Week 3: Vectors in context

Text Analytics and Natural Language Processing
Instructor: Benjamin Batorsky

Note on what to expect on midterm

- Aiming for understanding intuition and caveats to different designs/methods
- Short answer/applications will be focused on coding, not equations

Example question:

If you want to test the “performance” of a Latent Dirichlet Allocation (LDA) model, what is the preferred method (i.e. the one sklearn uses)?

Is this value directly comparable to non-negative matrix factorization’s reconstruction error?

Note on what to expect on midterm

- Aiming for understanding intuition and caveats to different designs/methods
- Short answer/applications will be focused on coding, not equations

Example question:

If you want to test the “performance” of a Latent Dirichlet Allocation (LDA) model, what is the preferred method (i.e. the one sklearn uses)?

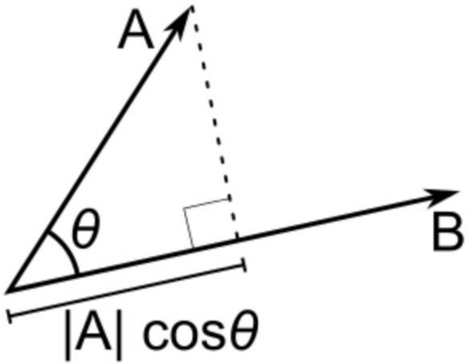
Evidence Lower BOund (ELBO)

Is this value directly comparable to non-negative matrix factorization’s reconstruction error?

No.

Cosine similarity

- Each data point is a vector in multi-dimensional space
  - TF = “Vocabulary space”
  - NMF = “Topic space”
- Cosine similarity
  - Compare the orientation of two vectors, regardless of magnitude (e.g. total word count)
  - Scale from 0 to 1, completely dissimilar to completely identical
- You CANNOT compare two vectors from two different spaces
- You CAN compare cosine similarity measures



http://blog.christianperone.com/2013/09/machine-learning-cosine-similarity-for-vector-space-models-part-iii/

Final project outline (Assignment 4)

- Will be posted on Canvas this week
- Due (**latest!**): Week 6, Monday, 7/27, 6:30pm ET
- Earlier submission = earlier feedback
  - NOTE: We are covering models these coming weeks and scoping in week 5
- Must include
  - Names of project members (no more than 3!)
  - Overview of sections
- Can include
  - Any questions you have (besides is this do-able)

Outline must detail your strategy around the following sections of the project

- A clear research question or problem to address
- A description of the dataset and justification of why it is appropriate to solve the problem
- Exploratory analyses of the dataset
- A description of the methodology to be applied and justification of its use
- Deployment strategy
  - (optional, if appropriate)

Assignment 2 review

- Will release instructor version tomorrow
- TF-IDF “most frequent” words
    - .sum() makes sense here, but is there a better measure?
- Intra-class and inter-class similarity
    - How similar are negative reviews to each other on average?
    - Is that value lower than the average similarity between negative and positive?
    - Looking for interpretation, provides info for next step
- Prediction
    - LDA/NMF don’t seem to outperform TF or TF-IDF
        - Why might that be?
- Remember: This is a real-world dataset, things don’t always work cleanly
    - Also: SUBSET of actual dataset, 50k in real set: https://ai.stanford.edu/~amaas/data/sentiment/

Review: Tokenization

- Token: Useful semantic unit
- "Useful" = informative
- Non-standard tokens
  - URLs
  - Entities
- (Potentially) non-informative tokens
  - Stop words
  - Rare, overly common words
  - Multiple forms/tenses of individual words
- Tokenization strategies
  - Input = Raw text
  - Output = Tokenized text

In June 2020, I took a course at Harvard Extension School in Cambridge.

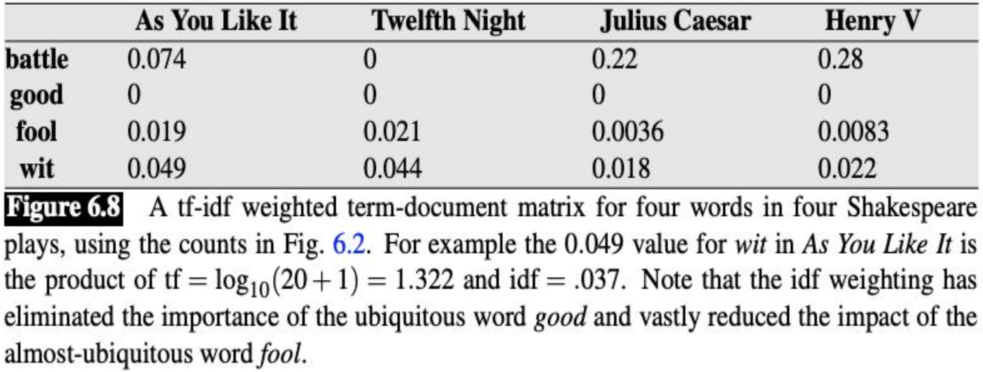
June 2020 took course Harvard Extension School Cambridge

June 2020 took course Harvard Extension School Cambridge .

Review: From tokens to vectors

- Create informative representations
  - Input: Tokenized text/tokenization strategy
  - Output: Word and document-level vectors
- Count vectors
  - Raw count (term frequencies)
  - IDF Weighted count (TF-IDF)
- Topic models
  - Documents = combination of topics/archetypes = combination of words
  - LDA: Probabilistic framework
  - NMF: Matrix factorization with non-negativity constraint

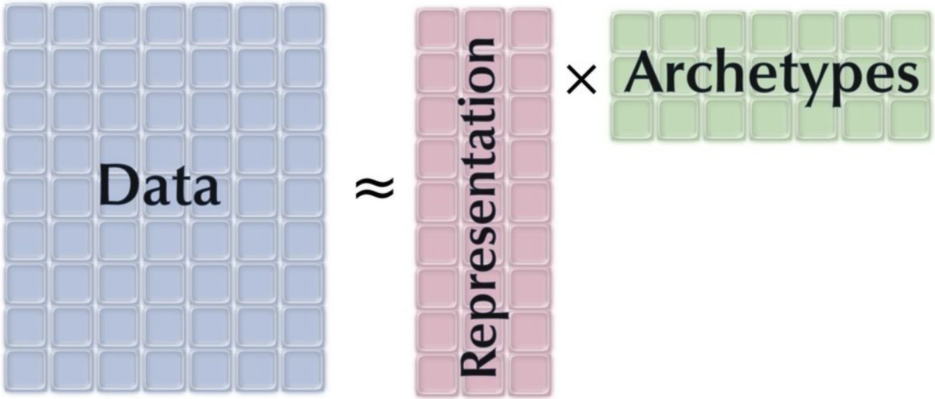
If the goal is to created an informative representation, what is the main thing we're missing?



	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

Figure 6.8 A tf-idf weighted term-document matrix for four words in four Shakespeare plays, using the counts in Fig. 6.2. For example the 0.049 value for *wit* in *As You Like It* is the product of tf $= \log_{10}(20+1) = 1.322$ and idf $= .037$. Note that the idf weighting has eliminated the importance of the ubiquitous word *good* and vastly reduced the impact of the almost-ubiquitous word *fool*.

https://web.stanford.edu/~jurafsky/slp3/6.pdf

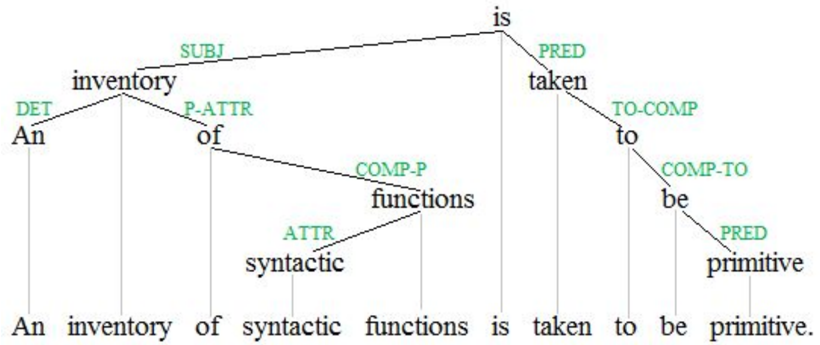


Leland McInnes: Bluffer's Guide to Matrix Factorization

Bag of words vs natural language

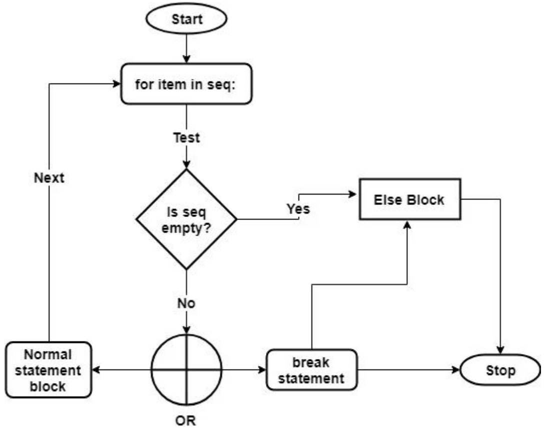
	the	movie	was	good	not	it	bad
“The movie was good”	1	1	1	1	0	0	0
“The movie was not bad, it was good”	1	1	1	1	1	1	1
“The movie was bad”	1	1	1	0	0	0	1

Review: Natural language inherently sequential



https://en.wikipedia.org/wiki/Dependency_grammar

“A language that has developed naturally in use (as contrasted with an artificial language or computer code).” (Oxford Dictionary definition)

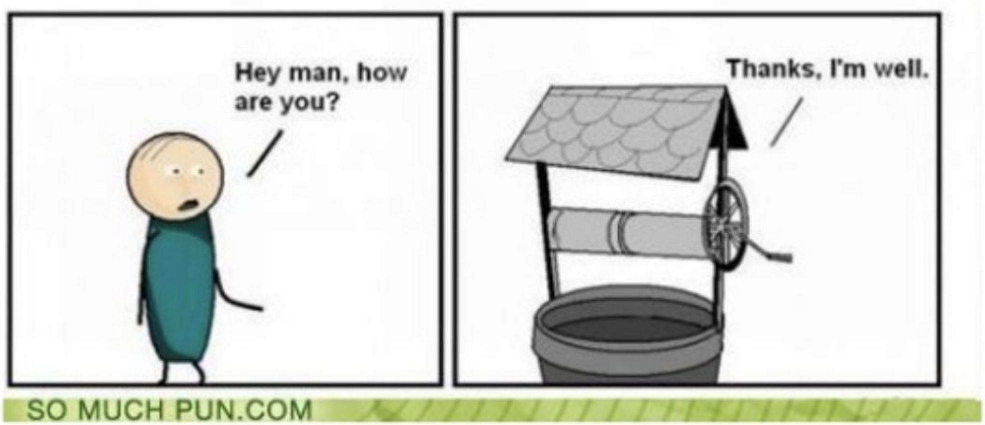


https://www.techbeamers.com/python-for-loop/

What do we lose with bag-of-word representations?

