STOR566: Introduction to Deep Learning

Lecture 9: Recurrent Neural Networks

Yao Li
UNC Chapel Hill

Sep 15, 2022

Materials are from *Deep Learning (UCLA)*

Representation for sentence/document

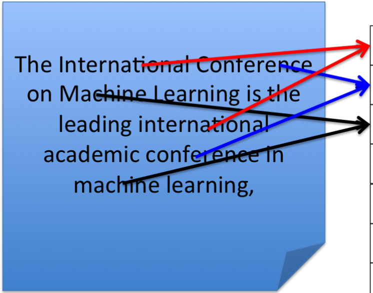
Bag of Words

- A classical way to represent NLP data
- Text $\rightarrow$ Vector/Matrices
- Problem: length not fixed

Bag of Words

- A classical way to represent NLP data
- Text $\rightarrow$ Vector/Matrices
- Problem: length not fixed
- Bag of words:

  Sentence $\rightarrow$ d-dimensional vector $\boldsymbol{x}$



(international)	2
(conference)	2
(machine)	2
(train)	0
(learning)	2
(leading)	1
(totoro)	0

d = number of potential words (very large)

Bag of Words: Processing Steps

- Step 1: Collect Data

It was the best of times,
it was the worst of times,
it was the age of wisdom,
it was the age of foolishness,

Bag of Words: Processing Steps

- Step 1: Collect Data

It was the best of times,
it was the worst of times,
it was the age of wisdom,
it was the age of foolishness,

- Step 2: Build the Vocabulary

{it, was, the, best, of, times, worst, age, wisdom, foolishness}

Bag of Words: Processing Steps

- Step 1: Collect Data

  It was the best of times,
  it was the worst of times,
  it was the age of wisdom,
  it was the age of foolishness,

- Step 2: Build the Vocabulary

  {it, was, the, best, of, times, worst, age, wisdom, foolishness}

- Step 3: Create Document/Sentence Vectors

	it	was	the	best	of	times	worst	age	wisdom	foolishness
d_1	1	1	1	1	1	1	0	0	0	0
d_2	1	1	1	0	1	1	1	0	0	0
d_3	1	1	1	0	1	0	0	1	1	0
d_4	1	1	1	0	1	0	0	1	0	1

Bag of n-gram

- Bag of n-gram features ($n = 2$):

> The International Conference on Machine Learning is the leading international academic conference in machine learning,

(international)	2
(conference)	2
(machine)	2
(train)	0
(learning)	2
(leading)	1
(totoro)	0

(international conference)	1
(machine learning)	2
(leading international)	1
(totoro tiger)	0
(tiger woods)	0
(international academic)	1
(academic conference)	1

TF-IDF

- Use the bag-of-word matrix or the normalized version (TF-IDF) for a dataset (denoted by D):

$$\text{tfidf}(\text{doc}, \text{word}, \text{D}) = \mathit{tf}(\text{doc}, \text{word}) \cdot \mathit{idf}(\text{word}, \text{D})$$

  - tf (doc, word): term frequency

    (word count in the document)/(total number of terms in the document)

  - idf (word, Dataset): inverse document frequency
    log((Number of documents)/(Number of documents with this word))

Data Matrix (document)

$$\text{tfidf}(\text{doc}, \text{word}, \text{D}) = \mathit{tf}(\text{doc}, \text{word}) \cdot \mathit{idf}(\text{word}, \text{D})$$

$$\text{TF} = (\text{word count in the doc})/(\text{total number of terms in the doc})$$

$$\text{IDF} = \log((\text{Number of docs})/(\text{Number of docs with this word}))$$

	angeles	los	new	post	times	york
d1	0	0	1	0	1	1
d2	0	0	1	1	0	1
d3	1	1	0	0	1	0

tf-idf

	angeles	los	new	post	times	york
d1	0	0	$\frac{1}{3} \times \log\left(\frac{3}{2}\right)$ =0.135	0	0.135	0.135
d2	0	0	0.135	$\frac{1}{3} \times \log(3)$ =0.366	0	0.135
d3	0.366	0.366	0	0	0.135	0

Bag of word + linear model

- Example: text classification (e.g., sentiment prediction, review score prediction)
- Linear model: $y \approx \text{sign}(\boldsymbol{w}^T \boldsymbol{x})$

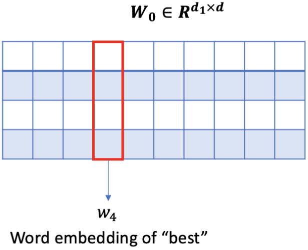
  (e.g., by linear SVM/logistic regression)
- w_i: the "contribution" of each word

