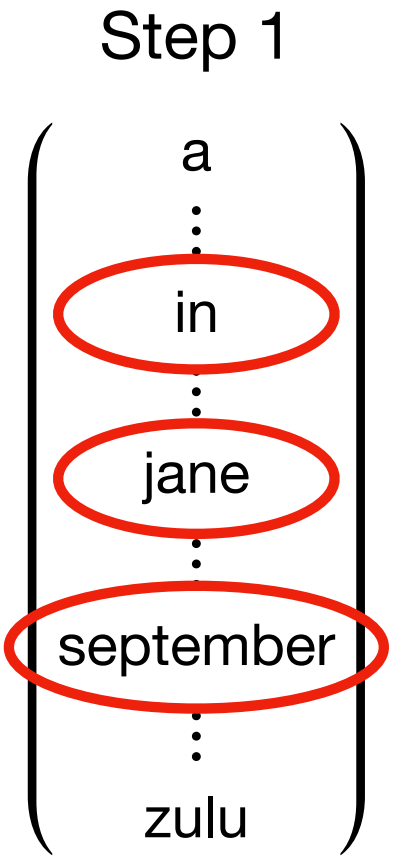


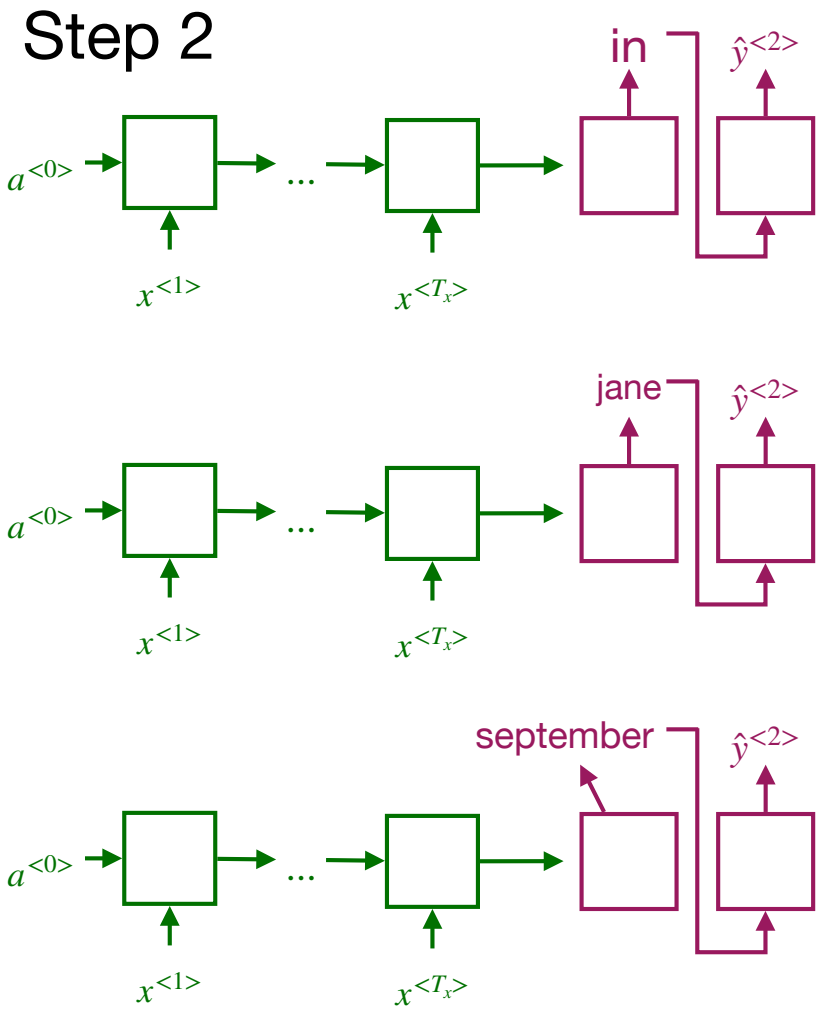


$$\mathbb{P}(y^{<1>} \quad x)$$

Beam width = 3

Beam search





$\mathbb{P}(y^{<2>} \quad x, \text{"in"})$

$\mathbb{P}(y^{<2>} \quad x, \text{"jane"})$

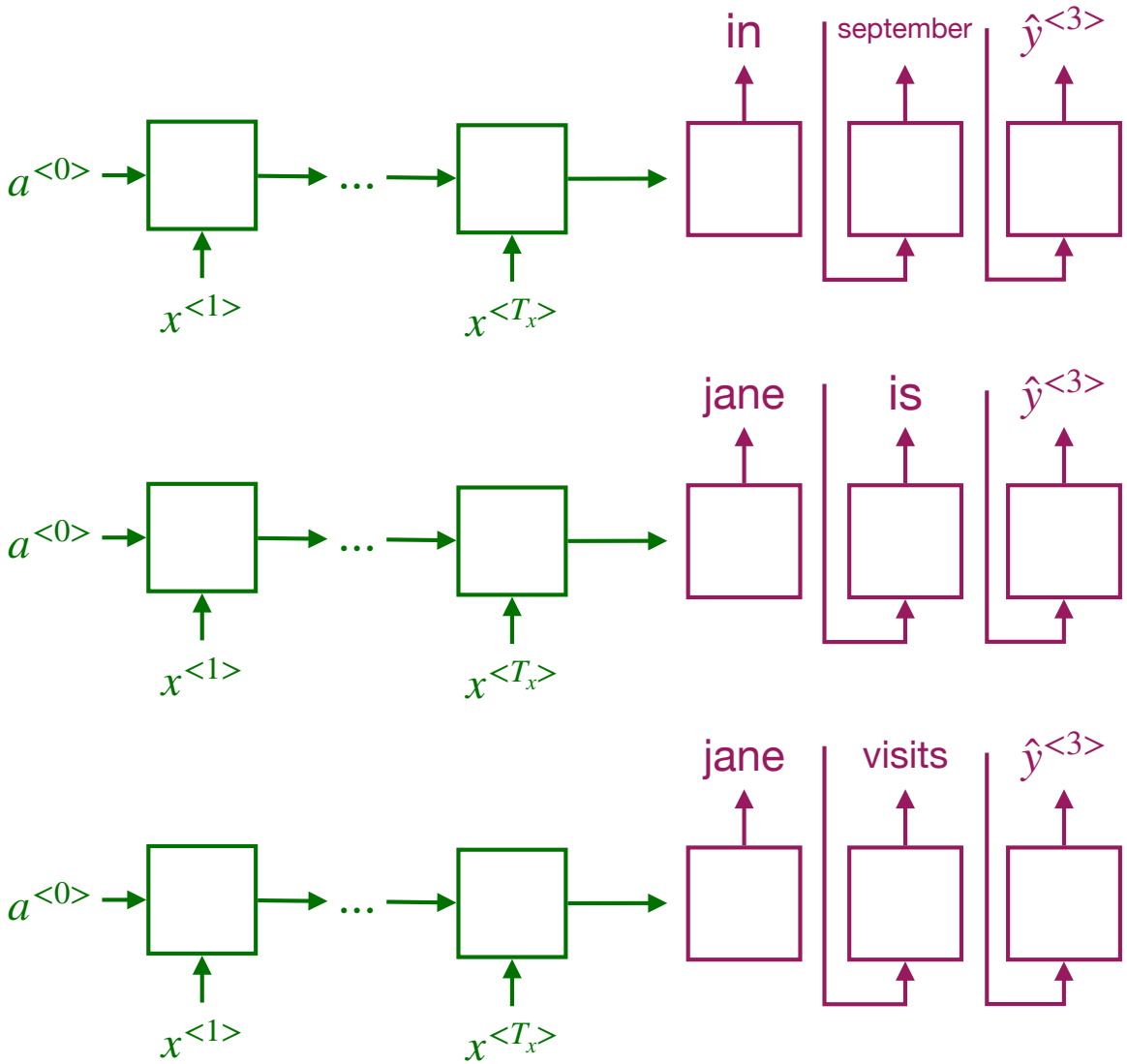
$\mathbb{P}(y^{<2>} \quad x, \text{"september"})$

Beam search

in september

jane in

jane visits



Practical issues

- Length normalization

$$\arg \max_{y^{<1>},\ldots,y^{<T_y>}} \mathbb{P}(y^{<1>},\ldots,y^{<T_y>} \mid x) = \arg\max_{y} \prod_{t=1}^{T_y} \mathbb{P}(y^{<t>} \mid x, y^{<1>},\ldots,y^{<y-1>})$$

$$\downarrow$$

$$\arg\max_{y} \sum_{t=1}^{T_y} \log \mathbb{P}(y^{<t>} \mid x, y^{<1>},\ldots,y^{<t-1>})$$

$$\downarrow$$

$$\frac{1}{T_y^{\alpha}} \sum_{t=1}^{T_y} \log \mathbb{P}(y^{<t>} \mid x, y^{<1>},\ldots,y^{<t-1>})$$

Practical issues

- Error analysis
  - Jane visite l'Afrique en Septembre
  - Human: Jane visits Africa in September.
  - Algorithm: Jane visited Africa last September.