What do we lose with bag-of-word representations?

- Negation (as shown in notebook)
- Coreferences
  - “I saw the movie. It was bad”
- Homonyms
  - “Doing well” vs “wishing well”
- Entailment/Contradiction
  - “I liked the actors. I didn’t like the story”
- Directionality/Causation
  - “Dog bit man” vs “Man bit dog”



(Some) Methods for including context

- Adding context features
  - N-grams
  - Document-level info (e.g. length)
- Auto-regression models
  - Prediction at state x = info from state x + info from state x-1, etc
- Recurrent models
  - Learned “hidden state” passed to each state
- Word representations learned from context
  - Word2Vec, GloVe

“I liked this movie”

1-grams (Unigrams):

[“I”, “liked”, “this”, “movie”]

2-grams (Bigrams):

[“I liked”, “liked this”, “this movie”]

4-grams: “I liked this movie”

Issues with context features and autoregression

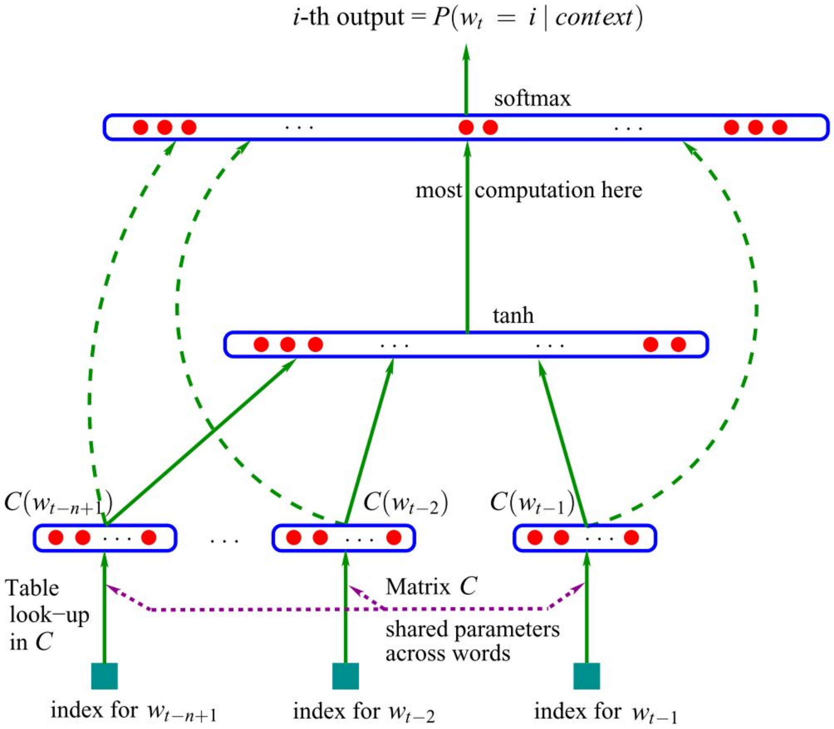
- Tradeoff between representation and complexity
  - Number of n-grams = number of words - (n-1)
  - Total possible n-grams = n(n+1)/2
  - For a 100-word document: 4,950
  - 1000-word: 499,500
- Extensive feature engineering = time consuming
  - Choosing the N for n-grams
  - Document-level features
  - How many steps back to include

Review: History of NLP

- 40s-50s: Machine translation era
- 60-70s: Shift towards semantic-driven processing
- 70s to 80s: Community expansion
- 90s-00s: Probabilistic/Statistical models
  - Also expansion of available data
- 2000s: Neural Language models
- 2008: Multi-task learning
- 2013: Word embeddings
- 2014: Expansion of Neural models
- 2015: Attention
- 2018 and beyond: Language model advancements

2000s: Advent of Neural Models for NLP

- Focused on language modelling task
  - Given context words, predict next word
  - “Do you want to go out for <mask>”
  - “I ate so much I am so <mask>”
  - “I’m so tired, I think I’ll take a <mask>”
- Apply Neural Net architectures to language modelling task
  - Neural Net: Model that connects inputs to outputs through sets of computations
- Bengio et. al. (2001) A Neural Probabilistic Language Model
  - Context words fed into a matrix that “represents” the information
  - These representations then fed into a computation layer
  - Output: Prediction of the target word
    - “Full”: 70% likely, “tired” 20% likely, etc



A Neural Probabilistic Language Model
(https://papers.nips.cc/paper/1839-a-neural-probabilistic-language-model.pdf)

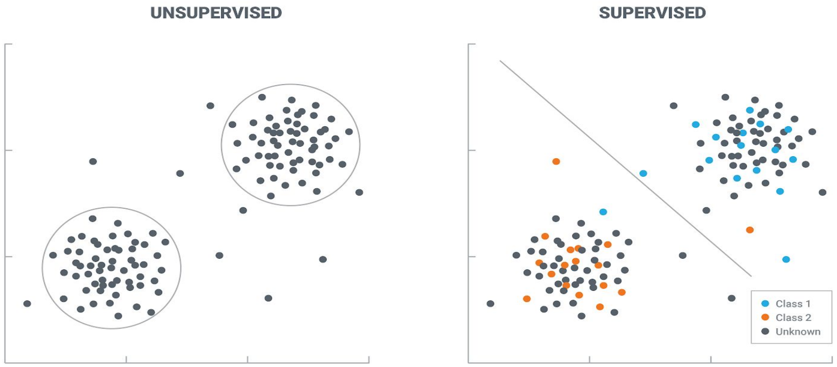
Review: Supervised vs unsupervised learning

Unsupervised learning

Unsupervised learning is a type of machine learning that looks for previously undetected patterns in a data set with no pre-existing labels and with a minimum (

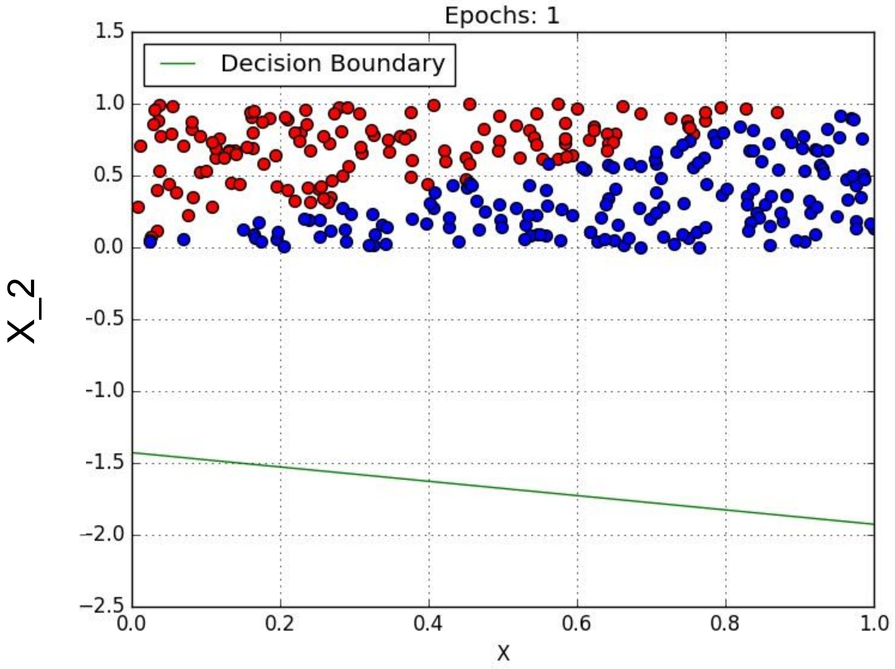
Supervised Learning

Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs.



https://lawtomated.com/supervised-vs-unsupervised-learning-which-is-better/

Review: Binary case - Finding a decision boundary

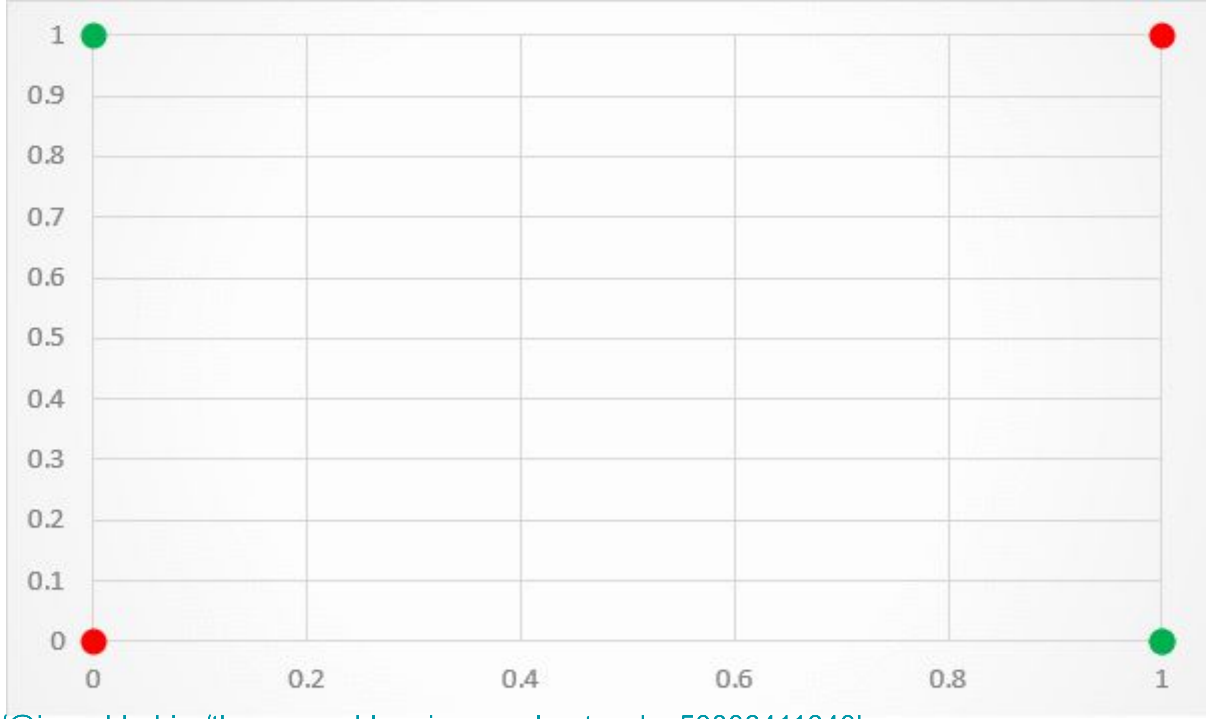


Blue = Positive
Red = Negative

https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148

How would you (linearly) separate these two classes?

