Attention and Transformers

CISC 7026: Introduction to Deep Learning

University of Macau

Agenda

1. GNN Review
2. VAE Review and Coding
3. Attention
4. Keys and Queries
5. Transformer
6. Positional Encoding
7. Coding

Agenda

1. **GNN Review**
2. VAE Review and Coding
3. Attention
4. Keys and Queries
5. Transformer
6. Positional Encoding
7. Coding

GNN Review

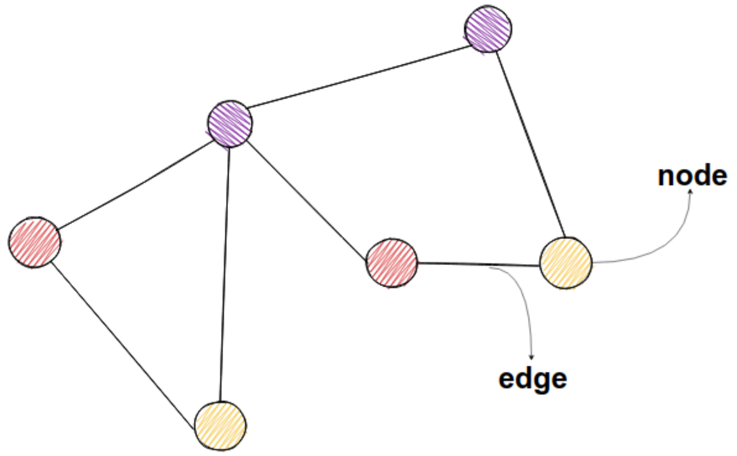
Graph is a structure of nodes (vertices) and edges

$$G = (\boldsymbol{X}, \boldsymbol{E}) \qquad \boldsymbol{X} \in \mathbb{R}^{T \times d_x} \qquad \boldsymbol{E} \in \mathcal{P}(\mathbb{Z}_T \times \mathbb{Z}_T)$$

GNN Review

Graph is a structure of nodes (vertices) and edges

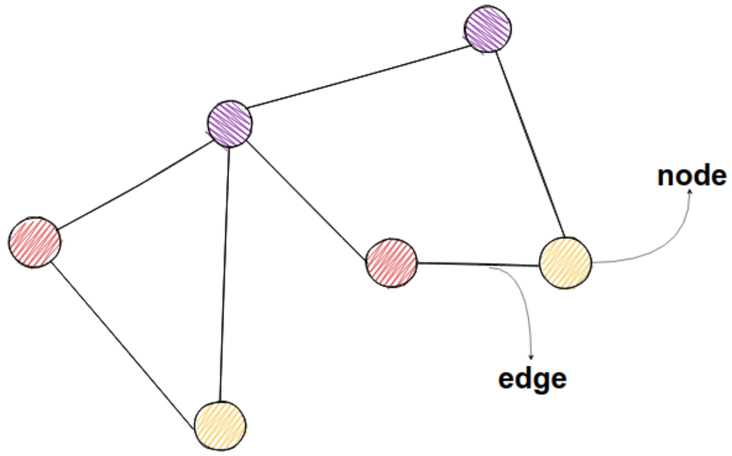
$$G = (\boldsymbol{X}, \boldsymbol{E}) \qquad \boldsymbol{X} \in \mathbb{R}^{T \times d_x} \qquad \boldsymbol{E} \in \mathcal{P}(\mathbb{Z}_T \times \mathbb{Z}_T)$$



GNN Review

Graph is a structure of nodes (vertices) and edges

$$G = (\boldsymbol{X}, \boldsymbol{E}) \qquad \boldsymbol{X} \in \mathbb{R}^{T \times d_x} \qquad \boldsymbol{E} \in \mathcal{P}(\mathbb{Z}_T \times \mathbb{Z}_T)$$

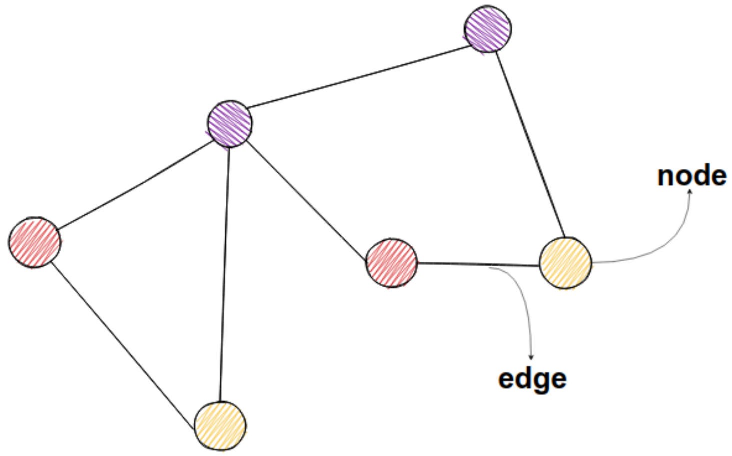


A **node** is a vector of information

GNN Review

Graph is a structure of nodes (vertices) and edges

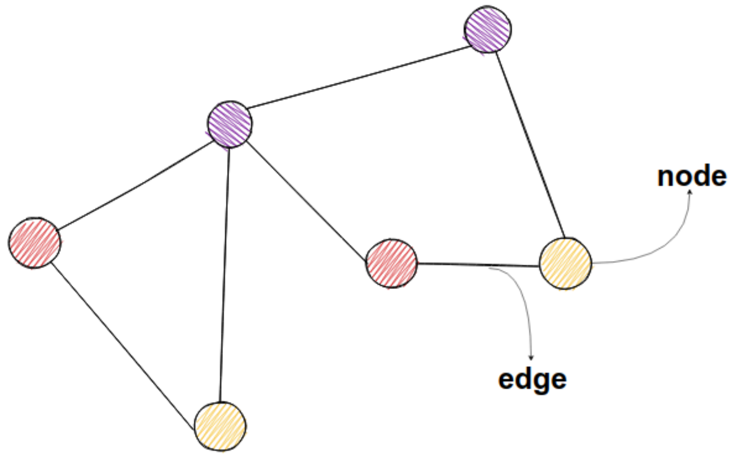
$$G = (\boldsymbol{X}, \boldsymbol{E}) \qquad \boldsymbol{X} \in \mathbb{R}^{T \times d_x} \qquad \boldsymbol{E} \in \mathcal{P}(\mathbb{Z}_T \times \mathbb{Z}_T)$$



A **node** is a vector of information

An **edge** connects two nodes

GNN Review

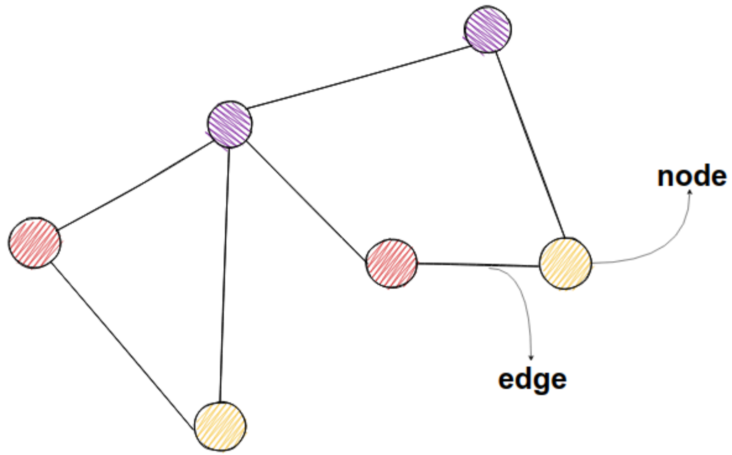


A **node** is a vector of information

An **edge** connects two nodes

If we connect nodes i and j with edge (i, j), then i and j are **neighbors**

GNN Review



A **node** is a vector of information

An **edge** connects two nodes

If we connect nodes i and j with edge (i, j), then i and j are **neighbors**

The **neighborhood** $N(j)$ contains all neighbors of node j

GNN Review

Prof. Li introduced the **graph convolution layer**

GNN Review

Prof. Li introduced the **graph convolution layer**

For a node j, the graph convolution layer is:

GNN Review

Prof. Li introduced the **graph convolution layer**

For a node j, the graph convolution layer is:

$$f(\boldsymbol{X}, \boldsymbol{E}, \boldsymbol{\theta})_j = \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_j + \sum_{i \in N(j)} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right)$$

GNN Review

Prof. Li introduced the **graph convolution layer**

For a node j, the graph convolution layer is:

$$f(\boldsymbol{X}, \boldsymbol{E}, \boldsymbol{\theta})_j = \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_j + \sum_{i \in N(j)} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right)$$

Combine information from current node $\boldsymbol{x}_j$ with neighbors $\boldsymbol{x}_i$

GNN Review

Prof. Li introduced the **graph convolution layer**

For a node j, the graph convolution layer is:

$$f(\boldsymbol{X}, \boldsymbol{E}, \boldsymbol{\theta})_j = \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_j + \sum_{i \in N(j)} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right)$$

Combine information from current node $\boldsymbol{x}_j$ with neighbors $\boldsymbol{x}_i$

This is just one node, we use this graph layer for all nodes in the graph

GNN Review

Graph convolution over all nodes in the graph

GNN Review

Graph convolution over all nodes in the graph

$$f(\boldsymbol{X}, \boldsymbol{E}, \boldsymbol{\theta}) = \begin{bmatrix} f(\boldsymbol{X}, \boldsymbol{E}, \boldsymbol{\theta})_1 \\ f(\boldsymbol{X}, \boldsymbol{E}, \boldsymbol{\theta})_2 \\ \vdots \\ f(\boldsymbol{X}, \boldsymbol{E}, \boldsymbol{\theta})_T \end{bmatrix} = \begin{bmatrix} \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_1 + \sum_{i \in N(1)} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \\ \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_2 + \sum_{i \in N(2)} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \\ \vdots \\ \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_T + \sum_{i \in N(T)} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \end{bmatrix}$$

GNN Review

Graph convolution over all nodes in the graph

$$f(\boldsymbol{X}, \boldsymbol{E}, \boldsymbol{\theta}) = \begin{bmatrix} f(\boldsymbol{X}, \boldsymbol{E}, \boldsymbol{\theta})_1 \\ f(\boldsymbol{X}, \boldsymbol{E}, \boldsymbol{\theta})_2 \\ \vdots \\ f(\boldsymbol{X}, \boldsymbol{E}, \boldsymbol{\theta})_T \end{bmatrix} = \begin{bmatrix} \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_1 + \sum_{i \in N(1)} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \\ \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_2 + \sum_{i \in N(2)} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \\ \vdots \\ \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_T + \sum_{i \in N(T)} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \end{bmatrix}$$

How does this compare to regular convolution (images, sound, etc)?

GNN Review

Standard convolution

$$\begin{bmatrix} \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_1 + \sum_{i=1}^{k} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \\ \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_2 + \sum_{i=2}^{k+1} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \\ \vdots \\ \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_T + \sum_{i=T-k}^{T} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \end{bmatrix}$$

GNN Review

Standard convolution

$$\begin{bmatrix} \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_1 + \sum_{i=1}^{k} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \\ \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_2 + \sum_{i=2}^{k+1} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \\ \vdots \\ \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_T + \sum_{i=T-k}^{T} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \end{bmatrix}$$

Graph convolution

$$\begin{bmatrix} \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_1 + \sum_{i \in N(1)} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \\ \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_2 + \sum_{i \in N(2)} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \\ \vdots \\ \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_T + \sum_{i \in N(T)} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \end{bmatrix}$$

GNN Review

Standard convolution

$$\begin{bmatrix} \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_1 + \sum_{i=1}^{k} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \\ \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_2 + \sum_{i=2}^{k+1} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \\ \vdots \\ \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_T + \sum_{i=T-k}^{T} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \end{bmatrix}$$

Graph convolution

$$\begin{bmatrix} \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_1 + \sum_{i \in N(1)} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \\ \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_2 + \sum_{i \in N(2)} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \\ \vdots \\ \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_T + \sum_{i \in N(T)} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \end{bmatrix}$$

Question: What is the output size of standard convolution?

GNN Review

Standard convolution

$$\begin{bmatrix} \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_1 + \sum_{i=1}^{k} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \\ \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_2 + \sum_{i=2}^{k+1} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \\ \vdots \\ \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_T + \sum_{i=T-k}^{T} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \end{bmatrix}$$

Graph convolution

$$\begin{bmatrix} \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_1 + \sum_{i \in N(1)} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \\ \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_2 + \sum_{i \in N(2)} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \\ \vdots \\ \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_T + \sum_{i \in N(T)} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \end{bmatrix}$$

Question: What is the output size of standard convolution?

Answer: $T - k - 1 \times d_h$

GNN Review

Standard convolution

$$\begin{bmatrix} \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_1 + \sum_{i=1}^{k} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \\ \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_2 + \sum_{i=2}^{k+1} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \\ \vdots \\ \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_T + \sum_{i=T-k}^{T} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \end{bmatrix}$$

Graph convolution

$$\begin{bmatrix} \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_1 + \sum_{i \in N(1)} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \\ \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_2 + \sum_{i \in N(2)} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \\ \vdots \\ \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_T + \sum_{i \in N(T)} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \end{bmatrix}$$

Question: What is the output size of graph convolution?

GNN Review

Standard convolution

$$\begin{bmatrix} \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_1 + \sum_{i=1}^{k} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \\ \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_2 + \sum_{i=2}^{k+1} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \\ \vdots \\ \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_T + \sum_{i=T-k}^{T} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \end{bmatrix}$$

Graph convolution

$$\begin{bmatrix} \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_1 + \sum_{i \in N(1)} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \\ \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_2 + \sum_{i \in N(2)} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \\ \vdots \\ \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_T + \sum_{i \in N(T)} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \end{bmatrix}$$

Question: What is the output size of graph convolution?