Practical issues

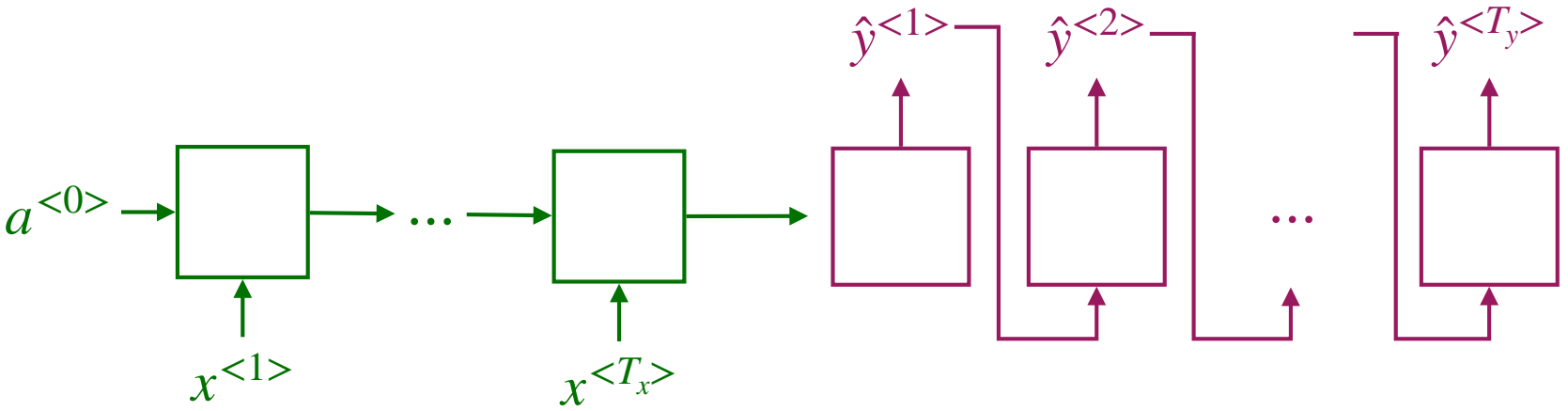
- Error analysis
  - Human: Jane visits Africa in September. (y^*)
  - Algorithm: Jane visited Africa last September. $(\hat{y})$
- Case 1: $\mathbb{P}(y^* \mid x) > \mathbb{P}(\hat{y} \mid x)$
  - Beam search chose $\hat{y}$. But, y^* attains higher $\mathbb{P}(y \mid x)$.
  - Conclusion: Beam search is at fault.
- Case 2: $\mathbb{P}(y^* \mid x) \leq \mathbb{P}(\hat{y} \mid x)$
  - y^* is a better translation than $\hat{y}$. But RNN predicted $\mathbb{P}(y^* \mid x) < \mathbb{P}(\hat{y} \mid x)$
  - Conclusion: RNN model is at fault.


Recent developments
Advanced Machine Learning

Attention mechanism

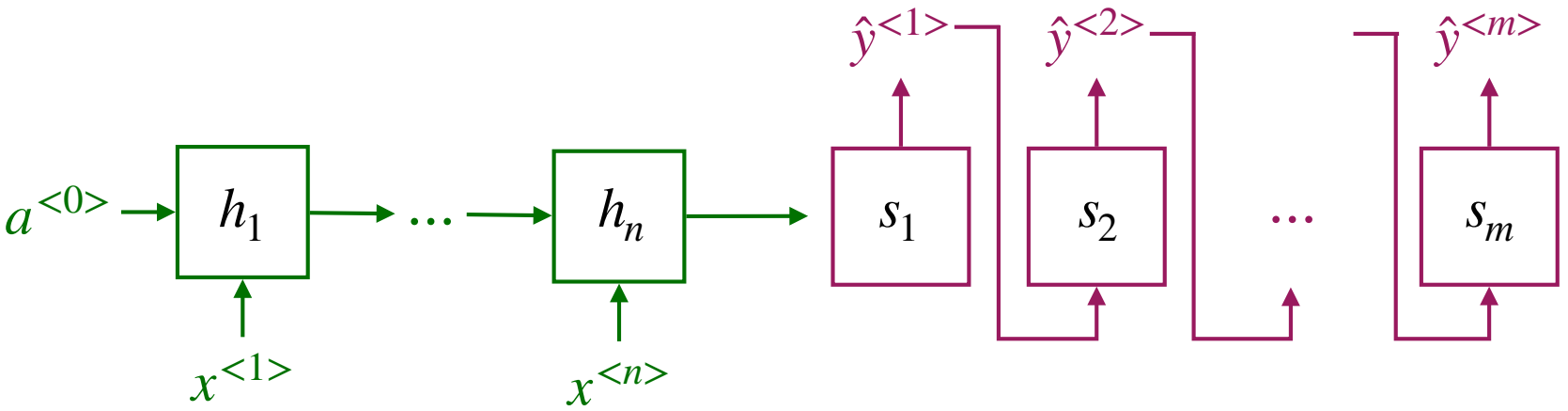
- Seq2Seq model (2014) → encoder-decoder representation

- Major drawback:
  - Fixed length representation of LSTM cell, loss of memory
  - Very hard to train



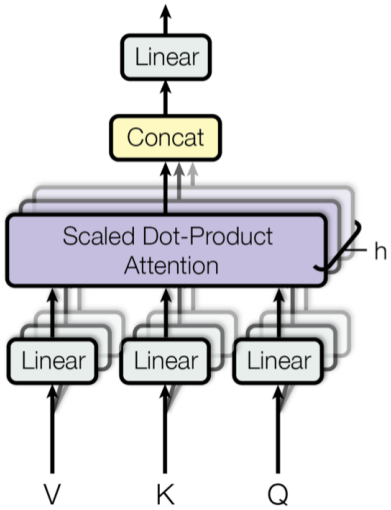
Attention mechanism

- Given an input $\mathbf{x} = [x_1, \ldots, x_n]$ and output $\mathbf{y} = [y_1, \ldots, y_m]$
- Define the context as $c_i = \sum_{i=1}^{n} \alpha_{t,i} h_i$
- The weights $\alpha_{t,i} = \text{align}(y_t, x_i) = \frac{\exp(\text{score}(s_{t-1}, h_i))}{\sum_{j=1}^{n} \exp(\text{score}(s_{t-1}, h_j))}$



Attention mechanism

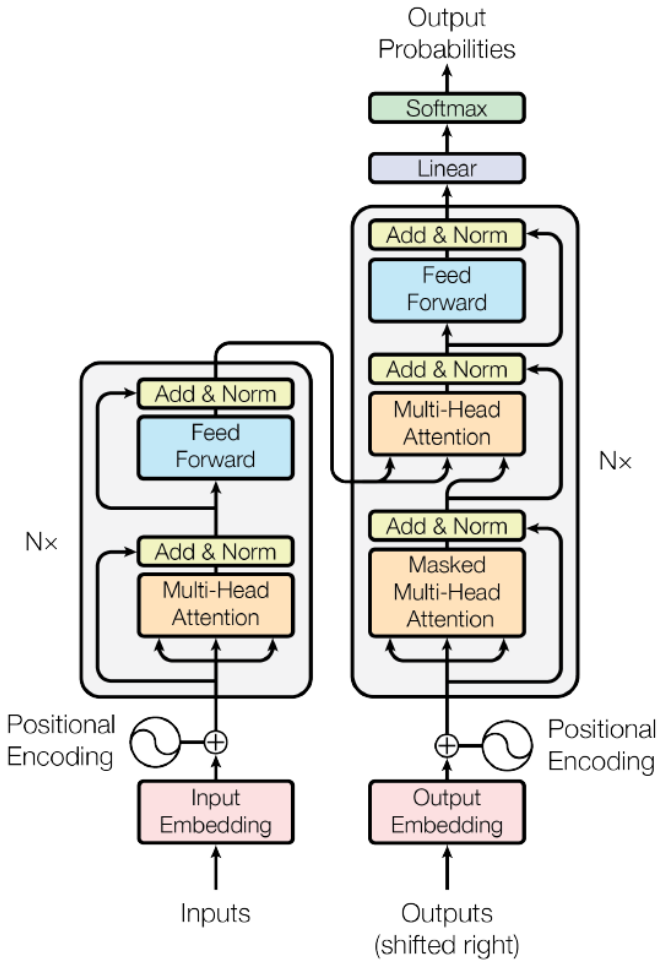
- Multi-head self attention (2017)
- Q(uery), K(ey) and V(alue) operators
- Attention is given by $\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{n}}\right)V$



Transformer (2017)

- Paper “Attention is all you need”
- Transformer is an architecture for transforming one sequence into another one
- Without any Recurrent Networks!

- Positional encoding takes over part of the RNN functionality



BERT (Google, 2018)

- Bidirectional Encoder Representations from Transformers
- 340 million parameters

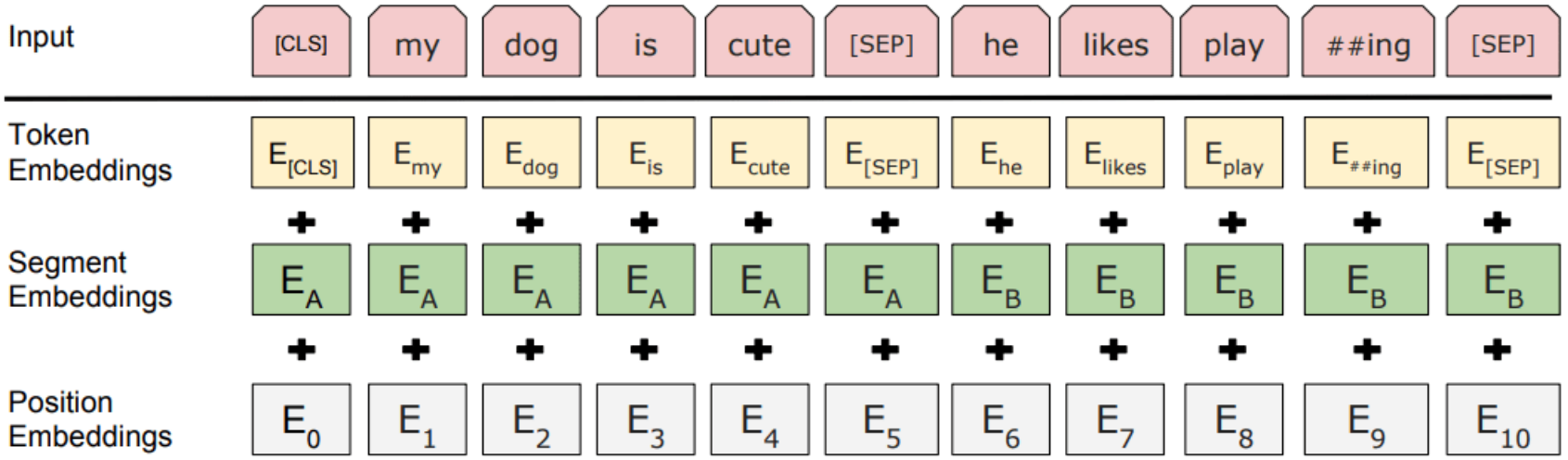
- Traditional models:
  - I want a glass of orange _____.
  - Trained with one-directional models
  - Word embeddings are context-free

