intel Nervana™ AI Academy

TEXT / WORD VECTORS

MOTIVATION

- Have shown how to use Neural Networks with structured numerical data
- Images can be upsampled / downsampled to be a certain size
- Image values are numbers (greyscale, RGB)
- But how do we work with text?
- Issue 1: How to deal with pieces of text (sequences of words that vary in length)?
- Issue 2: How to convert words into something numerical?

ISSUE: VARIABLE LENGTH SEQUENCES OF WORDS

- With images, we forced them into a specific input dimension
- Not obvious how to do this with text
- We will use a new structure of network called a “Recurrent Neural Network” which will be discussed next lecture

TOKENIZATION

- Need to convert word into something numerical
- First approach: Tokenization
- Treat as a categorical variable with huge number of categories (one hot encoding)
- Deal with some details around casing, punctuation, etc.

“The cat in the hat.”

```
[‘the’,’cat’,’in’,’the’,’hat’,’<EOS>’]
```

TOKENIZATION

- Use tokens to build a vocabulary
- Vocabulary is a one-to-one mapping from index # to a token
- Usually represented by a list and a dictionary

index → word

```
[
  ‘<EOS>’,
  ‘the’,
  ‘cat’,
  ‘in’,
  ‘hat’,
  ‘.’
]
```

index → word

```
{
  ‘<EOS>’: 0,
  ‘the’:      1,
  ‘cat’:      2,
  ‘in’:       3,
  ‘hat’:      4,
  ‘.’:    5
}
```

ISSUES WITH TOKENIZATION

- Tokenization loses a lot of information about words:
  - Part of speech
  - Synonymy (distinct words with same or similar meaning)
  - Polysemy (single word with multiple meanings)
  - General context in which word is likely to appear
    (e.g. “unemployment” and “inflation”) are both about economics
- Increasing vocabulary size is difficult (would require re-training the model)
- Vector length is huge -> large number of weights
- Yet information in vector is very sparse

WORD VECTORS

- Goal: represent a word by an m-dimensional vector (for medium-sized m, say, m=300)
- Have “similar” words be represented by “nearby” vectors in this m-dimensional space
- Words in a particular domain (economics, science, sports) could be closer to one another than words in other domains.
- Could help with synonymy
  - e.g. “big” and ”large” have nearby vectors
- Could help with polysemy
  - “Java” and ”Indonesia” could be close in some dimensions
  - “Java” and “Python” are close in other dimensions

WORD VECTORS

- Vectors would be shorter length and information-dense, rather than very long and information-sparse
- Would require fewer weights and parameters
- Fortunately, there are existing mappings which can be downloaded and used
- These were trained on big corpora for a long time
- Let’s understand how they were developed and trained

WHAT MAKES TWO WORDS SIMILAR?

- Idea: similar words occur in similar contexts
- For a given word, look at the words in a “window” around it
- Consider trying to predict a word given the context
- This is exactly the CBOW (continuous bag of words) model

“We hold these truths to be self-evident, that all men are created equal”