Answer: $T \times d_h$

GNN Review

We can use pooling with graph convolutions too

$$\text{SumPool}\left(\begin{bmatrix} \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_1 + \sum_{i \in N(1)} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \\ \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_2 + \sum_{i \in N(2)} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \\ \vdots \\ \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_T + \sum_{i \in N(T)} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) \end{bmatrix}\right) =$$

$$\sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_1 + \sum_{i \in N(1)} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) + \sigma\left(\boldsymbol{\theta}_1^\top \overline{\boldsymbol{x}}_2 + \sum_{i \in N(2)} \boldsymbol{\theta}_2^\top \boldsymbol{x}_i\right) + \ldots$$

Agenda

1. **GNN Review**
2. VAE Review and Coding
3. Attention
4. Keys and Queries
5. Transformer
6. Positional Encoding
7. Coding

Agenda

1. GNN Review
2. **VAE Review and Coding**
3. Attention
4. Keys and Queries
5. Transformer
6. Positional Encoding
7. Coding

VAE Review and Coding

Autoencoders are useful for compression and denoising

VAE Review and Coding

Autoencoders are useful for compression and denoising

But we can also use them as **generative models**

VAE Review and Coding

Autoencoders are useful for compression and denoising

But we can also use them as **generative models**

A generative model learns the structure of data

VAE Review and Coding

Autoencoders are useful for compression and denoising

But we can also use them as **generative models**

A generative model learns the structure of data

Using this structure, it generates **new** data

VAE Review and Coding

Autoencoders are useful for compression and denoising

But we can also use them as **generative models**

A generative model learns the structure of data

Using this structure, it generates **new** data

- Train on face dataset, generate **new** pictures

VAE Review and Coding

Autoencoders are useful for compression and denoising

But we can also use them as **generative models**

A generative model learns the structure of data

Using this structure, it generates **new** data

- Train on face dataset, generate **new** pictures
- Train on book dataset, write a **new** book

VAE Review and Coding

Autoencoders are useful for compression and denoising

But we can also use them as **generative models**

A generative model learns the structure of data

Using this structure, it generates **new** data

- Train on face dataset, generate **new** pictures
- Train on book dataset, write a **new** book
- Train on protein dataset, create **new** proteins

VAE Review and Coding

Autoencoders are useful for compression and denoising

But we can also use them as **generative models**

A generative model learns the structure of data

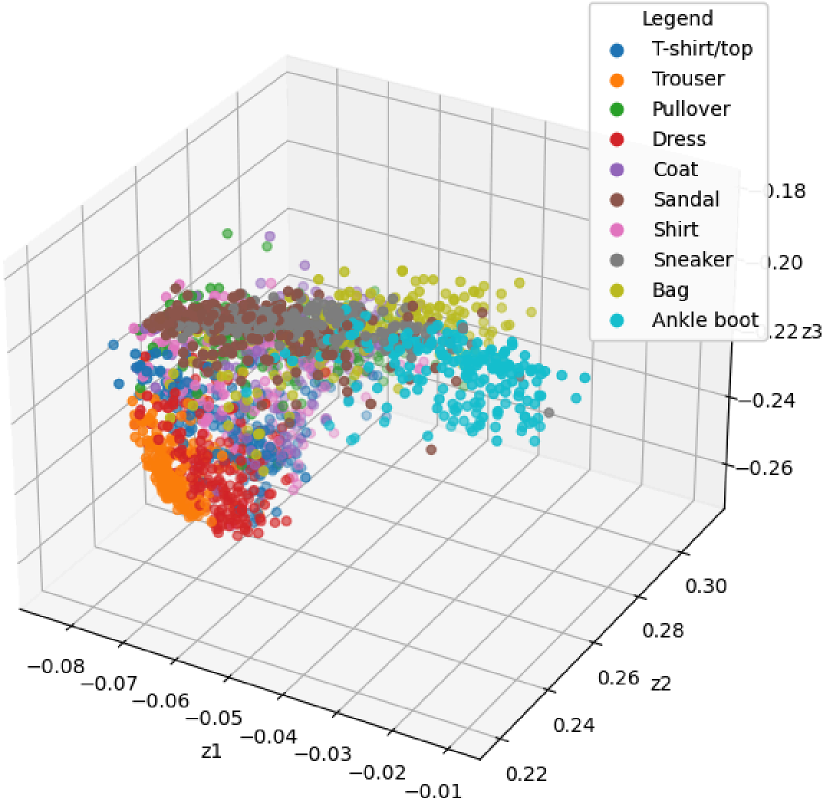
Using this structure, it generates **new** data

- Train on face dataset, generate **new** pictures
- Train on book dataset, write a **new** book
- Train on protein dataset, create **new** proteins

How does this work?

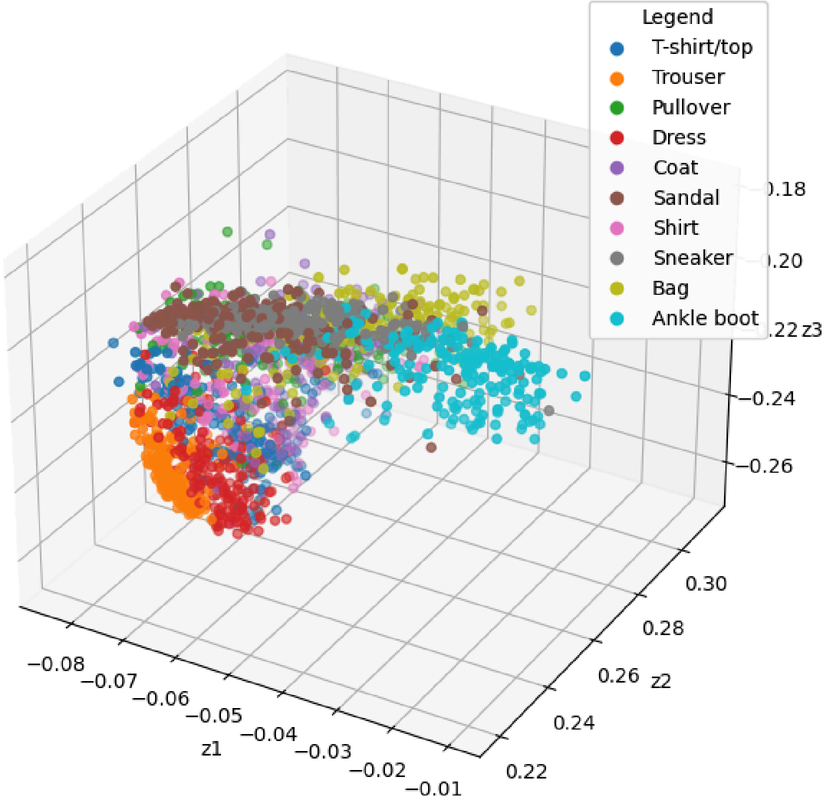
VAE Review and Coding

Latent space Z for autoencoder on the clothes dataset with $d_z = 3$

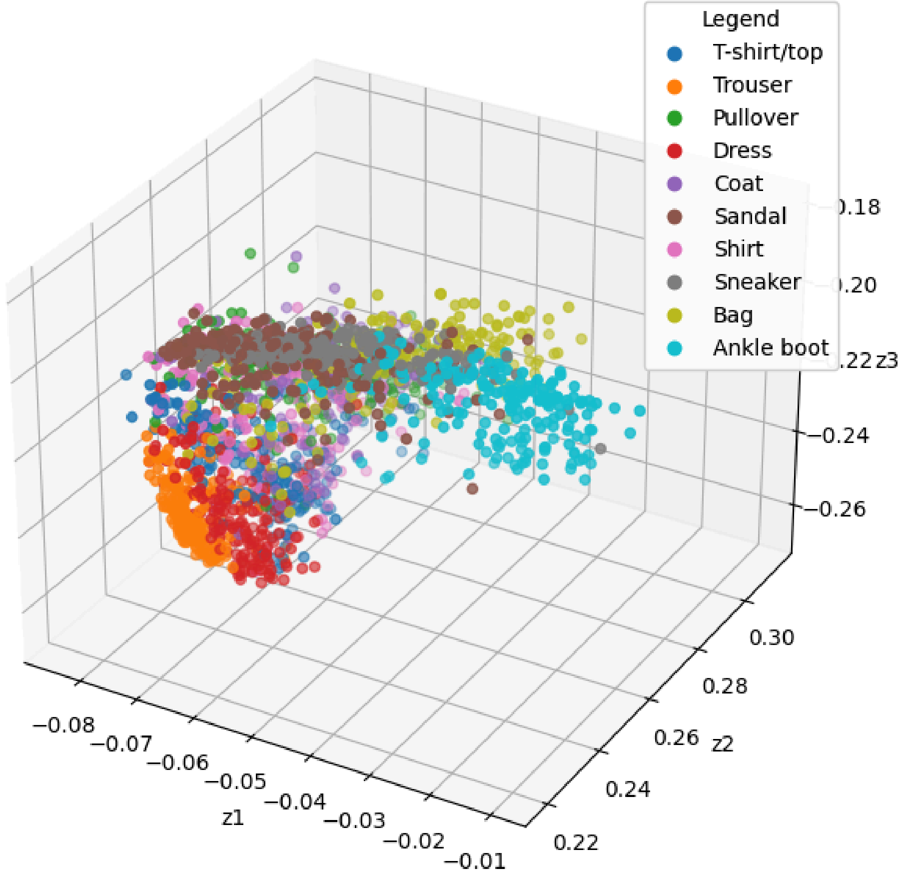


VAE Review and Coding

What happens if we decode a new point?

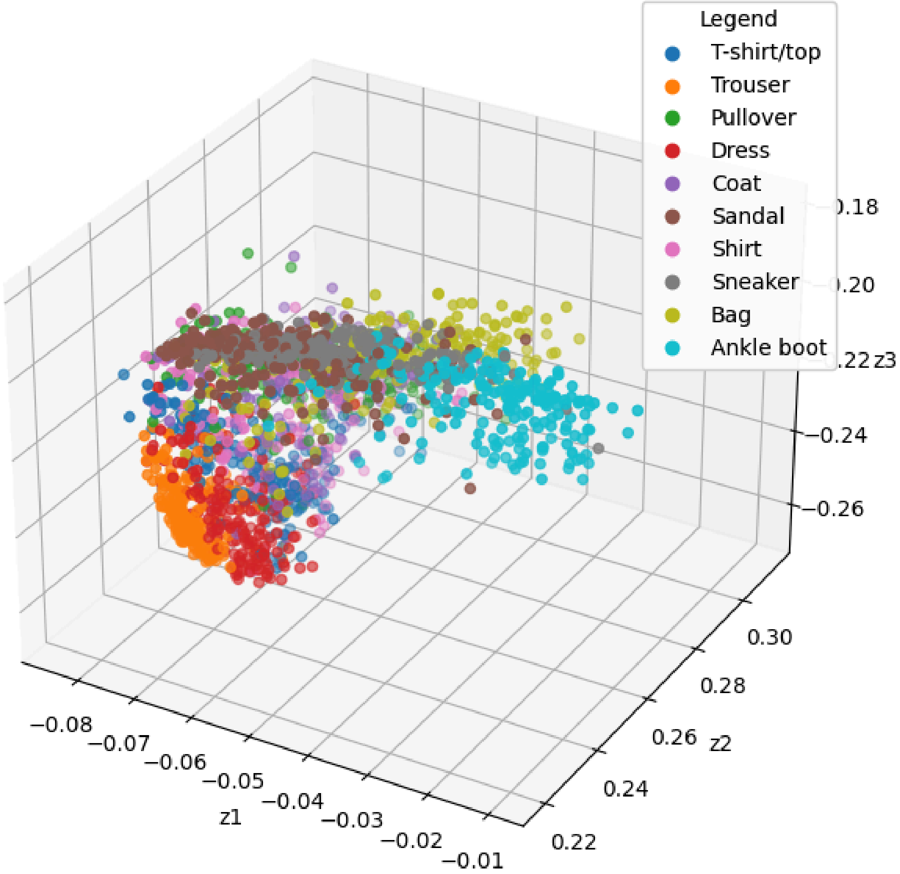


VAE Review and Coding



Autoencoder generative model:

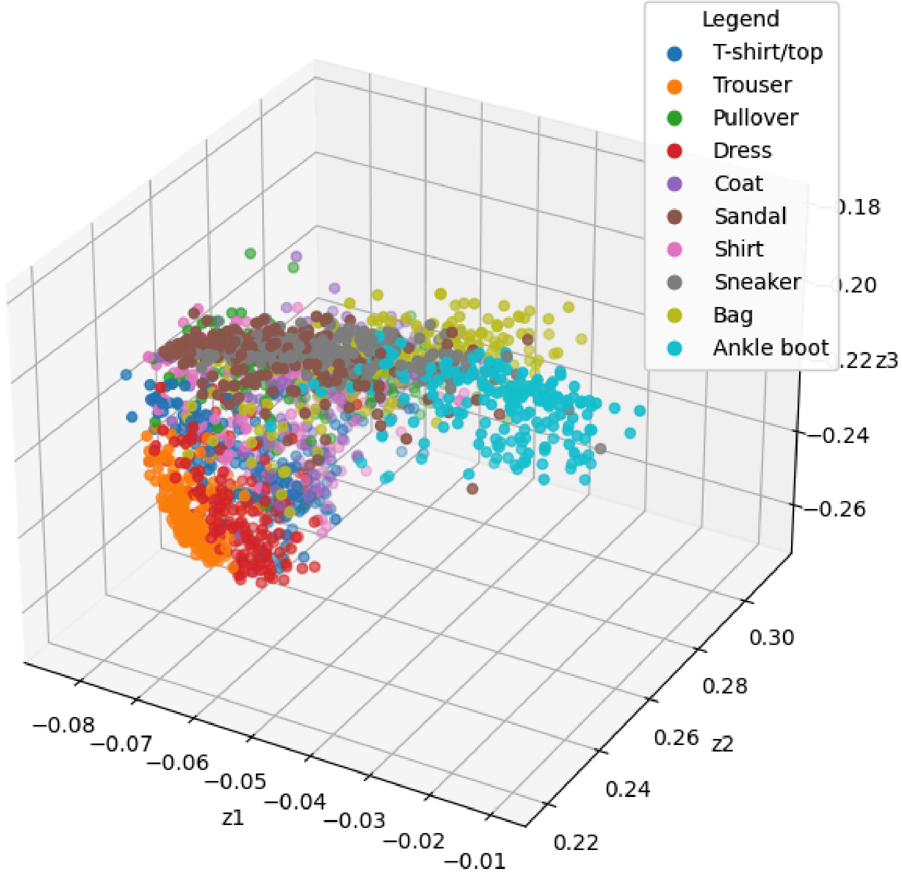
VAE Review and Coding



Autoencoder generative model:

$$\text{Encode} \begin{bmatrix} \boldsymbol{x}_{[1]} \\ \vdots \\ \boldsymbol{x}_{[n]} \end{bmatrix} \text{ into } \begin{bmatrix} \boldsymbol{z}_{[1]} \\ \vdots \\ \boldsymbol{z}_{[n]} \end{bmatrix}$$