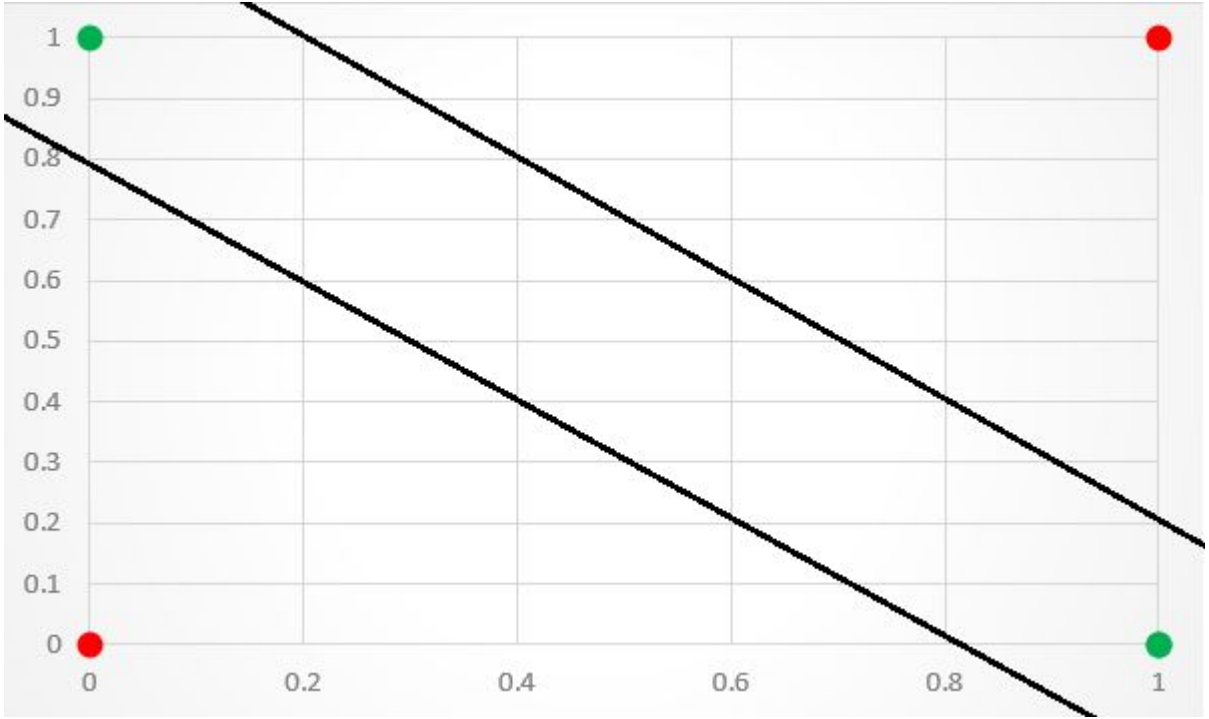
Green = positive class, red = negative class



https://medium.com/@jayeshbahire/the-xor-problem-in-neural-networks-50006411840b

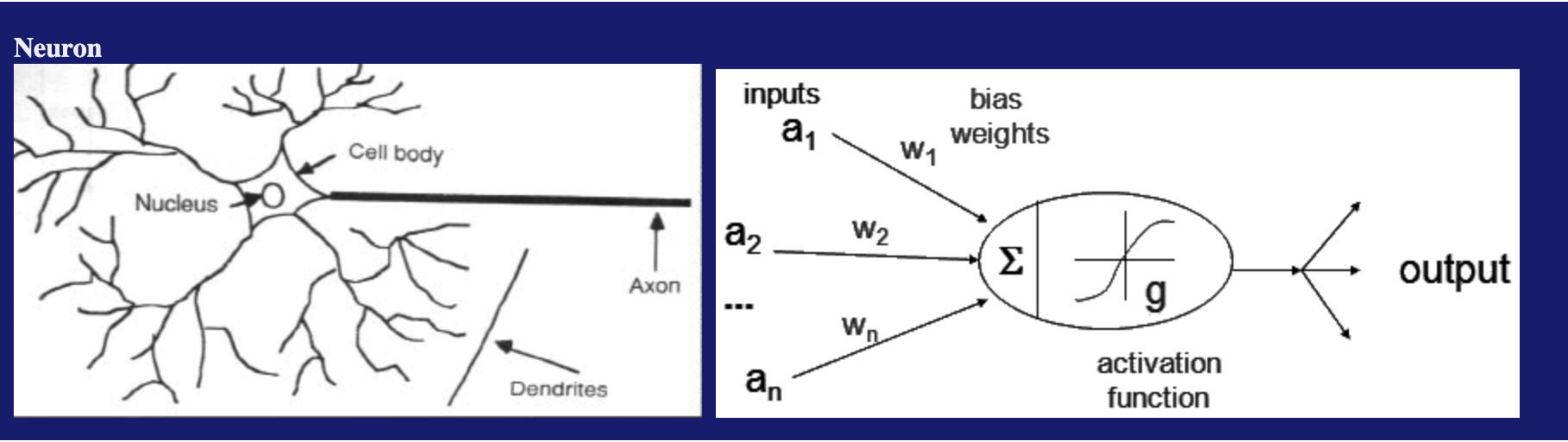
You need TWO lines; nonlinear function

Green = positive class, red = negative class



https://medium.com/@jayeshbahire/the-xor-problem-in-neural-networks-50006411840b

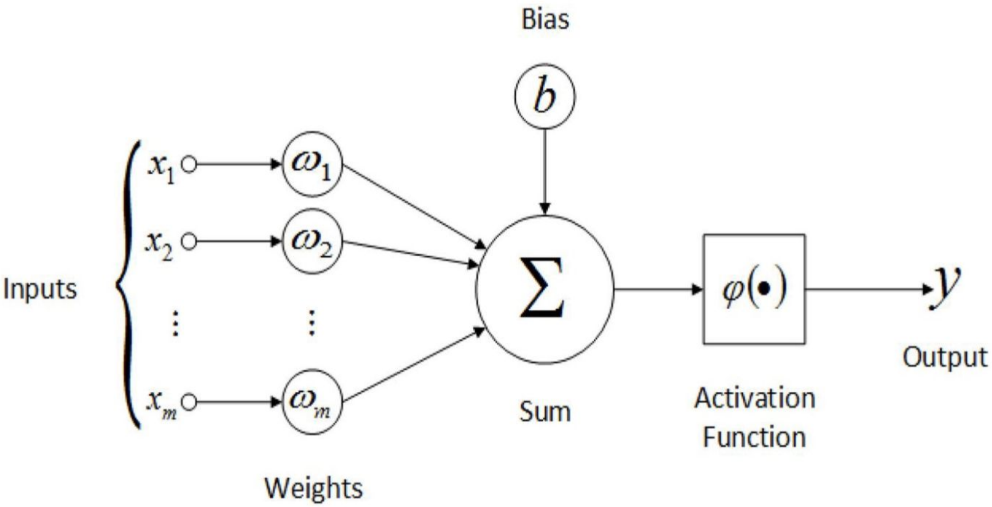
What is a neural network?



https://www.cs.iusb.edu/~danav/teach/c463/12_nn.html

Basic neural network design

Single node:



The Artificial Neuron

Loss computation:

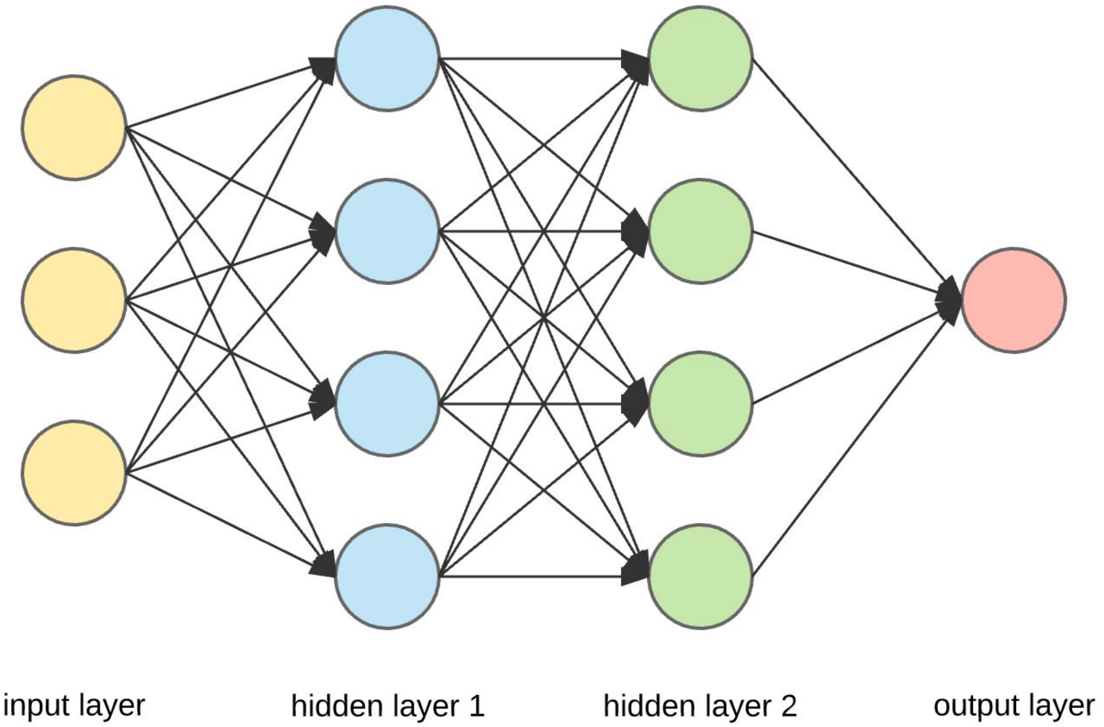
$$L_{\text{cross-entropy}}(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_i y_i \log(\hat{y}_i)$$

