Machine Learning

11. Representation learning

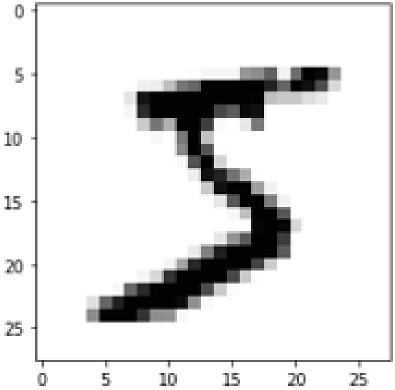
Yannick Le Cacheux

CentraleSupélec - Université Paris Saclay

September 2024

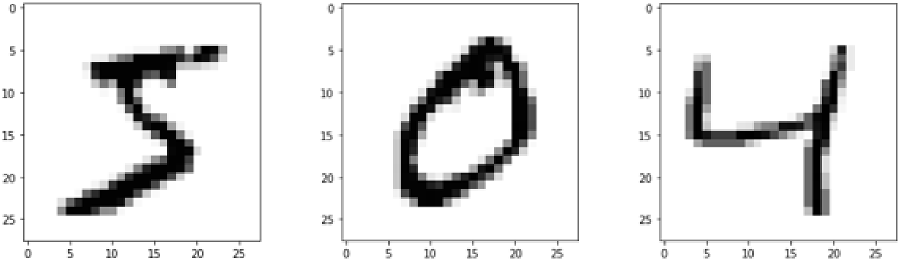
How do we represent images?

- Most obvious solution: as an array (or a tensor) of pixels




```
[  0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
   0.,   3.,  18.,  18.,  18., 126., 136., 175.,  26., 166., 255.,
 247., 127.,   0.,   0.,   0.,   0.],
[  0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,  30.,  36.,  94.,
 154., 170., 253., 253., 253., 253., 253., 225., 172., 253., 242.,
 195.,  64.,   0.,   0.,   0.,   0.],
[  0.,   0.,   0.,   0.,   0.,   0.,   0.,  49., 238., 253., 253.,
 253., 253., 253., 253., 253., 253., 251.,  93.,  82.,  82.,  56.,
  39.,   0.,   0.,   0.,   0.,   0.],
[  0.,   0.,   0.,   0.,   0.,   0.,   0.,  18., 219., 253., 253.,
 253., 253., 253., 198., 182., 247., 241.,   0.,   0.,   0.,   0.,
   0.,   0.,   0.,   0.,   0.,   0.],
[  0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,  80., 156., 107.,
 253., 253., 205.,  11.,   0.,  43., 154.,   0.,   0.,   0.,   0.,
   0.,   0.,   0.,   0.,   0.,   0.],
[  0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,  14.,   1.,
 154., 253.,  90.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
   0.,   0.,   0.,   0.,   0.,   0.],
[  0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
 139., 253., 190.,   2.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
   0.,   0.,   0.,   0.,   0.,   0.],
[  0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
  11., 190., 253.,  70.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
   0.,   0.,   0.,   0.,   0.,   0.],
[  0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,   0.,
   0.,  35., 241., 225., 160., 108.,   1.,   0.,   0.,   0.,   0.,
   0.,   0.,   0.,   0.,   0.,   0.],
```

How do we represent images?



- We need an additional condition to train most machine learning algorithms we have seen so far (*e.g.* logistic regression, SVM, PCA. . .)

How do we represent images?

- All images should have the same size if we want to be able to train a model with a fixed number of parameters
- This is the same as representing images as a fixed-size vector $\mathbf{x} \in \mathbb{R}^D$

How do we represent words?

tiger crocodile supercalifragilisticexpialidocious

- In general, as a string, *i.e.* a list of characters
- But can we train machine learning algorithms from a list of characters with variable size?
  - $\rightarrow$ (Actually, yes, we can with architectures such as recurrent neural networks. But this is not always convenient or suitable)
- The problem is the same with sentences
- Ideally, we would like to represent a word as a fixed-size vector $\mathbf{x} \in \mathbb{R}^D$