VAE Review and Coding

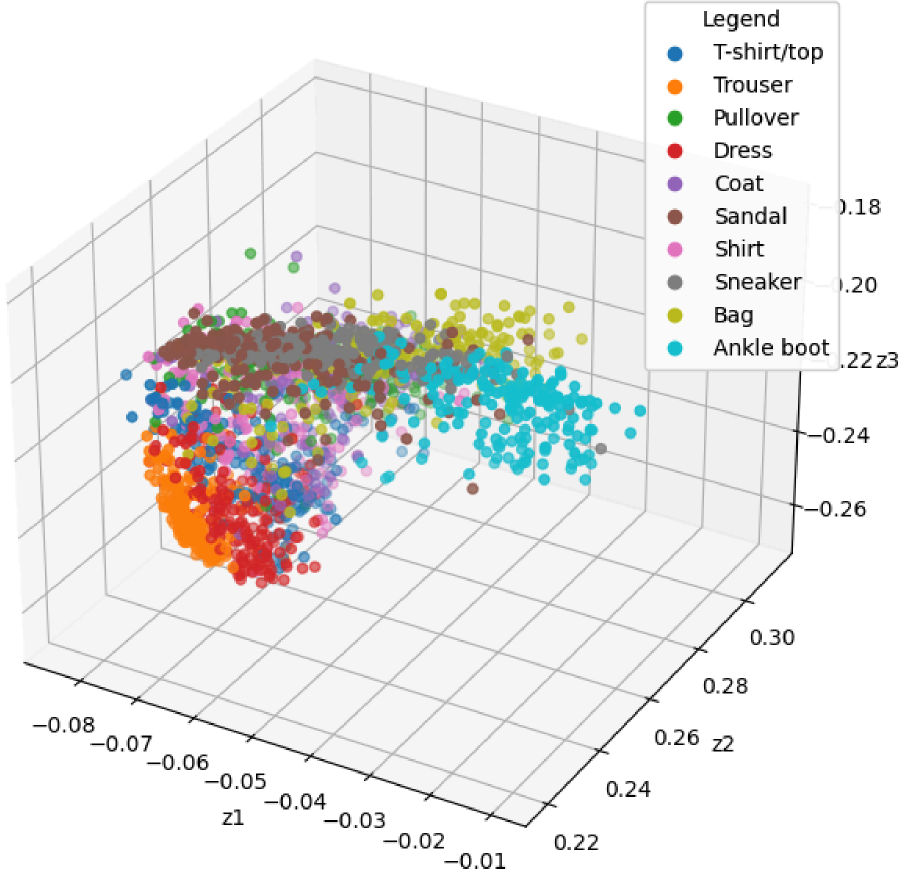


Autoencoder generative model:

Encode $\begin{bmatrix} \boldsymbol{x}_{[1]} \\ \vdots \\ \boldsymbol{x}_{[n]} \end{bmatrix}$ into $\begin{bmatrix} \boldsymbol{z}_{[1]} \\ \vdots \\ \boldsymbol{z}_{[n]} \end{bmatrix}$

Pick a point $\boldsymbol{z}_{[k]}$

VAE Review and Coding



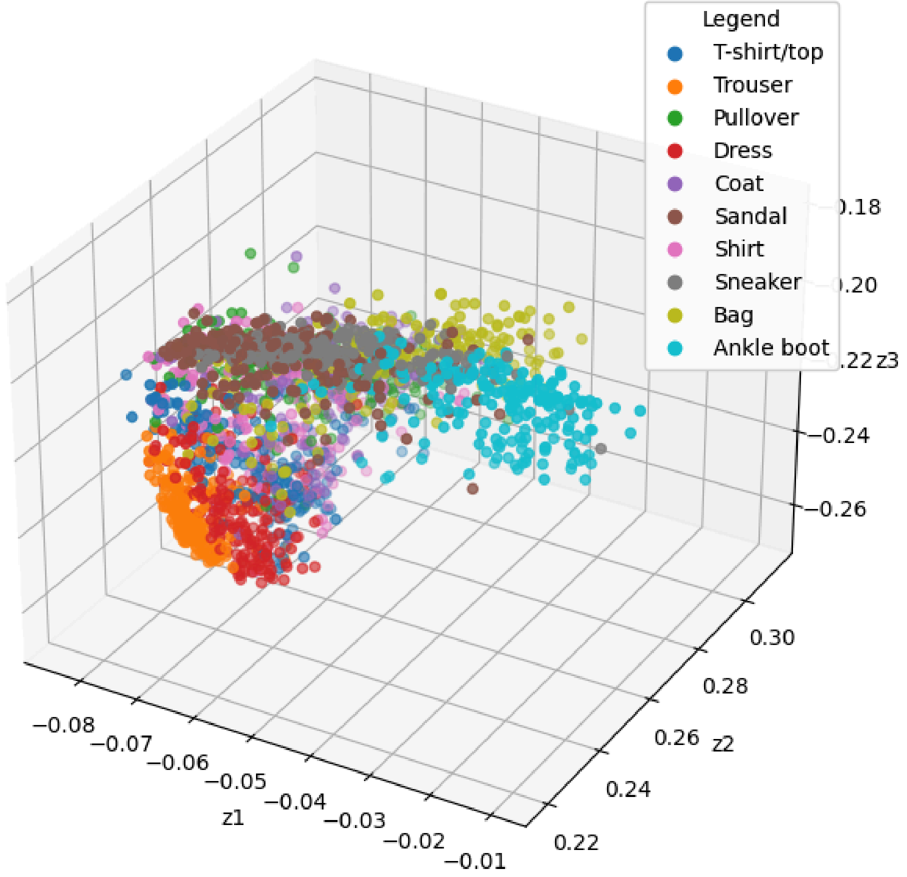
Autoencoder generative model:

Encode $\begin{bmatrix} \boldsymbol{x}_{[1]} \\ \vdots \\ \boldsymbol{x}_{[n]} \end{bmatrix}$ into $\begin{bmatrix} \boldsymbol{z}_{[1]} \\ \vdots \\ \boldsymbol{z}_{[n]} \end{bmatrix}$

Pick a point $\boldsymbol{z}_{[k]}$

Add some noise $\boldsymbol{z}_{\text{new}} = \boldsymbol{z}_{[k]} + \boldsymbol{\varepsilon}$

VAE Review and Coding



Autoencoder generative model:

Encode $\begin{bmatrix} \boldsymbol{x}_{[1]} \\ \vdots \\ \boldsymbol{x}_{[n]} \end{bmatrix}$ into $\begin{bmatrix} \boldsymbol{z}_{[1]} \\ \vdots \\ \boldsymbol{z}_{[n]} \end{bmatrix}$

Pick a point $\boldsymbol{z}_{[k]}$

Add some noise $\boldsymbol{z}_{\text{new}} = \boldsymbol{z}_{[k]} + \boldsymbol{\varepsilon}$

Decode $\boldsymbol{z}_{\text{new}}$ into $\boldsymbol{x}_{\text{new}}$

VAE Review and Coding

VAE Review and Coding

$$f^{-1}(\boldsymbol{z}_k + \varepsilon, \boldsymbol{\theta}_d)$$

VAE Review and Coding

But there is a problem, the **curse of dimensionality**

VAE Review and Coding

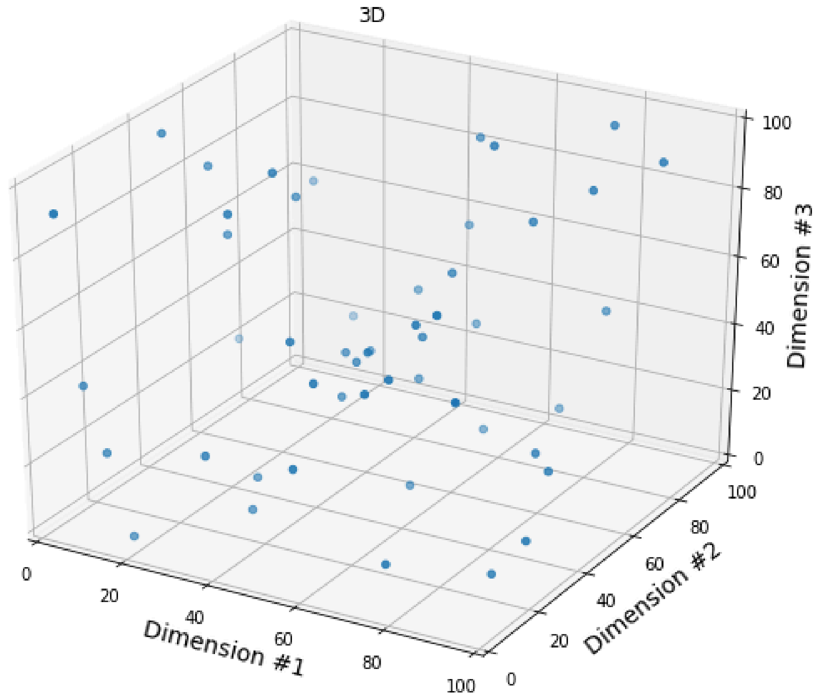
But there is a problem, the **curse of dimensionality**

As d_z increases, points move further and further apart

VAE Review and Coding

But there is a problem, the **curse of dimensionality**

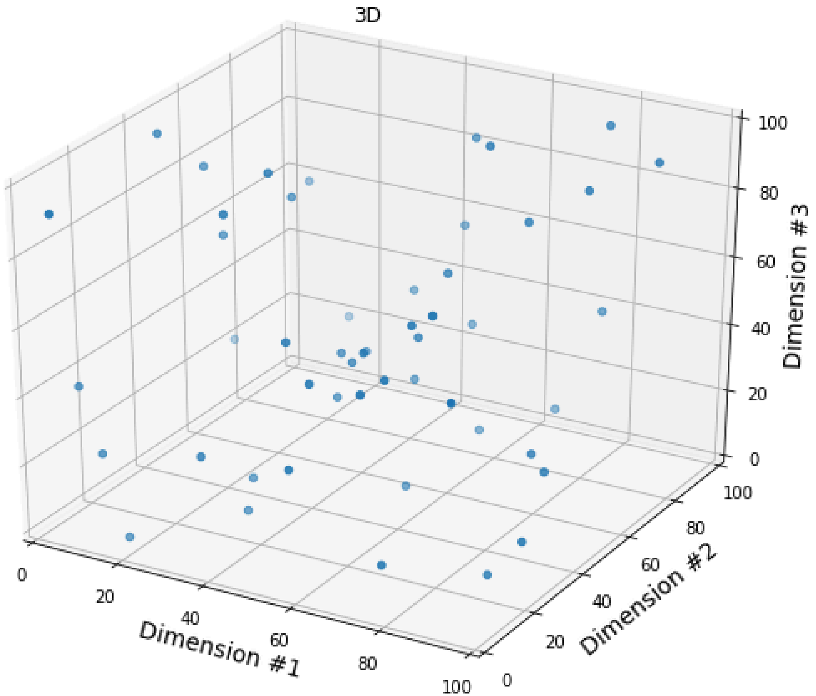
As d_z increases, points move further and further apart



VAE Review and Coding

But there is a problem, the **curse of dimensionality**

As d_z increases, points move further and further apart



$f^{-1}(\boldsymbol{z} + \varepsilon)$ will produce either garbage, or $\boldsymbol{z}$

VAE Review and Coding

Variational autoencoders (VAEs) do three things:

VAE Review and Coding

Variational autoencoders (VAEs) do three things:

1. Make it easy to sample random z

VAE Review and Coding

Variational autoencoders (VAEs) do three things:

1. Make it easy to sample random $\boldsymbol{z}$
2. Keep all $\boldsymbol{z}_{[1]}, ...\boldsymbol{z}_{[n]}$ close together in a small region

VAE Review and Coding

Variational autoencoders (VAEs) do three things:

1. Make it easy to sample random $\boldsymbol{z}$
2. Keep all $\boldsymbol{z}_{[1]}, ...\boldsymbol{z}_{[n]}$ close together in a small region
3. Ensure that $\boldsymbol{z} + \boldsymbol{\varepsilon}$ is always meaningful

VAE Review and Coding

Variational autoencoders (VAEs) do three things:

1. Make it easy to sample random $\boldsymbol{z}$
2. Keep all $\boldsymbol{z}_{[1]}, ...\boldsymbol{z}_{[n]}$ close together in a small region
3. Ensure that $\boldsymbol{z} + \boldsymbol{\varepsilon}$ is always meaningful

How?

VAE Review and Coding

Variational autoencoders (VAEs) do three things:

1. Make it easy to sample random $\boldsymbol{z}$
2. Keep all $\boldsymbol{z}_{[1]}, ...\boldsymbol{z}_{[n]}$ close together in a small region
3. Ensure that $\boldsymbol{z} + \boldsymbol{\varepsilon}$ is always meaningful

How?

Make $\boldsymbol{z}_{[1]}, ..., \boldsymbol{z}_{[n]}$ normally distributed

VAE Review and Coding

Variational autoencoders (VAEs) do three things:

1. Make it easy to sample random $\boldsymbol{z}$
2. Keep all $\boldsymbol{z}_{[1]}, ...\boldsymbol{z}_{[n]}$ close together in a small region
3. Ensure that $\boldsymbol{z} + \boldsymbol{\varepsilon}$ is always meaningful

How?

