COE548: LARGE LANGUAGE MODELS

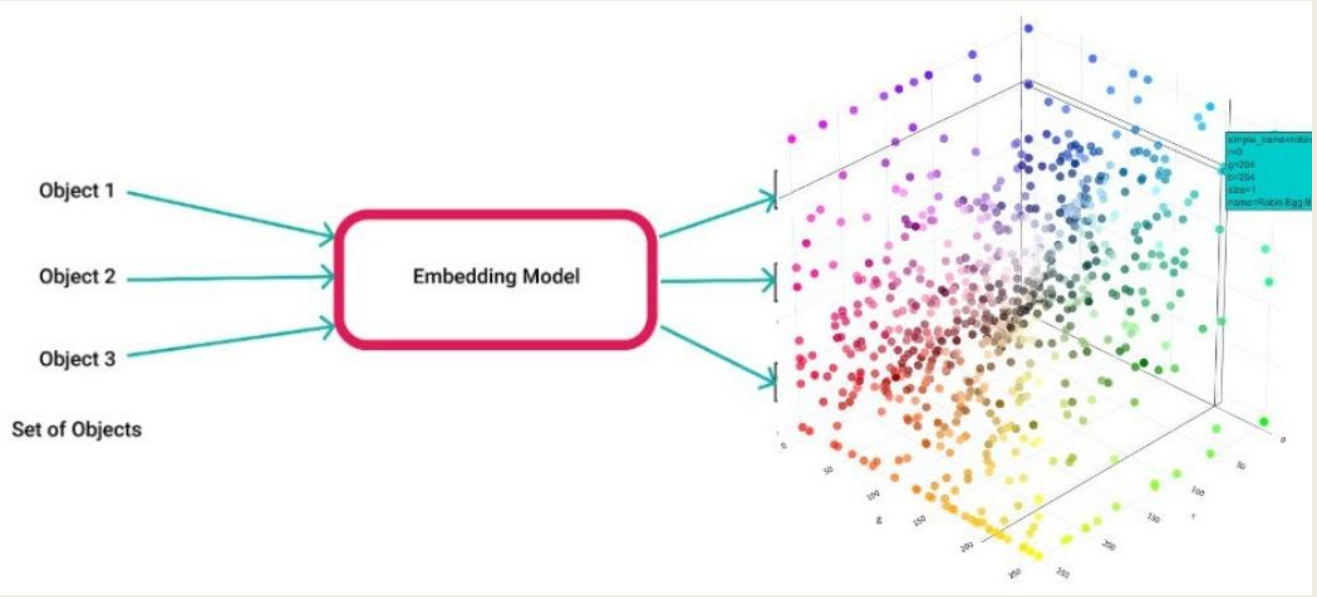
Word Embeddings

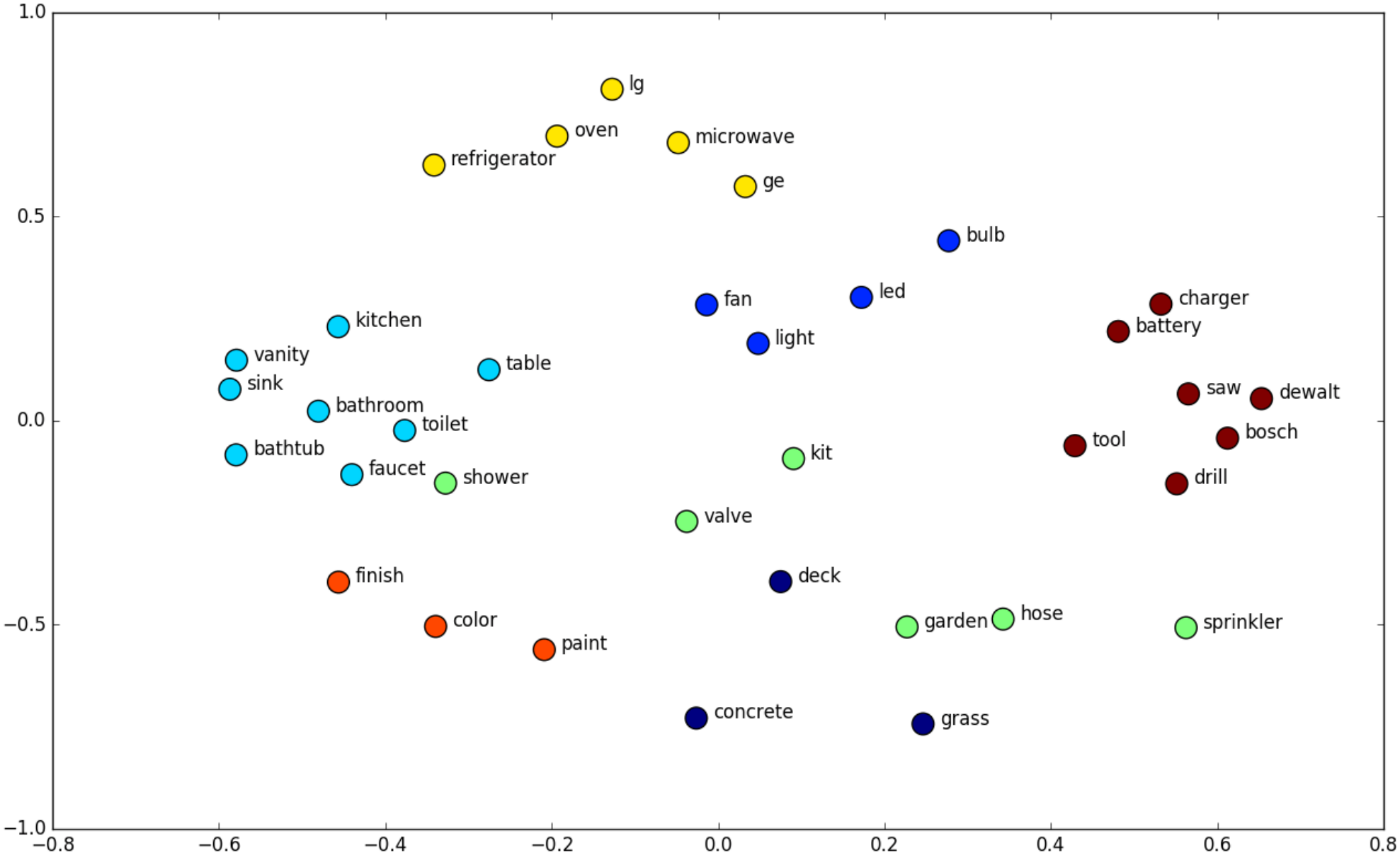
Outline

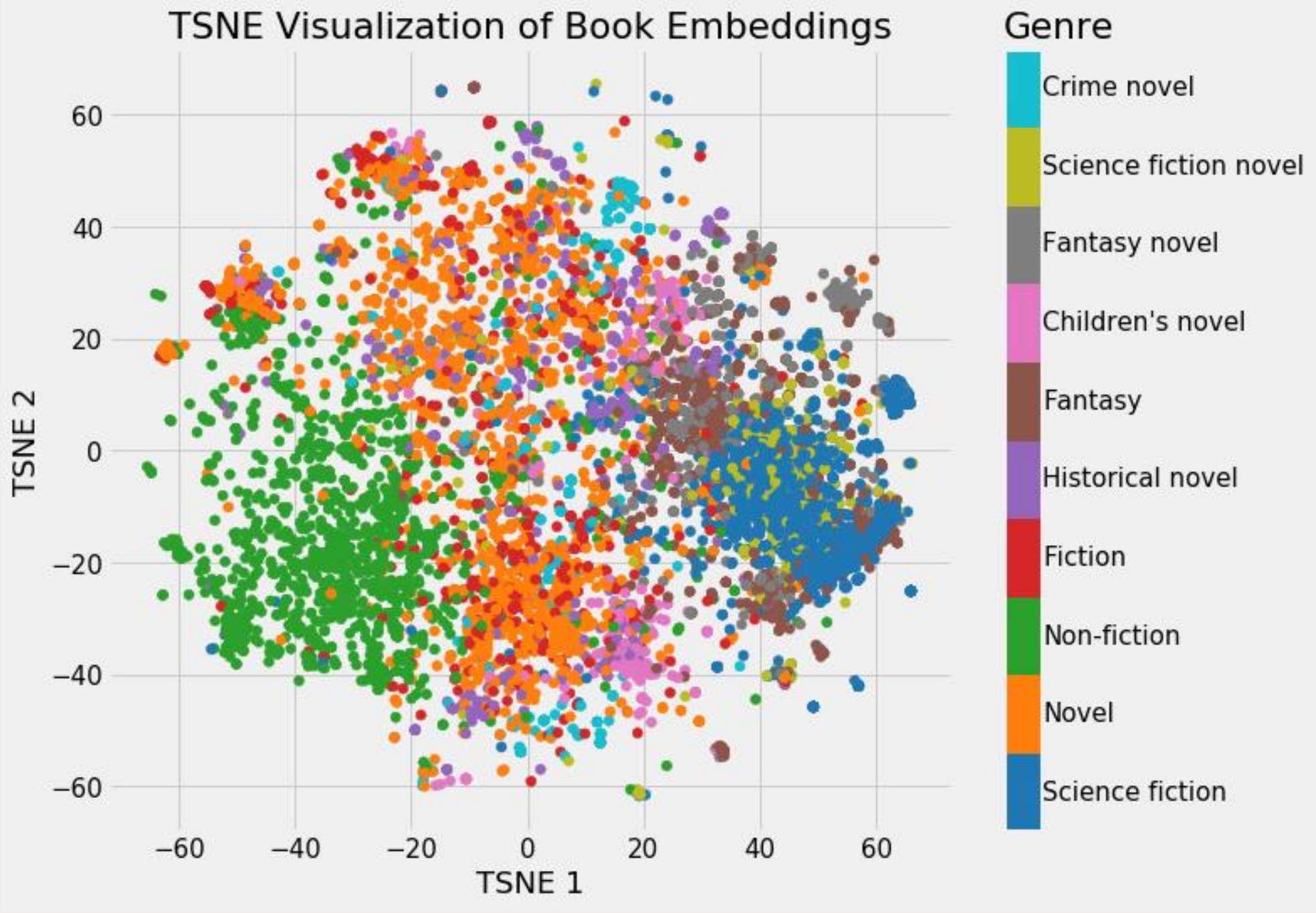
- 1-hot Vectors
- Vector Representations as Features
- Distributional Semantics and Co-occurrence Matrices
- Neural Network (very) Brief Overview
- Word2vec
- Global Vectores (GloVe) for Word Representations

Word Embeddings

- Word embeddings helps make numeric representations of words make some sense by allowing similar words to have similar numeric values.
- By definition, word embeddings are the mapping of words to a vector space.




1.0
0.5
0.0
−0.5
−1.0
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6
0.8
lg
oven
microwave
refrigerator
ge
bulb
led
fan
charger
kitchen
battery
light
vanity
table
sink
saw
dewalt
bathroom
toilet
tool
bosch
bathtub
kit
faucet
shower
drill
valve
finish
deck
color
garden
hose
sprinkler
paint
concrete
grass



TSNE: t-distributed Stochastic Neighbor Embedding

Word Embeddings

- Simplest way, assign random numbers to words.
  - *However, with such a method, the NLP model would need a lot of complexity and training to overcome the large numeric differences of all words (the large differences in words that have similar meanings).*
  - *This is because the words are represented as independent unrelated entities.*
    - For example king and queen would seem completely unrelated whereas as humans we know there is some relation between the two words.

1-hot Vectors

- Another way to represented tokens, still as independent and uncorrelated entities, is through 1-hot vectors, or standard basis vectors.

$$v_a = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, v_{an} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, v_{aardvark} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{bmatrix}, \dots, v_{zyzzyva} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

- Using vectors helps with better compute with the tokens, however as 1-hot vectors there is no meaningful notion of similarity or relationship amongst the tokens (e.g., see through dot product of two words, always equals zero).

1-hot Vectors

- The other problem with 1-hot encoded vector representations of words is the inefficient compute cost.

- For any given language with n words, the vector representation for any one word will exist in $v_{word} \in \mathbb{R}^n$.
  - *Example: English language has around 170,000 words + 47,000 obsolete words. Basically $n > 200{,}000$.*
  - *Example: Arabic language has over 12 million distinct words.*
- Performing operations on such large vectors is computationally expensive and unfeasible.

Vector Representations as Collection of Features

- For any word, say runners, there is a wealth of information we can annotate about that word.
- There is grammatical information, like plurality