- Innovation:
  - Bi-directional training (actually, non-directional)
  - Masked Language Models (Masked LM)

BERT (Google, 2018)

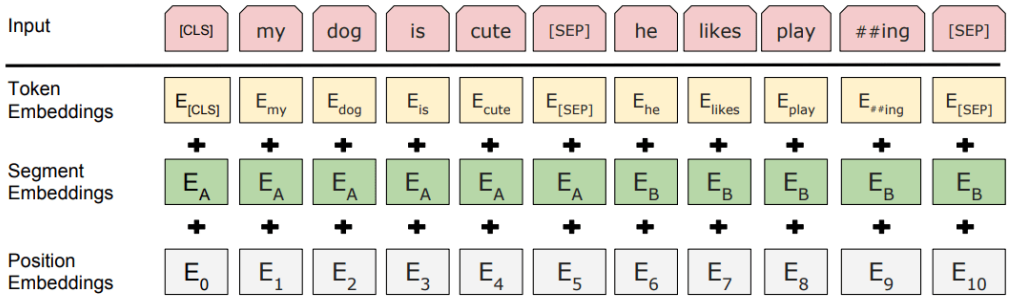
Input processing

- Token embeddings
- Segment embeddings
- Positional embeddings



BERT (Google, 2018)

- During training the model is fed with two input sentences at a time such that
  - 50% of the time, sentence comes after the first one
  - 50% of the time, random sentence from the full corpus.
- Masked LM (MLM): Randomly mask out 15% of the words in the input
- Next Sentence Prediction

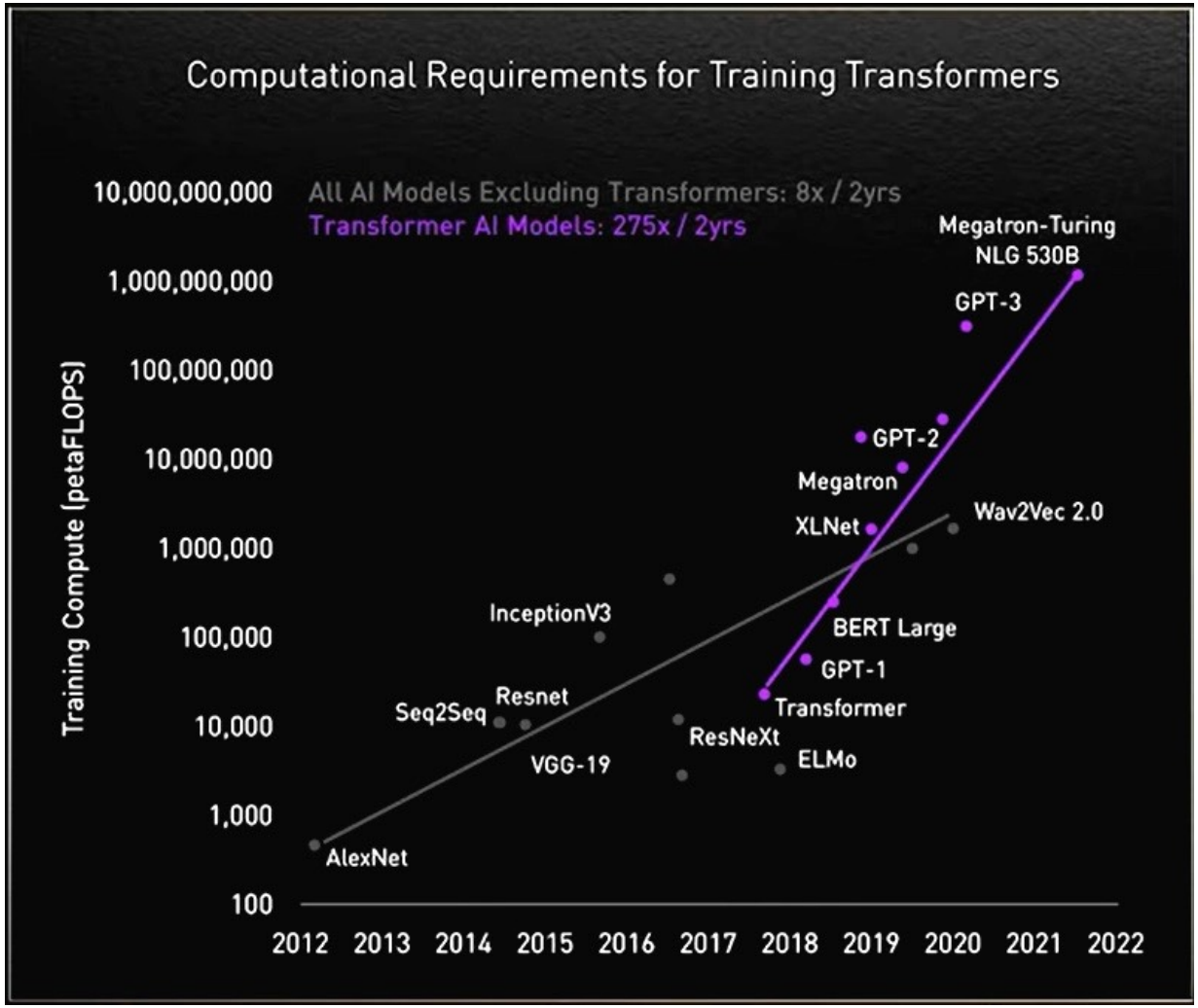


GPT-3 (OpenAI, 2020)

- Generative Pretrained Transformer version 3
- GPT-3 uses 175 billion parameters

- The model can perform “few-shot,” “one-shot,” or “zero-shot” learning, i.e., no fine-tuning is necessary
- OpenAI has not disclosed details about it yet
- GPT-3 175B model required 3.14E23 FLOPS of computing for training
- training: at theoretical 28 TFLOPS for V100, this will take 355 GPU-years and cost $4.6M

Computational power



Advanced Machine Learning
Lecture 10: Hidden Markov models

Sandjai Bhulai
Vrije Universiteit Amsterdam

s.bhulai@vu.nl
06 October 2023

Markov models

Advanced Machine Learning

Markov models

- All previously discussed methods assume i.i.d. data
  - Likelihood function is expressed as a product

- Assumption is poor for sequence data
  - Time series data
  - Sequence of characters / speech
  - DNA analysis

Markov models

- Intuitively, recent observations are likely to be more informative
  - Markov models

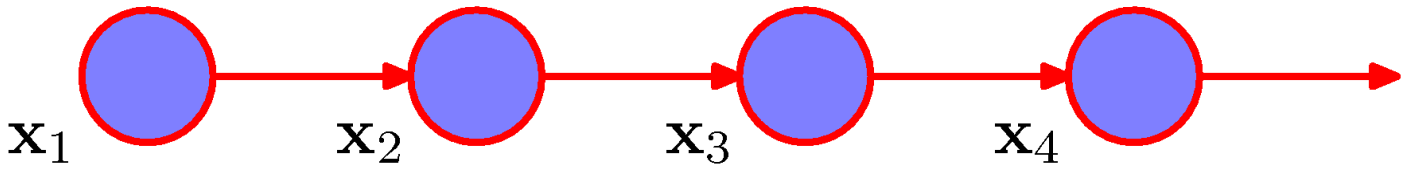
- More general framework: state space models
  - Includes latent variables
  - Hidden Markov models
  - Linear dynamical systems

Markov models

- Joint distribution is given by

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N) = \prod_{n=1}^{N} p(\mathbf{x}_n \mid \mathbf{x}_1, \ldots, \mathbf{x}_{n-1})$$

- If conditional distributions are independent of all previous observations except for the most recent → first-order Markov chain



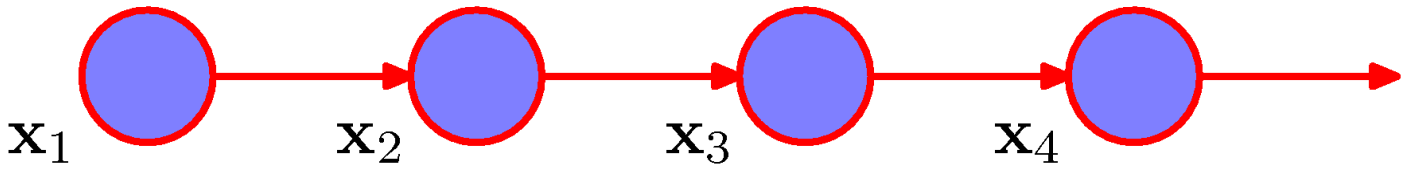
Markov models

- Joint distribution is given by

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N) = p(\mathbf{x}_1) \prod_{n=2}^{N} p(\mathbf{x}_n \mid \mathbf{x}_{n-1})$$

