APS360 Fundamentals of AI

Lisa Zhang

Lecture 9; June 6, 2019

Agenda

Last time:

- Preventing Overfitting
- Transpose Convolutions
- Autoencoder

Today:

- Lab 4; One-hot encoding of categorical variables
- Review autoencoder
- Word Embeddings

Lab 4

Lab 4 Task

- The task in **Lab 4** is to train a variation of an autoencoder on **categorical** and **continuous features**.
- Not images!
- Machine learning practitioners like using images because **humans** have good intuitions about images, and can verify neural network results

One-hot encoding

A way to convert categorical features into numerical features:

One-hot encoding

A way to convert categorical features into numerical features:

Example: At UofT, the categorical feature "Term" can take on three possible values: Fall, Winter, Summer

```
Fall   -> [1, 0, 0]
Winter -> [0, 1, 0]
Summer -> [0, 0, 1]
```

One-hot encoding

A way to convert categorical features into numerical features:

Example: At UofT, the categorical feature "Term" can take on three possible values: Fall, Winter, Summer

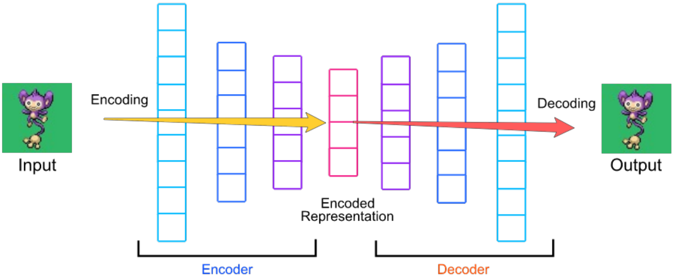
```
Fall   -> [1, 0, 0]
Winter -> [0, 1, 0]
Summer -> [0, 0, 1]
```

Use three numerical features to represent the categorical variable "Term"

We already used one-hot encodings in multi-class classification, to encode the target label

Review Autoencoder

Autoencoder Representation



The output of the encoder is a reduced dimension representation of some data (e.g. the image of an MNIST digit)

Each point in this embedding space (latent space) represents an MNIST digit, which can be recovered using the **decoder**.

Structure in the Autoencoder Representation

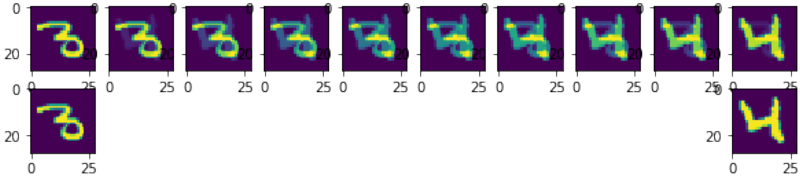
- Points that are close to each other in the latent embedding space will have similar reconstructions (continuity of the decoder)
- So, the encoder will learn to map "similar" images close to each other (due to the bottleneck)

Therefore, distances in the autoencoder representation will become meaningful

- Example: interpolation example from last class
- Example: https://www.youtube.com/watch?v=XNZIN7Jh3Sg

Interpolating in the Latent Embedding Space

- Compared with interpolating in the **pixel** space
- When we interpolate the pixels of an image, the interpolated images do not look like elements of the training set



Size of the latent space: too small

The **size of the latent space** is the number of output neurons that the encoder has.

Q: What if the size of the latent space of the autoencoders is too small?

Size of the latent space: too small

The **size of the latent space** is the number of output neurons that the encoder has.

Q: What if the size of the latent space of the autoencoders is too small?

Poor reconstruction (underfitting)

Size of the latent space: too large

Q: What if the size of the latent space of the autoencoders is too large?

Hint: What if the size of the latent space is equal to the size of the training set?

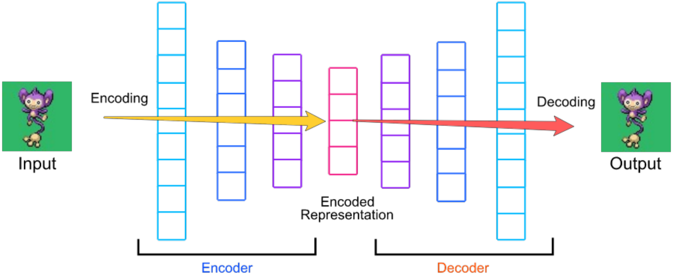
Size of the latent space: too large

Q: What if the size of the latent space of the autoencoders is too large?