- There's derivational information, like how the runners is something like the verb to run plus a notion of "doer" or agent (think one who runs)
- There's also semantic information, like how runners might be a hyponym of humans, or animals, or entities.

Vector Representations as Collection of Features

- There are many resources for annotated information about words.
  - *WordNet [Miller, 1995] annotates for synonyms, hyponyms, and other semantic relations*
  - *UniMorph [Batsuren et al., 2022] annotates for morphology (subword structure) information across many languages.*
- With such resources, one could build word vectors that look something like

$$v_{tea} = \begin{bmatrix} 0 \\ 1 \\ \vdots \\ 1 \end{bmatrix} \begin{matrix} plural \\ hyponym\ of\ beverage \\ \vdots \\ synonym\ of\ chai \end{matrix}$$

Vector Representations as Collection of Features

- One main failure is that human-annotated resources (or discrete representations) are always lacking in vocabulary compared to methods that can draw a vocabulary from a naturally occurring text source.
- Updating these resources is costly and they're always incomplete.
- Another failure is a tradeoff between dimensionality and utility of the embedding.
  - *It takes a very high-dimensional vector (think much larger than the vocabulary size) to represent all of these categories*
  - *Modern neural methods that tend to operate on dense vectors do not behave well with such vectors*

Distributional Semantics and Co-occurrence Matrices

- “You shall know a word by the company it keeps” – J.R. Firth, 1957.
- Unsupervised (or lately, “self-supervised”) learning takes data and attempts to learn properties of the elements of that data, often by taking part of the data (maybe a word in a sentence) and attempting to predict other parts of the data (other words) with it.
- At a high level, you can think of the distribution of words that show up around the word tea as a way to define the meaning of that word.
- So, tea shows up around drank, the, pot, kettle, bag, delicious, oolong, hot, steam,. . .
- It should become clear that words similar to tea (like coffee) will have similar distributions of surrounding words.

Distributional Semantics and Co-occurrence Matrices

- The distributional hypothesis: the meaning of a word can be derived from the distribution of contexts in which it appears.

- This idea can be quantized using co-occurrence matrices. The process of creating one may look like:
  - *Determine a vocabulary V.*
  - *Make a zero matrix of size |V|x|V|*
  - *Walk through a sequence of documents. For each document, for each word w in the document, add all the counts of the surrounding words w′ in the document to the row corresponding to w at the column corresponding to w′.*
  - *Normalize the rows by the sum.*

- Turns out while creating this matrix, shorter windows of context encode more syntactic properties, whereas longer windows encode more semantic properties.