Make $\boldsymbol{z}_{[1]}, ..., \boldsymbol{z}_{[n]}$ normally distributed

$$\boldsymbol{z} \sim \mathcal{N}(\mu, \sigma), \quad \mu = 0, \sigma = 1$$

VAE Review and Coding

If $\boldsymbol{z}_{[1]}, ..., \boldsymbol{z}_{[n]}$ are distributed following $\mathcal{N}(0, 1)$:

VAE Review and Coding

If $\boldsymbol{z}_{[1]}, ..., \boldsymbol{z}_{[n]}$ are distributed following $\mathcal{N}(0, 1)$:

1. 99.7% of $\boldsymbol{z}_{[1]}, ..., \boldsymbol{z}_{[n]}$ lie within $3\sigma = [-3, 3]$

VAE Review and Coding

If $\boldsymbol{z}_{[1]}, ..., \boldsymbol{z}_{[n]}$ are distributed following $\mathcal{N}(0, 1)$:

1. 99.7% of $\boldsymbol{z}_{[1]}, ..., \boldsymbol{z}_{[n]}$ lie within $3\sigma = [-3, 3]$
2. Make it easy to generate new $\boldsymbol{z}$, just sample $\boldsymbol{z} \sim \mathcal{N}(0, 1)$

VAE Review and Coding

Key idea 1: We want to model the distribution over the dataset X

$$P(\boldsymbol{x}; \boldsymbol{\theta}), \quad \boldsymbol{x} \sim \boldsymbol{X}$$

VAE Review and Coding

Key idea 1: We want to model the distribution over the dataset X

$$P(\boldsymbol{x}; \boldsymbol{\theta}), \quad \boldsymbol{x} \sim \boldsymbol{X}$$

We want to learn $\boldsymbol{\theta}$ that best models the distribution of possible faces

VAE Review and Coding

Key idea 1: We want to model the distribution over the dataset X

$$P(\boldsymbol{x}; \boldsymbol{\theta}), \quad \boldsymbol{x} \sim \boldsymbol{X}$$

We want to learn $\boldsymbol{\theta}$ that best models the distribution of possible faces

Large $P(\boldsymbol{x}; \boldsymbol{\theta})$

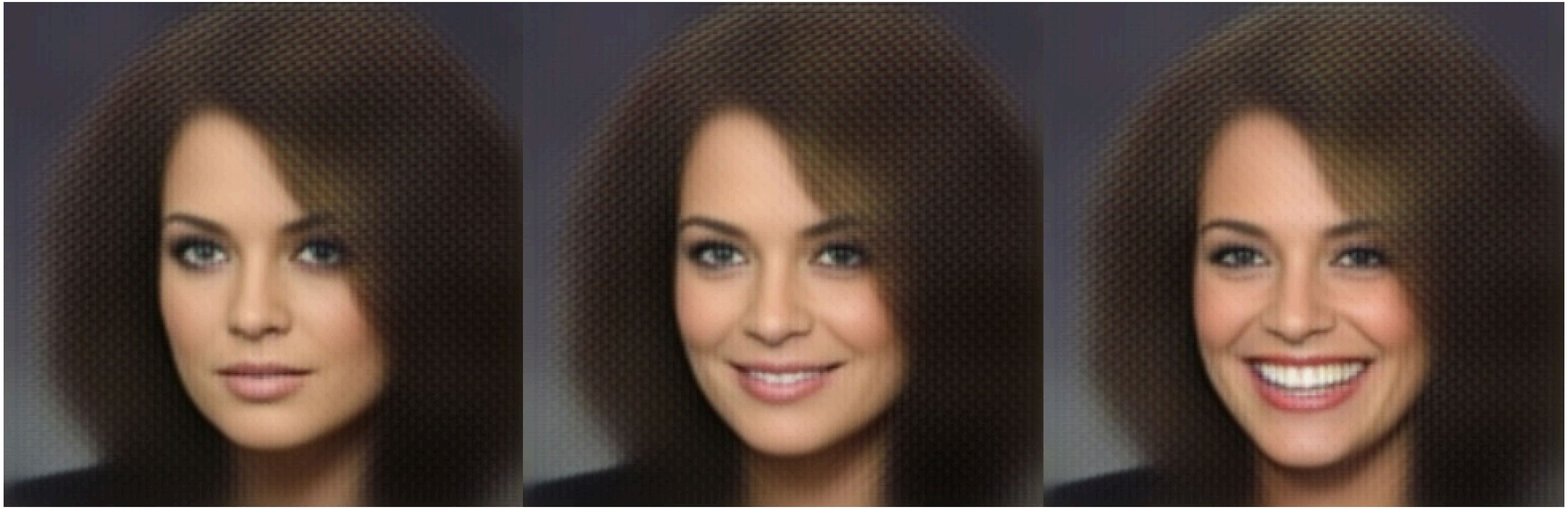
$P(\boldsymbol{x}; \boldsymbol{\theta}) \approx 0$

VAE Review and Coding

Key idea 2: There is some latent variable $\boldsymbol{z}$ which generates data $\boldsymbol{x}$

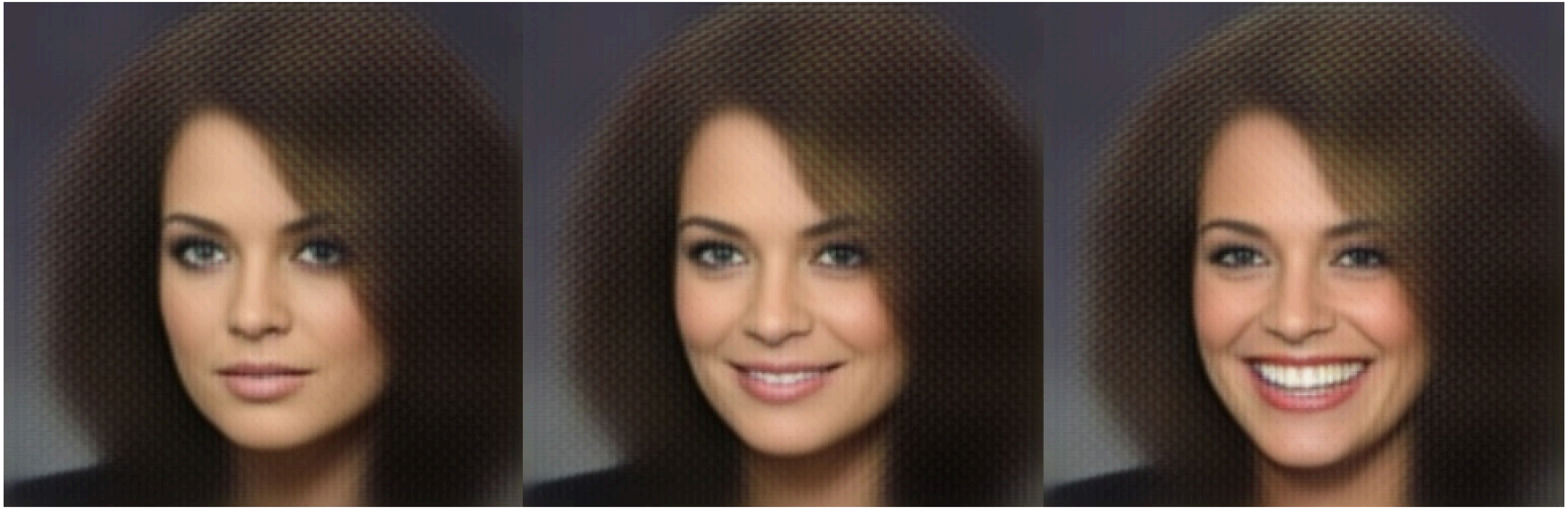
VAE Review and Coding

Key idea 2: There is some latent variable $\boldsymbol{z}$ which generates data $\boldsymbol{x}$

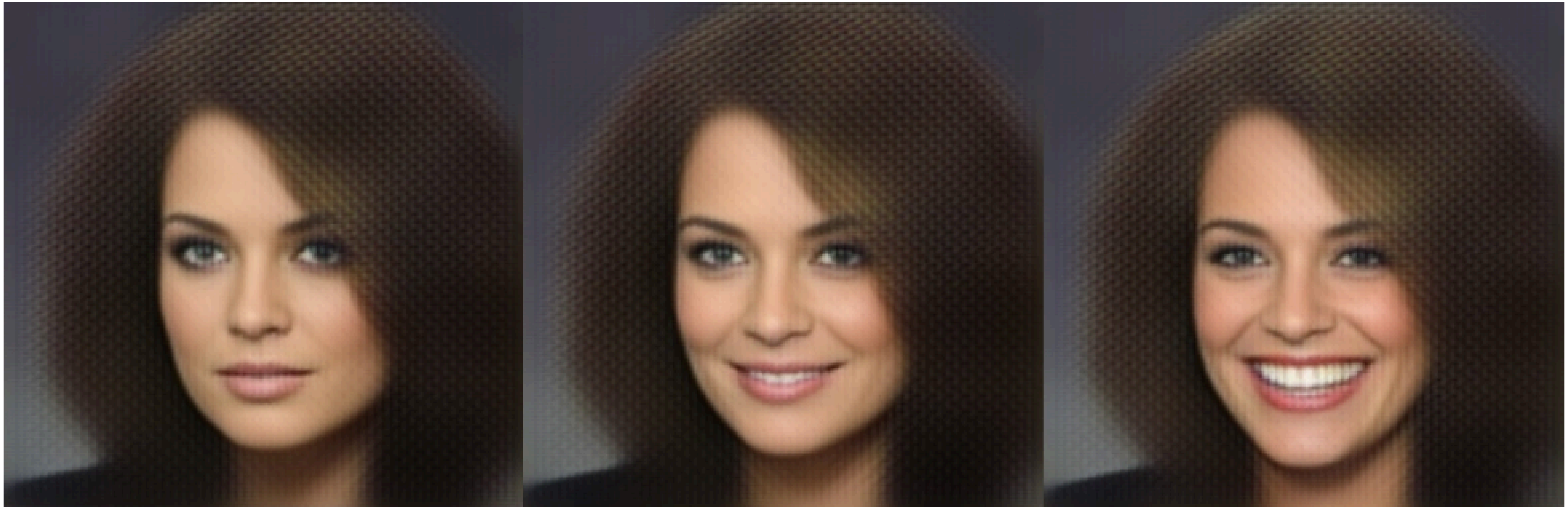
x : 

$$z : [\text{woman} \;\; \text{brown hair} \;\; (\text{frown} \mid \text{smile})]$$

VAE Review and Coding

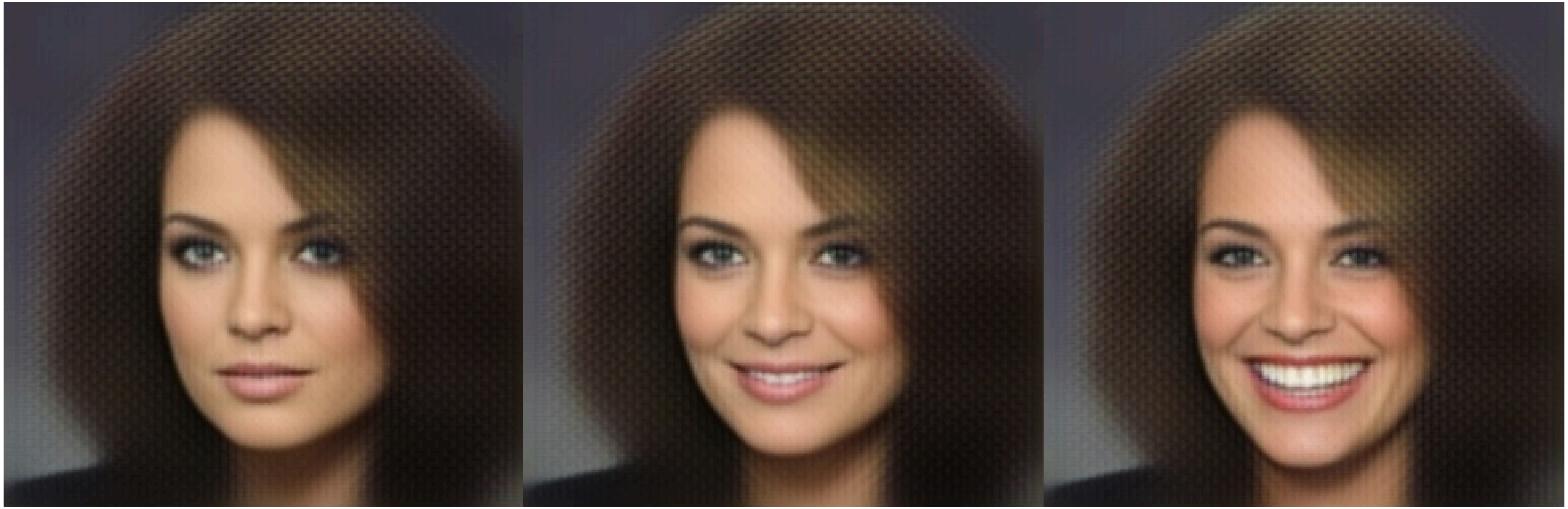


VAE Review and Coding



Network can only see $\boldsymbol{x}$, it cannot directly observe $\boldsymbol{z}$

VAE Review and Coding



Network can only see $\boldsymbol{x}$, it cannot directly observe $\boldsymbol{z}$