Hint: What if the size of the latent space is equal to the size of the training set?

Decoder memorizes training set (overfitting)

Autoencoder Reconstruction

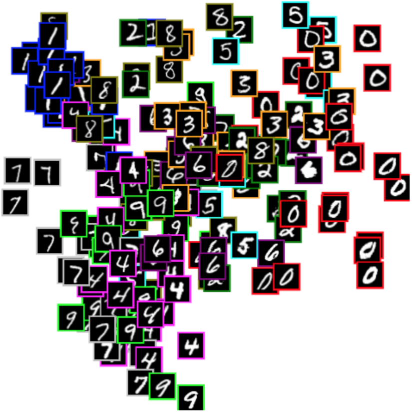


Q: Will an arbitrary image (very different from images in the training set) have a good reconstruction?

Autoencoder Use Cases

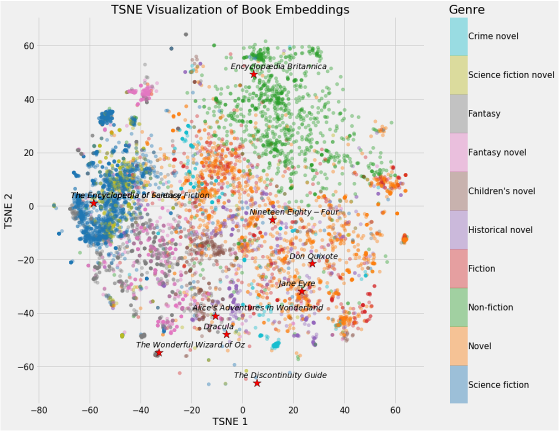
- Generate new data (using the decoder)
- Transfer learning (using the encoder similar to the way we used AlexNet)
  - encoder weights are trained to retain a lot of information (for reconstruction)
  - AlexNet weights are trained to retain only information relevant to classification
- Clustering in the latent space (using encoder)
- Denoising an image (encoder-decoder)

Denoising Autoencoder Example



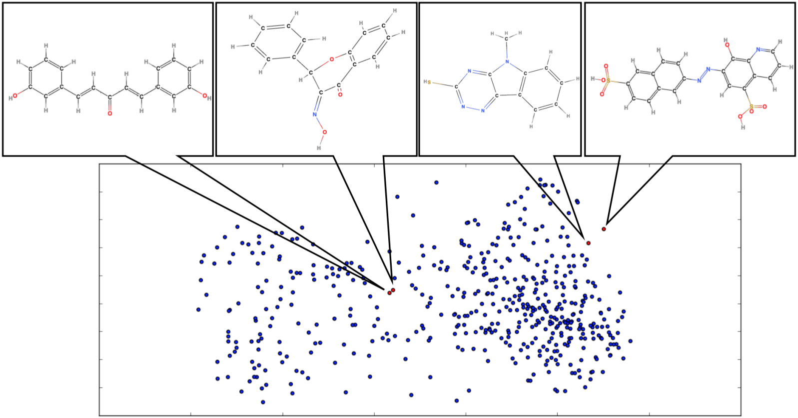
https://cs.stanford.edu/people/karpathy/convnetjs/demo/autoencod

Embedding of Books



https://towardsdatascience.com/neural-network-embeddings-explain

Embedding of Molecules



https://openreview.net/pdf?id=BkSqjHqxg

How to train embeddings

- **Encoder**: data -> embedding
- **Decoder**: embedding -> data

How to train embeddings (alternative encoder-decoder architecture)

- **Encoder**: data -> embedding
- **Decoder**: embedding -> ~~data~~ some *feature* of the data

How to train embeddings (denoising)

- **Encoder**: noisy data -> embedding
- **Decoder**: embedding -> denoised data

Word Embeddings

History

- The term “Word embedding” coined in 2003 (Bengio et al.)
- word2vec model proposed in 2013 (Mikolov et al.)
- GloVe vectors released in 2014 (Pennington et al.)