Co-occurrence Matrices: Example

"A large language model … capable of language generation or other natural language processing tasks."

a	…	generation	…	language	…	large	…	model	…	natural	…	of	…	processing	…
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
generation	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
language	0	0	0	0	0	0+1	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
large	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
model	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
natural	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
of	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
processing	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Co-occurrence Matrices: Example

"A large language model … capable of language generation or other natural language processing tasks."

a	…	generation	…	language	…	large	…	model	…	natural	…	of	…	processing	…
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
generation	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
language	0	0	0	0	0	1+1	0	0+1	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
large	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
model	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
natural	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
of	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
processing	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Co-occurrence Matrices: Example

"A large language model … capable of language generation or other natural language processing tasks."

a	…	generation	…	language	…	large	…	model	…	natural	…	of	…	processing	…
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
generation	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
language	0	0	0	0	0	2	0	1+1	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
large	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
model	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
natural	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
of	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
processing	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Co-occurrence Matrices: Example

"A large language model … capable of language generation or other natural language processing tasks."

a	…	generation	…	language	…	large	…	model	…	natural	…	of	…	processing	…
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
generation	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
language	0	0	0	0	0	2	0	2	0	0	0	0+1	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
large	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
model	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
natural	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
of	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
processing	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Co-occurrence Matrices: Example

"A large language model … capable of language generation or other natural language processing tasks."

a	…	generation	…	language	…	large	…	model	…	natural	…	of	…	processing	…
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
generation	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
language	0	0+1	0	0	0	2	0	2	0	0	0	1+1	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
large	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
model	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
natural	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
of	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
processing	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Co-occurrence Matrices: Example

"A large language model … capable of language generation or other natural language processing tasks."

a	…	generation	…	language	…	large	…	model	…	natural	…	of	…	processing	…
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
generation	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
language	0	1+1	0	0	0	2	0	2	0	0	0	2	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
large	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
model	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
natural	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
of	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
processing	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Co-occurrence Matrices: Example

"A large language model … capable of language generation or other natural language processing tasks."

a	…	generation	…	language	…	large	…	model	…	natural	…	of	…	processing	…
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
generation	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
language	0	2	0	0	0	2	0	2	0	0+1	0	2	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
large	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
model	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
natural	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
of	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
processing	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Co-occurrence Matrices: Example

"A large language model … capable of language generation or other natural language processing tasks."

a	…	generation	…	language	…	large	…	model	…	natural	…	of	…	processing	…
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
generation	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
language	0	2	0	0	0	2	0	2	0	1+1	0	2	0	0+1	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
large	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
model	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
natural	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
of	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
processing	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Co-occurrence Matrices: Example

"A large language model … capable of language generation or other natural language processing tasks."

a	…	generation	…	language	…	large	…	model	…	natural	…	of	…	processing	…
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
generation	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
language	0	2	0	0	0	2	0	2	0	2	0	2	0	1+1	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
large	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
model	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
natural	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
of	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
processing	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Co-occurrence Matrices: Example

"A large language model … capable of language generation or other natural language processing tasks."

a	…	generation	…	language	…	large	…	model	…	natural	…	of	…	processing	…
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
generation	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
language	0	2	0	0	0	2	0	2	0	2	0	2	0	2	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
large	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
model	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
natural	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
of	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
processing	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Divide by row sum

Co-occurrence Matrices: Example

"A large language model … capable of language generation or other natural language processing tasks."

a	…	generation	…	language	…	large	…	model	…	natural	…	of	…	processing	…
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
generation	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
language	0	0.167	0	0	0	0.167	0	0.167	0	0.167	0	0.167	0	0.167	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
large	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
model	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
natural	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
of	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
processing	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
…	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

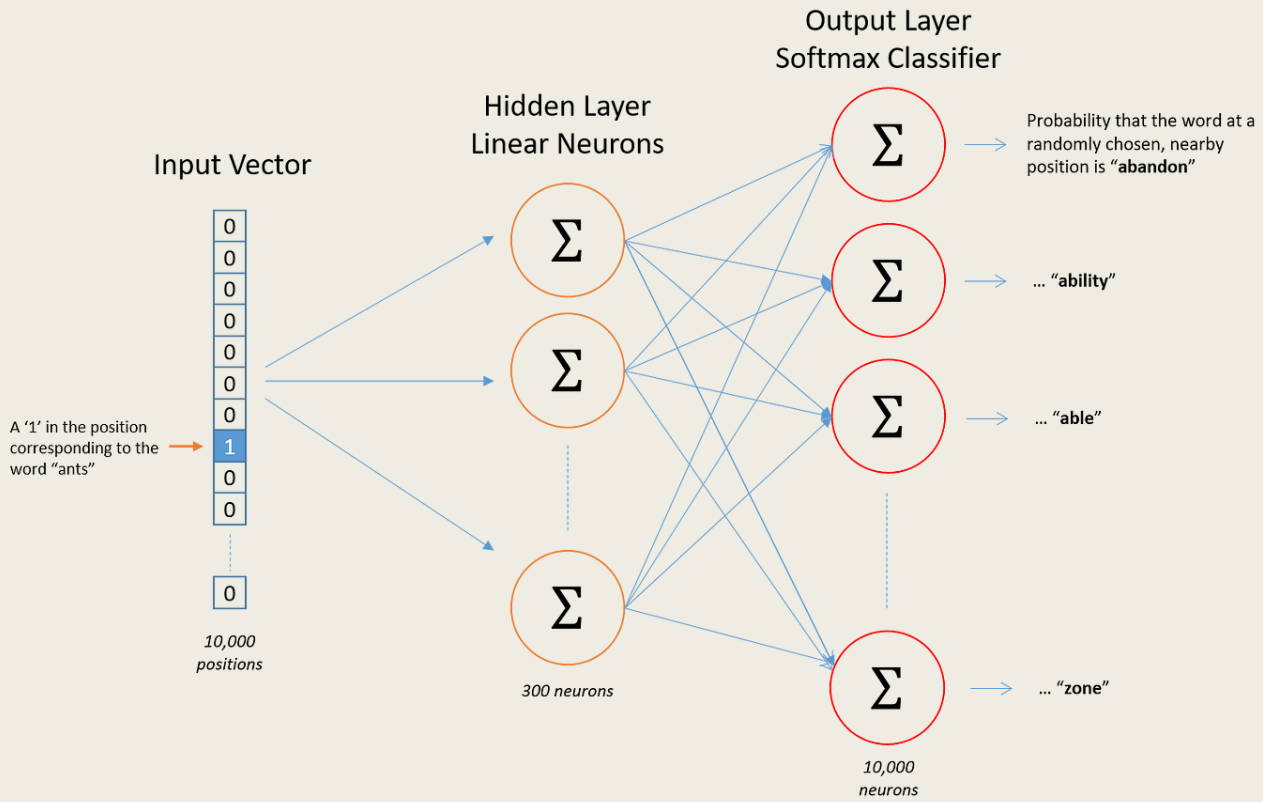
Divide by row sum

Co-occurrence Matrices

- Problem to note: very frequent words, such as “the”, may get very high frequency counts, over emphasizing them.
- To get around this, can cap the count for words (e.g., min(X, 100)).
- Or can give words closer to center word higher weight, say 1, and further away words less weight, say 0.5.

Word2Vec

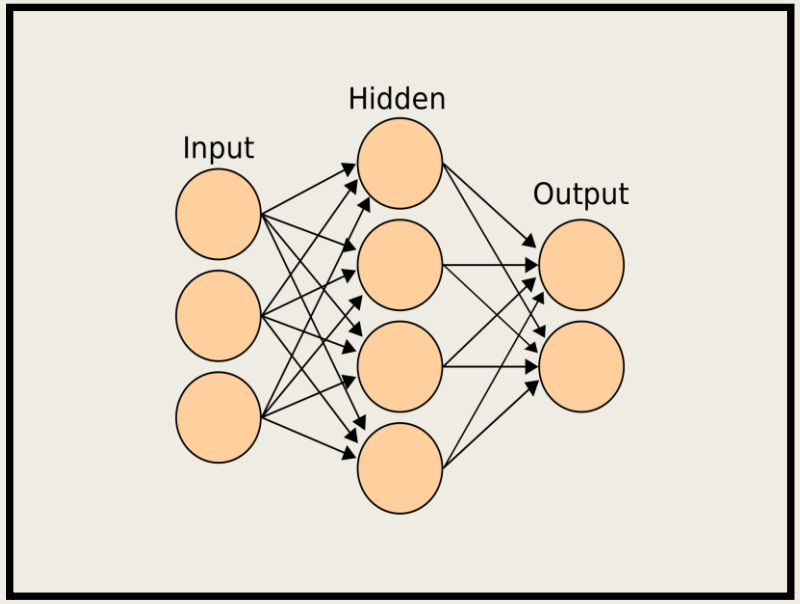
- Word2Vec uses a 2-layer neural network to generate word embeddings for a corpus of text.



