Deep Learning for NLP

Week 4: QA & word embeddings 1

Sharid Loáiciga — May 12th, 2020

Reminder

- A1 is due today

Data splitting

- Always split your data into:
    - training set (usually 70%)
    - development set (15%)
        - you need to predict something during training to know how the model is doing, error analysis
        - parameter choice, final model choice
    - test set (15%)
        - forget about it during training
        - don’t use it until the very end
        - measures (unbiased) performance
- Pytorch (mostly) takes care of this for us; scikit-learn too.

Why is the bias always 1?

- The bias is an additional neuron added (as in addition) to the layers and set to 1.
- Each neuron takes the input and multiplies it by a weight. So if you input 0, it’s not possible to output 2. By adding the bias (+1) you move the entire value of the activation function.

SGD or Adam?,

is there a non-stochastic GD?

- SGD is an optimization algorithm:
  - gradient-based optimizers
  - derivative-free optimizers

- Adam is also a gradient-based optimizer, https://arxiv.org/pdf/1412.6980.pdf
  - variant of SGD
  - supposedly more robust (better results independently of other hyper-parameters)
  - converges faster

- If you're interested in optimizers, I recommend:

https://ruder.io/optimizing-gradient-descent/index.html

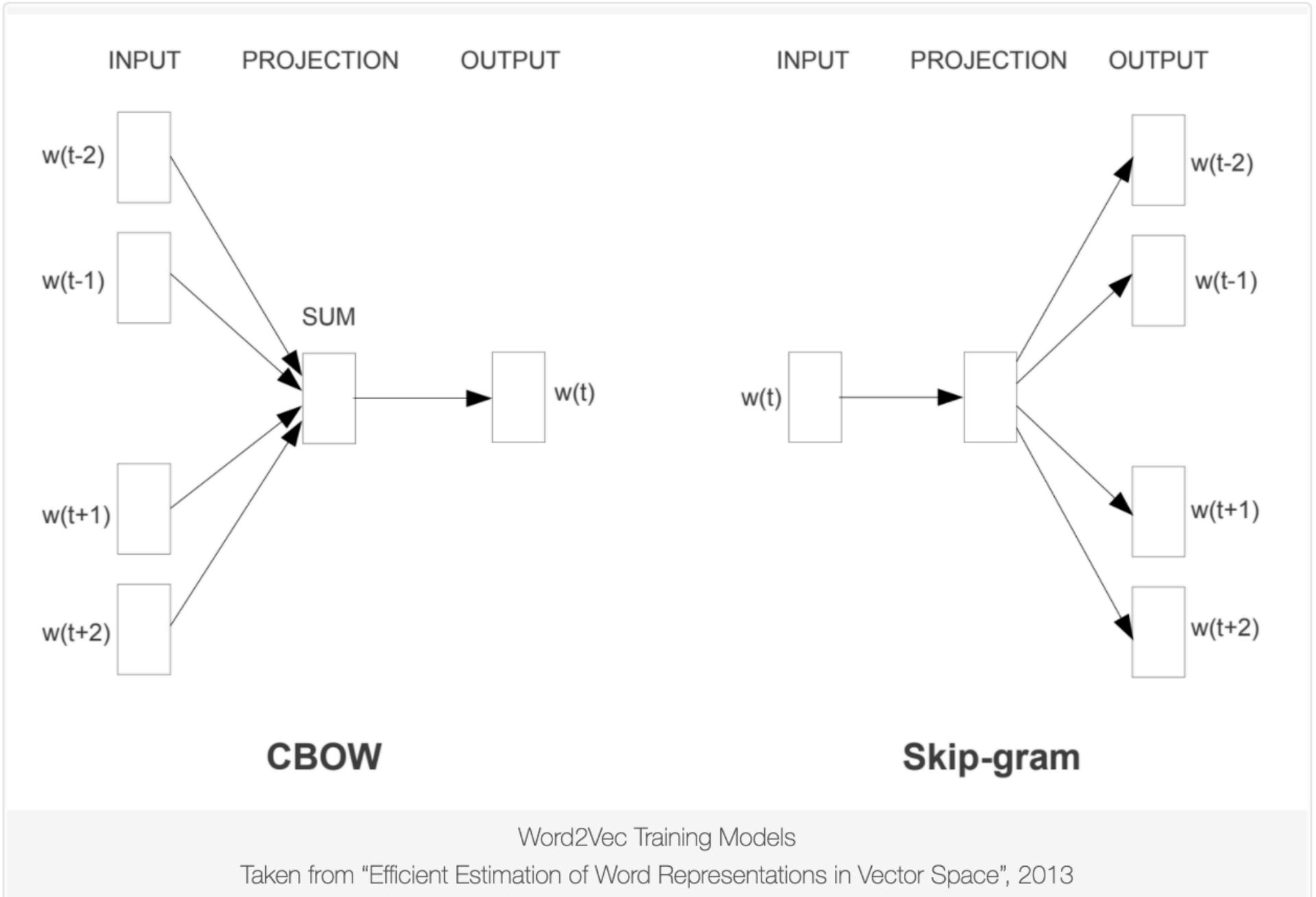
Why can’t the loss be 0?