(binary case)

$$-(ylog(\hat{y}) + (1 - y)log(1 - \hat{y}))$$

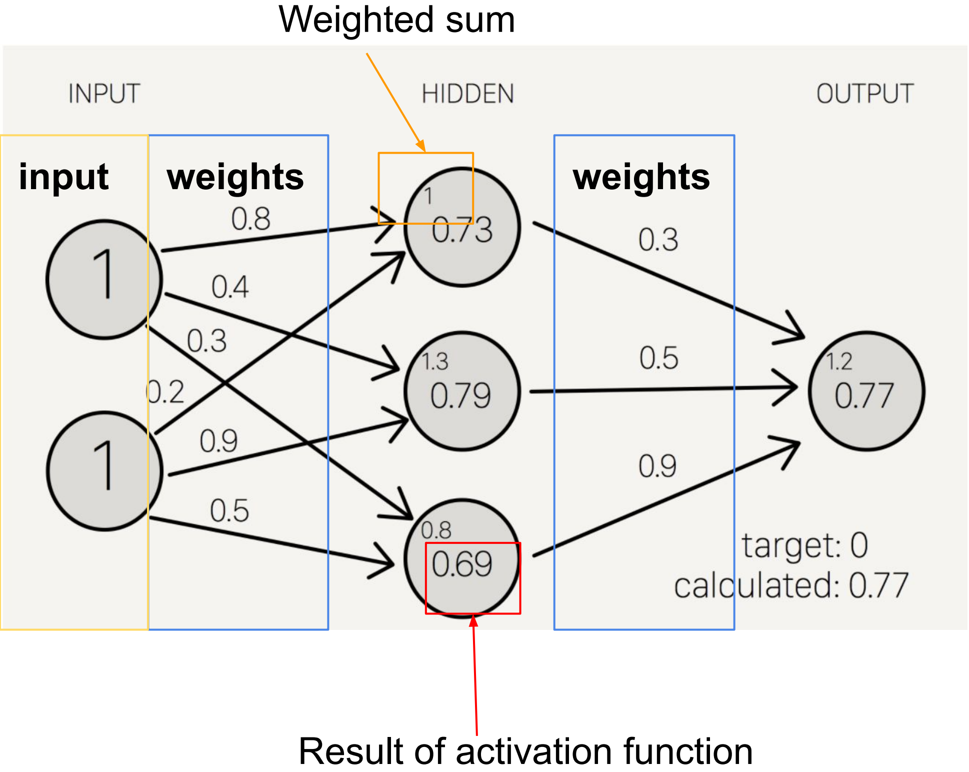
https://medium.com/@jayeshbahire/the-artificial-neural-networks-handbook-part-4-d2087d1f583e

Multi-layered “fully connected” network



https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6

Forward pass



https://stevenmiller888.github.io/mind-how-to-build-a-neural-network/

Take input, apply weights (randomly initialized)

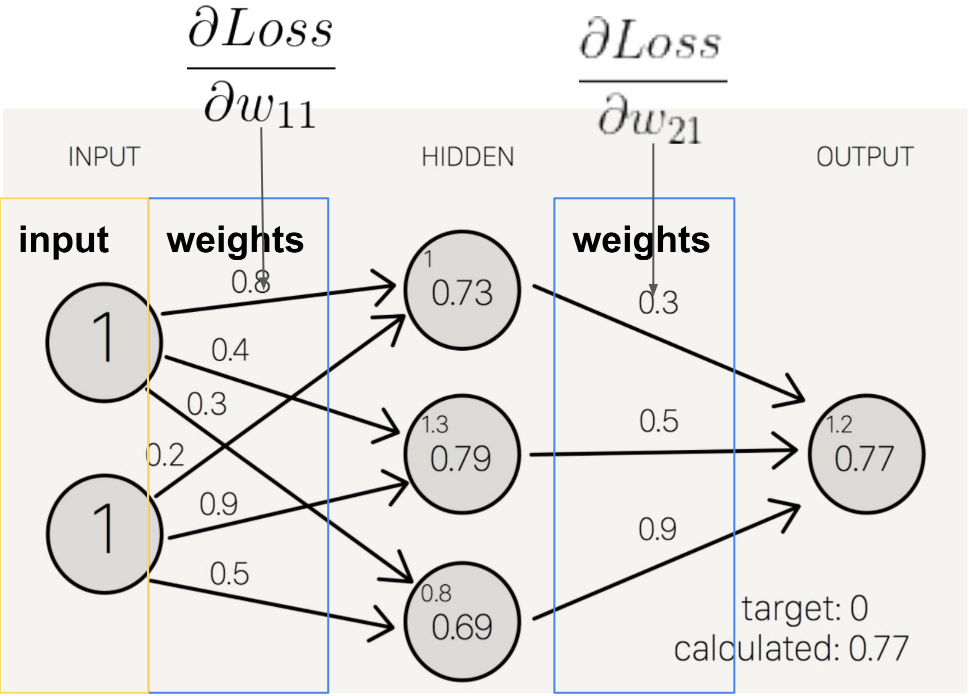
Weighted sum, apply activation get “hidden layer” representation

Each have weights, weighted sum + activation = prediction (calculated)

Loss here: (see previous slide) 0.64

How should we change the weights to reduce this loss?

Backward pass (back propagation)



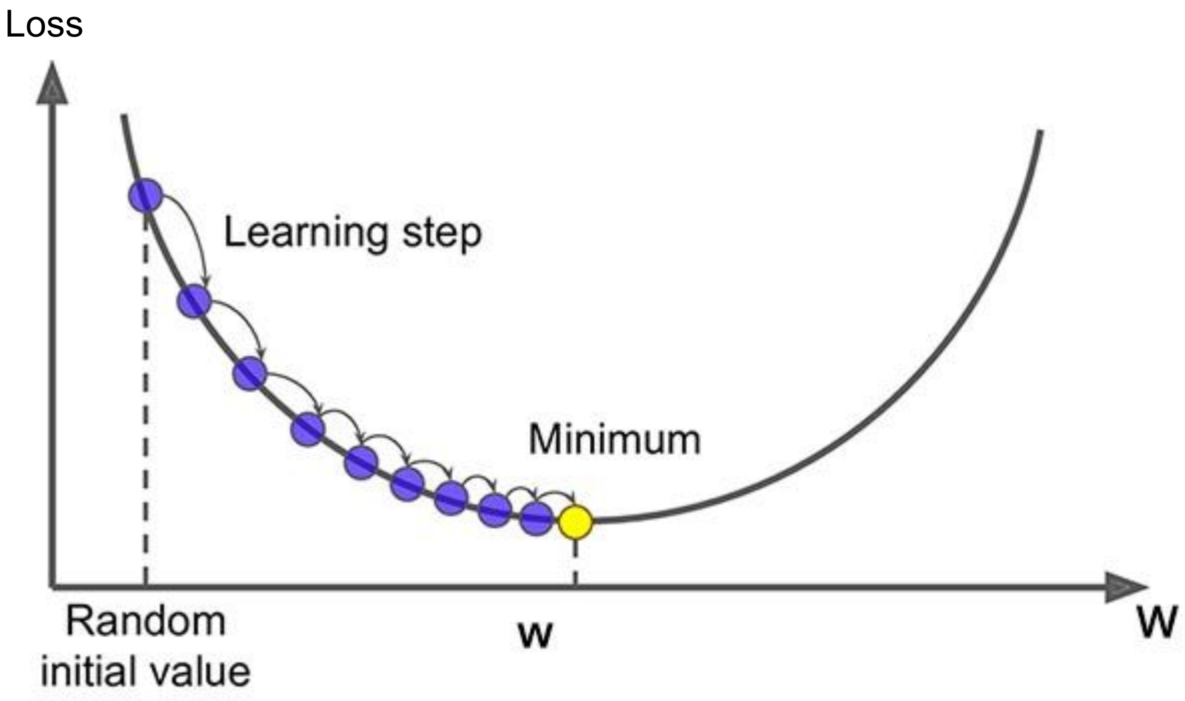
How should we change the weights to reduce this loss?

For each weight: Adjust based on the change in loss with respect to a change in weight

Want to move the weights in a direction that reduces loss

https://stevenmiller888.github.io/mind-how-to-build-a-neural-network/

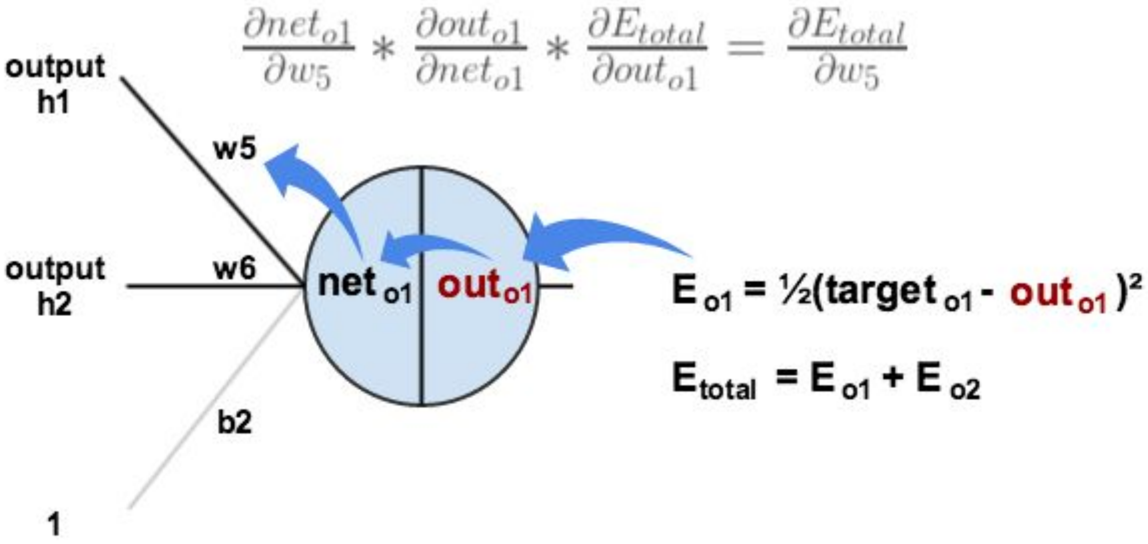
Visualization of loss with respect to weight