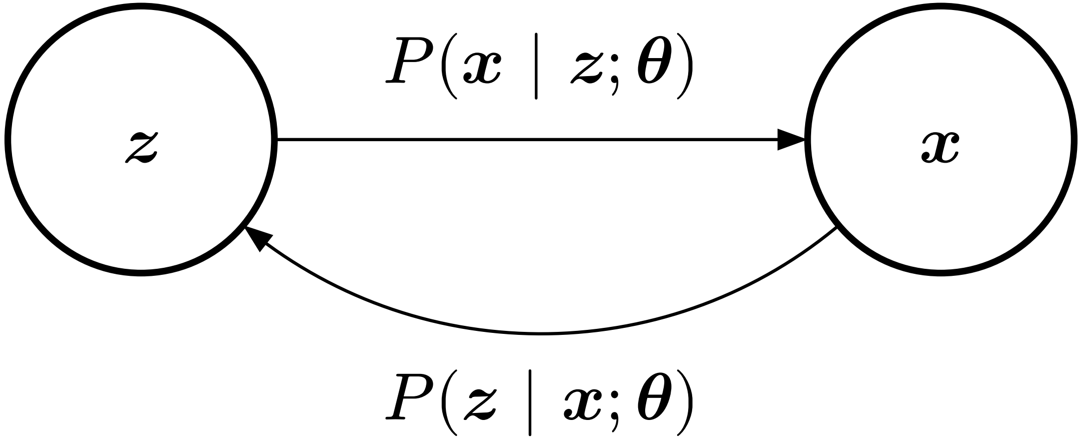
Given $\boldsymbol{x}$, find the probability that the person is smiling $P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta})$

VAE Review and Coding

We cast the autoencoding task as a **variational inference** problem

VAE Review and Coding

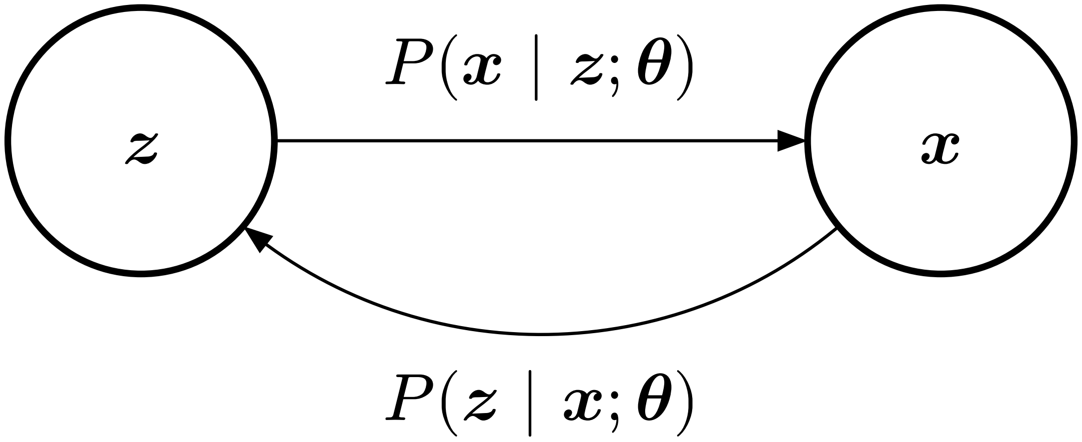
We cast the autoencoding task as a **variational inference** problem



Decoder	Encoder
$P(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta})$	$P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta})$

VAE Review and Coding

We cast the autoencoding task as a **variational inference** problem



Decoder	Encoder
$P(\boldsymbol{x} \mid \boldsymbol{z}; \boldsymbol{\theta})$	$P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta})$

We want to learn both the encoder and decoder: $P(\boldsymbol{z}, \boldsymbol{x}; \boldsymbol{\theta})$

VAE Review and Coding

How do we implement f (i.e., $P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta})$)?

VAE Review and Coding

How do we implement f (i.e., $P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta})$)?

$$f : X \times \Theta \mapsto \Delta Z$$

VAE Review and Coding

How do we implement f (i.e., $P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta})$)?

$$f : X \times \Theta \mapsto \Delta Z$$

Normal distribution has a mean $\mu \in \mathbb{R}$ and standard deviation $\sigma \in \mathbb{R}_+$

VAE Review and Coding

How do we implement f (i.e., $P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta})$)?

$$f : X \times \Theta \mapsto \Delta Z$$

Normal distribution has a mean $\mu \in \mathbb{R}$ and standard deviation $\sigma \in \mathbb{R}_+$

Our encoder should output d_z means and d_z standard deviations

VAE Review and Coding

How do we implement f (i.e., $P(\boldsymbol{z} \mid \boldsymbol{x}; \boldsymbol{\theta})$)?

$$f : X \times \Theta \mapsto \Delta Z$$

Normal distribution has a mean $\mu \in \mathbb{R}$ and standard deviation $\sigma \in \mathbb{R}_+$

Our encoder should output d_z means and d_z standard deviations

$$f : X \times \Theta \mapsto \mathbb{R}^{d_z} \times \mathbb{R}_+^{d_z}$$

VAE Review and Coding

```
core = nn.Sequential(...)
mu_layer = nn.Linear(d_h, d_z)
# Neural networks output real numbers
# But sigma must be positive
# Output log sigma, because e^(sigma) is always positive
log_sigma_layer = nn.Linear(d_h, d_z)
# Alternatively, one sigma for all data
log_sigma = jnp.ones((d_z,))

tmp = core(x)
mu = mu_layer(tmp)
log_sigma = log_sigma_layer(tmp)
distribution = (mu, exp(sigma))
```

VAE Review and Coding

We covered the encoder

$$f : X \times \Theta \mapsto \Delta Z$$

We can use the same decoder as a standard autoencoder

VAE Review and Coding

We covered the encoder

$$f : X \times \Theta \mapsto \Delta Z$$

We can use the same decoder as a standard autoencoder

$$f^{-1} : Z \times \Theta \mapsto X$$

Encoder outputs a distribution ΔZ but decoder input is Z

VAE Review and Coding

We covered the encoder

$$f : X \times \Theta \mapsto \Delta Z$$

We can use the same decoder as a standard autoencoder

$$f^{-1} : Z \times \Theta \mapsto X$$

Encoder outputs a distribution ΔZ but decoder input is Z

Solution: Sample a vector $\boldsymbol{z}$ from the distribution ΔZ

VAE Review and Coding

Put it all together

VAE Review and Coding

Put it all together

Step 1: Encode the input to a normal distribution

$$\boldsymbol{\mu}, \boldsymbol{\sigma} = f(\boldsymbol{x}, \boldsymbol{\theta}_e)$$

VAE Review and Coding

Put it all together

Step 1: Encode the input to a normal distribution

$$\boldsymbol{\mu}, \boldsymbol{\sigma} = f(\boldsymbol{x}, \boldsymbol{\theta}_e)$$

Step 2: Generate a sample from distribution

$$\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\varepsilon}$$

VAE Review and Coding

Put it all together

Step 1: Encode the input to a normal distribution

$$\boldsymbol{\mu}, \boldsymbol{\sigma} = f(\boldsymbol{x}, \boldsymbol{\theta}_e)$$

Step 2: Generate a sample from distribution

$$\boldsymbol{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\varepsilon}$$

Step 3: Decode the sample

$$\boldsymbol{x} = f^{-1}(\boldsymbol{z}, \boldsymbol{\theta}_d)$$

VAE Review and Coding

```python
# Create normal distribution from input
mu, sigma = model.f(x)

# Randomly sample a z vector from our distribution
epsilon = jax.random.normal(key, x.shape[0])
z = mu + sigma * epsilon

# Decode/reconstruct z back into x
pred_x = model.f_inverse(z)
```

VAE Review and Coding

Now, all we must do is find $\boldsymbol{\theta}$ that best explains the dataset distribution

VAE Review and Coding

Now, all we must do is find $\boldsymbol{\theta}$ that best explains the dataset distribution

Learned distribution $P(\boldsymbol{x}; \boldsymbol{\theta})$ to be close to dataset $P(\boldsymbol{x}), \quad \boldsymbol{x} \sim X$

VAE Review and Coding

Now, all we must do is find $\boldsymbol{\theta}$ that best explains the dataset distribution

Learned distribution $P(\boldsymbol{x}; \boldsymbol{\theta})$ to be close to dataset $P(\boldsymbol{x}), \quad \boldsymbol{x} \sim X$

We started with the KL divergence

$$\arg \min_{\boldsymbol{\theta}} \mathrm{KL}(P(\boldsymbol{x}), P(\boldsymbol{x}; \boldsymbol{\theta}))$$

VAE Review and Coding

Now, all we must do is find $\boldsymbol{\theta}$ that best explains the dataset distribution

Learned distribution $P(\boldsymbol{x}; \boldsymbol{\theta})$ to be close to dataset $P(\boldsymbol{x}), \quad \boldsymbol{x} \sim X$

We started with the KL divergence

$$\arg\min_{\boldsymbol{\theta}} \mathrm{KL}(P(\boldsymbol{x}), P(\boldsymbol{x}; \boldsymbol{\theta}))$$

VAE Review and Coding

From the KL divergence, we derived the **ELBO** loss for the VAE

$$\mathcal{L}(\boldsymbol{X}, \boldsymbol{\theta}) = \underbrace{\frac{m}{n} \sum_{i=1}^{n} \sum_{j=1}^{d_z} \left(x_{[i],j} - f^{-1}\left(f\left(\boldsymbol{x}_{[i]}, \boldsymbol{\theta}_e \right), \boldsymbol{\theta}_d \right)_j \right)^2}_{\text{Reconstruct } \boldsymbol{x}} -$$

$$\underbrace{\beta \left(\sum_{i=1}^{n} \sum_{j=1}^{d_z} \mu_{[i],j}^2 + \sigma_{[i],j}^2 - \log\left(\sigma_{[i],j}^2 \right) - 1 \right)}_{\text{Make } \boldsymbol{z} \text{ normally distributed}}$$

VAE Review and Coding

```
def L(model, x, m, n, key):
  mu, sigma = model.f(x) # Encode input into distribution
  # Sample from distribution
  z = mu + sigma * jax.random.normal(key, x.shape[0])
  # Reconstruct input
  pred_x = model.f_inverse(z)
  # Compute reconstruction and kl loss terms
  recon = jnp.sum((x - pred_x) ** 2)
  kl = jnp.sum(mu ** 2 + sigma ** 2 - jnp.log(sigma ** 2) -
1)
  # Loss function contains reconstruction and kl terms
  return m / n * recon + kl
```

VAE Review and Coding

https://colab.research.google.com/drive/1UyR_W6NDIujaJXYlHZh6O3NfaCAMscpH#scrollTo=nmyQ8aE2pSbb

Agenda

1. GNN Review
2. **VAE Review and Coding**
3. Attention
4. Keys and Queries
5. Transformer
6. Positional Encoding
7. Coding

Agenda

1. GNN Review
2. VAE Review and Coding
3. **Attention**
4. Keys and Queries
5. Transformer
6. Positional Encoding
7. Coding

Attention

Today, we will investigate attention and transformers

Attention

Today, we will investigate attention and transformers

Attention and transformers are the “hottest” topic in deep learning

Attention

Today, we will investigate attention and transformers

Attention and transformers are the "hottest" topic in deep learning

People use them for almost every task (even if they shouldn't!)

Attention

Today, we will investigate attention and transformers

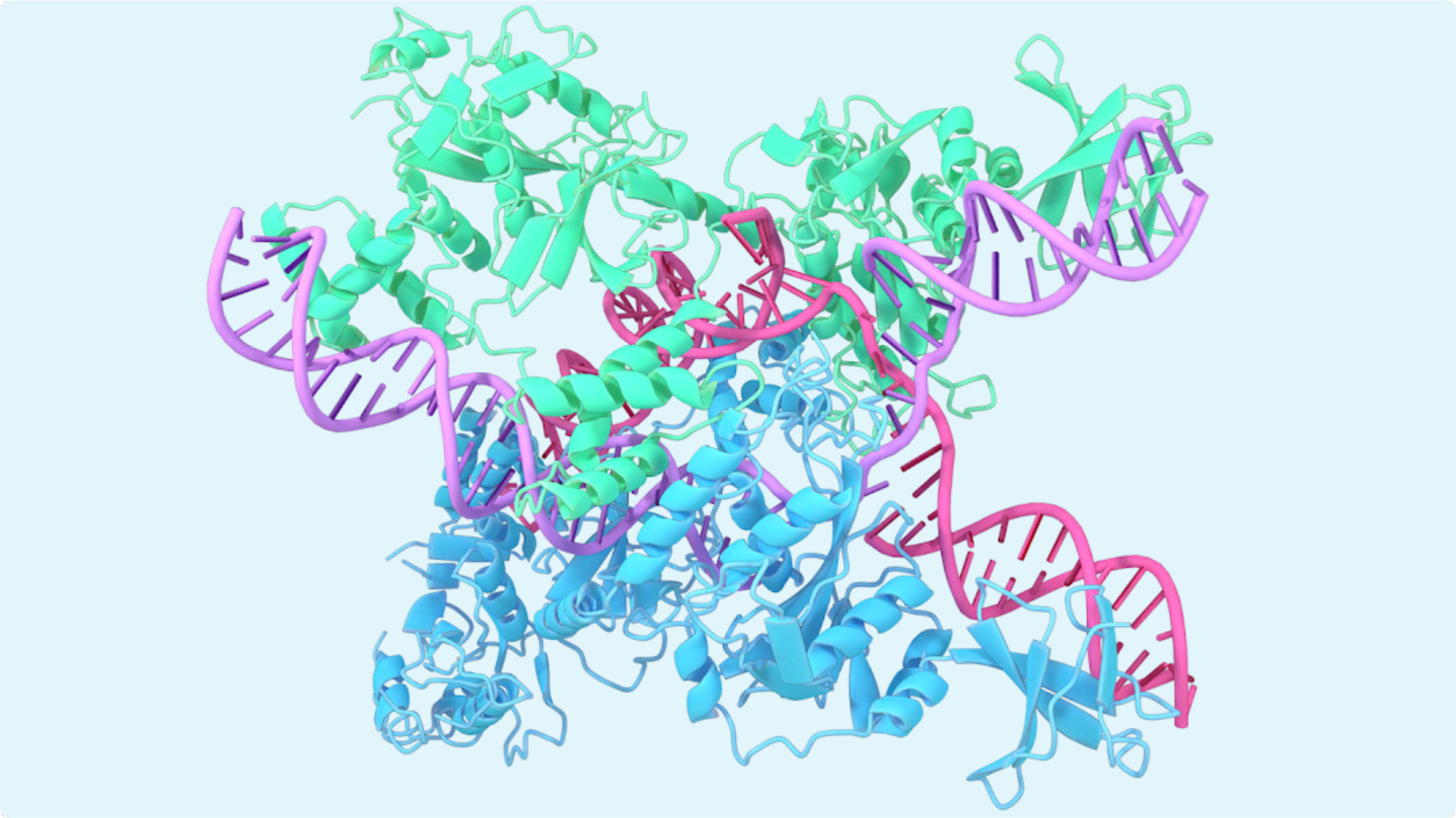
Attention and transformers are the "hottest" topic in deep learning

People use them for almost every task (even if they shouldn't!)