- loss = 0, means that $\hat{y} = y$, you’ve memorized your data
- keep a margin of loss
- early stopping as regularization technique

Pytorch basics

https://stackoverflow.com/questions/57248777/backward-function-in-pytorch

CBOW vs Skip-gram



Word2Vec Training Models

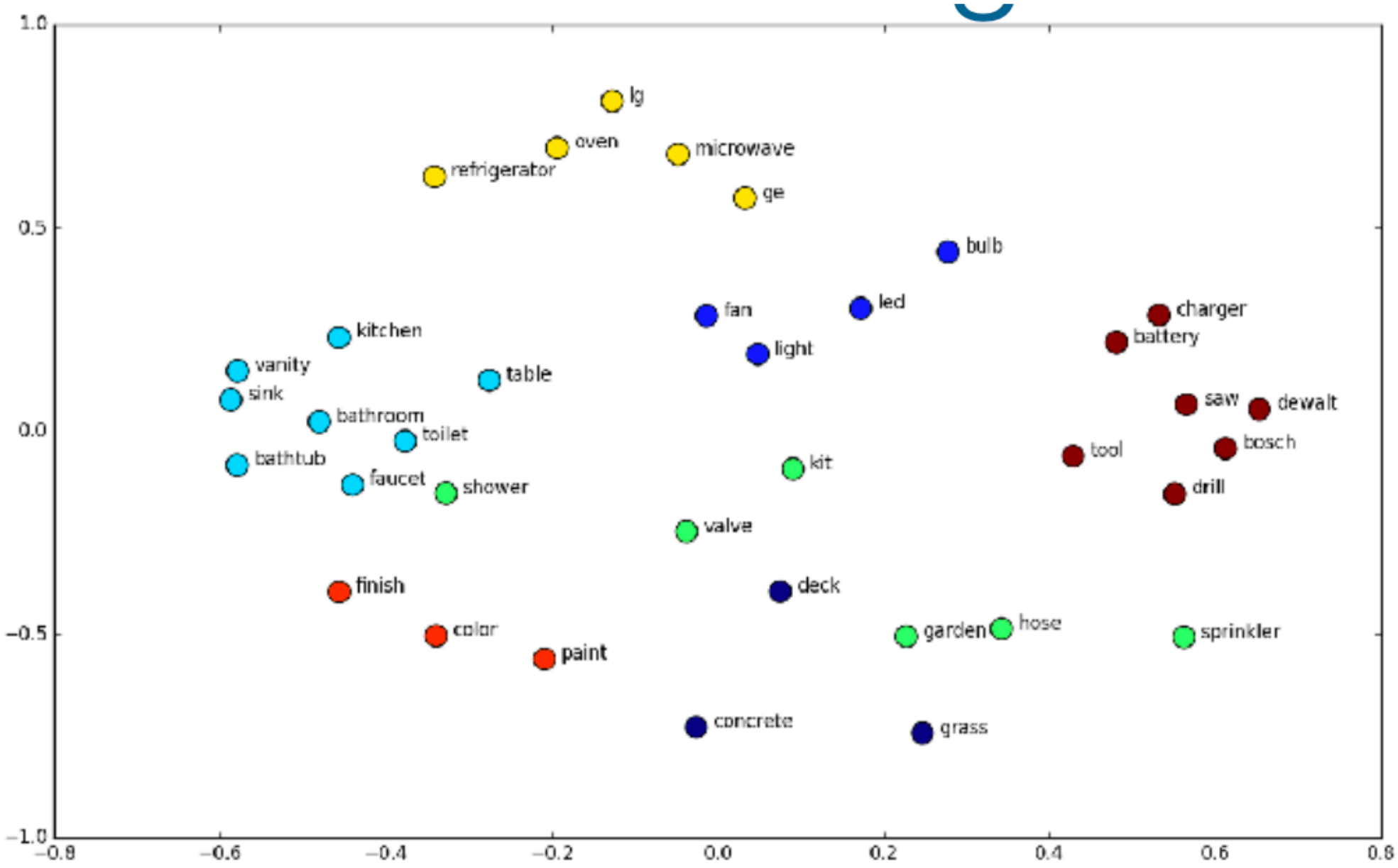
Taken from "Efficient Estimation of Word Representations in Vector Space", 2013

https://machinelearningmastery.com/what-are-word-embeddings/

Word Embeddings

Slides credits: Miikka Silfverberg & Hande Celikkanat

Geometrical representation of meaning



Word embeddings are geometrical representations of word meaning

Context encodes meaning

During domestication, cats have undergone only minor changes in anatomy and behavior

cats are similar in anatomy to the other felid species

Compared to other felines, domestic cats have narrowly spaced canine teeth, which is an adaptation to their preferred prey of small rodents, which have small vertebrae.