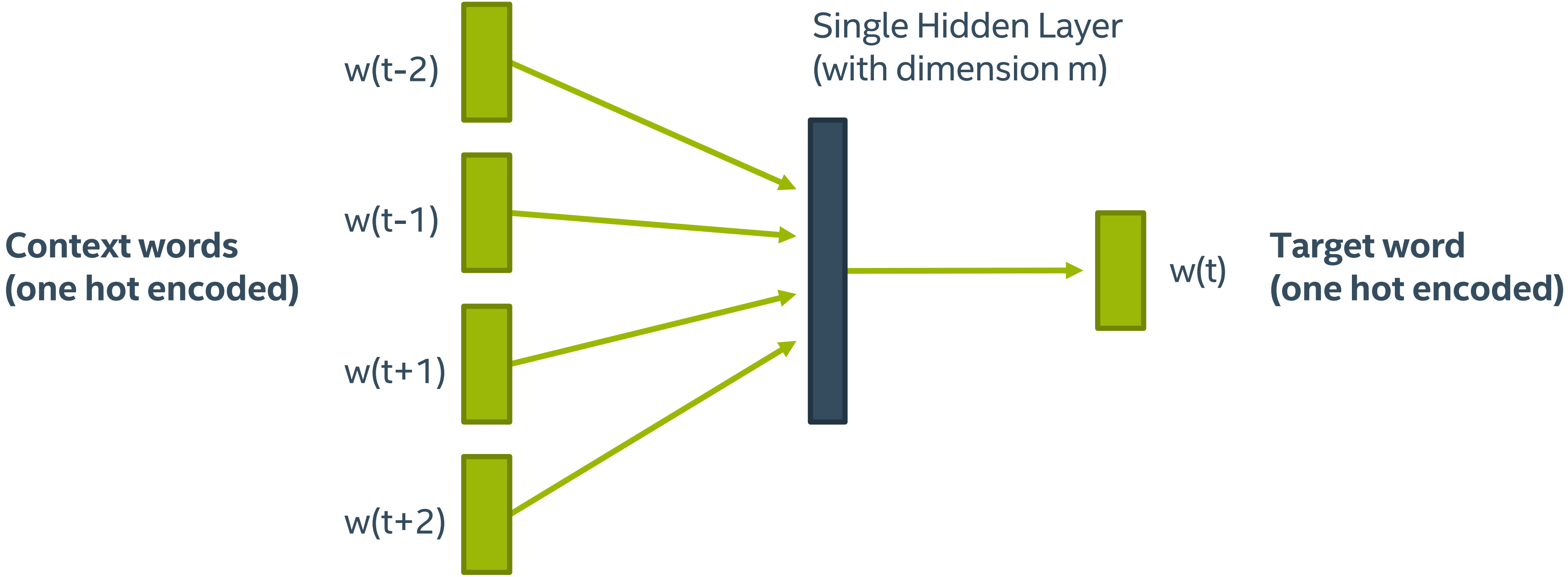
```
([‘truths’, ‘to’, ‘be’, ‘that’, ’all’, ‘men’], ‘self-evident’)
```

context

target word

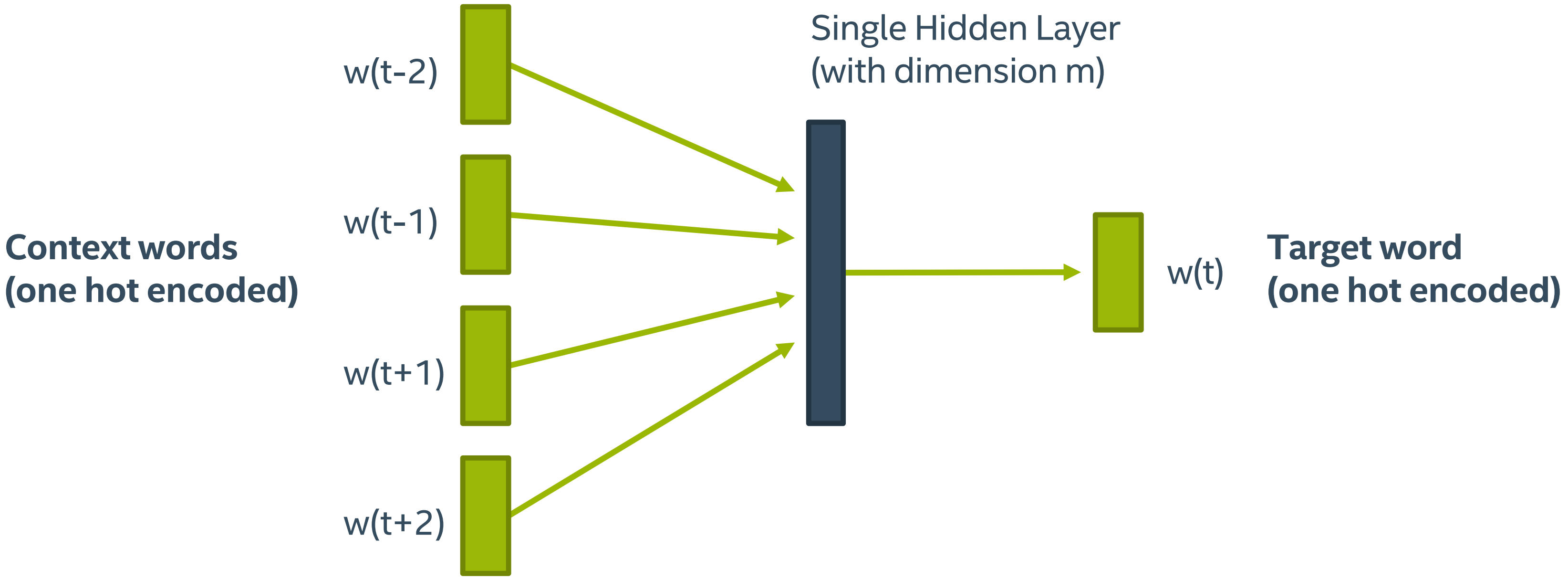
CBOW MODEL

Train a neural network on a large corpus of data.