Let's review some products based on attention

Attention

AlphaFold (Nobel prize)

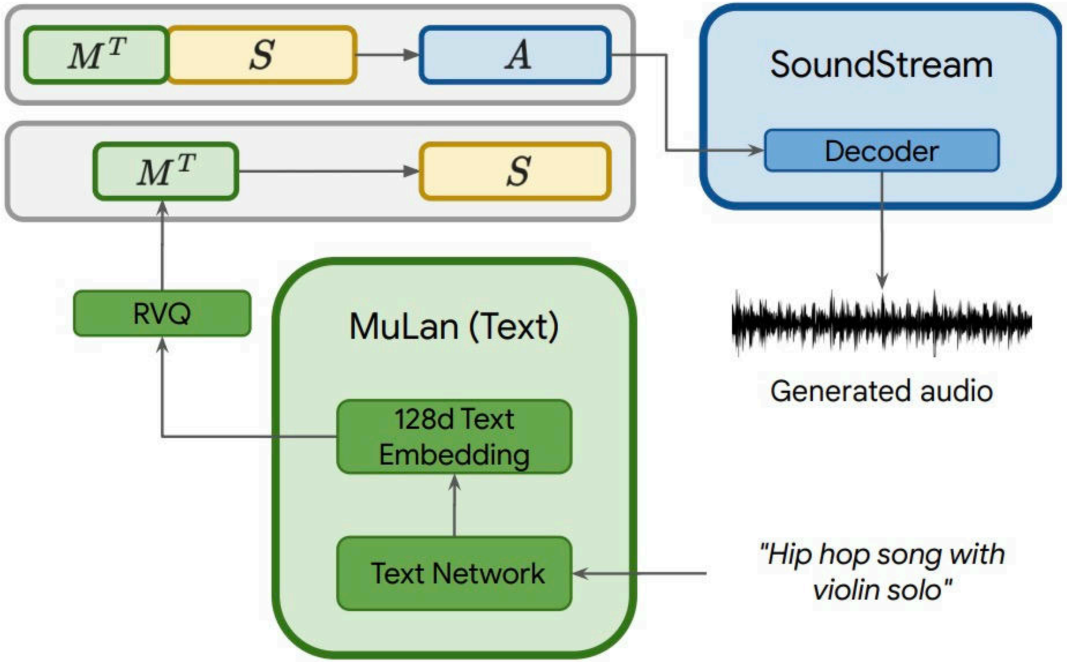


Attention

ChatGPT, Qwen, LLaMA, Mistral, Doubou, Ernie chatbots

Attention

MusicTransformer, MuLan

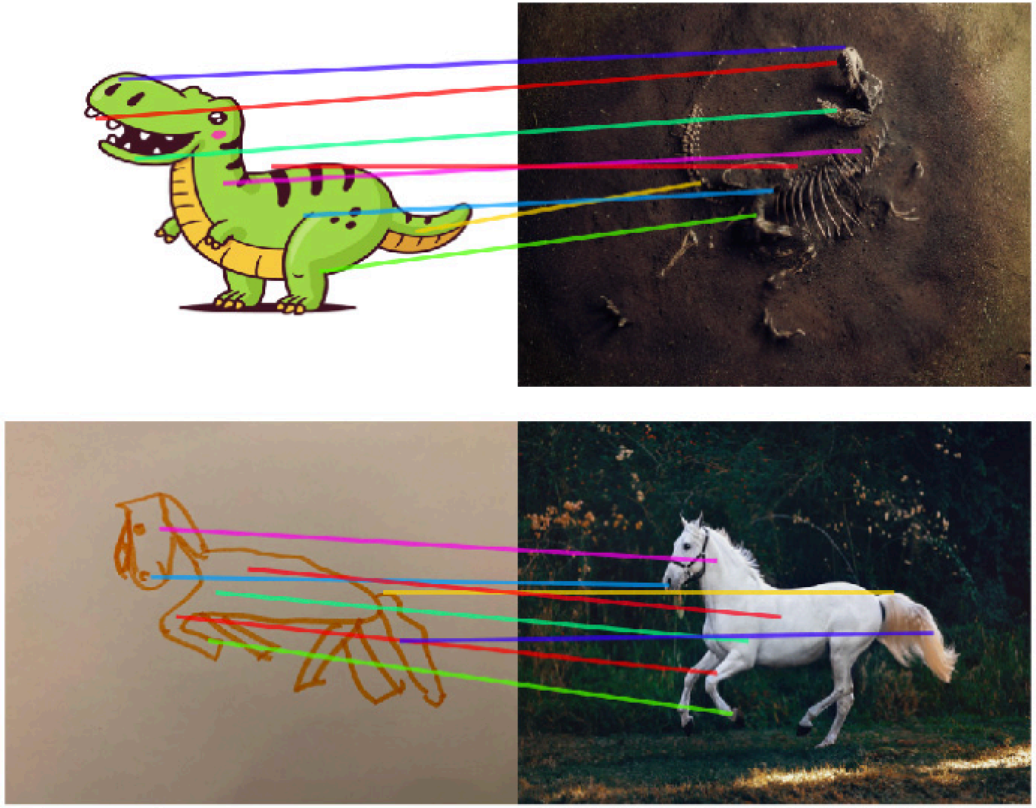


Attention

Google Translate, Baidu Translate, Apple Translate

Attention

ViT, DinoV2



Attention

All these models are **transformers**

Attention

All these models are **transformers**

At the core of each transformer is **attention**

Attention

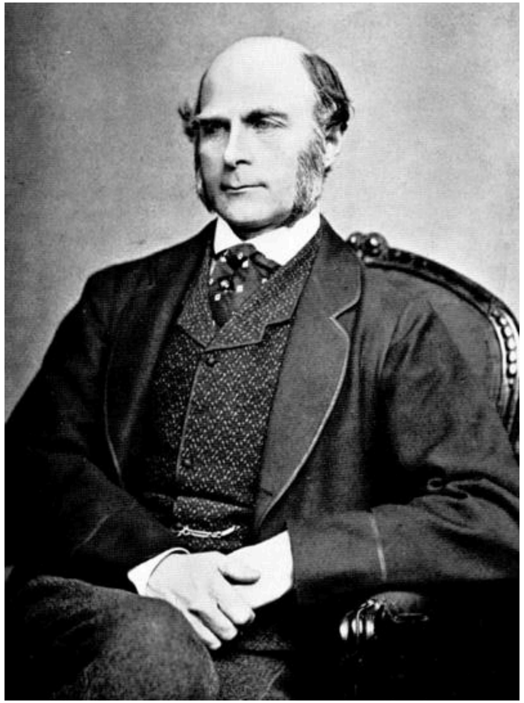
All these models are **transformers**

At the core of each transformer is **attention**

We can derive attention from composite memory

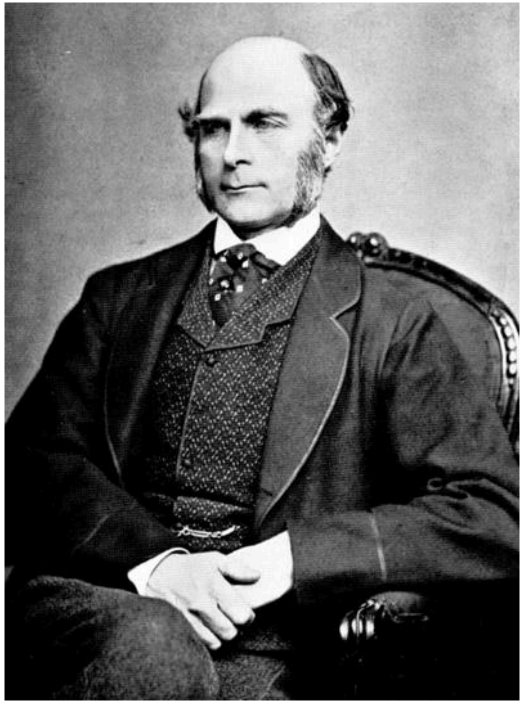
Attention

Francis Galton (1822-1911)
photo composite memory

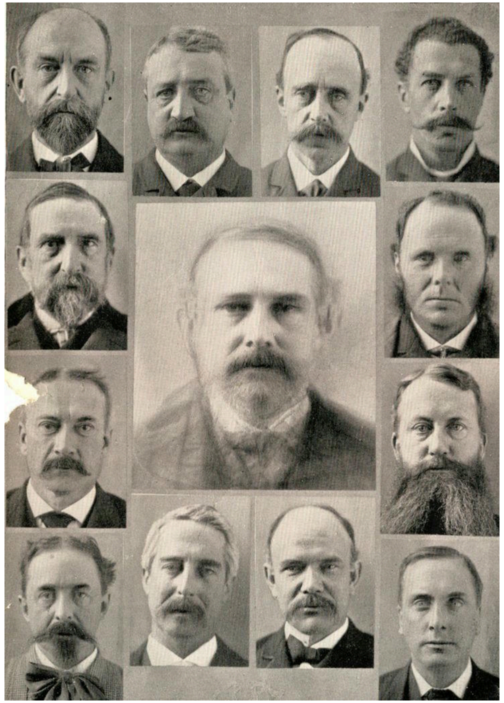


Attention

Francis Galton (1822-1911) photo composite memory



Composite photo of members of a party



Attention

Task: Find a mathematical model of how our mind represents memories

Attention

Task: Find a mathematical model of how our mind represents memories

$$X : \mathbb{R}^{h \times w} \quad \text{People you see at the party}$$

Attention

Task: Find a mathematical model of how our mind represents memories

$$X : \mathbb{R}^{h \times w} \quad \text{People you see at the party}$$

$$H : \mathbb{R}^{h \times w} \quad \text{The image in your mind}$$

Attention

Task: Find a mathematical model of how our mind represents memories

$$X : \mathbb{R}^{h \times w} \quad \text{People you see at the party}$$

$$H : \mathbb{R}^{h \times w} \quad \text{The image in your mind}$$

$$f : X^T \times \Theta \mapsto H$$

Attention

Task: Find a mathematical model of how our mind represents memories

$$X : \mathbb{R}^{h \times w} \quad \text{People you see at the party}$$

$$H : \mathbb{R}^{h \times w} \quad \text{The image in your mind}$$

$$f : X^T \times \Theta \mapsto H$$

Composite photography/memory uses a weighted sum

Attention

Task: Find a mathematical model of how our mind represents memories

$$X : \mathbb{R}^{h \times w} \quad \text{People you see at the party}$$

$$H : \mathbb{R}^{h \times w} \quad \text{The image in your mind}$$

$$f : X^T \times \Theta \mapsto H$$

Composite photography/memory uses a weighted sum

$$f(\boldsymbol{x}, \boldsymbol{\theta}) = \sum_{i=1}^{T} \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_i$$

Attention

Limited space, cannot remember everything

Attention

Limited space, cannot remember everything

Introduced forgetting

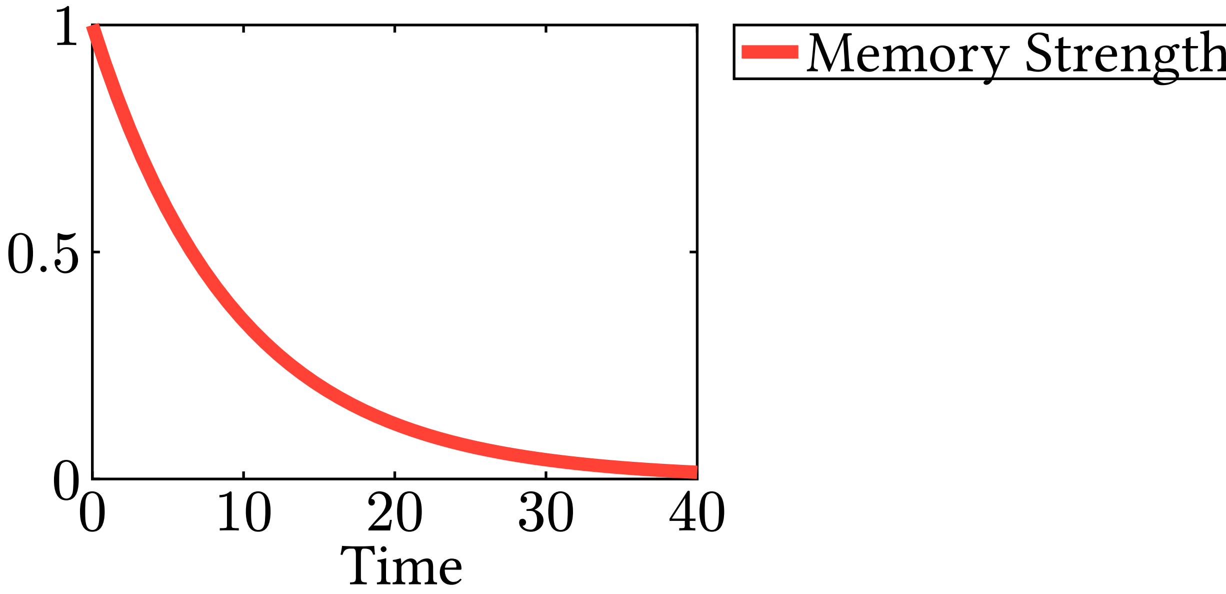
$$\sum_{i=1}^{T} \gamma^{T-i} \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_i$$

Attention

Limited space, cannot remember everything

Introduced forgetting

$$\sum_{i=1}^{T} \gamma^{T-i} \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_i$$