What's the missing word?

"You shall know a word by the company it keeps" (Firth, J. R. 1957:11)

Context window representation

`Context window: 5`

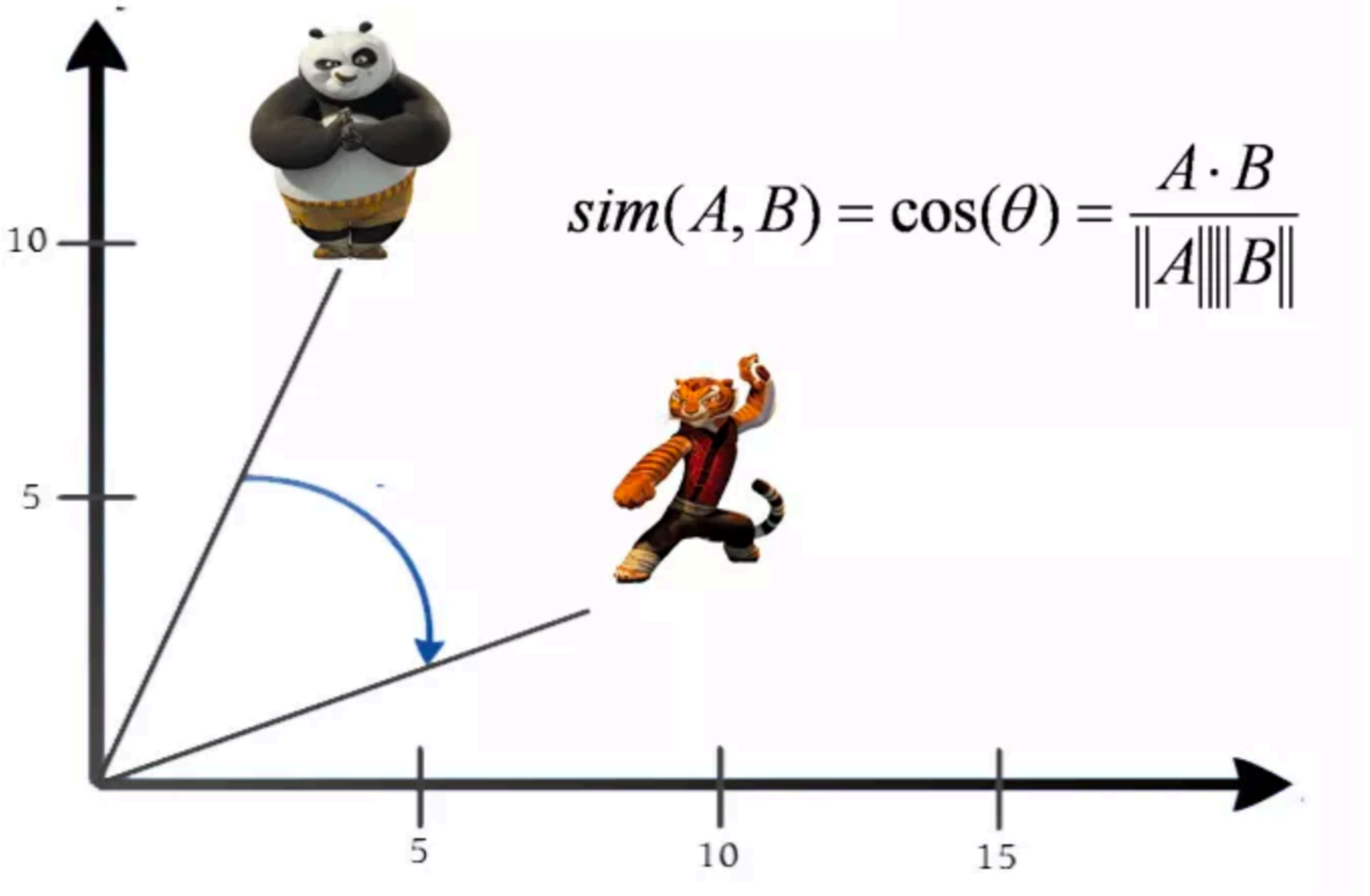
During domestication, cats **have undergone** only minor changes in anatomy and behavior

Compared to other **felines**, **domestic** cats **have narrowly** spaced canine teeth, which is an adaptation to their preferred prey of small rodents, which have small vertebrae. …

	during	among	domestication	work	have	undergone	minor	domestic	…
cats	1	0	1	0	2	1	0	1	…
dogs	1	0	3	2	2	1	0	2	…

Semantically related words get similar vectors.