Bag of word + Fully connected network

- $f(\boldsymbol{x}) = W_L \sigma(W_{L-1} \cdots \sigma(W_0 \boldsymbol{x}))$

Bag of word + Fully connected network

- $f(\boldsymbol{x}) = W_L\sigma(W_{L-1}\cdots\sigma(W_0\boldsymbol{x}))$



$\boldsymbol{x} \in \boldsymbol{R}^d$ | Bag of words

$\boldsymbol{x}$		
1	1	it
1	1	was
1	1	the
1	1	best
1	1	of
1	1	times
0	0	worst
0	0	age
0	0	wisdom
0	0	foolishness

It was the best of times

- W_0 is also called the word embedding matrix
- $\boldsymbol{w}_i$: d_1 dimensional representation of i-th word
- $W_0\boldsymbol{x} = x_1\boldsymbol{w}_1 + x_2\boldsymbol{w}_2 + \cdots + x_d\boldsymbol{w}_d$

  is a linear combination of these vectors

Recurrent Neural Network

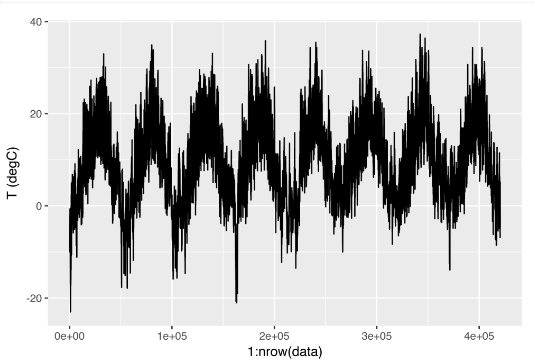
Time Series/Sequence Data

- Input: $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_T\}$
  - Each $\boldsymbol{x}_t$ is the feature at time step t
  - Each $\boldsymbol{x}_t$ can be a d-dimensional vector
- Output: $\{y_1, y_2, \cdots, y_T\}$
  - Each y_t is the output at step t
  - Multi-class output or Regression output:

$$y_t \in \{1, 2, \cdots, L\} \quad \text{or} \quad y_t \in \mathbb{R}$$

Example: Time Series Prediction

- Climate Data:
  - $\boldsymbol{x}_t$: temperature at time t
  - y_t: temperature (or temperature change) at time $t+1$
- Stock Price: Predicting stock price



Example: Language Modeling

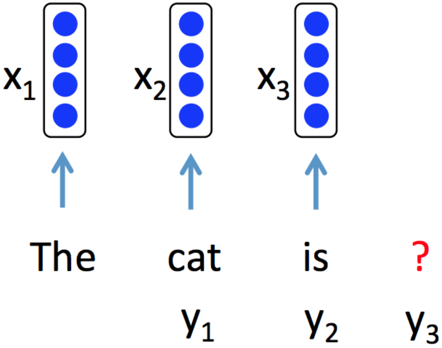
The cat is ?

Example: Language Modeling

The cat is ?

- $\boldsymbol{x}_t$: one-hot encoding to represent the word at step t $([0, \ldots, 0, 1, 0, \ldots, 0])$
- y_t: the next word

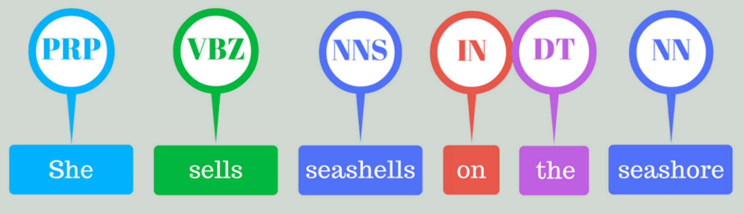
$$y_t \in \{1, \cdots, V\} \quad \text{V: Vocabulary size}$$



Example: POS Tagging

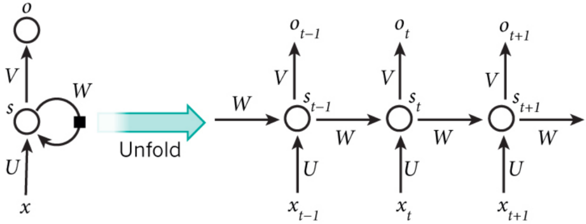
- Part of Speech Tagging:

  Labeling words with their Part-Of-Speech (Noun, Verb, Adjective, $\cdots$)
- $\boldsymbol{x}_t$: a vector to represent the word at step t
- y_t: label of word t



picture from `https://medium.com/analytics-vidhya/pos-tagging-using-conditional-random-fields-92077e5eaa31`

