Day 11: Recurrent Neural Networks

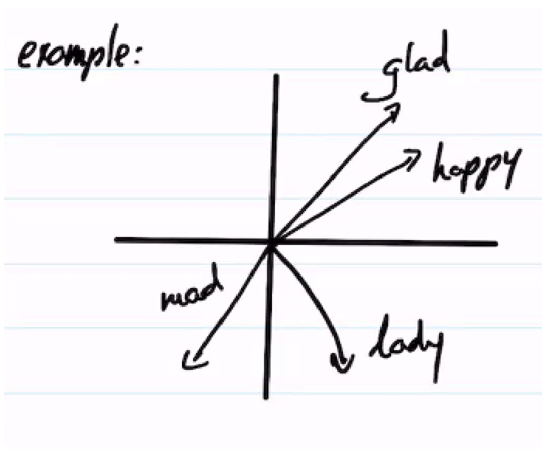
Word Embeddings

- n-gram models
- Bag of Words
    - Ignores word order
    - No information on relationships between words
- TF-IDF

Can we represent words with vectors such that words with similar meanings are closer together?

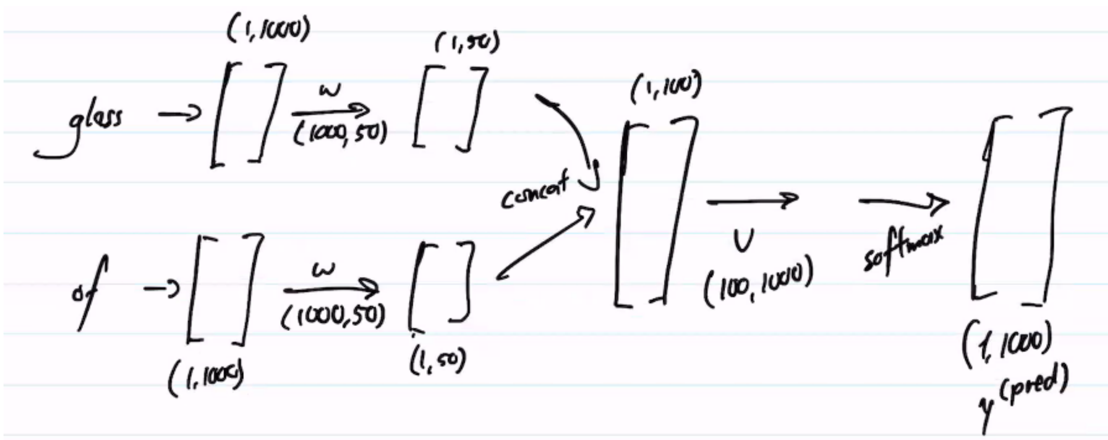
* (think of these like face descriptors)

- Example: “That lady was happy”; “That lady was glad”, “That lady was mad”
    - Ideally, the vector forms of “happy” and “glad” would be close, while “mad” would be further away.



- Word embedding - representation of words as vectors in a continuous space (hopefully such that words with similar meaning are close together)
    - We want to design/learn an embedding that captures common words relations (e.g: $\vec{w}_{doctor} + \vec{w}_{teeth} - \vec{w}_{body} = \vec{w}_{dentist}$)
    - Start with a one-hot encoding (vectors with all zeros, except one 1 where the index of the word/part is present in the vocabulary); e.g: vocab “the”, “black”, “cat” → $\vec{x}_{the} = [1, 0, 0], \vec{x}_{black} = [0, 1, 0], \vec{x}_{cat} = [0, 0, 1]$
        - We want to find a function $f_{embed}(\vec{x}_{cat}) = \vec{w}_{cat}$ where $\vec{x}_{cat}$ has shape `N_vocab` and $\vec{w}_{cat}$ has shape `D_embed`
    - GLoVE-50 is a f_{embed} with `D_embed==50`