- From the d-separation theorem, we can derive

$$p(\mathbf{x}_n \mid \mathbf{x}_1, \ldots, \mathbf{x}_{n-1}) = p(\mathbf{x}_n \mid \mathbf{x}_{n-1})$$

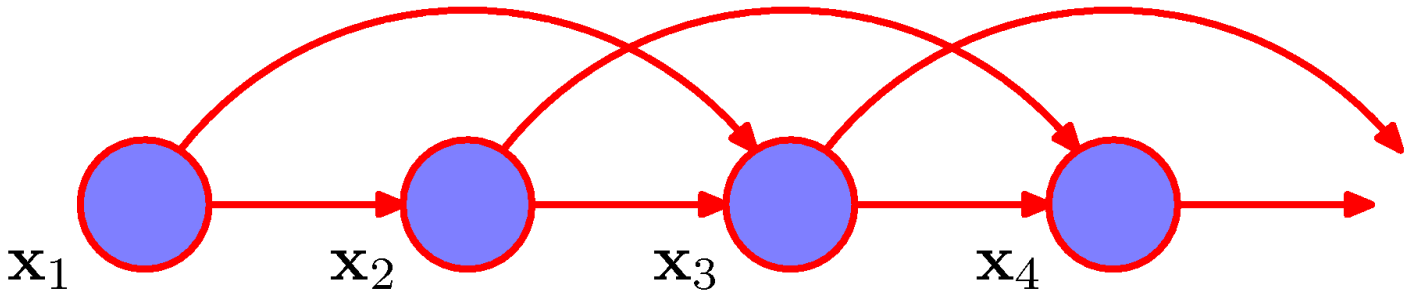


Markov models

- Second-order Markov chain

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N) = p(\mathbf{x}_1)p(\mathbf{x}_2 \mid \mathbf{x}_1)\prod_{n=3}^{N} p(\mathbf{x}_n \mid \mathbf{x}_{n-1}, \mathbf{x}_{n-2})$$

- M-th order Markov chain has $K^{M-1}(K-1)$ parameters for a discrete K-valued state



Markov models

- Objective:
  - Build models not limited by the Markov assumption
  - Model that can be specified with a limited number of free parameters

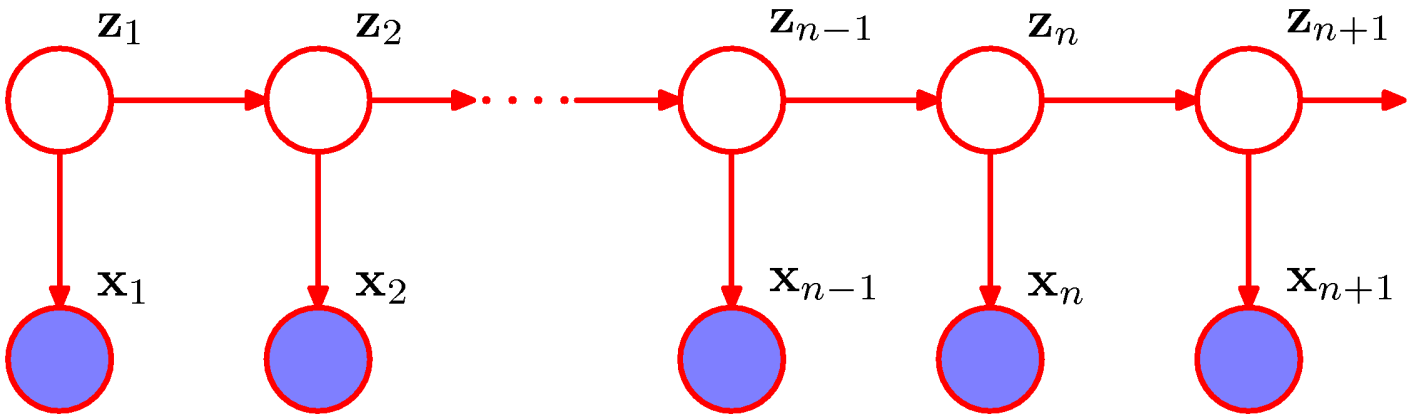
- Solution: introduce latent variables

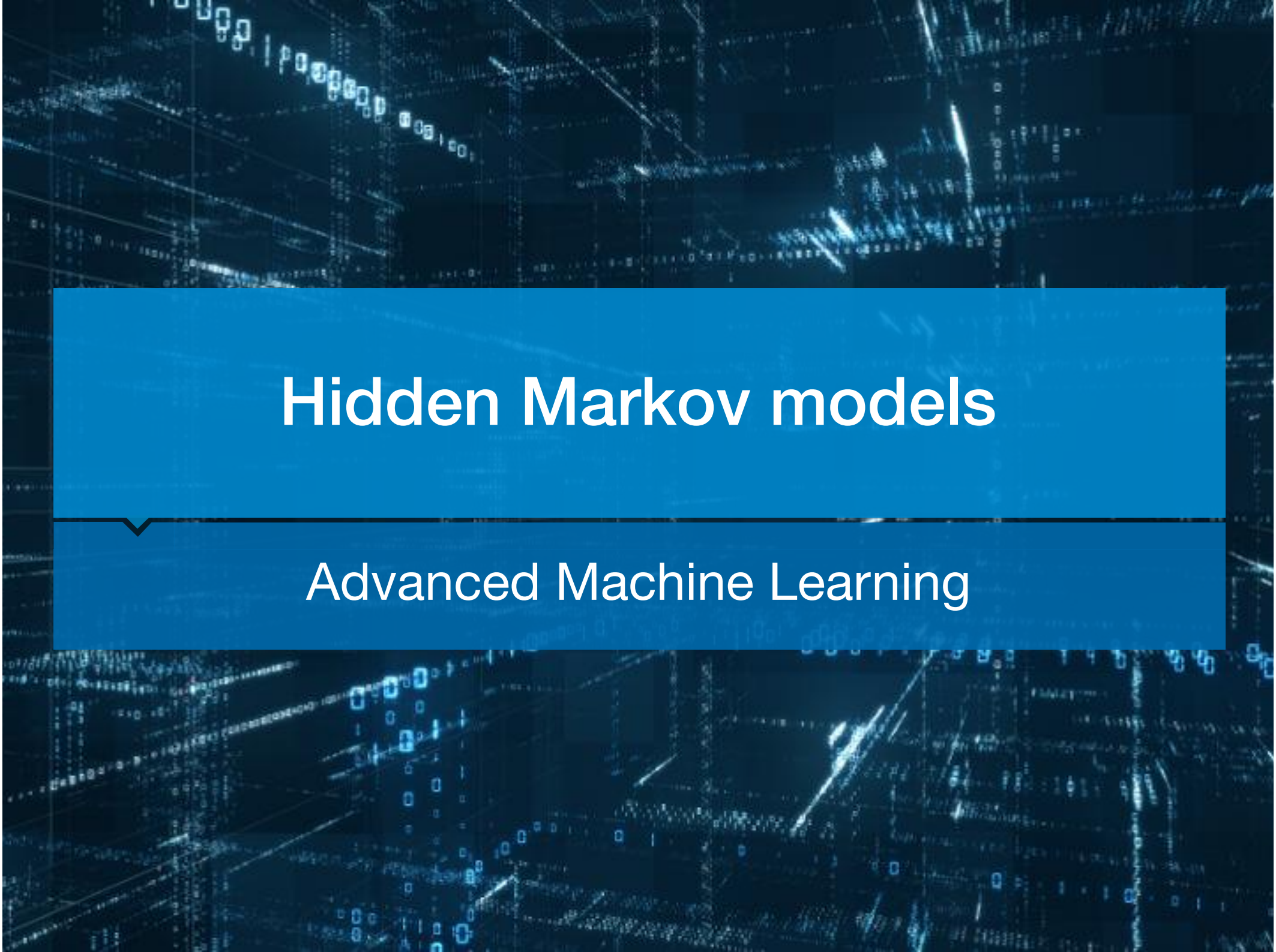
Markov models

- Introduce latent variables such that

$$z_{n+1} \perp\!\!\!\perp z_{n-1} \mid z_n$$

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_N, \mathbf{z}_1, \ldots, \mathbf{z}_N) = p(\mathbf{z}_1) \left[\prod_{n=2}^{N} p(\mathbf{z}_n \mid \mathbf{z}_{n-1}) \right] \prod_{n=1}^{N} p(\mathbf{x}_n \mid \mathbf{z}_n)$$