$$w^* = w - \alpha(\frac{\partial loss}{\partial w})$$

https://mc.ai/gradient-descent-and-its-types/

More detailed outline of backpropagation



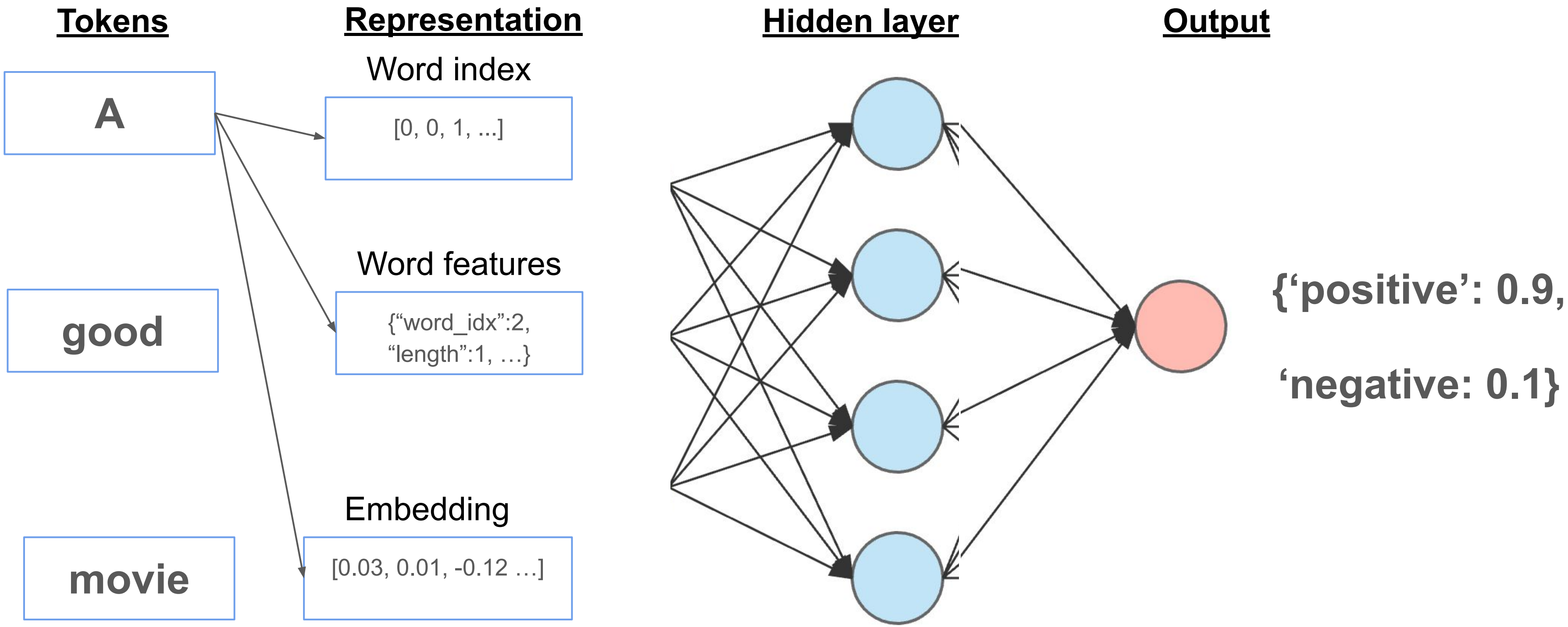
$$\frac{\partial net_{o1}}{\partial w_5} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial E_{total}}{\partial out_{o1}} = \frac{\partial E_{total}}{\partial w_5}$$

$E_{o1} = ½(target_{o1} - out_{o1})^2$

$E_{total} = E_{o1} + E_{o2}$

https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/

Single-layer fully-connected NN with text



Tokens

A

good

movie

Representation

Word index

[0, 0, 1, ...]

Word features

{“word_idx”:2,
“length”:1, …}

Embedding

[0.03, 0.01, -0.12 …]

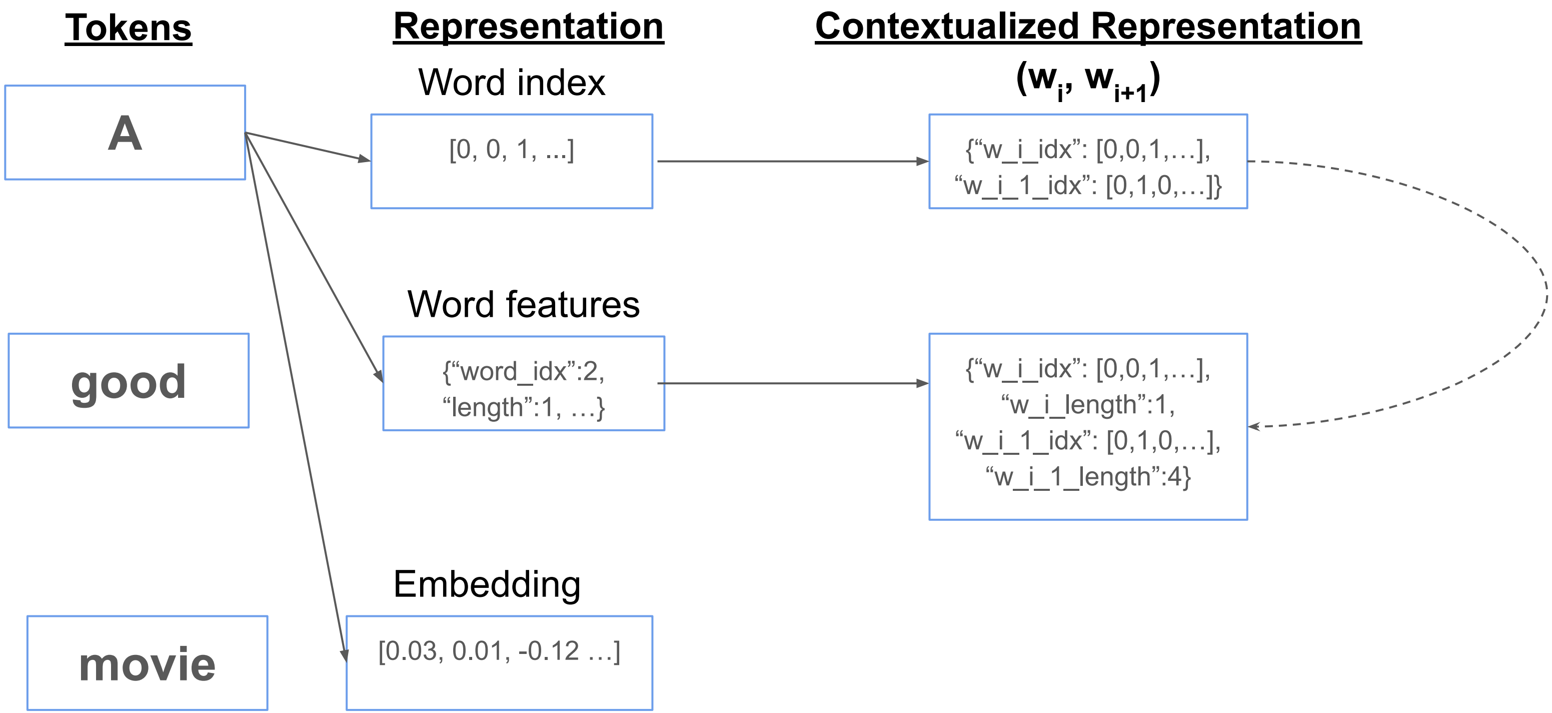
Hidden layer

Output

{‘positive’: 0.9,

‘negative: 0.1}

But we’re still missing context!



Tokens

A

good

movie

Representation

Word index

[0, 0, 1, ...]

Word features

{“word_idx”:2, “length”:1, …}

Embedding

[0.03, 0.01, -0.12 …]

Contextualized Representation

(w_i, w_{i+1})

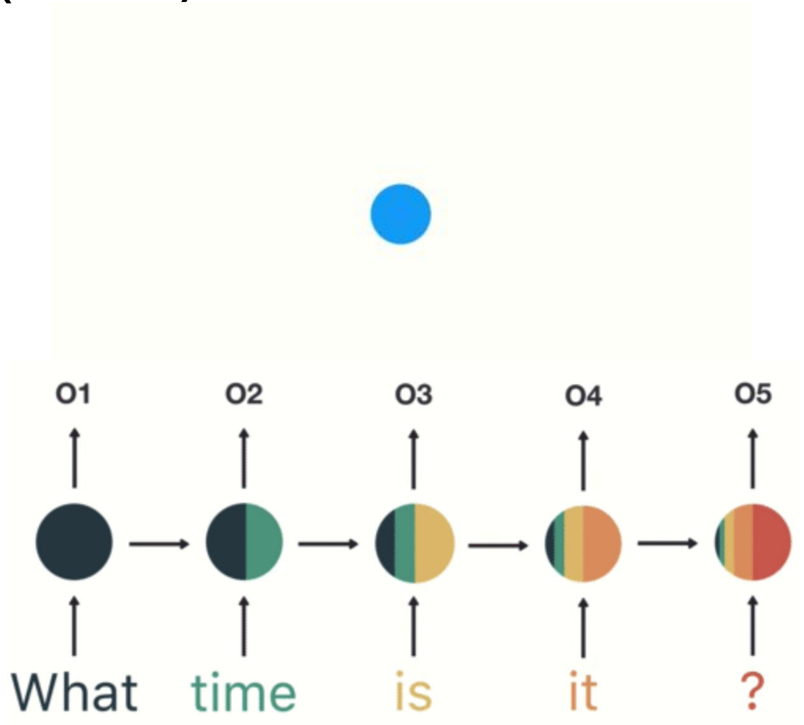
{“w_i_idx”: [0,0,1,…], “w_i_1_idx”: [0,1,0,…]}

{“w_i_idx”: [0,0,1,…], “w_i_length”:1, “w_i_1_idx”: [0,1,0,…], “w_i_1_length”:4}

What if we could share representations across states?