- all words in its vocabulary are represented with a 50-D vector
- **Method #1: One-hot Encoding**
  - Use one-hot encoding itself
- **Method #2: Learning Embedding via Language Modeling (Bengro et al. 2003)**
  - Example: “glass of ?”, where “?” is 1 of 1000 possible words `N_vocab=1000`
  - Go through a series of dense layers to transform the data across dimensions, along with concatenation with the next word (which has also been passed through multi-layer perceptrons), to ultimately convert to a softmax layer and obtain a set of predictions



  - w ~ learned embedding weights
  - $L_{cross-entropy}(y^{(pred)}, y^{(true)})$
  - Applications
    - Autocompletion
    - Language ID (from grammatical structure)
    - Language translation
- **Method 3: word2vec → skip-gram**
  - Predict surrounding words (e.g: “? ? orange ? ?”
  - We want the $P(\text{"the tall, tree was"} \mid \text{"orange"})$
- **Method 4: word2vec → continuous bag of words**
  - E.g.: “glass of ? orange juice”
  - “glass” → $\vec{w} = []$, `(1, 50)`
  - “of” → $\vec{w} = []$, `(1, 50)`
  - “orange” → $\vec{w} = []$, `(1, 50)`
  - “juice” → $\vec{w} = []$, `(1, 50)`
  - These values are summed, transformed, and put through softmax for final probabilities

- **Method 5: GLoVE (Penmington et al. 2014)**

  $\sum_{ij} f(x_{ij})(\vec{w}_i * \vec{w}_j + b_i + b_j - log(X_{ij}))^2$

  1. Collect co-occurrence statistics X_{ij}
     a. Needs window size
     b. Decay based on offset: $\frac{1}{offset}$
     c. Example: “I am a black cat.”

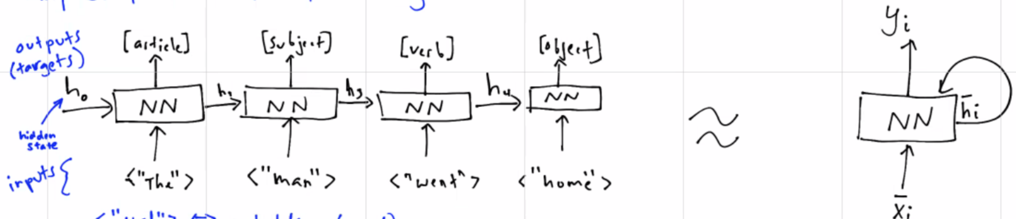
	a	am	black	cat	I
a					1/2
am					1
black					
cat					
I					

  2. We want: $\vec{w}_i \cdot \vec{w}_j + b_i + b_j = logX_{ij}$
  3. $f(X_{ij})$ to downweight extremely common pairs
     - Scales to huge corpera
     - Effective with small corpera
     - Good matches with more words

Recurrent Neural Networks

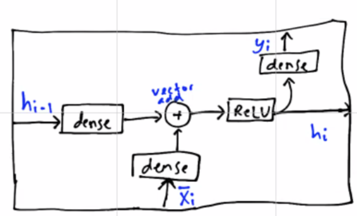
Concept: For sequential tasks, can we reuse the same neural network?

Example: part of speech labeling

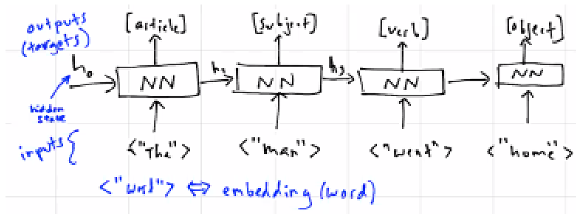


<"word"> ⇔ embedding(word)