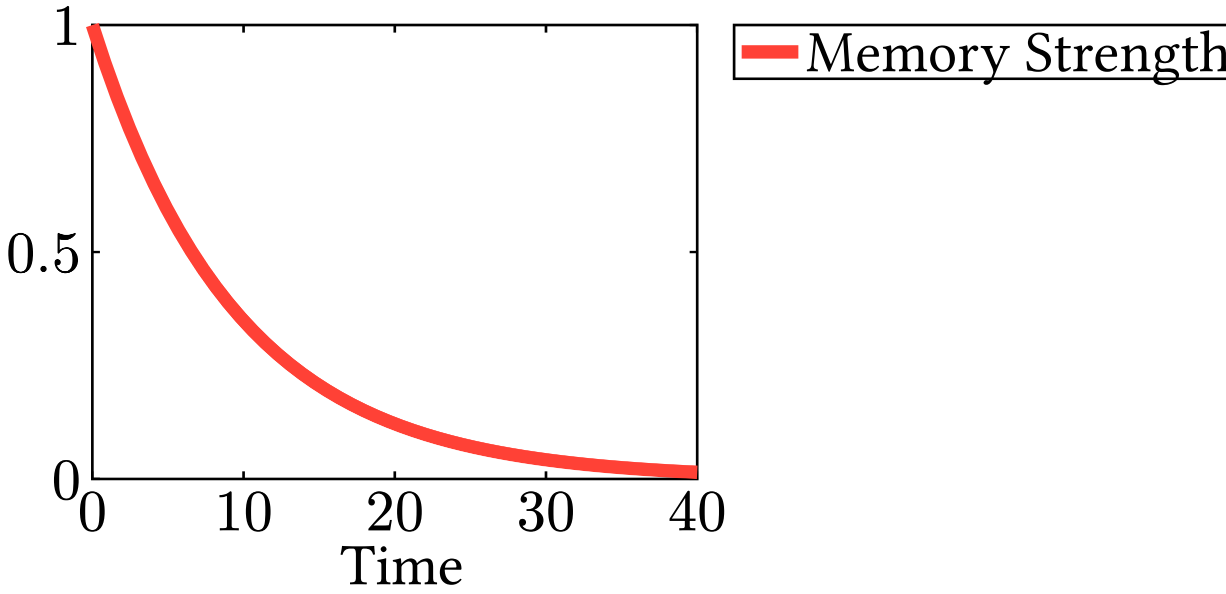


Attention

Limited space, cannot remember everything

Introduced forgetting

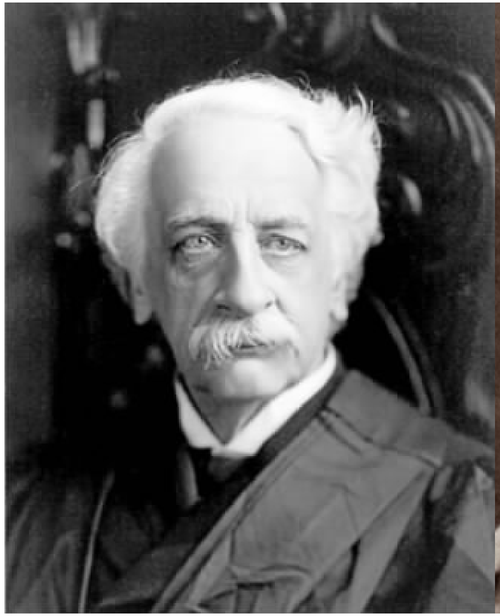
$$\sum_{i=1}^{T} \gamma^{T-i} \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_i$$

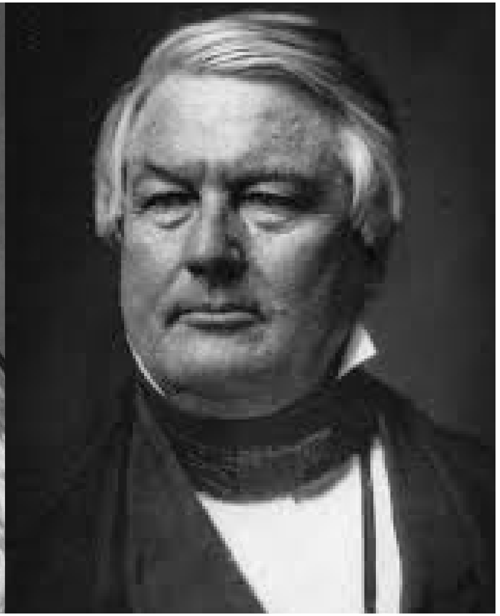


Question: Does this accurately model what you remember?

Attention

You go to a party and meet these people in order



Attention

According to forgetting, the memories should fade with time

$$\gamma^3 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$$

Attention

According to forgetting, the memories should fade with time

$\gamma^3 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$ $\gamma^2 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2$

Attention

According to forgetting, the memories should fade with time

$\gamma^3 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$ $\quad$ $\gamma^2 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2$ $\quad$ $\gamma^1 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3$

Attention

According to forgetting, the memories should fade with time

$\gamma^3 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$ $\quad$ $\gamma^2 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2$ $\quad$ $\gamma^1 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3$ $\quad$ $\gamma^0 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_4$

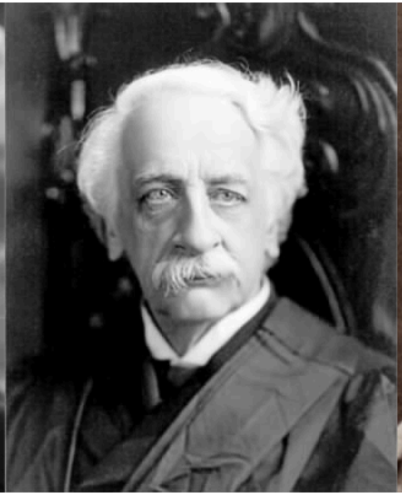
Attention

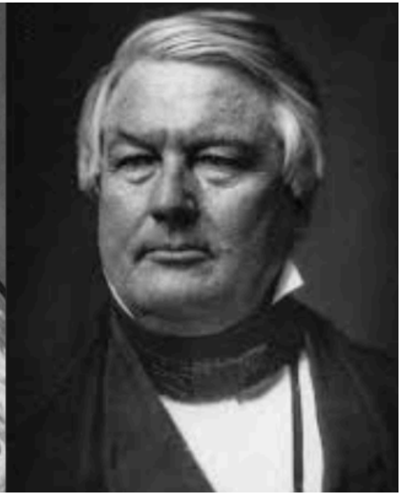
Consider another party

Attention

Consider another party

Consider another party

$\gamma^4 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$ $\gamma^3 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2$ $\gamma^2 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3$ $\gamma^1 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_4$ $\gamma^0 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_5$

Attention

Consider another party

$\gamma^4 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$ $\gamma^3 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2$ $\gamma^2 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3$ $\gamma^1 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_4$ $\gamma^0 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_5$

Question: What will happen to Taylor Swift?

Attention

$$\gamma^4 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1 \qquad \gamma^3 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2 \qquad \gamma^2 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3 \qquad \gamma^1 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_4 \qquad \gamma^0 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_5$$

Attention

$\gamma^4 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$ $\gamma^3 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2$ $\gamma^2 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3$ $\gamma^1 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_4$ $\gamma^0 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_5$

We will forget meeting her!

Attention

$$\gamma^4 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1 \qquad \gamma^3 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2 \qquad \gamma^2 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3 \qquad \gamma^1 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_4 \qquad \gamma^0 \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_5$$

We will forget meeting her!

Question: Would you forget meeting Taylor Swift?

Attention

Our model of memory is incomplete

Attention

Our model of memory is incomplete

Memories are not created equal, some are more important than others

Attention

Our model of memory is incomplete

Memories are not created equal, some are more important than others

We will **pay more attention** to certain memories

My memory might actually be

$$1.0 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$$

My memory might actually be

$1.0 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1 \qquad 0.1 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2$

My memory might actually be

$$1.0 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1 \qquad 0.1 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2 \qquad 0.1 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3$$

My memory might actually be

$1.0 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$ $\quad 0.1 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2$ $\quad 0.1 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3$ $\quad 0.5 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_4$

My memory might actually be

$1.0 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$ $\quad 0.1 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2$ $\quad 0.1 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3$ $\quad 0.5 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_4$ $\quad 0.1 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_5$

My memory might actually be

$1.0 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1$ $0.1 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2$ $0.1 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3$ $0.5 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_4$ $0.1 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_5$

Question: How can we achieve this forgetting?

Attention

In our composite model, forgetting is a function of time

Attention

In our composite model, forgetting is a function of time

Question: Any forgetting mechanism that is not a function of time?

Attention

In our composite model, forgetting is a function of time

Question: Any forgetting mechanism that is not a function of time?

Answer: Forgetting in recurrent neural network is function of input!

Attention

In our composite model, forgetting is a function of time

Question: Any forgetting mechanism that is not a function of time?

Answer: Forgetting in recurrent neural network is function of input!

$$f_{\text{forget}}(\boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_\lambda^\top \overline{\boldsymbol{x}}\right)$$

Attention

In our composite model, forgetting is a function of time

Question: Any forgetting mechanism that is not a function of time?

Answer: Forgetting in recurrent neural network is function of input!

$$f_{\text{forget}}(\boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_\lambda^\top \overline{\boldsymbol{x}}\right)$$

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = f_{\text{forget}}(\boldsymbol{x}, \boldsymbol{\theta}) \odot \boldsymbol{h} + \boldsymbol{\theta}_x^\top \overline{\boldsymbol{x}}$$

Attention

In our composite model, forgetting is a function of time

Question: Any forgetting mechanism that is not a function of time?

Answer: Forgetting in recurrent neural network is function of input!

$$f_{\text{forget}}(\boldsymbol{x}, \boldsymbol{\theta}) = \sigma\left(\boldsymbol{\theta}_\lambda^\top \overline{\boldsymbol{x}}\right)$$

$$f(\boldsymbol{h}, \boldsymbol{x}, \boldsymbol{\theta}) = f_{\text{forget}}(\boldsymbol{x}, \boldsymbol{\theta}) \odot \boldsymbol{h} + \boldsymbol{\theta}_x^\top \overline{\boldsymbol{x}}$$

Attention

First, write our forgetting function

Attention

First, write our forgetting function

$$\lambda(\boldsymbol{x}, \boldsymbol{\theta}_\lambda) = \sigma\left(\boldsymbol{\theta}_\lambda^\top \overline{\boldsymbol{x}}\right); \quad \boldsymbol{\theta}_\lambda \in \mathbb{R}^{(d_x+1) \times 1}$$

Attention

First, write our forgetting function

$$\lambda(\boldsymbol{x}, \boldsymbol{\theta}_\lambda) = \sigma\left(\boldsymbol{\theta}_\lambda^\top \overline{\boldsymbol{x}}\right); \quad \boldsymbol{\theta}_\lambda \in \mathbb{R}^{(d_x+1) \times 1}$$

Then, write our composite memory model with forgetting

Attention

First, write our forgetting function

$$\lambda(\boldsymbol{x}, \boldsymbol{\theta}_\lambda) = \sigma\left(\boldsymbol{\theta}_\lambda^\top \overline{\boldsymbol{x}}\right); \quad \boldsymbol{\theta}_\lambda \in \mathbb{R}^{(d_x+1)\times 1}$$

Then, write our composite memory model with forgetting

$$f\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}\right) = \sum_{i=1}^{T} \boldsymbol{\theta}^\top \boldsymbol{x}_i \cdot \lambda(\boldsymbol{x}_i, \boldsymbol{\theta}_\lambda)$$

Attention

First, write our forgetting function

$$\lambda(\boldsymbol{x}, \boldsymbol{\theta}_\lambda) = \sigma\left(\boldsymbol{\theta}_\lambda^\top \overline{\boldsymbol{x}}\right); \quad \boldsymbol{\theta}_\lambda \in \mathbb{R}^{(d_x+1)\times 1}$$

Then, write our composite memory model with forgetting

$$f\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}\right) = \sum_{i=1}^{T} \boldsymbol{\theta}^\top \boldsymbol{x}_i \cdot \lambda(\boldsymbol{x}_i, \boldsymbol{\theta}_\lambda)$$

This is one form of **attention**

Attention

First, write our forgetting function

$$\lambda(\boldsymbol{x}, \boldsymbol{\theta}_\lambda) = \sigma\left(\boldsymbol{\theta}_\lambda^\top \overline{\boldsymbol{x}}\right); \quad \boldsymbol{\theta}_\lambda \in \mathbb{R}^{(d_x+1) \times 1}$$

Then, write our composite memory model with forgetting

$$f\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}\right) = \sum_{i=1}^{T} \boldsymbol{\theta}^\top \boldsymbol{x}_i \cdot \lambda(\boldsymbol{x}_i, \boldsymbol{\theta}_\lambda)$$

This is one form of **attention**

We only pay attention to specific inputs

Attention

We can use this simple form of attention to pay attention to Taylor Swift

Attention

We can use this simple form of attention to pay attention to Taylor Swift

Attention

$$\lambda(\boldsymbol{x}_1, \boldsymbol{\theta}_\lambda) \quad \lambda(\boldsymbol{x}_2, \boldsymbol{\theta}_\lambda) \quad \lambda(\boldsymbol{x}_3, \boldsymbol{\theta}_\lambda) \quad \lambda(\boldsymbol{x}_4, \boldsymbol{\theta}_\lambda) \quad \lambda(\boldsymbol{x}_5, \boldsymbol{\theta}_\lambda)$$

$$\cdot\, \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1 \quad \cdot\, \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2 \quad \cdot\, \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3 \quad \cdot\, \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_4 \quad \cdot\, \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_5$$

Attention

$$\lambda(\boldsymbol{x}_1, \boldsymbol{\theta}_\lambda) \quad \lambda(\boldsymbol{x}_2, \boldsymbol{\theta}_\lambda) \quad \lambda(\boldsymbol{x}_3, \boldsymbol{\theta}_\lambda) \quad \lambda(\boldsymbol{x}_4, \boldsymbol{\theta}_\lambda) \quad \lambda(\boldsymbol{x}_5, \boldsymbol{\theta}_\lambda)$$

$$\cdot\, \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1 \quad \cdot\, \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2 \quad \cdot\, \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3 \quad \cdot\, \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_4 \quad \cdot\, \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_5$$

Question: What do the images look like now?