Hidden Markov models

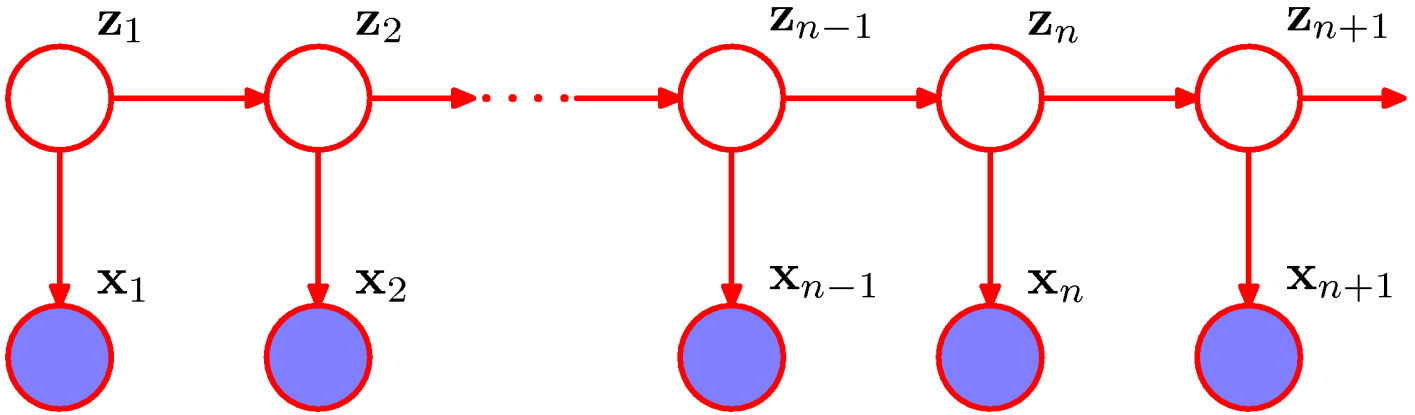
Advanced Machine Learning

Hidden Markov models

- Define $A_{jk} \equiv p(z_{nk} = 1 \mid z_{n-1,j} = 1)$
- Then

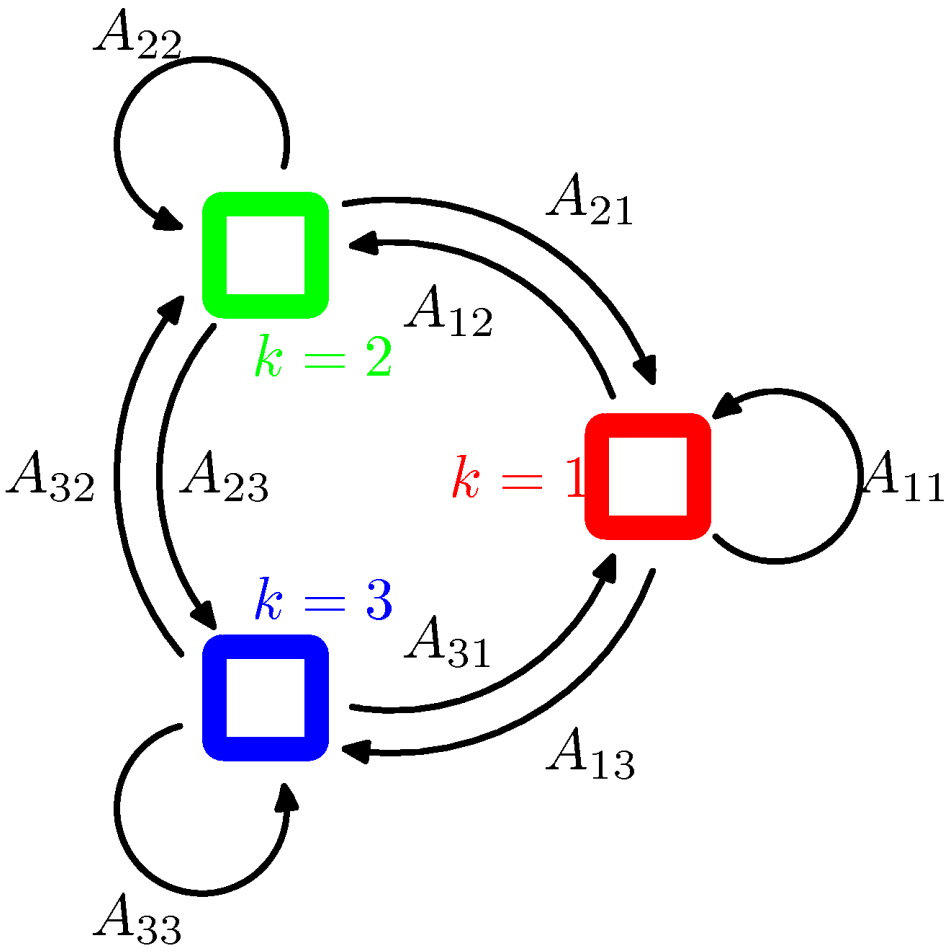
$$p(\mathbf{z}_n \mid \mathbf{z}_{n-1}, \mathbf{A}) = \prod_{k=1}^{K} \prod_{j=1}^{K} A_{jk}^{z_{n-1,j} z_{nk}}$$

$$p(\mathbf{z}_1 \mid \pi) = \prod_{k=1}^{K} \pi_k^{z_{1k}}$$



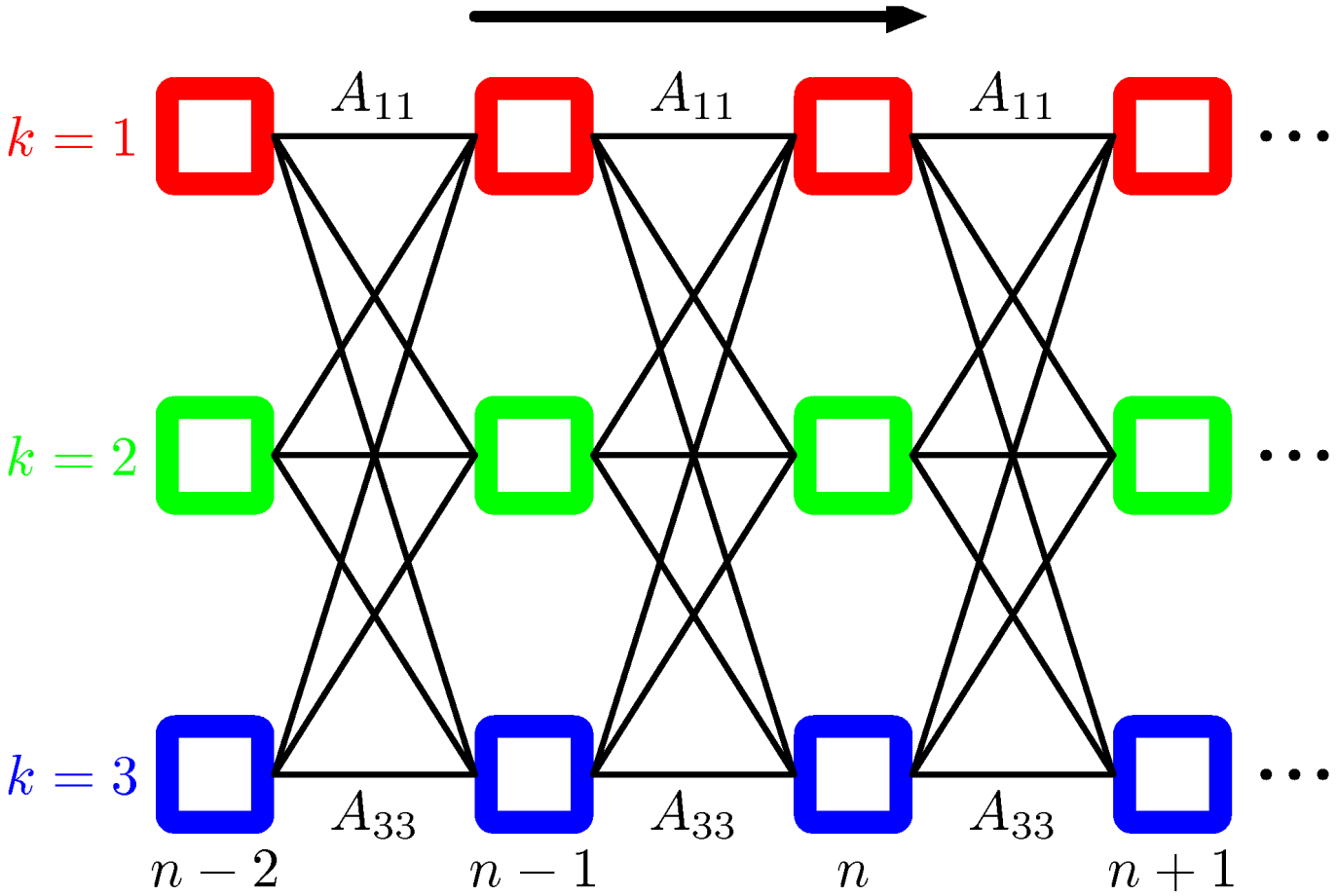
Hidden Markov models

- Example: Markov chain on latent variables



Hidden Markov models

- Example: Markov chain on latent variables



Hidden Markov models

- Description is completed by defining emission probabilities

$$p(\mathbf{x}_n \mid \mathbf{z}_n, \phi)$$

- Then

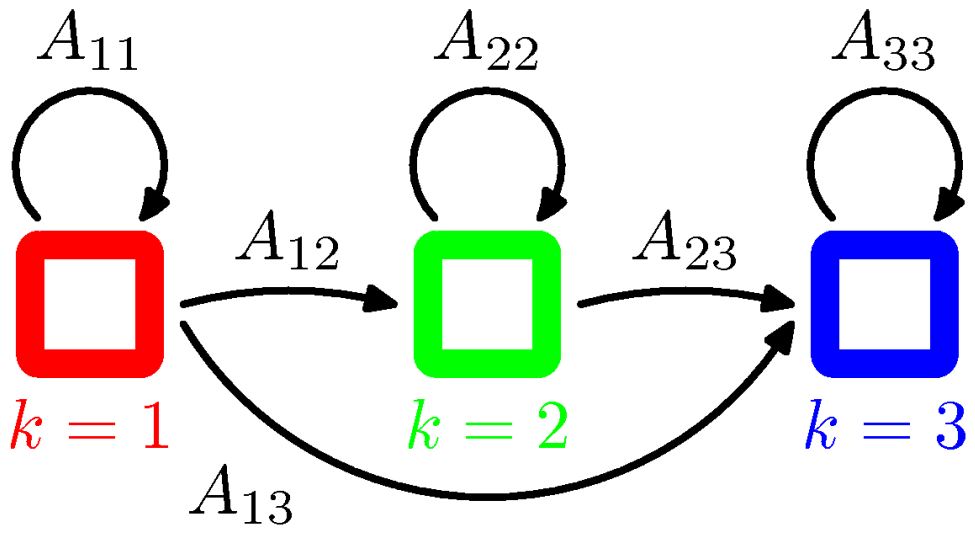
$$p(\mathbf{x}_n \mid \mathbf{z}_n, \phi) = \prod_{k=1}^{K} p(x_n \mid \phi_k)^{z_{nk}}$$