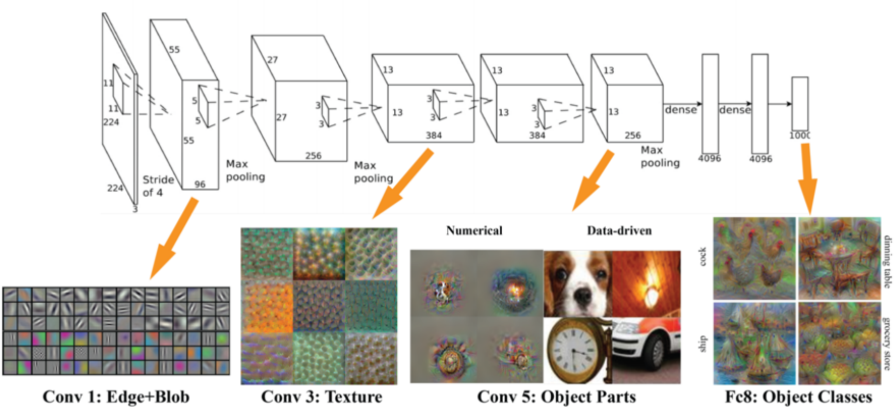
Transfer learning

Main idea

We first train a model on source task different from the target task. The idea is that the model may still acquire useful abilities on this task.

Transfer learning with CNN

These high-level representation
may be useful for similar tasks



Conv 1: Edge+Blob
Conv 3: Texture
Conv 5: Object Parts
Fc8: Object Classes

These kernels may be useful for many tasks

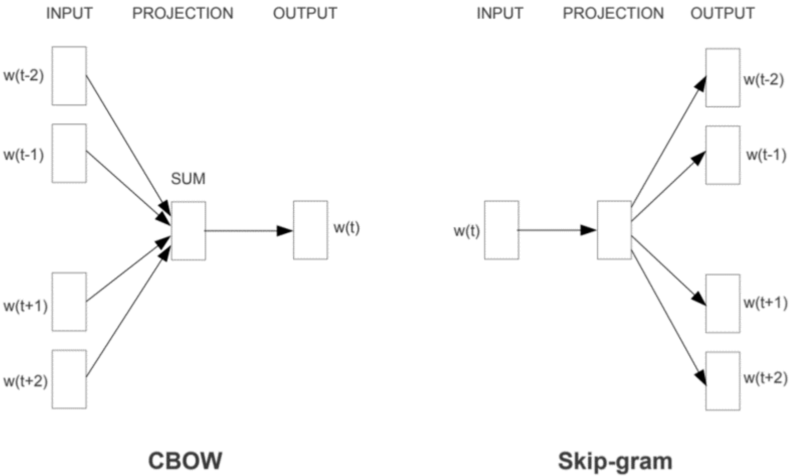
Word embedding

- Main idea: learn to predict a word from its context, or the opposite

Long live the <u>king</u>! Long live the <u>queen</u>!

- Synonyms or similar words should have very similar contexts
- We hope that representations of words learned for this specific task will be useful for other tasks
- One big advantage of this task: it doesn't require any human annotation, we can just use large text corpora

CBOW and skipgram

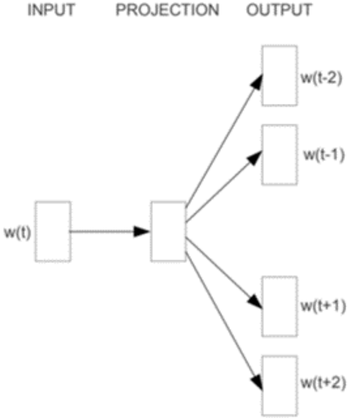


Continuous Bag of Words (CBOW) vs Skip-gram architectures

Skipgram objective

- Given a corpus of T words and a context window we want to maximize

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



Skip-gram

Skipgram architecture

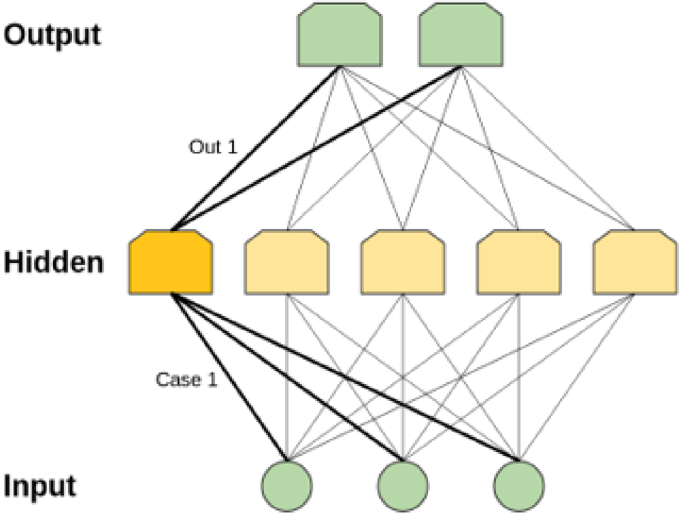
- For each word w, we learn an "input" vector $\mathbf{v}_w$ and an "output" vector $\mathbf{v}'_w$
- The input vector will be the word embedding

$$p(w_i|w_t) = \frac{\exp(\mathbf{v}_{w_t}^\top \mathbf{v}'_{w_i})}{\sum_w \exp(\mathbf{v}_{w_t}^\top \mathbf{v}'_w)}$$