Recurrent Neural Networks (RNN)

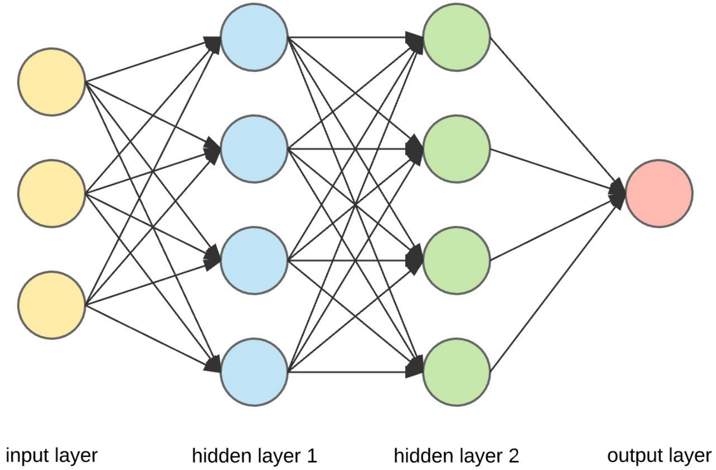
- Specialized structure for handling sequence information
  - Audio
  - Video
  - Text
- Information from previous states maintained in "hidden state"
- Have representations at each state and representation at the end of the sequence



https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9

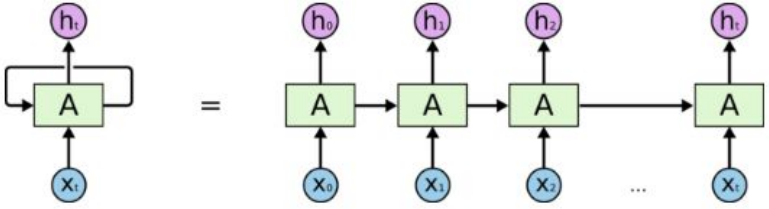
Recurrent vs Fully-Connected

FCNN



https://towardsdatascience.com/applied-deep-learning-part-1-artificial-neural-networks-d7834f67a4f6

RNN

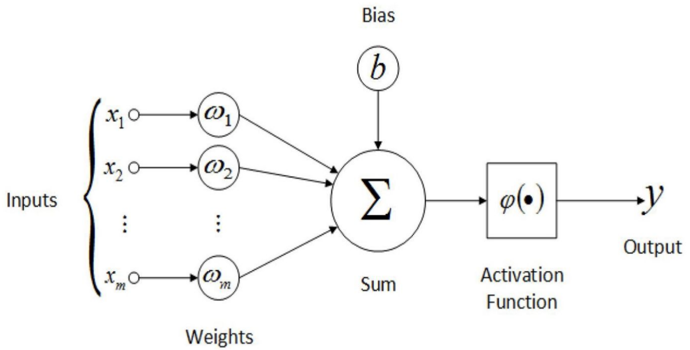


An unrolled recurrent neural network.

https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e

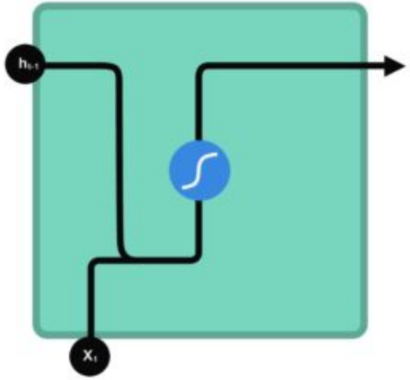
How RNNs “learn”

- NNs:
  - Weight vectors for each input value, adjust based gradient wrt loss
  - Bias vectors (also learned)
  - 1 vector of each per node/cell
  - activation(Weight * input + bias) = output
  - Pass to next layer
- RNNs
  - Hidden layer reflects “state”
  - Combine output (hidden representation) from previous state with current sequence element = input
  - activation(Weight * input + bias) = output (hidden representation)
  - Repeat for each element in the sequence



The Artificial Neuron

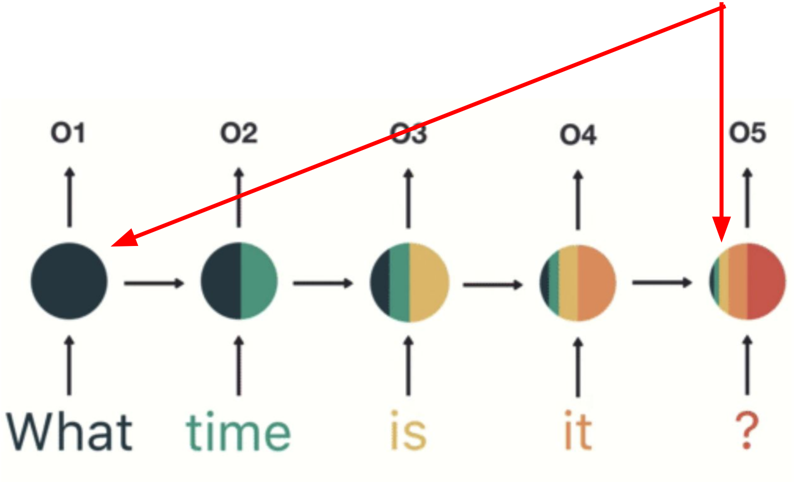
https://medium.com/@jayeshbahire/the-artificial-neural-networks-handbook-part-4-d2087d1f583e



https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

Issues with RNN

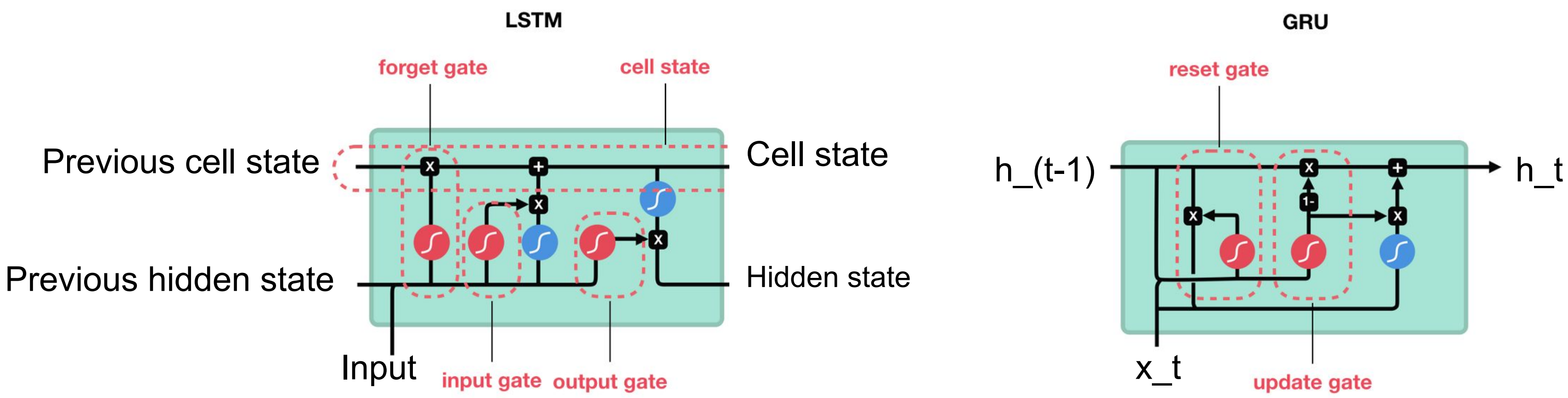
- Language often has long-term dependencies
- Backpropagation in RNN
  - Update weights on hidden state at each step
  - Moving further back, smaller updates
- Need to carry through some information, discard other information
- Redesign RNN units to accomplish this task



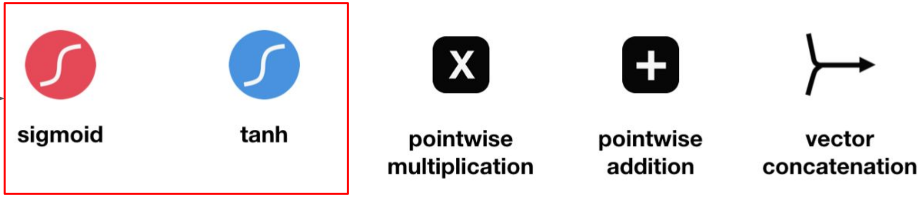
First state contribution to last state

https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9

Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU)



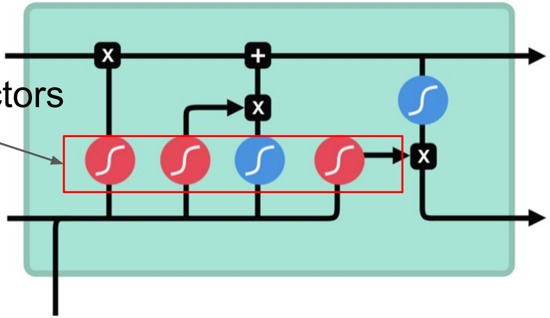
Functions for transforming values



https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

How LSTMs “learn”

- RNNs
  - Hidden layer reflects “state”
  - Combine output (hidden representation) from previous state with current sequence element = input
  - activation(Weight * input + bias) = output (hidden representation)
  - Repeat for each element in the sequence
- LSTMs
  - 4 weight/bias vectors
    - Forget: Weight “useful” info
    - Input: Weight cell candidate info
    - Cell: Cell candidate info
    - Output: Determining new hidden
  - Cell state has info from input and forget
  - Hidden state has info cell state and output
  - Repeat for each element in sequence