- And thus,

$$p(\mathbf{X}, \mathbf{Z} \mid \theta) = p(\mathbf{z}_1 \mid \pi) \left[\prod_{n=2}^{N} p(\mathbf{z}_n \mid \mathbf{z}_{n-1}, \mathbf{A}) \right] \prod_{m=1}^{N} p(\mathbf{x}_m \mid \mathbf{z}_m, \phi)$$

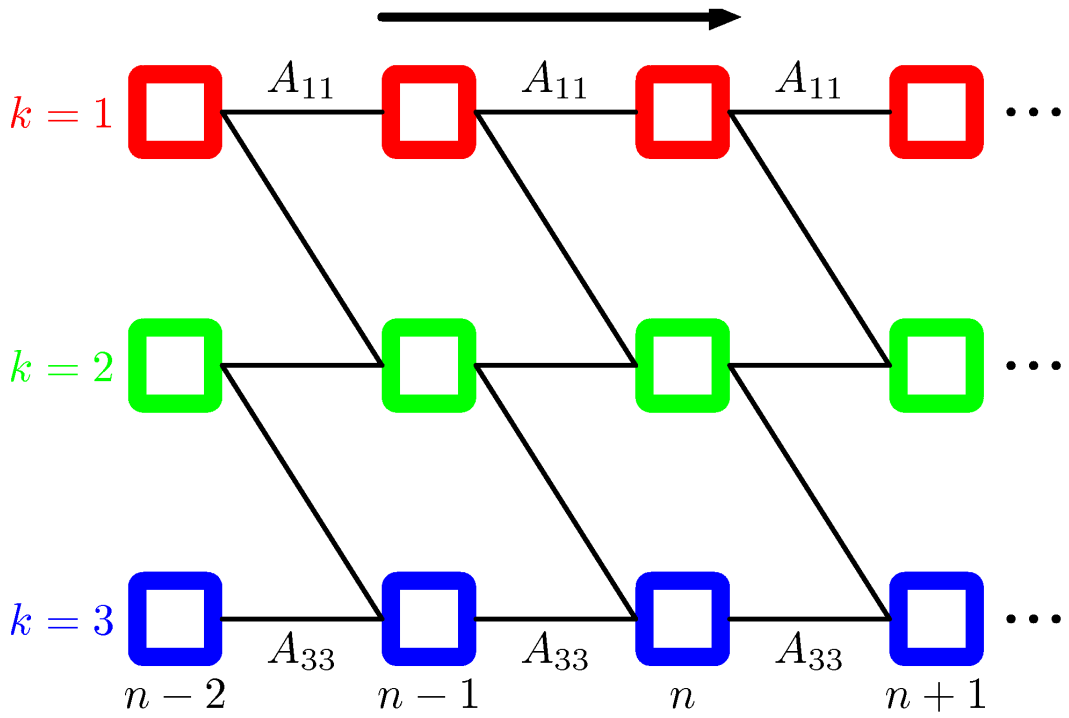
Hidden Markov models

- Left-to-right HMM
- Set $A_{jk} = 0$ if $k < j$



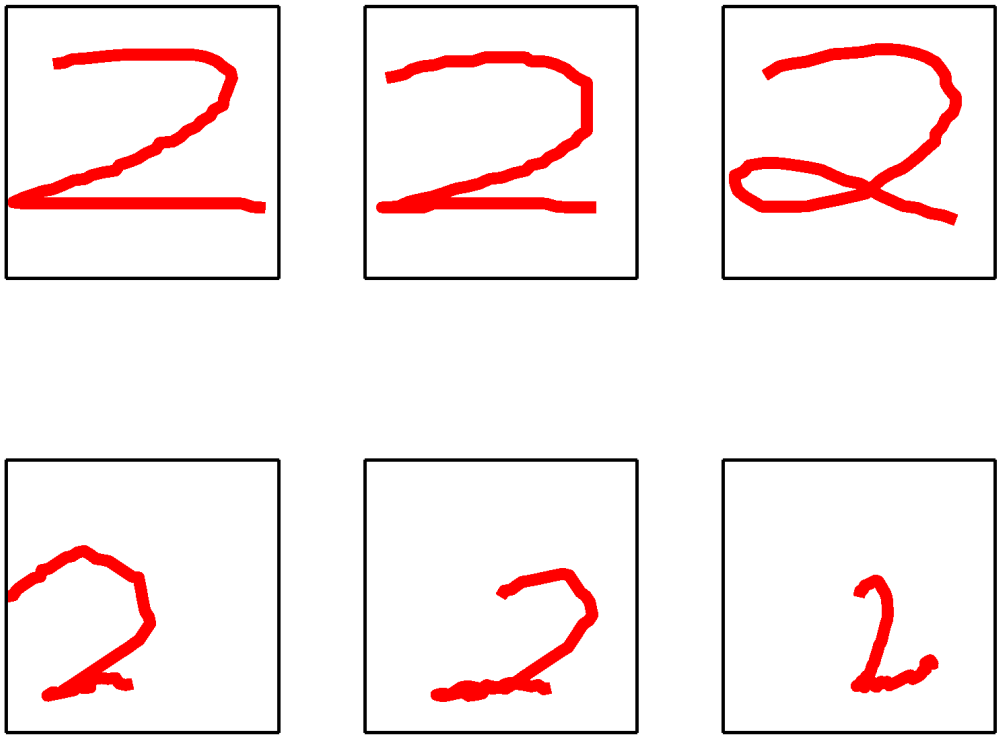
Hidden Markov models

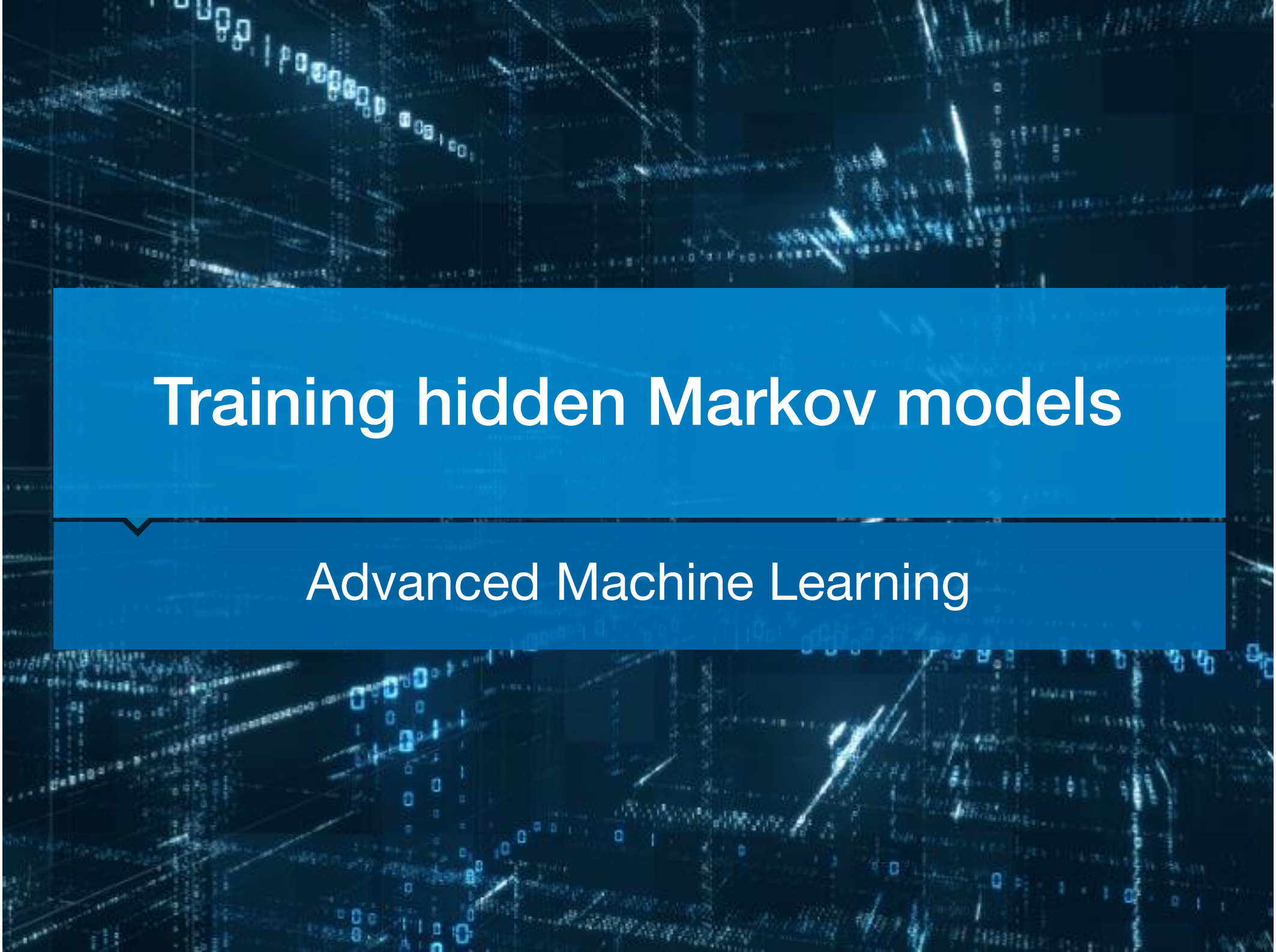
- Left-to-right HMM
- Set $A_{jk} = 0$ if $k < j$



Hidden Markov models

- Example: handwritten digits
- $K = 16$, representing angles of the pen





Training hidden Markov models

Advanced Machine Learning

EM algorithm

- Given $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, consider the joint distribution

$$p(\mathbf{X} \ \ \theta) = \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z} \ \ \theta)$$