Recurrent Neural Network (RNN)



- $\boldsymbol{x}_t$: t-th input
- $\boldsymbol{s}_t$: hidden state at time t ("memory" of the network)

$$\boldsymbol{s}_t = f(U\boldsymbol{x}_t + W\boldsymbol{s}_{t-1})$$

  W: transition matrix, U : word embedding matrix
  $\boldsymbol{s}_0$ usually set to be 0, f: activation function
- Predicted output at time t:

$$o_t = \arg\max_i (V\boldsymbol{s}_t)_i$$

Recurrent Neural Network (RNN)

- Training: Find U, W, V to minimize empirical loss:
- Loss of a sequence:

$$\sum_{t=1}^{T} \text{loss}(V\boldsymbol{s}_t, y_t)$$

($\boldsymbol{s}_t$ is a function of U, W, V)

Recurrent Neural Network (RNN)

- Training: Find U, W, V to minimize empirical loss:
- Loss of a sequence:

$$\sum_{t=1}^{T} \text{loss}(V\boldsymbol{s}_t, y_t)$$

  ($\boldsymbol{s}_t$ is a function of U, W, V)
- Loss on the whole dataset:

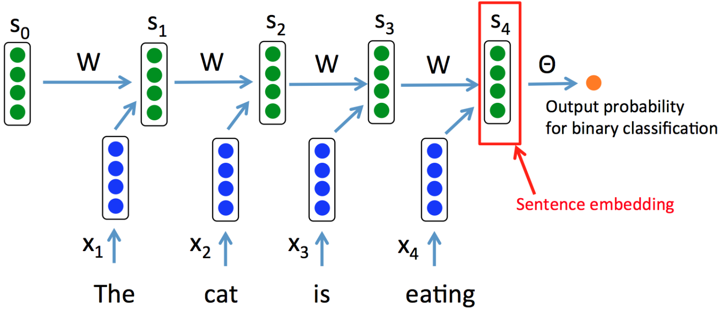
  Average loss over all sequences
- Solved by SGD/Adam

RNN: Text Classification

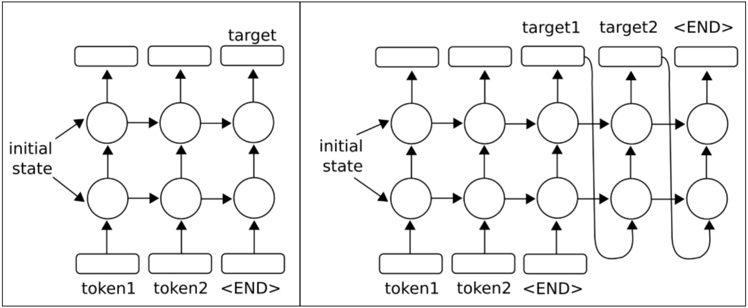
- Not necessary to output at each step
- Text Classification:

$$\text{sentence} \rightarrow \text{category}$$

  Output only at the final step
- Model: add a fully connected network to the final embedding



Multi-layer RNN

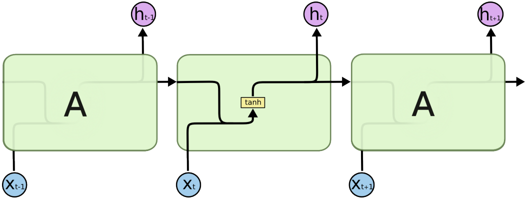


(Figure from
`https://subscription.packtpub.com/book/big_data_and_business_intelligence`)

Problems of Classical RNN

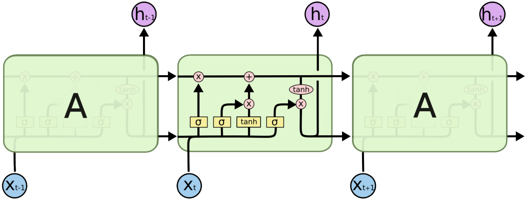
- Hard to capture long-term dependencies
- Hard to solve (vanishing gradient problem)
- Solution:
  - LSTM (Long Short Term Memory networks)
  - GRU (Gated Recurrent Unit)
  - ...