Neural Network (Brief) Overview

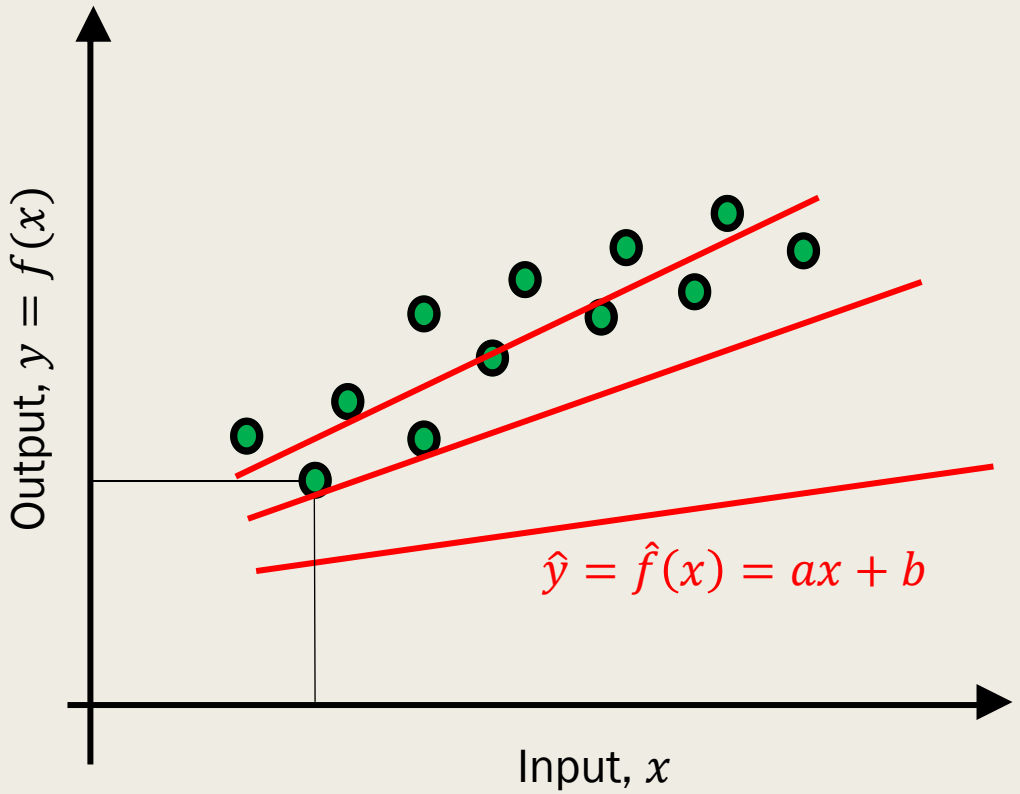
- Neural network is a computational system inspired by the human brain, which learns from large amounts of data.

- Input layer: Receives the input (e.g., picture, feature vector, etc.)
- Hidden layer: Performs computations through neurons that process inputs and pass their output to the next layer.
- Output layer: Produces the final predictions or classification.



Neural Network (Brief) Overview

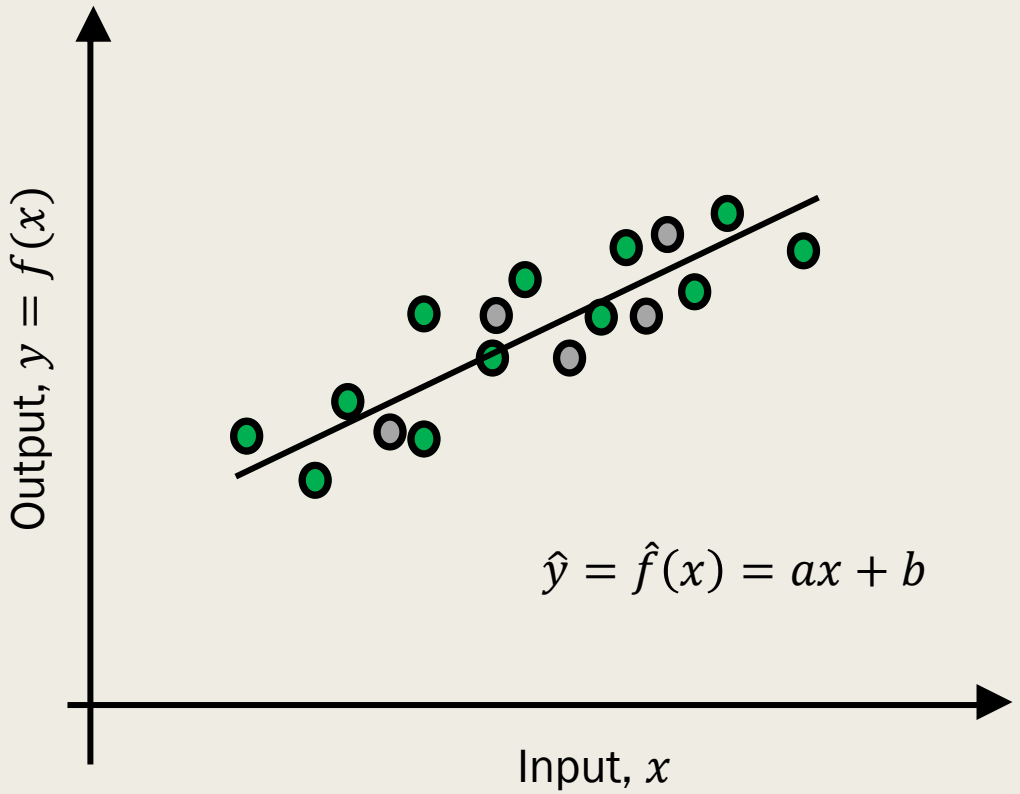
Neural network is some function, e.g., $f(x) = ax + b$



- Training data sample
- Unseen test data sample

Neural Network (Brief) Overview

Neural network is some function, e.g., $f(x) = ax + b$

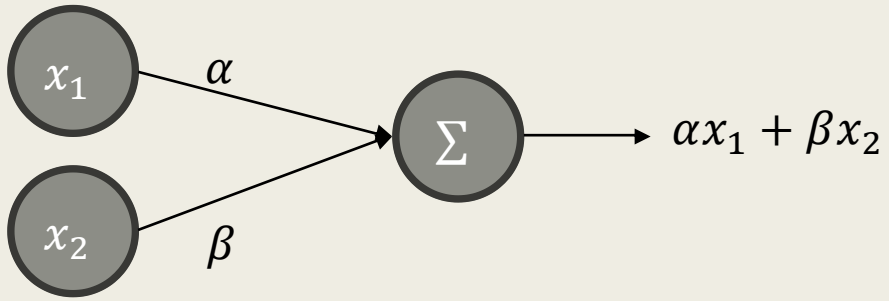


- Training data sample
- Unseen test data sample

Neural Network (Brief) Overview

- The neural network function could also take vectors as input:

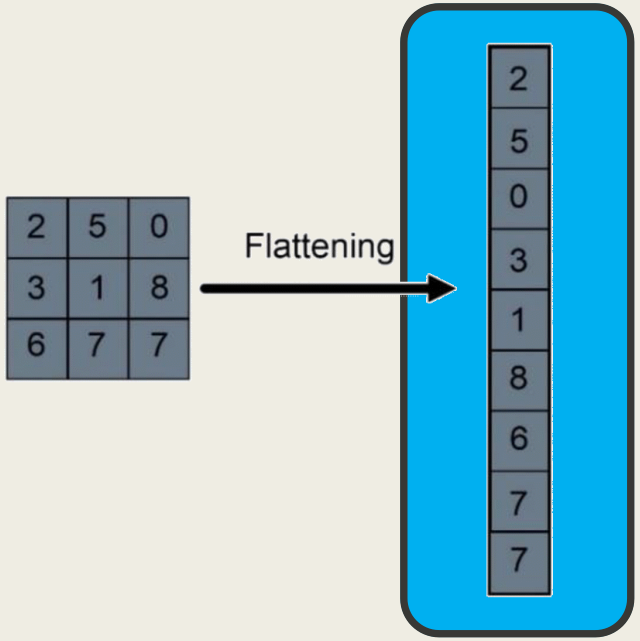
$$y = f\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = \alpha x_1 + \beta x_2$$



In practice, neural networks are just much larger and more complicated with many nested functions $y = f\left(g\left(h(x)\right)\right)$, where the functions are non-linear, and there are many more parameters than just α and β, and tons of tricks to make them perform well on different tasks.