- Likelihood maximization will not work!
- Joint distribution does not factorize over n
- Summation requires K^n terms

EM algorithm

- Alternative: expectation maximization (EM)
- Take initial selection of parameters θ^{old}
- Calculate posterior distribution $p(\mathbf{Z} \mid \mathbf{X}, \theta^{\text{old}})$
- Define

$$Q(\theta, \theta^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z} \mid \mathbf{X}, \theta^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z} \mid \theta)$$

- and

$$\gamma(\mathbf{z}_n) = p(\mathbf{z}_n \mid \mathbf{X}, \theta^{\text{old}})$$

$$\xi(\mathbf{z}_{n-1}, \mathbf{z}_n) = p(\mathbf{z}_{n-1}, \mathbf{z}_n \mid \mathbf{X}, \theta^{\text{old}})$$

EM algorithm

- Now,

$$\gamma(z_{nk}) = \mathbb{E}[z_{nk}] = \sum_{\mathbf{z}} \gamma(\mathbf{z}) z_{nk}$$

$$\xi(z_{n-1,j}, z_{nk}) = \mathbb{E}[z_{n-1,j}, z_{nk}] = \sum_{\mathbf{z}} \gamma(\mathbf{z}) z_{n-1,j} z_{nk}$$

- Then the E step is given by,

$$Q(\theta, \theta^{\text{old}}) = \sum_{\mathbf{z}} p(\mathbf{Z} \mid \mathbf{X}, \theta^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z} \mid \theta)$$

$$= \sum_{k=1}^{K} \gamma(z_{1k}) \ln \pi_k + \sum_{n=2}^{N} \sum_{j=1}^{K} \sum_{k=1}^{K} \xi(z_{n-1,j}, z_{nk}) \ln A_{jk}$$

$$+ \sum_{n=1}^{N} \sum_{k=1}^{K} \gamma(z_{nk}) \ln p(\mathbf{x}_n \mid \phi_k)$$

EM algorithm

- In the M step, we maximize $Q(\theta, \theta^{\text{old}})$
- Using Lagrange multipliers, we get

$$\pi_k = \frac{\gamma(z_{1k})}{\sum_{j=1}^{K} \gamma(z_{1j})}$$

$$A_{jk} = \frac{\sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nk})}{\sum_{l=1}^{K} \sum_{n=2}^{N} \xi(z_{n-1,j}, z_{nl})}$$

Forward-backward algorithm

$$p(\mathbf{X} \mid \mathbf{z}_n) = p(\mathbf{x}_1, \ldots, \mathbf{x}_n \mid \mathbf{z}_n) p(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N \mid \mathbf{z}_n)$$

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_{n-1} \mid \mathbf{x}_n, \mathbf{z}_n) = p(\mathbf{x}_1, \ldots, \mathbf{x}_{n-1} \mid \mathbf{z}_n)$$

$$p(\mathbf{x}_1, \ldots, \mathbf{x}_{n-1} \mid \mathbf{z}_{n-1}, \mathbf{z}_n) = p(\mathbf{x}_1, \ldots, \mathbf{x}_{n-1} \mid \mathbf{z}_{n-1})$$

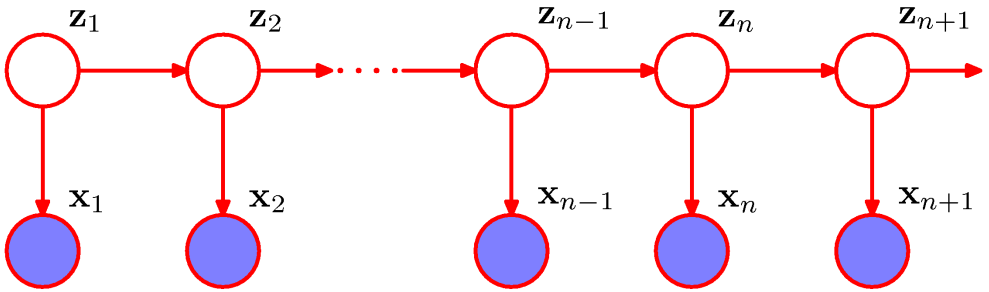
$$p(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N \mid \mathbf{z}_n, \mathbf{z}_{n+1}) = p(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N \mid \mathbf{z}_{n+1})$$

$$p(\mathbf{x}_{n+2}, \ldots, \mathbf{x}_N \mid \mathbf{z}_{n+1}, \mathbf{x}_{n+1}) = p(\mathbf{x}_{n+2}, \ldots, \mathbf{x}_N \mid \mathbf{z}_{n+1})$$

$$p(\mathbf{X} \mid \mathbf{z}_{n-1}, \mathbf{z}_n) = p(\mathbf{x}_1, \ldots, \mathbf{x}_{n-1} \mid \mathbf{z}_{n-1}) p(\mathbf{x}_n \mid \mathbf{z}_n) p(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N \mid \mathbf{z}_n)$$

$$p(\mathbf{x}_{N+1} \mid \mathbf{X}, \mathbf{z}_{N+1}) = p(\mathbf{x}_{N+1} \mid \mathbf{z}_{N+1})$$

$$p(\mathbf{z}_{N+1} \mid \mathbf{z}_N, \mathbf{X}) = p(\mathbf{z}_{N+1} \mid \mathbf{z}_N)$$



Forward-backward algorithm

- Focus attention on $\gamma(z_{nk})$

$$\gamma(\mathbf{z}_n) = p(\mathbf{z}_n \mid \mathbf{X}) = \frac{p(\mathbf{X} \mid \mathbf{z}_n)p(\mathbf{z}_n)}{p(\mathbf{X})}$$

- Using the rules, we get

$$\gamma(\mathbf{z}_n) = \frac{p(\mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{z}_n)p(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N \mid \mathbf{z}_n)}{p(\mathbf{X})} = \frac{\alpha(\mathbf{z}_n)\beta(\mathbf{z}_n)}{p(\mathbf{X})}$$

with

$$\alpha(\mathbf{z}_n) \equiv p(\mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{z}_n)$$

$$\beta(\mathbf{z}_n) \equiv p(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N \mid \mathbf{z}_n)$$

Forward-backward algorithm

$$\alpha(\mathbf{z}_n) = p(\mathbf{x}_1, \ldots, \mathbf{x}_n, \mathbf{z}_n) = p(\mathbf{x}_1, \ldots, \mathbf{x}_n \mid \mathbf{z}_n) p(\mathbf{z}_n)$$

$$= p(\mathbf{x}_n \mid \mathbf{z}_n) p(\mathbf{x}_1, \ldots, \mathbf{x}_{n-1} \mid \mathbf{z}_n) p(\mathbf{z}_n) = p(\mathbf{x}_n \mid \mathbf{z}_n) p(\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}, \mathbf{z}_n)$$

$$= p(\mathbf{x}_n \mid \mathbf{z}_n) \sum p(\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}, \mathbf{z}_{n-1}, \mathbf{z}_n)$$

$$= p(\mathbf{x}_n \mid \mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} p(\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}, \mathbf{z}_n \mid \mathbf{z}_{n-1}) p(\mathbf{z}_{n-1})$$

$$= p(\mathbf{x}_n \mid \mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} p(\mathbf{x}_1, \ldots, \mathbf{x}_{n-1} \mid \mathbf{z}_{n-1}) p(\mathbf{z}_n \mid \mathbf{z}_{n-1}) p(\mathbf{z}_{n-1})$$

$$= p(\mathbf{x}_n \mid \mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} p(\mathbf{x}_1, \ldots, \mathbf{x}_{n-1}, \mathbf{z}_{n-1}) p(\mathbf{z}_n \mid \mathbf{z}_{n-1})$$

$$= p(\mathbf{x}_n \mid \mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} \alpha(\mathbf{z}_{n-1}) p(\mathbf{z}_n \mid \mathbf{z}_{n-1})$$

Forward-backward algorithm

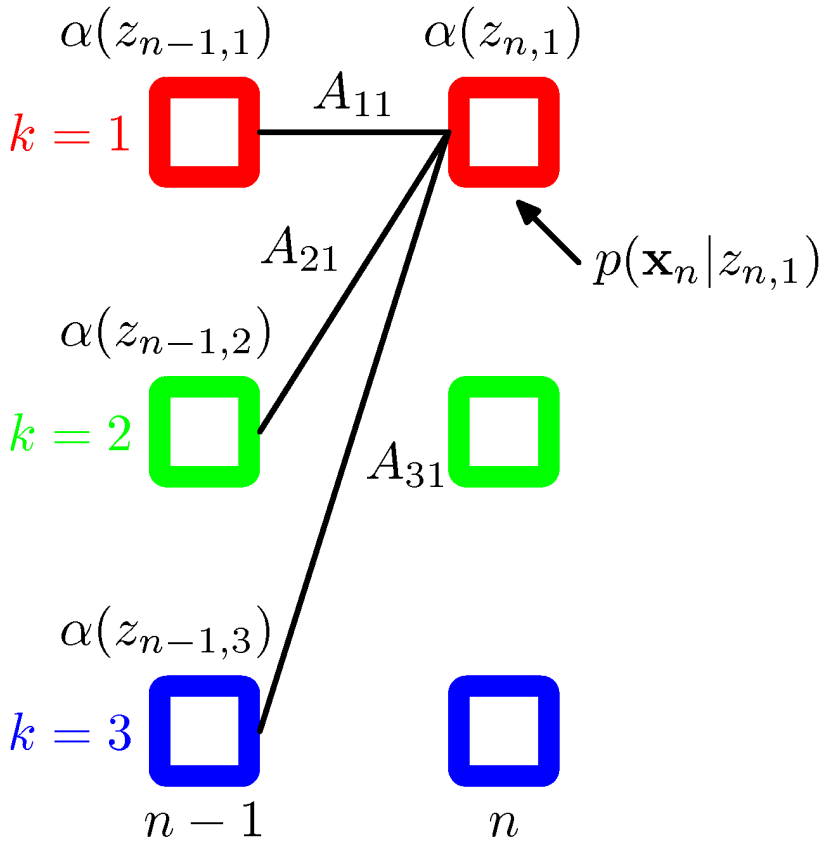
- This leads to the recursion

$$\alpha(\mathbf{z}_n) = p(\mathbf{x}_n \mid \mathbf{z}_n) \sum_{\mathbf{z}_{n-1}} \alpha(\mathbf{z}_{n-1}) p(\mathbf{z}_n \mid \mathbf{z}_{n-1})$$