4 weight, bias vectors

https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

$$f_t = \sigma((W_{hf} \times h_{t-1}) + (W_{xf} \times x_t) + b_f)$$

$$u_t = \sigma((W_{hu} \times h_{t-1}) + (W_{xu} \times x_t) + b_i)$$

$$\tilde{c}_t = \tanh((W_{hc} \times h_{t-1}) + (W_{xc} \times x_t) + b_c)$$

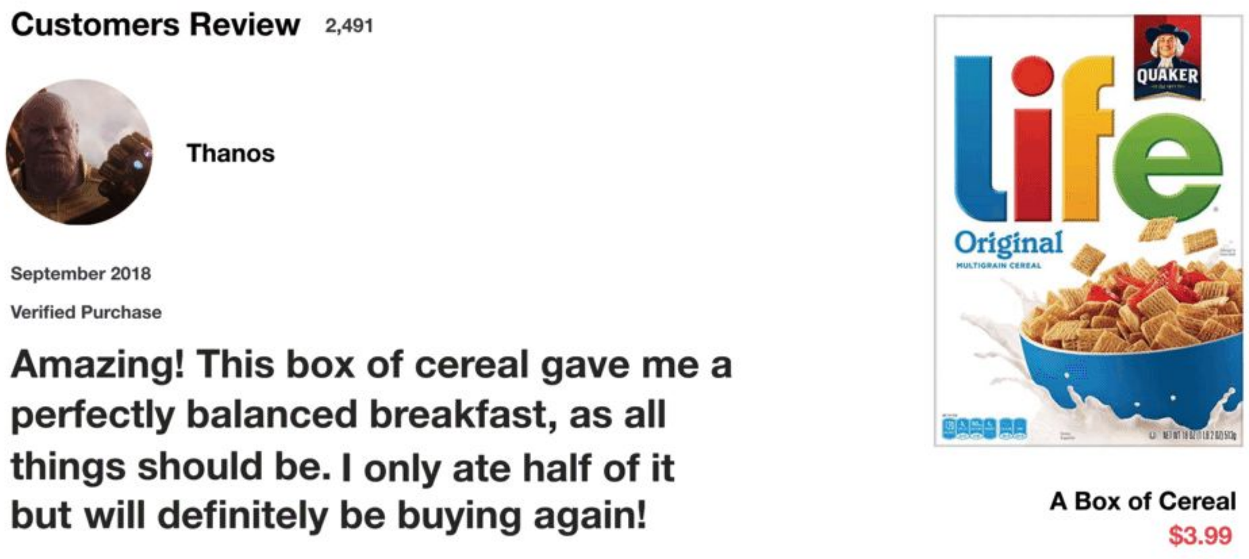
$$o_t = \sigma((W_{ho} \times h_{t-1}) + (W_{xo} \times x_t) + b_o)$$

$$c_t = [\tilde{c}_t \times u_t] + [c_{t-1} \times f_t]$$

h_t $= [o_t \times \tanh c_t]$

https://medium.com/analytics-vidhya/demystifying-lstm-weights-and-biases-dimensions-c47dbd39b30a

Example with product review



Customers Review 2,491

Thanos

September 2018

Verified Purchase

Amazing! This box of cereal gave me a perfectly balanced breakfast, as all things should be. I only ate half of it but will definitely be buying again!

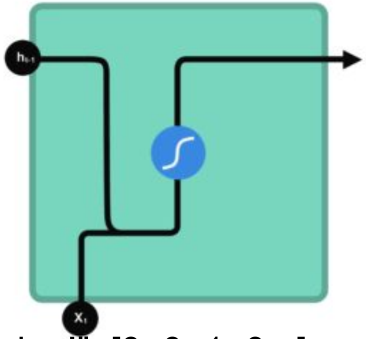
A Box of Cereal

$3.99

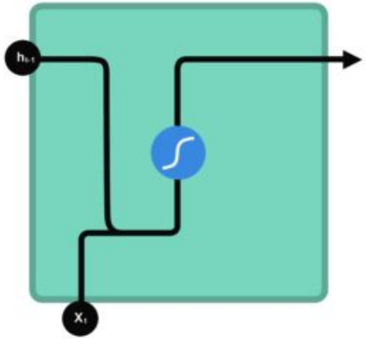
https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

Example with product review (RNN)

h_0
0
0
0
...



h_1
-0.1
0.1
-0.3
...



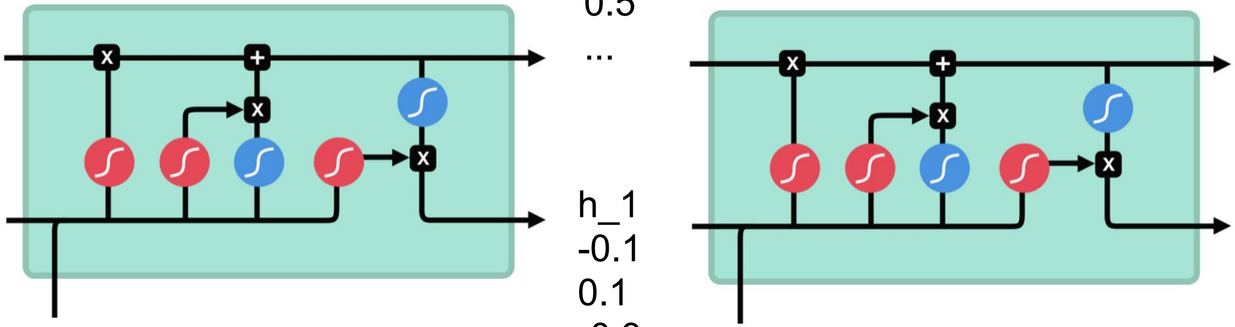
h_2
-0.3
0.5
-0.1
...

“Amazing!”: [0, 0, 1, 0 ..]

“This”: [0, 0, 0, 1 ..]

https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

Example with product review (LSTM)



c_0
0
0
0
...

h_0
0
0
0
...

“Amazing!”: [0, 0, 1, 0 ..]

c_1
-0.1
-0.3
0.5
...

h_1
-0.1
0.1
-0.3
...

“This”: [0, 0, 0, 1 ..]

c_1
0.3
-0.1
0.8
...

h_2
-0.3
0.5
-0.1
...

https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

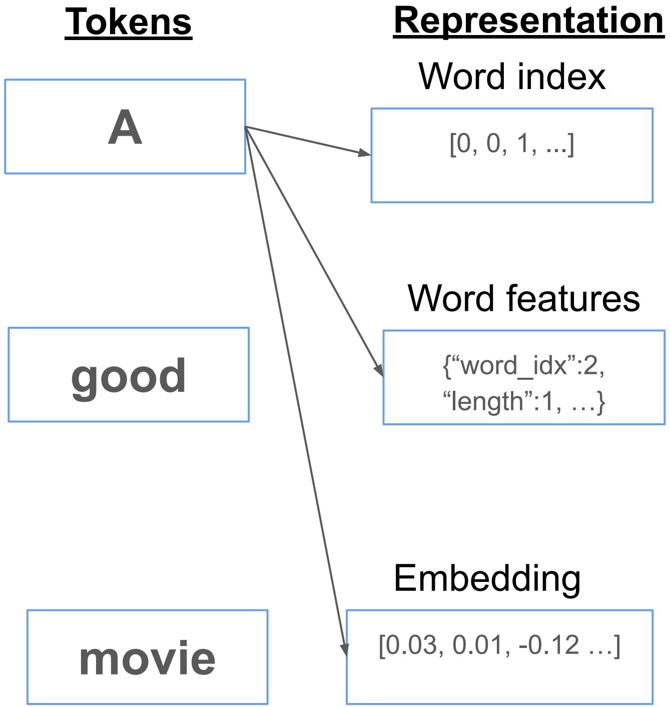
LSTM implementation in PyTorch

Sequence length considerations

- For batch processing, need to have consistent “maximum length”
  - If running one observation at a time, can be of any length
- Formatting batches:
  - Greater than max length = Truncation/splitting
    - E.g. Sentence-wise splitting (BERT trained around sentences)
  - Less than this value = Padding
    - Max length = 4
    - Sequence = [“I”, “like”, “NLP”]
    - Prepared input = [<index for “I”>, <index>, <index>, <index for padding token>]
- To batch or not to batch?
  - Running single observation: Any length, but slower and updating weights for each observation
  - Batching: Need to choose a max length or group by same length

How to represent words

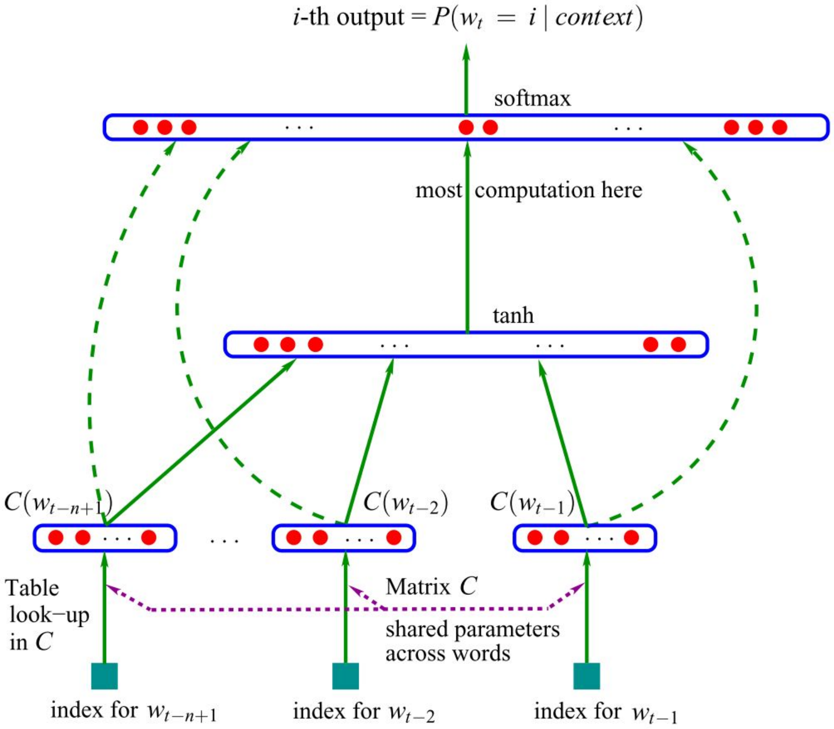
- In notebook: Words as sparse vectors (one-hot encoded)
- “Car” vs “automobile” totally different vectors
- Model needs to learn weights for every word in vocabulary
- Condensed, informative representation
  - Word-level representations from topic models
  - Embeddings