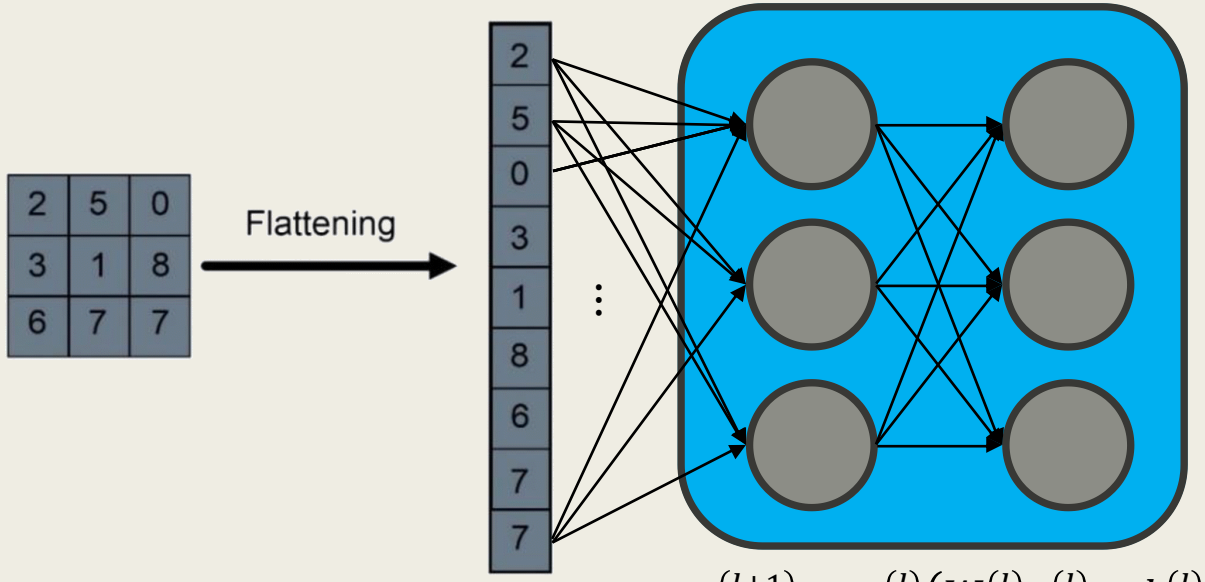
Neural Network (Brief) Overview

- Input layer
  - *Might involve some pre-processing step to convert whatever input we are looking at into a vector.*
  - *For sake of this overview, the input needs to be a fixed size.*



Neural Network (Brief) Overview

- Hidden layers
  - *Affine transformations (linear transformations) of previous layer*
    - E.g. if $x^{(l)}$ is the vector representation at layer l, then we have $W^{(l)}x^{(l)} + b^{(l)}$
  - *Neurons: Activation (nonlinear) functions that take $W^{(l)}x^{(l)} + b^{(l)}$ as input*



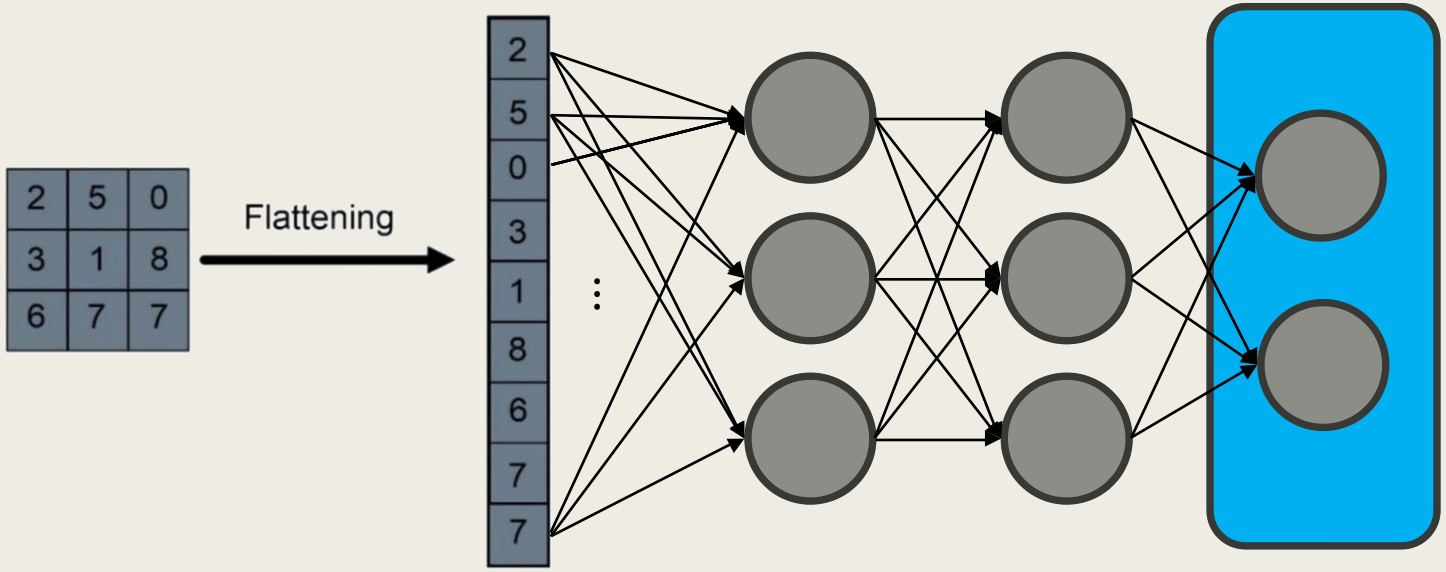
$$x^{(l+1)} = \varphi^{(l)}\left(W^{(l)}x^{(l)} + b^{(l)}\right)$$

Common Activation Functions

- Sigmoid: $f(x) = \frac{1}{1+e^{-x}} \in [0, 1]$
- Tanh: $f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \in [-1, 1]$
- ReLU: $f(x) = x^{+}$ or $\max(0, x)$

Neural Network (Brief) Overview

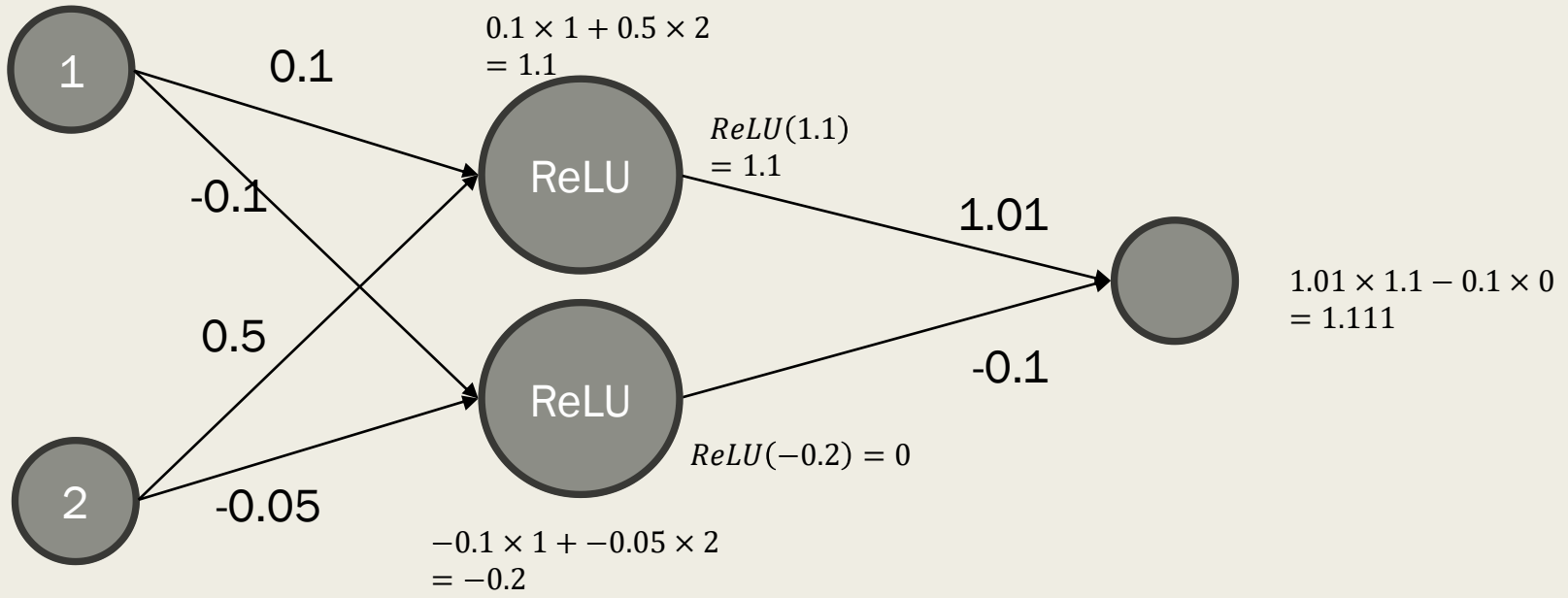
- Output layer
  - *Affine transformations (linear transformations) of last hidden layer.*
  - *Does not necessarily need an activation function (depends on task).*