- RNN:

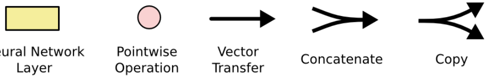


ht-1
ht
ht+1
A
tanh
A
Xt-1
Xt
Xt+1

- LSTM:



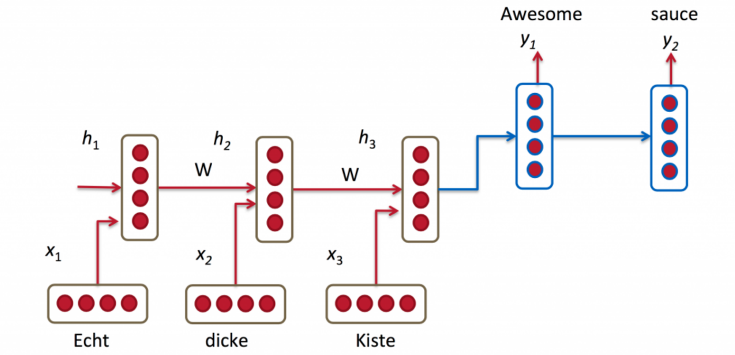
ht-1
ht
ht+1
A
tanh
σ
σ
tanh
σ
A
Xt-1
Xt
Xt+1



Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy

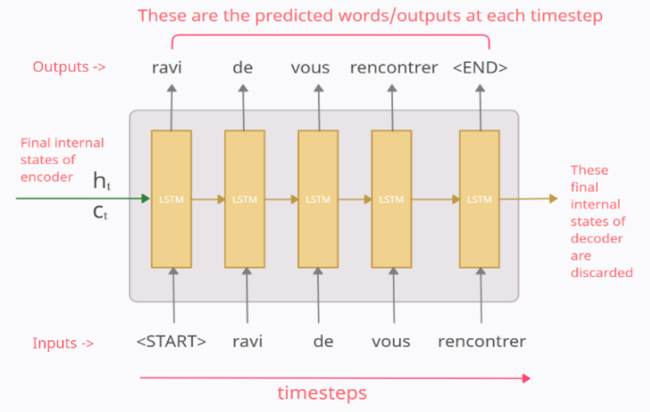
Neural Machine Translation (NMT)

- Out the translated sentence from an input sentence
- Training data: a set of input-output pairs (supervised setting)
- Encoder-decoder approach:
  - Encoder: Use (RNN/LSTM) to encode the input sentence input a latent vector
  - Decoder: Use (RNN/LSTM) to generate a sentence based on the latent vector



RNN: Neural Machine Translation

- Start input of the decoder?
- When to stop?



picture from https://medium.com/analytics-vidhya/encoder-decoder-seq2seq-models-clearly-explained-c34186fbf49b

Problems

- Only the last hidden state is used in decoding.
- Do not work well on long sequences.
- Solution:
    - Attention Mechanism:

      How about if we give a vector representation from every encoder step to the decoder model?

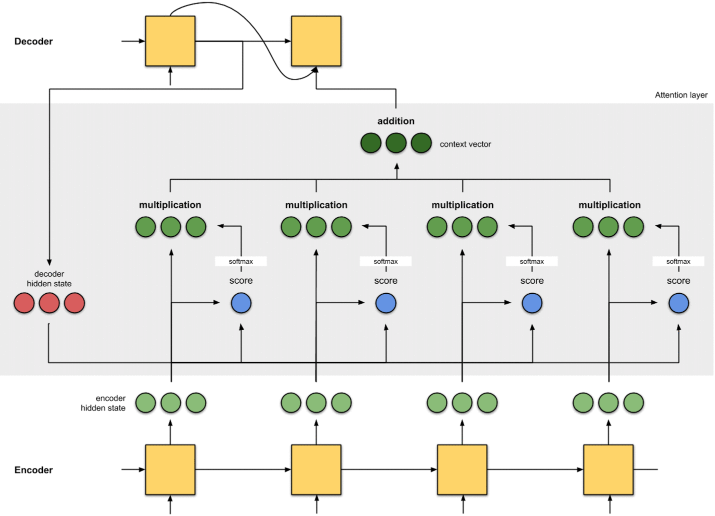
Attention in NMT

- Usually, each output word is only related to a subset of input words (e.g., for machine translation)
- Let $\boldsymbol{u}$ be the current decoder hidden state
  $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n$ be the hidden sate for each input word
- Compute the weight of each state by

$$\boldsymbol{p} = \text{Softmax}(\boldsymbol{u}^T \boldsymbol{v}_1, \ldots, \boldsymbol{u}^T \boldsymbol{v}_n)$$

- Compute the context vector by $V\boldsymbol{p} = p_1 \boldsymbol{v}_1 + \cdots + p_n \boldsymbol{v}_n$

Attention in NMT



(Figure from `https://towardsdatascience.com/neural-machine-translation-nmt-with-attention-mechanism`)

Conclusions

- Bag of words
- RNN
- Attention in NMT

Questions?