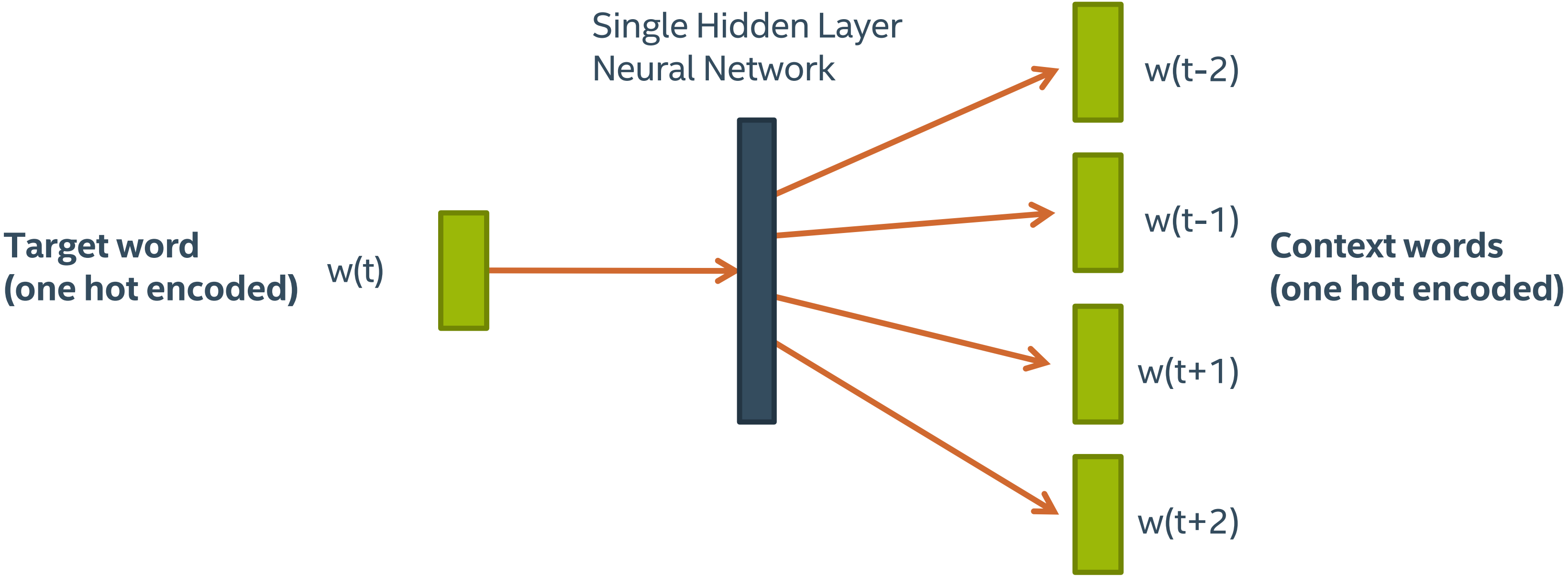
CBOW MODEL

Once the network is trained, weights -> word vectors.



SKIP-GRAM MODEL

Same idea, except we predict the context from the target.



WORD2VEC

- *Distributed Representations of Words and Phrases and Their Compositionality*—Mikolov et al.
- Uses a Skip-gram model to train on a large corpus
- Lots of details to make it work better
  - Aggregation of multi-word phrases (e.g. Boston Globe)
  - Subsampling (i.e. oversample less common words)
  - Negative Sampling (give network examples of wrong words)

GLOVE

- Global Vectors for Word Representation (GloVe)
- Use co-occurrence matrix with neighboring words to determine similarity

$$J = \frac{1}{2} \sum_{i,j=1}^{W} f(P_{ij})\left(u_i^T v_j - \log(P_{ij})\right)^2$$

f → **frequency of a word, with a maximum cap**

P_ij → **probability words i and j occur together**

GLOVE

- GloVe is publicly available
- Developed at Stanford: https://nlp.stanford.edu/projects/glove/
- Trained on huge corpora

Nervana™ AI Academy