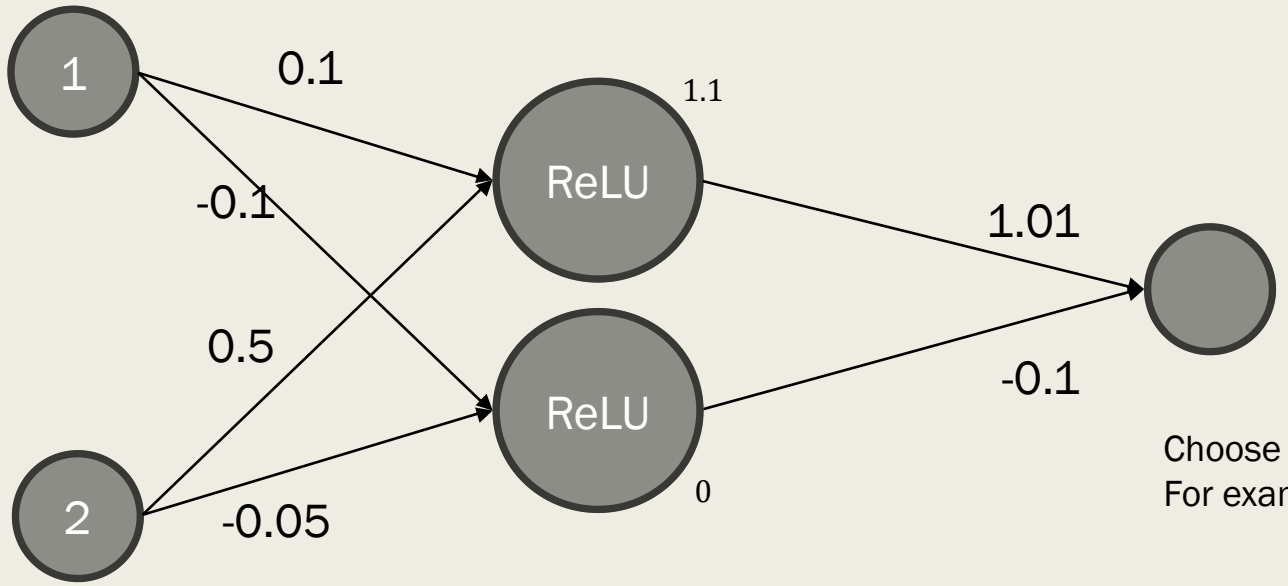
Neural Network (Brief) Overview

- **Forward Propagation**: Data propagates from the input to the output layer.
- **Loss Function**: Measures the difference between the network's prediction and the actual target values.
- **Backpropagation**: The process of updating the weights of the network by calculating the gradient of the loss function with respect to each weight.
- **Optimization**: The process of gradually adjusting the weights (or parameters) of the neural network to minimize the loss.