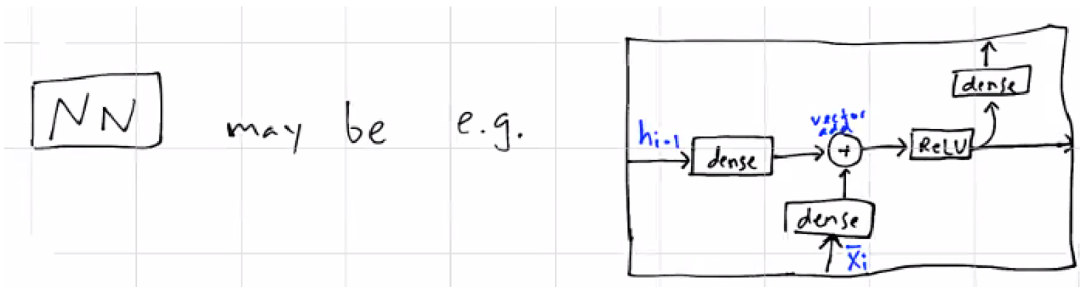
$\boxed{NN}$ may be e.g.



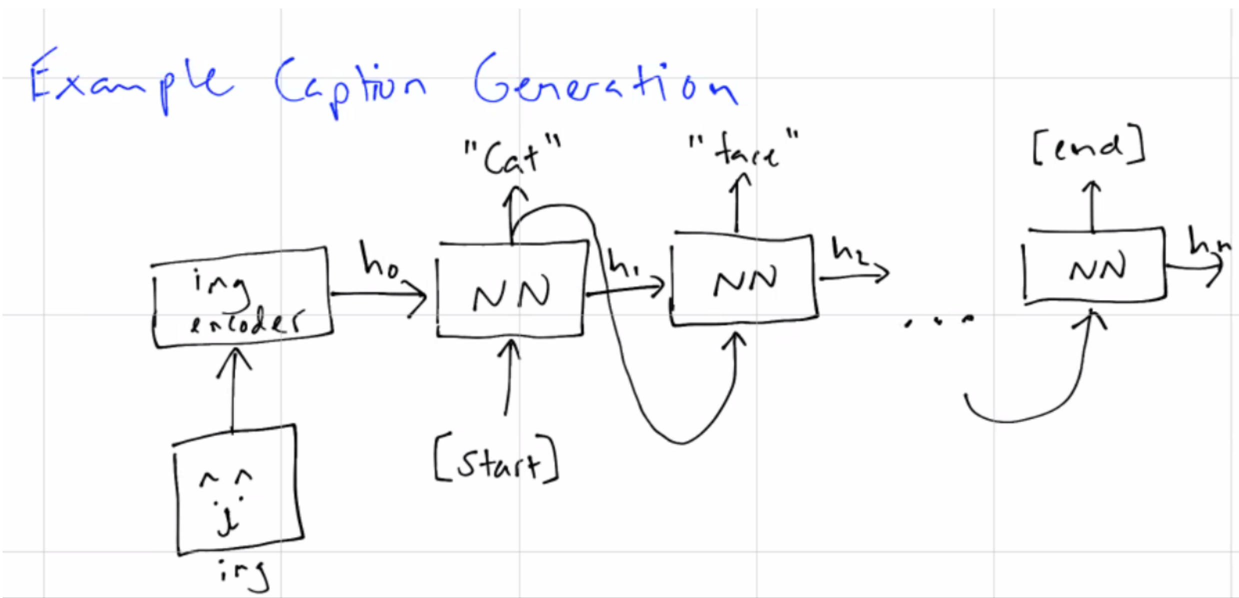
- Concept: For sequential tasks, we can reuse the same neural network
- Example: part of speech labeling



- Network may be a series of transformations and layers in its own right



- Example Caption Generation



- RNN's can model:
  - one-to-many
  - many-to-one
  - many-to-many
- **The XOR Problem (RNN motivating example)**

- Example inputs: $\vec{x}_1 = [0, 0, 0, 1, 1, 0]; \vec{x}_2 = [0, 1, 1, 0, 1, 0]$
- Example outputs: $y_{1,true} = 1(even); y_{2,true} = 0(odd)$

- Consider fram