- Question: does it remind you of something?

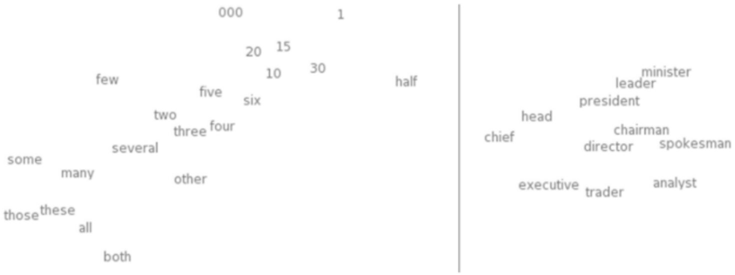
Skipgram architecture: an alternative view

- Answer: a one hidden layer neural net



Word representations

- Resulting word embeddings have very interesting properties

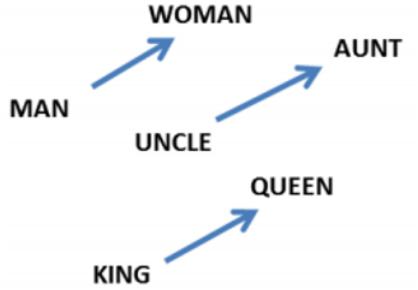


t-SNE visualizations of word embeddings. Left: Number Region; Right: Jobs Region. From Turian *et al.* (2010)

t-SNE visualizations of word embeddings. Left: Number Region; Right: Jobs Region. From Turian *et al.* (2010)

Word representations

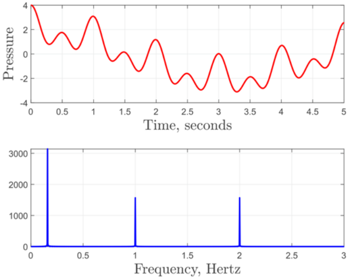
- Resulting word embeddings have very interesting properties



$$W(\text{"woman"}) - W(\text{"man"}) \simeq W(\text{"aunt"}) - W(\text{"uncle"})$$

$$W(\text{"woman"}) - W(\text{"man"}) \simeq W(\text{"queen"}) - W(\text{"king"})$$

Frequency analysis

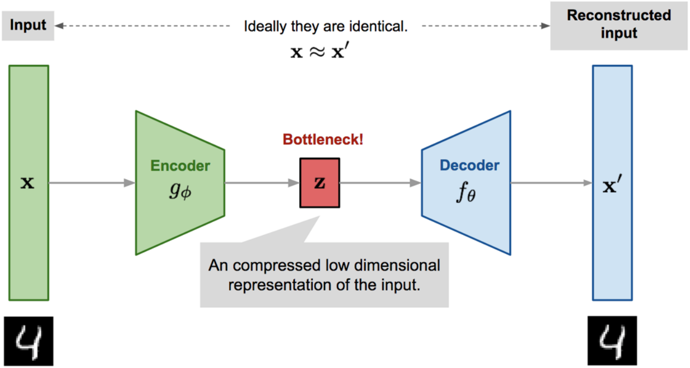


From www.mwmresearchgroup.org/blog/key-concepts-fourier-transforms-and-signal-processing

- Fourier transforms can summarize periodic information in a signal

Autoencoder

- An auto-encoder aims to map and reconstruct the input to and from a "compressed" low-dim representation



From `lilianweng.github.io/posts/2018-08-12-vae/`

- PCA is a form of auto-encoder

Self- and semi-supervised learning

Self-supervised learning

A machine learning approach where the model learns useful representations from unlabeled data by solving pretext tasks that generate labels automatically from the data itself

Semi-supervised learning

A learning paradigm that uses a small amount of labeled data alongside a large amount of unlabeled data to improve the model's performance by leveraging the structure in the unlabeled data.