NN Forward Pass Example



NN Loss Example

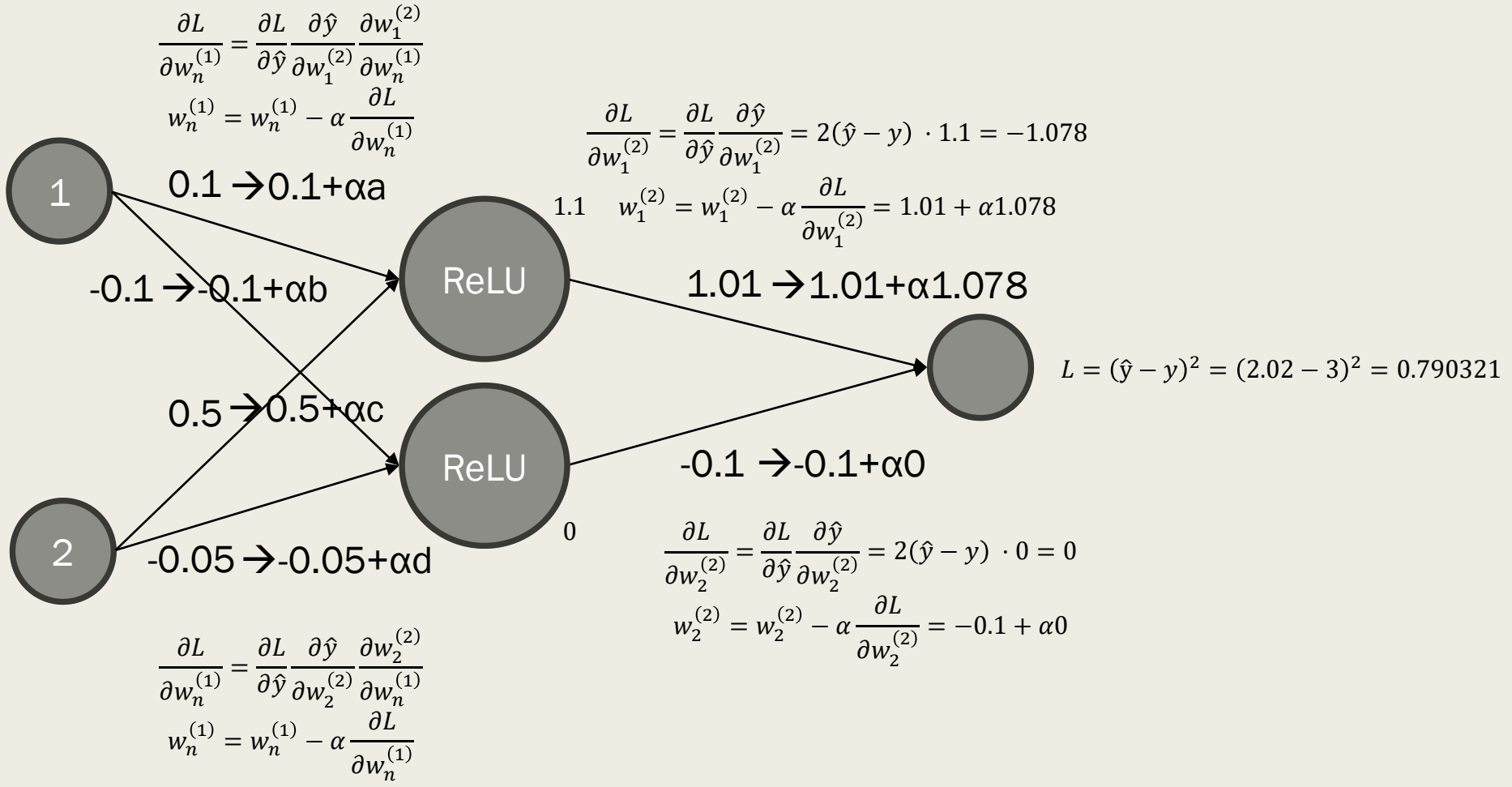


Choose a loss function

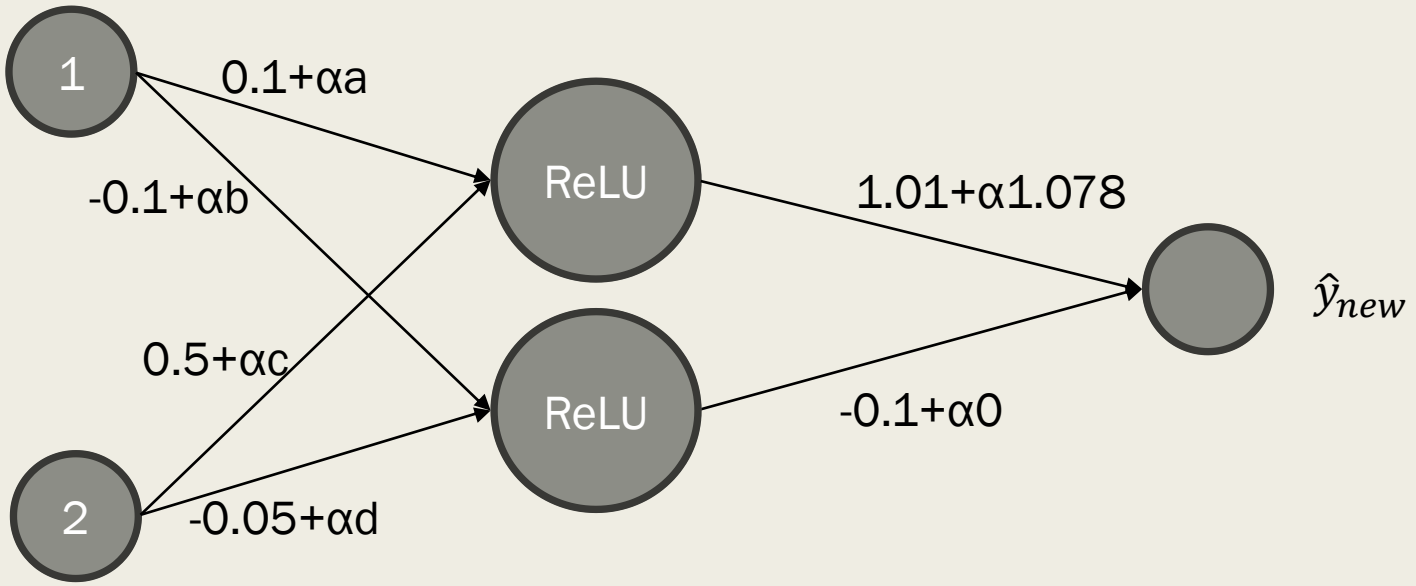
For example, the squared-loss function:

$$L(\hat{y}, y) = (\hat{y} - y)^2$$

NN Backward Pass Example



NN “Trained” Example

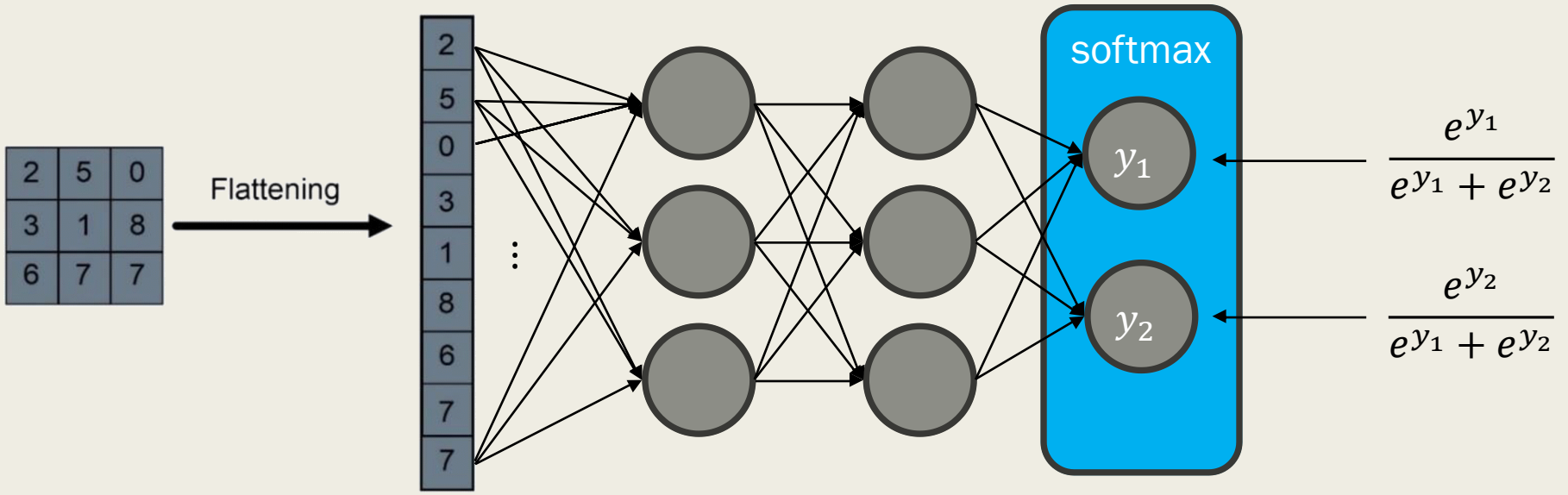


Softmax

- Output layer
  - *You will commonly see the output layer equipped with the softmax function*

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

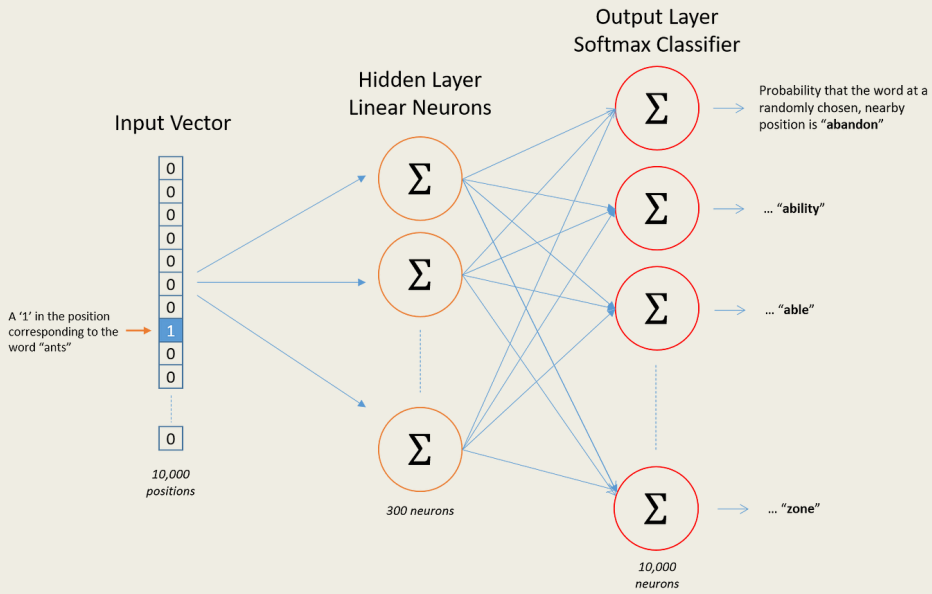
This makes it such that the output vector is a probability distribution.



Word2Vec

- Word2Vec uses a 2-layer neural network to generate word embeddings for a corpus of text.