- Initial condition

$$\alpha(\mathbf{z}_1) = p(\mathbf{x}_1, \mathbf{z}_1) = p(\mathbf{z}_1) p(\mathbf{x}_1 \mid \mathbf{z}_1)$$

$$= \prod_{k=1}^{K} \{\pi_k p(\mathbf{x}_1 \mid \phi_k)\}^{z_{1k}}$$



Forward-backward algorithm

$$
\begin{aligned}
\beta(\mathbf{z}_n) &= p(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N \mid \mathbf{z}_n) \\
&= \sum_{\mathbf{z}_{n+1}} p(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N, \mathbf{z}_{n+1} \mid \mathbf{z}_n) \\
&= \sum_{\mathbf{z}_{n+1}} p(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N \mid \mathbf{z}_n, \mathbf{z}_{n+1}) p(\mathbf{z}_{n+1} \mid \mathbf{z}_n) \\
&= \sum_{\mathbf{z}_{n+1}} p(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N \mid \mathbf{z}_{n+1}) p(\mathbf{z}_{n+1} \mid \mathbf{z}_n) \\
&= \sum_{\mathbf{z}_{n+1}} p(\mathbf{x}_{n+2}, \ldots, \mathbf{x}_N \mid \mathbf{z}_{n+1}) p(\mathbf{x}_{n+1} \mid \mathbf{z}_{n+1}) p(\mathbf{z}_{n+1} \mid \mathbf{z}_n) \\
&= \sum_{\mathbf{z}_{n+1}} \beta(\mathbf{z}_{n+1}) p(\mathbf{x}_{n+1} \mid \mathbf{z}_{n+1}) p(\mathbf{z}_{n+1} \mid \mathbf{z}_n)
\end{aligned}
$$

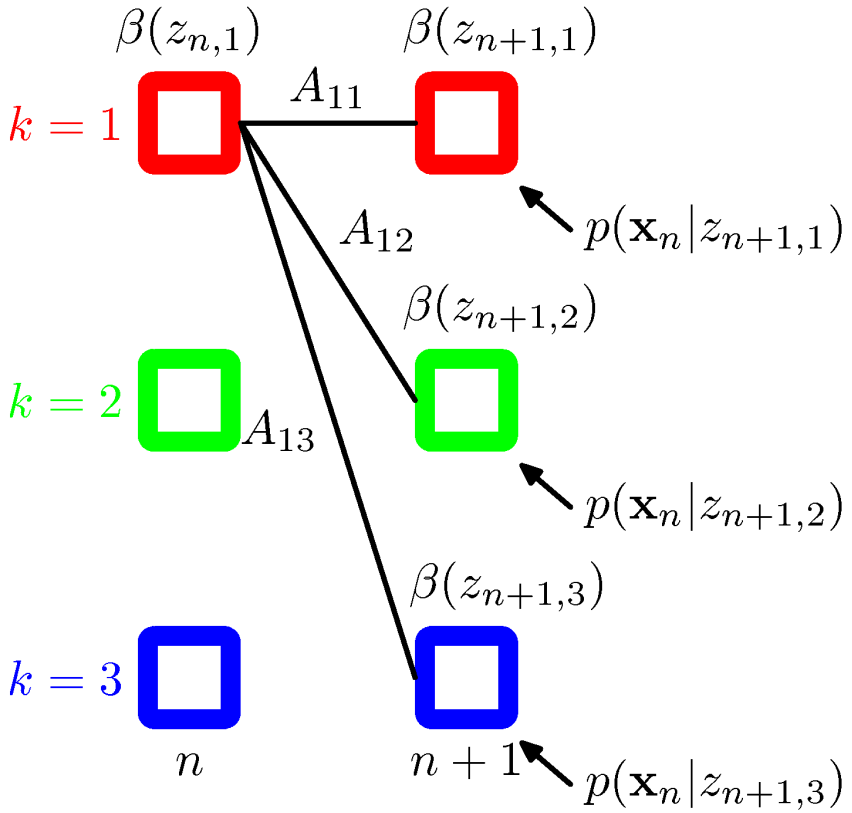
Forward-backward algorithm

- This leads to the recursion

$$\beta(\mathbf{z}_n) = \sum_{\mathbf{z}_{n+1}} \beta(\mathbf{z}_{n+1}) p(\mathbf{x}_{n+1} \mid \mathbf{z}_{n+1}) p(\mathbf{z}_{n+1} \mid \mathbf{z}_n)$$

- Initial condition

$$\beta(\mathbf{z}_N) = 1$$



Forward-backward algorithm

- Now, focus attention on

$$\xi(\mathbf{z}_{n-1}, \mathbf{z}_n) = p(\mathbf{z}_{n-1}, \mathbf{z}_n \mid \mathbf{X}) = \frac{p(\mathbf{X} \mid \mathbf{z}_{n-1}, \mathbf{z}_n) p(\mathbf{z}_{n-1}, \mathbf{z}_n)}{p(\mathbf{X})}$$

$$= \frac{p(\mathbf{x}_1, \ldots, \mathbf{x}_{n-1} \mid \mathbf{z}_{n-1}) p(\mathbf{x}_n \mid \mathbf{z}_n) p(\mathbf{x}_{n+1}, \ldots, \mathbf{x}_N \mid \mathbf{z}_n) p(\mathbf{z}_n \mid \mathbf{z}_{n-1}) p(\mathbf{z}_{n-1})}{p(\mathbf{X})}$$

$$= \frac{\alpha(\mathbf{z}_{n-1}) p(\mathbf{x}_n \mid \mathbf{z}_n) p(\mathbf{z}_n \mid \mathbf{z}_{n-1}) \beta(\mathbf{z}_n)}{p(\mathbf{X})}$$

- This is easily found using $\alpha(\mathbf{z}_n)$ and $\beta(\mathbf{z}_n)$

Forward-backward algorithm

- Summary of the EM algorithm
- Choose initial selection of $\theta^{\text{old}} = (\pi, \mathbf{A}, \phi)$
- Run forward α, and backward β recursion
- Evaluate $\gamma(\mathbf{z}_n)$ and $\xi(\mathbf{z}_{n-1}, \mathbf{z}_n)$
- Find θ^{new} using the M step

Forward-backward algorithm

- The predictive distribution

$$
\begin{aligned}
p(\mathbf{x}_{N+1} \mid \mathbf{X}) &= \sum_{\mathbf{z}_{N+1}} p(\mathbf{x}_{N+1}, \mathbf{z}_{N+1} \mid \mathbf{X}) = \sum_{\mathbf{z}_{N+1}} p(\mathbf{x}_{N+1} \mid \mathbf{z}_{N+1}) p(\mathbf{z}_{N+1} \mid \mathbf{X}) \\
&= \sum_{\mathbf{z}_{N+1}} p(\mathbf{x}_{N+1} \mid \mathbf{z}_{N+1}) \sum_{\mathbf{z}_N} p(\mathbf{z}_{N+1}, \mathbf{z}_N \mid \mathbf{X}) \\
&= \sum_{\mathbf{z}_{N+1}} p(\mathbf{x}_{N+1} \mid \mathbf{z}_{N+1}) \sum_{\mathbf{z}_N} p(\mathbf{z}_{N+1} \mid \mathbf{z}_N) p(\mathbf{z}_N \mid \mathbf{X}) \\
&= \sum_{\mathbf{z}_{N+1}} p(\mathbf{x}_{N+1} \mid \mathbf{z}_{N+1}) \sum_{\mathbf{z}_N} p(\mathbf{z}_{N+1} \mid \mathbf{z}_N) \frac{p(\mathbf{z}_N, \mathbf{X})}{p(\mathbf{X})} \\
&= \frac{1}{p(\mathbf{X})} \sum_{\mathbf{z}_{N+1}} p(\mathbf{x}_{N+1} \mid \mathbf{z}_{N+1}) \sum_{\mathbf{z}_N} p(\mathbf{z}_{N+1} \mid \mathbf{z}_N) \alpha(\mathbf{z}_N)
\end{aligned}
$$

Forward-backward algorithm

- The forward-backward algorithm as presented suffers from numerical problems
- Problem needs to be scaled

$$\hat{\alpha}(\mathbf{z}_n) = p(\mathbf{z}_n \mid \mathbf{x}_1, \ldots, \mathbf{x}_n) = \frac{\alpha(\mathbf{z}_n)}{p(\mathbf{x}_1, \ldots, \mathbf{x}_n)}$$

- Define

$$c_n = p(\mathbf{x}_n \mid \mathbf{x}_1, \ldots, \mathbf{x}_{n-1})$$

- Then

$$\alpha(\mathbf{z}_n) = p(\mathbf{z}_n \mid \mathbf{x}_1, \ldots, \mathbf{x}_n) p(\mathbf{x}_1, \ldots, \mathbf{x}_n) = \left(\prod_{m=1}^{n} c_m \right) \hat{\alpha}(\mathbf{z}_n)$$