Cosine similarity



$$sim(A, B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

We can measure the similarity of embedding vectors

Positive point-wise mutual information

Function words dominate raw counts

Problem: These representations are adversely affected by very common words like “a”, “the”.

	during	among	domestication	work	have	undergone	minor	domestic	and	…
cats	1	2	15	0	30	3	0	34	492	…

Solution: Weight words more when they seem to be strongly associated with the target word:

$$\text{PPMI}(\text{domestic}, \text{cats}) = \log \frac{\text{p}(\text{domestic},\text{cats})}{\text{p}(\text{domestic})\cdot\text{p}(\text{cats})} = 4.96$$

$$\text{PPMI}(\text{and}, \text{cats}) = \log \frac{\text{p}(\text{and},\text{cats})}{\text{p}(\text{and})\cdot\text{p}(\text{cats})} = 1.21$$

Content words dominate PPMI counts

	during	among	domestication	work	have	undergone	minor	domestic	and
cats	0.34	1.10	3.56	1.23	1.00	2.01	1.10	4.96	1.21

Problem with sparse representations

The main problem with sparse representations is that they have tens of thousands of dimensions.

It becomes slow to use them in a model.

Another problem:

Some words only occur with “big” and others with “great”.
Our model doesn’t consider the words similar because it doesn’t know that the **context words** “big” and “great” are related.

Dense representations

We can use dimensionality reduction techniques to go from sparse (maybe 50,000-dimensional) embedding vectors down to 100-300-dimensional ones.

Typically on uses Singular Value Decomposition

Problems with SVD

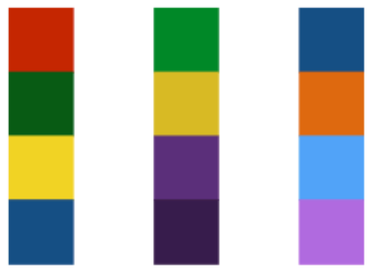
Unfortunately, SVD is very slow for large matrices.
If we have vocabularies of millions of words types,
SVD becomes too slow

For a m-by-n matrix (m > n), SVD has complexity $O(mn^2)$.

When m and n are around 1,000,000, this is huge.

A language modeling approach

A cat **sat on the** mat

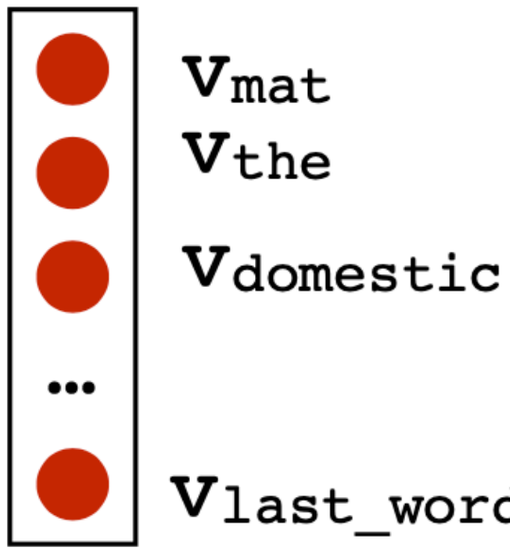


Optimize:

$$-\log \frac{\exp(c \cdot v_{\text{mat}})}{\sum_{w \in V} \exp(c \cdot v_{\text{w}})}$$

Context vector:

c



We can use v_w as word embeddings.

Problem: The sum in the denominator is **HUGE**!

Negative sampling skipgram model

word2vec

Trying to keep your clothing free of cat hair isn't worth the effort

Train the model to score actual context words w_O (free, hair) higher than randomly sampled words w_i (computer, thus, …)

No need to compute a pesky sum over the entire vocabulary!

http://bionlp-www.utu.fi/wv_demo/

Pretrained embeddings

Typically you train word embeddings on huge datasets:

Google 300 dimensional word embeddings trained on 100 billion tokens from GoogleNews.

This allows us to derive knowledge about words which do not occur in our own task-specific dataset.

We get knowledge about “awesome” because its embedding will be similar to “great”.