Architecture for Training Word Embedding

- **Encoder**: word(??) -> embedding
- **Decoder**: embedding -> ???

How do we encode the word?

What is our target?

One-hot encoding of words

- Each word has its own "index"
- If there are 10,000 words, there are 10,000 features

One-hot embedding as input to the encoder

- **Encoder**: one-hot embedding -> low-dim embedding
- **Decoder**: low-dim embedding -> ???

Text as sequences

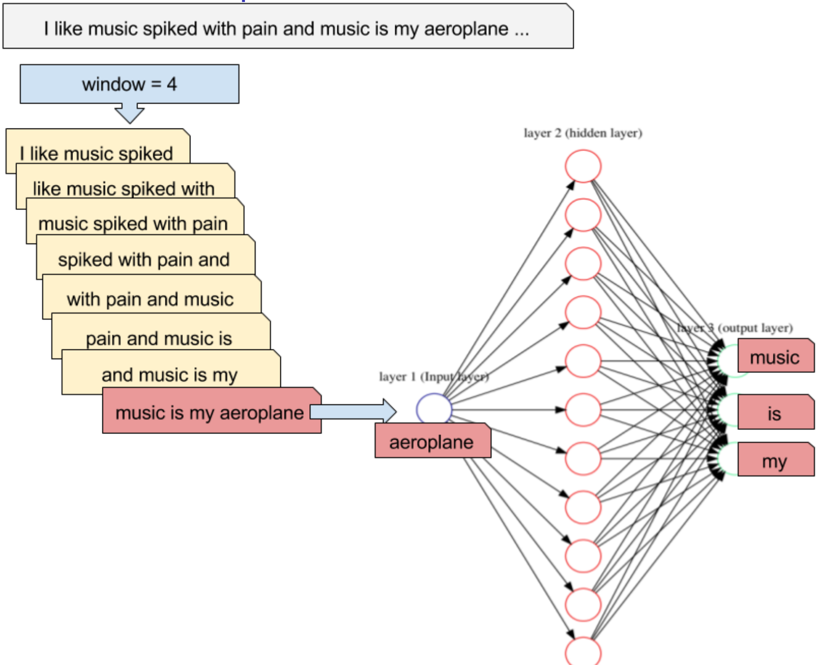
Key idea: the meaning of a word depends on its *context*, or other words that appear *nearby*

There is evidence that children learns new words based on their surrounding words.

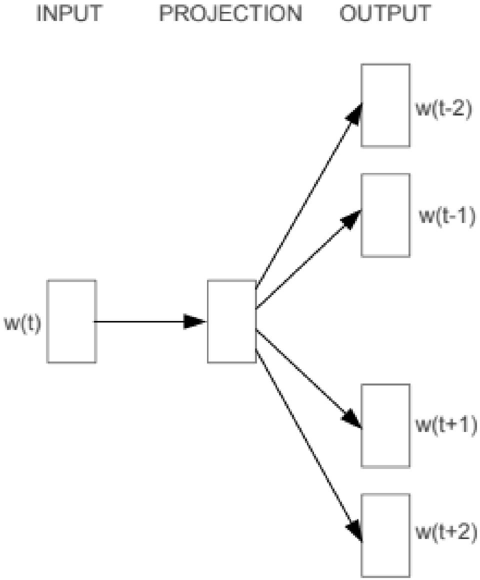
Architecture of a word2vec model

- **Encoder**: one-hot embedding -> low-dim embedding
- **Decoder**: low-dim embedding -> **nearby words**

Architecture Example



Architecture Example: Skip-Gram Model



Structure of the Embedding Space

- Words that have **similar context words** will be mapped to similar embeddings

GloVe Embeddings

- **word2vec** is a family of architecture used to learn word embeddings (i.e. a word2vec model)
- **GloVe** is a set of trained word embeddings that someone else already trained (i.e. like AlexNetweights)

Course Coverage

- We will not train our own word embeddings in this course
- We will not discuss the specific of word2vec models and their variations
- Instead, we will use pre-trained GloVe embeddings

You are not expected to know about specific word2vec model architectures.

You are expected to have intuition about GloVe embeddings, which we will talk about now. . .

GloVe Embeddings

Let's look at some!