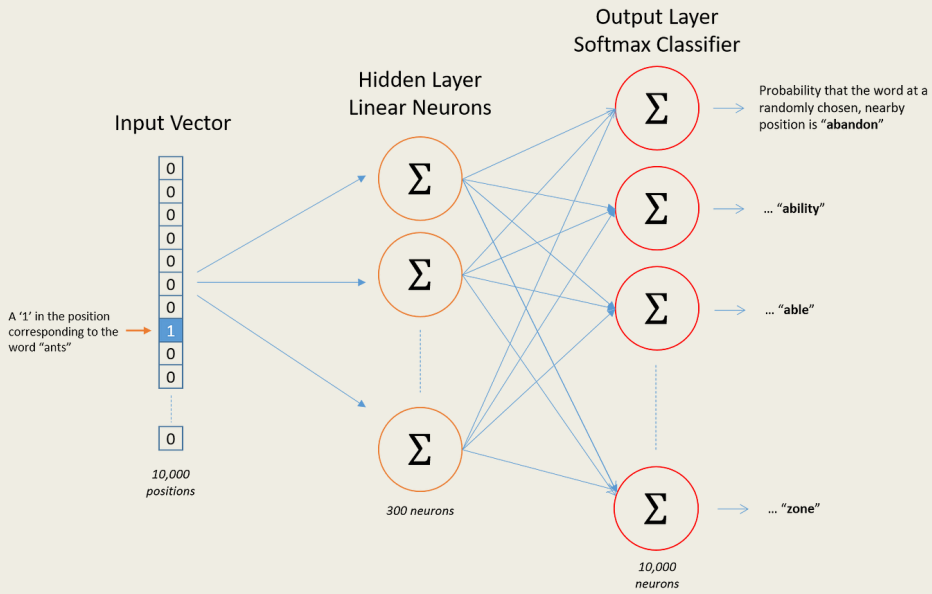
- To create word embeddings using neural networks, start off with a vector the size of your vocabulary |V|.
- This vector (one-hot encoded) is fully connected to a layer of neurons with linear/identity activation functions.
- The weights of the connections are the numeric representations of the words. The number of vector elements (or the embedding vector dimension) associated with each word is the number of neurons in the hidden layer.



Word2Vec

- Word2Vec uses a 2-layer neural network to generate word embeddings for a corpus of text.

- The weights are optimized through backpropagation.
- The output to the neural network is also a one-hot encoded vector that is the size of the vocabulary, and the activated element of the vector is the word that proceeds the input word.
- Training is done on a corpus of text.
- The output is a fully connected layer with softmax, which allows us to use the cross-entropy loss function.

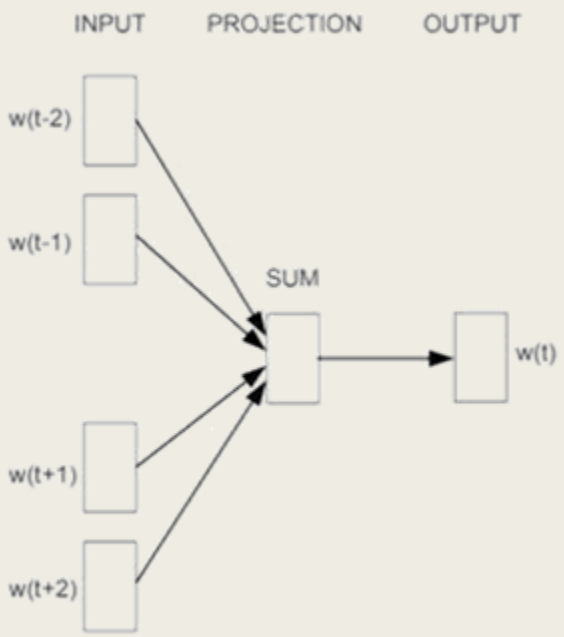


Word2Vec

- So far, this method helps in predicting just the next word which does not give a lot of context to understand each one.
- Thus we introduce the two common methods of Word2Vec
  - *Continuous bag of words (CBOW)*
  - *Skip-gram*

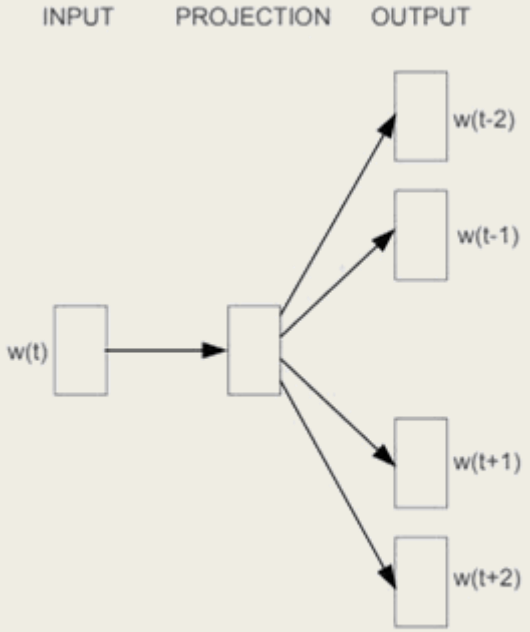
Word2Vec

Continuous Bag of Words (CBOW): Increases context by using the target word as the output, and use the surrounding words as the input.



CBOW

Skip-gram: Increases context by using the target word as the input, and use the surrounding words as the output.



Skip-gram

Word2Vec

- Notice, since the output layer of this Word2Vec neural network is a softmax classifier, then we are essentially trying to maximize the probability of the target word given the context (CBOW) or the probability of the context word given the target word (skip-gram).
  - *CBOW: maximizing* $P(w_t|w_{t+c};\theta)$
  - *Skip-gram: maximizing* $P(w_{t+c}|w_t;\theta)$
- More formally, (e.g., for skip-gram) our objective function is:

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-j\leq c\leq j, c\neq 0} \log P(w_{t+c}|w_t;\theta)$$

- Our loss function is:

$$L = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-j\leq c\leq j, c\neq 0} \log P(w_{t+c}|w_t;\theta)$$

Word2Vec: Pros and Cons

- Pros:
  - *Word2vec preserves relationships between words, and gets good results in deep learning applications*
  - *Quick and easy to train*
  - *Overcomes curse of dimensionality (selection of number of neurons)*
  - *Pre-trained embeddings used in a host of applications*
  - *Self-supervised (no need to manually label anything)*
- Cons:
  - *If new words are introduced to vocabulary, will require re-training*
  - *Global information is not preserved (solved by GloVe)*
  - *Doesn't work well for morphologically rich languages*

Global Vectors (GloVe)

- While Word2Vec is a predictive model, relying on local context windows, GloVe is a count-based model, relying on global co-occurrence counts to derive word embeddings.
- GloVe uses matrix factorization techniques on the word co-occurrence matrix.
- Matrix factorization is a process used to decompose a matrix into a product of two or more smaller matrices.
- For example can use SVD ($X = USV^{\mathrm{T}}$, thus can use U as the dimensionally reduced matrix and use the vectors therein as the word vectors).

Global Vectors (GloVe)

- However, SVD computational cost scales quadratically for n by m matrix, bad for millions of words or documents.
- To make this scalable, can use an objective function

$$J(\theta) = \sum_{i,j=1}^{V} f(P_{ij})(u_i^T v_j - log P_{ij})^2$$

  - P_{ij} *is the co-occurrence matrix*
  - f *is a weighing function to down-weight the influence of very frequent co-occurrent words*
  - u, v *are the word vectors for the target and context words*
- Final representation is $u + v$

Global Vectors (GloVe)

- An approach to evaluate the resulting word vectors:
  - *Intrinsic evaluation: Can use word vector analogies and measure the cosine distances to recover the word vector with the greatest analogy.*
  - *Cosine distance:* $d = 1 - \frac{a^T b}{\|a\|\|b\|} = 1 - \frac{\sum_{i=1}^{n} a_i b_i}{\sqrt{\sum_{i=1}^{n} a_i^2} \cdot \sqrt{\sum_{i=1}^{n} b_i^2}}$
  - *Example: man:woman :: king:?*
  - *Applying cosine distance:*

$$\underset{x_i \in X}{\operatorname{argmax}} \frac{\left(x_{woman} - x_{man} + x_{king}\right)^T x_i}{\left\|x_{woman} - x_{man} + x_{king}\right\| \|x_i\|}$$

We expect the result to be x_{queen}.

Notes/References

- Word2Vec + Code: https://towardsdatascience.com/skip-gram-neural-network-from-scratch-485f2e688238
- Word embedding and word2vec: https://www.youtube.com/watch?v=viZrOnJclY0&ab_channel=StatQuestwithJoshStarmer
- Word2vec – CBOW and skip-gram: https://www.youtube.com/watch?v=UqRCEmrv1gQ&ab_channel=TheSemicolon
- GloVe: https://www.youtube.com/watch?v=ASn7ExxLZws&t=2391s&ab_channel=StanfordUniversitySchoolofEngineering