Dense representations from topic models

Language model embeddings

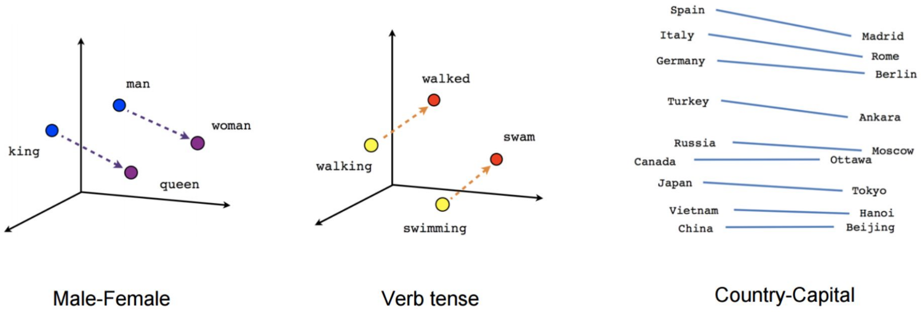
- Language model embeddings
  - Language modelling task: Predict masked word
  - Learned representations of input words
  - These representations then fed into a computation layer
  - Output: Prediction of the target word
    - “Full”: 70% likely, “tired” 20% likely, etc
- Issues here
  - Computationally expensive to train representation + prediction
  - Representation trained for task, rather than generalized word representation



A Neural Probabilistic Language Model
(https://papers.nips.cc/paper/1839-a-neural-probabilistic-language-model.pdf)

Word2Vec and GloVe embeddings

- Word2Vec (2013)
  - Directly estimated word representation
  - Proposed two implementations
    - Skip-Gram
    - Continuous Bag of Words
  - More efficient, generalized than language model-based
- GloVe (2014)
  - Based on the co-occurence of words
  - More efficient to compute, similar performance

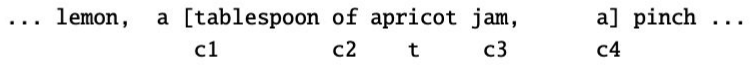


Male-Female Verb tense Country-Capital

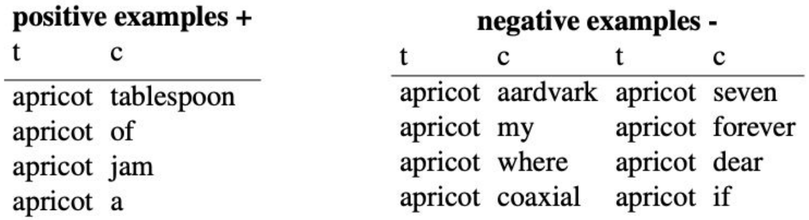
[[1301.3781] Efficient Estimation of Word Representations in Vector Space]

Basic structure of word embedding algorithms

- Using text as supervised learning problem
  - Positive class: target words and their context
  - Negative class: Target words (T) and “noise words”
- Binary classification: Predict whether word c is in context of word t
- Prediction is based on the similarity between the vector for t and vector for c
- Vectors initialized randomly, model “learns” embeddings
  - Maximize distance between t and noise words
  - Minimize distance between t and actual context



```
... lemon,  a [tablespoon of apricot jam,      a] pinch ...
               c1         c2    t    c3        c4
```

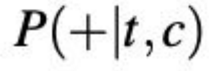


positive examples +

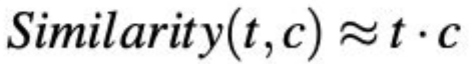
t	c
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

negative examples -

t	c	t	c
apricot	aardvark	apricot	seven
apricot	my	apricot	forever
apricot	where	apricot	dear
apricot	coaxial	apricot	if



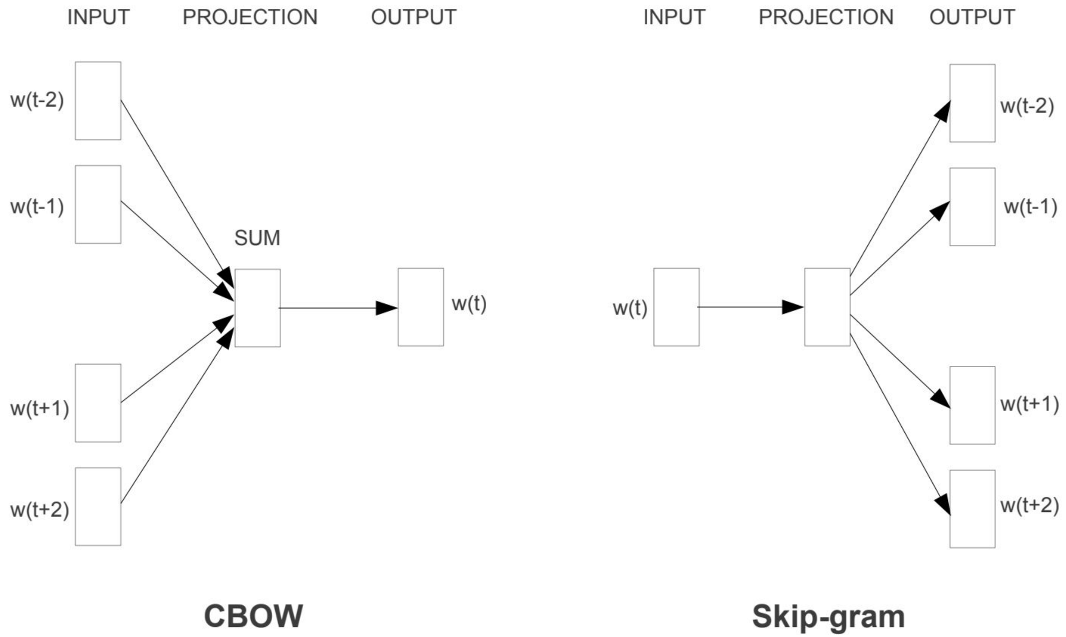
$$P(+|t,c)$$



$$Similarity(t,c) \approx t \cdot c$$

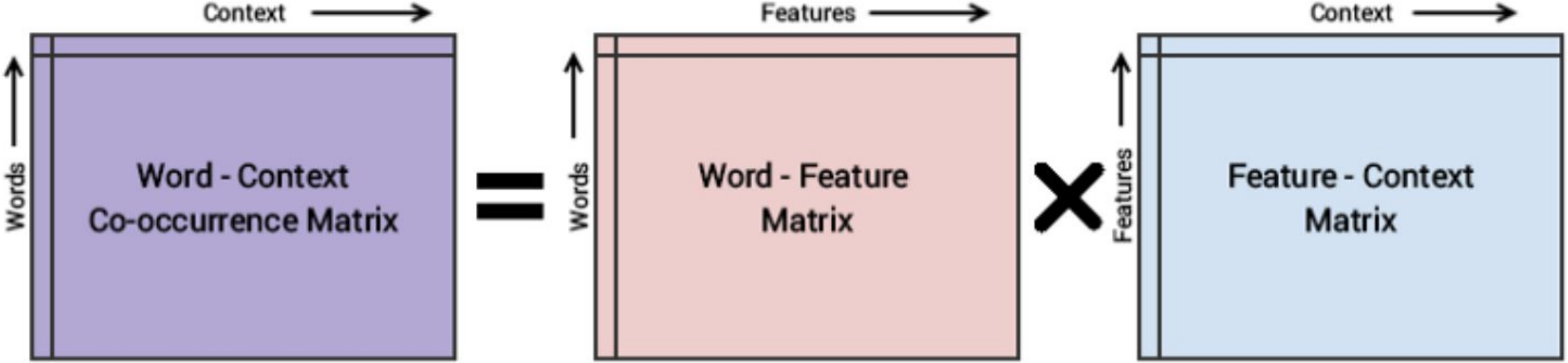
https://web.stanford.edu/~jurafsky/slp3/6.pdf

Skip-Gram vs CBOW



https://arxiv.org/pdf/1301.3781.pdf

Global Vectors for Word Representation (GloVe)



Conceptual model for the GloVe model's implementation

Implementing Deep Learning Methods and Feature Engineering for Text Data: The GloVe Model

Back to our RNN!

Considerations when choosing embeddings

- Availability
  - Usually can rely on pre-trained models
- Applicability
  - Specific domains may require specific embeddings
- Complexity
  - Highly complex likely need more data
- Parameters
  - Context window size
  - Over-sampling rare words
  - Representation dimensionality

Domain-specific word embeddings

https://allenai.github.io/scispacy/

Implementation of Word2vec

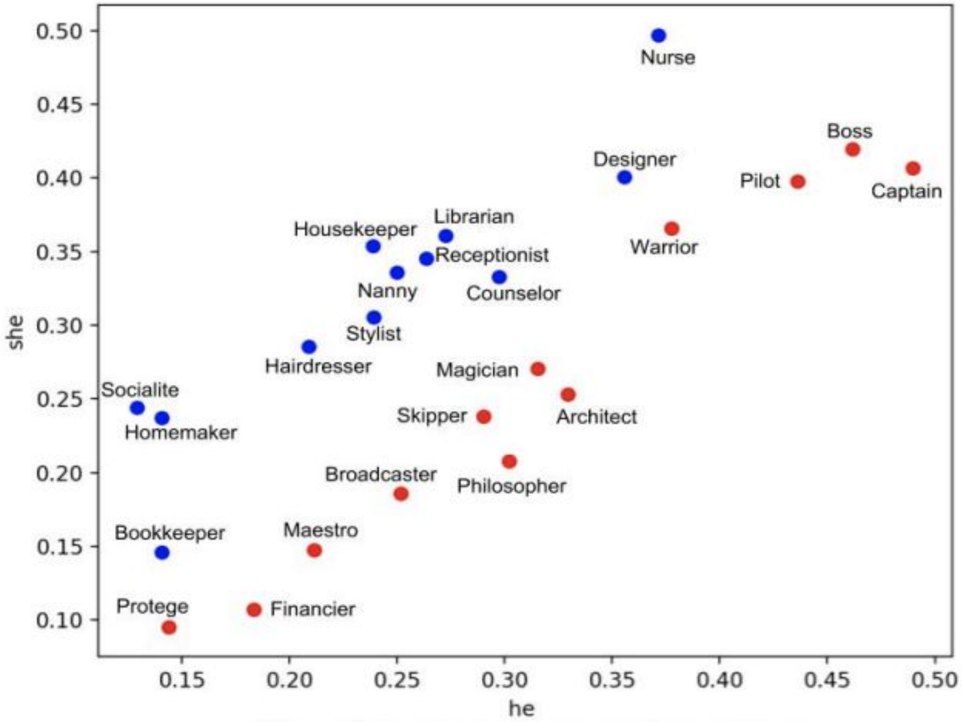
https://radimrehurek.com/gensim/models/word2vec.html

Implementation of GloVe

https://github.com/stanfordnlp/glove

Considerations around bias

Word vector similarity between gender words and occupations



[Exploring and Mitigating Gender Bias in GloVe Word Embeddings](#)