Attention

$$\lambda(\boldsymbol{x}_1, \boldsymbol{\theta}_\lambda) \qquad \lambda(\boldsymbol{x}_2, \boldsymbol{\theta}_\lambda) \qquad \lambda(\boldsymbol{x}_3, \boldsymbol{\theta}_\lambda) \qquad \lambda(\boldsymbol{x}_4, \boldsymbol{\theta}_\lambda) \qquad \lambda(\boldsymbol{x}_5, \boldsymbol{\theta}_\lambda)$$

$$\cdot\, \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1 \qquad \cdot\, \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2 \qquad \cdot\, \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3 \qquad \cdot\, \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_4 \qquad \cdot\, \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_5$$

Question: What do the images look like now?

Attention

This form of attention will learn to pay attention to everyone!

Attention

This form of attention will learn to pay attention to everyone!

$$1.0 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1 \qquad 1.0 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2 \qquad 1.0 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3 \qquad 1.0 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_4 \qquad 1.0 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_5$$

Attention

This form of attention will learn to pay attention to everyone!

$$1.0 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1 \quad 1.0 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2 \quad 1.0 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3 \quad 1.0 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_4 \quad 1.0 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_5$$

Attention

This form of attention will learn to pay attention to everyone!

$$1.0 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1 \qquad 1.0 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2 \qquad 1.0 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3 \qquad 1.0 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_4 \qquad 1.0 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_5$$

Not a good model of attention!

Attention

We should normalize $\lambda(\boldsymbol{x}, \boldsymbol{\theta}_\lambda)$ to model finite (human) attention span

Attention

We should normalize $\lambda(\boldsymbol{x}, \boldsymbol{\theta}_\lambda)$ to model finite (human) attention span

For example, normalize attention to sum to one

$$\sum_{i=1}^{T} \lambda(\boldsymbol{x}, \boldsymbol{\theta}_\lambda) = 1$$

Attention

We should normalize $\lambda(\boldsymbol{x}, \boldsymbol{\theta}_\lambda)$ to model finite (human) attention span

For example, normalize attention to sum to one

$$\sum_{i=1}^{T} \lambda(\boldsymbol{x}, \boldsymbol{\theta}_\lambda) = 1$$

Now the model must carefully choose who to remember!

Attention

We should normalize $\lambda(\boldsymbol{x}, \boldsymbol{\theta}_\lambda)$ to model finite (human) attention span

For example, normalize attention to sum to one

$$\sum_{i=1}^{T} \lambda(\boldsymbol{x}, \boldsymbol{\theta}_\lambda) = 1$$

Now the model must carefully choose who to remember!

Question: Do we know of any functions with this property?

Attention

We should normalize $\lambda(\boldsymbol{x}, \boldsymbol{\theta}_\lambda)$ to model finite (human) attention span

For example, normalize attention to sum to one

$$\sum_{i=1}^{T} \lambda(\boldsymbol{x}, \boldsymbol{\theta}_\lambda) = 1$$

Now the model must carefully choose who to remember!

Question: Do we know of any functions with this property?

Answer: Softmax!

Attention

The softmax function maps real numbers to the simplex (probabilities)

$$\text{softmax} : \mathbb{R}^k \mapsto \Delta^{k-1}$$

Attention

The softmax function maps real numbers to the simplex (probabilities)

$$\text{softmax} : \mathbb{R}^k \mapsto \Delta^{k-1}$$

$$\text{softmax}\left(\begin{bmatrix} x_1 \\ \vdots \\ x_k \end{bmatrix} \right) = \frac{\exp(\boldsymbol{x})}{\sum_{i=1}^{k} \exp(x_i)} = \begin{bmatrix} \frac{\exp(x_1)}{\exp(x_1) + \exp(x_2) + \ldots \exp(x_k)} \\ \frac{\exp(x_2)}{\exp(x_1) + e^{x_2} + \ldots \exp(x_k)} \\ \vdots \\ \frac{\exp(x_k)}{\exp(x_1) + \exp(x_2) + \ldots \exp(x_k)} \end{bmatrix}$$

Attention

Let us rewrite attention using softmax

Attention

Let us rewrite attention using softmax

The attention we pay to person i is

$$\lambda\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}_\lambda\right)_i = \frac{\exp\left(\boldsymbol{\theta}_\lambda^\top \overline{\boldsymbol{x}}_i\right)}{\sum_{j=1}^T \exp\left(\boldsymbol{\theta}_\lambda^\top \overline{\boldsymbol{x}}_j\right)}$$

Attention

$$\lambda\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_5 \end{bmatrix}, \boldsymbol{\theta}_\lambda \right)_1 \lambda\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_5 \end{bmatrix}, \boldsymbol{\theta}_\lambda \right)_2 \lambda\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_5 \end{bmatrix}, \boldsymbol{\theta}_\lambda \right)_3 \lambda\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_5 \end{bmatrix}, \boldsymbol{\theta}_\lambda \right)_4 \lambda\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_5 \end{bmatrix}, \boldsymbol{\theta}_\lambda \right)_5$$

$$\cdot\, \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1 \qquad \cdot\, \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2 \qquad \cdot\, \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3 \qquad \cdot\, \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_4 \qquad \cdot\, \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_5$$

Attention

$$\lambda\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_5 \end{bmatrix}, \boldsymbol{\theta}_\lambda\right)_1 \lambda\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_5 \end{bmatrix}, \boldsymbol{\theta}_\lambda\right)_2 \lambda\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_5 \end{bmatrix}, \boldsymbol{\theta}_\lambda\right)_3 \lambda\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_5 \end{bmatrix}, \boldsymbol{\theta}_\lambda\right)_4 \lambda\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_5 \end{bmatrix}, \boldsymbol{\theta}_\lambda\right)_5$$

$$\cdot\, \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1 \qquad \cdot\, \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2 \qquad \cdot\, \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3 \qquad \cdot\, \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_4 \qquad \cdot\, \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_5$$

Attention

$$0.70 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1 \quad 0.04 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2 \quad 0.03 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3 \quad 0.20 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_4 \quad 0.03 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_5$$

Attention

$$0.70 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1 \quad 0.04 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2 \quad 0.03 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3 \quad 0.20 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_4 \quad 0.03 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_5$$

Attention

$$0.70 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_1 \quad 0.04 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_2 \quad 0.03 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_3 \quad 0.20 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_4 \quad 0.03 \cdot \boldsymbol{\theta}^\top \overline{\boldsymbol{x}}_5$$

$$0.70 + 0.04 + 0.03 + 0.20 + 0.03 = 1.0$$

Attention

$$\lambda\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}_\lambda\right)_i = \frac{\exp\left(\boldsymbol{\theta}_\lambda^\top \overline{\boldsymbol{x}}\right)}{\sum_{j=1}^T \exp\left(\boldsymbol{\theta}_\lambda^\top \overline{\boldsymbol{x}}_j\right)}$$

$$\lambda\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}_\lambda\right) = \begin{bmatrix} \frac{\exp\left(\boldsymbol{\theta}_\lambda^\top \overline{\boldsymbol{x}}_{1}\right)}{\sum_{j=1}^T \exp\left(\boldsymbol{\theta}_\lambda^\top \overline{\boldsymbol{x}}_j\right)} \\ \vdots \\ \frac{\exp\left(\boldsymbol{\theta}_\lambda^\top \overline{\boldsymbol{x}}_{T}\right)}{\sum_{j=1}^T \exp\left(\boldsymbol{\theta}_\lambda^\top \overline{\boldsymbol{x}}_j\right)} \end{bmatrix}$$

Agenda

1. GNN Review
2. VAE Review and Coding
3. **Attention**
4. Keys and Queries
5. Transformer
6. Positional Encoding
7. Coding

Agenda

1. GNN Review
2. VAE Review and Coding
3. Attention
4. **Keys and Queries**
5. Transformer
6. Positional Encoding
7. Coding

Keys and Queries

The modern form of attention behaves like a database

Keys and Queries

The modern form of attention behaves like a database

We label each person at the party with a **key**

Keys and Queries

The modern form of attention behaves like a database

We label each person at the party with a **key**

The key describes the content of each $\boldsymbol{x}$

Keys and Queries

The modern form of attention behaves like a database

We label each person at the party with a **key**

The key describes the content of each $\boldsymbol{x}$

Keys and Queries

The modern form of attention behaves like a database

We label each person at the party with a **key**

The key describes the content of each $\boldsymbol{x}$

Musician

Keys and Queries

The modern form of attention behaves like a database

We label each person at the party with a **key**

The key describes the content of each $\boldsymbol{x}$

Musician Lawyer

Keys and Queries

The modern form of attention behaves like a database

We label each person at the party with a **key**

The key describes the content of each $\boldsymbol{x}$

Musician Lawyer Shopkeeper

Keys and Queries

The modern form of attention behaves like a database

We label each person at the party with a **key**

The key describes the content of each $\boldsymbol{x}$

Musician Lawyer Shopkeeper Chef

Keys and Queries

The modern form of attention behaves like a database

We label each person at the party with a **key**

The key describes the content of each $\boldsymbol{x}$

Musician Lawyer Shopkeeper Chef Scientist

Keys and Queries

We can search our keys using a **query**

Keys and Queries

We can search our keys using a **query**

Query: Which person will help me on my exam?

Keys and Queries

We can search our keys using a **query**

Query: Which person will help me on my exam?

Keys and Queries

We can search our keys using a **query**

Query: Which person will help me on my exam?

Musician	Lawyer	Shopkeeper	Chef	Scientist

Keys and Queries

We can search our keys using a **query**

Query: Which person will help me on my exam?

Musician Lawyer Shopkeeper Chef Scientist

Keys and Queries

Query: I want to have fun

Keys and Queries

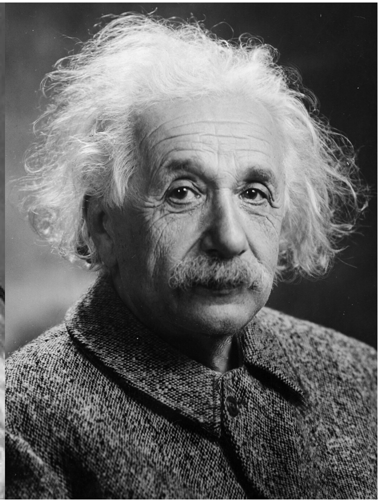
Query: I want to have fun

Musician

Lawyer

Shopkeeper

Chef



Scientist

Keys and Queries

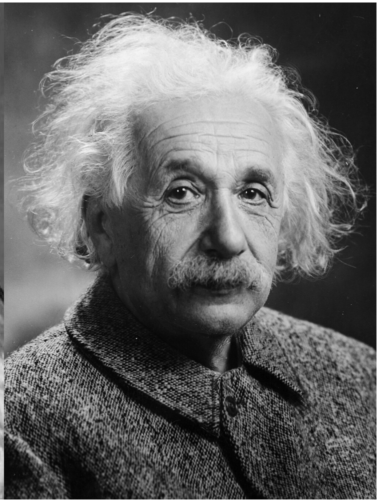
Query: I want to have fun

Musician

Lawyer

Shopkeeper

Chef



Scientist

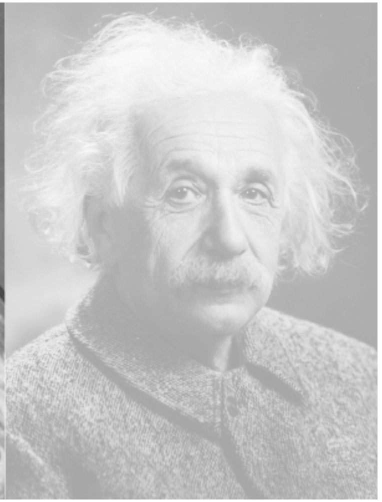
Keys and Queries

Query: I want to have fun

Musician Lawyer Shopkeeper Chef Scientist







How do we represent this mathematically?

Keys and Queries

For an input, we create a key $\boldsymbol{k}$

Keys and Queries

For an input, we create a key $\boldsymbol{k}$

$$\boldsymbol{k}_j = \boldsymbol{\theta}_K^\top \boldsymbol{x}_j, \quad \boldsymbol{\theta}_K \in \mathbb{R}^{d_x \times d_h}, \quad \boldsymbol{k}_j \in \mathbb{R}^{d_h}$$

Keys and Queries

For an input, we create a key $\boldsymbol{k}$

$$\boldsymbol{k}_j = \boldsymbol{\theta}_K^\top \boldsymbol{x}_j, \quad \boldsymbol{\theta}_K \in \mathbb{R}^{d_x \times d_h}, \quad \boldsymbol{k}_j \in \mathbb{R}^{d_h}$$

Create keys for all inputs

Keys and Queries

For an input, we create a key $\boldsymbol{k}$

$$\boldsymbol{k}_j = \boldsymbol{\theta}_K^\top \boldsymbol{x}_j, \quad \boldsymbol{\theta}_K \in \mathbb{R}^{d_x \times d_h}, \quad \boldsymbol{k}_j \in \mathbb{R}^{d_h}$$

Create keys for all inputs

$$\boldsymbol{K} = \begin{bmatrix} \boldsymbol{k}_1 \\ \boldsymbol{k}_2 \\ \vdots \\ \boldsymbol{k}_T \end{bmatrix} = \begin{bmatrix} \boldsymbol{\theta}_K^\top \boldsymbol{x}_1 \\ \boldsymbol{\theta}_K^\top \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{\theta}_K^\top \boldsymbol{x}_T \end{bmatrix}, \quad \boldsymbol{K} \in \mathbb{R}^{T \times d_h}$$

Keys and Queries

Now, create a query from some $\boldsymbol{x}_q$

Keys and Queries

Now, create a query from some $\boldsymbol{x}_q$

$$\boldsymbol{q} = \boldsymbol{\theta}_Q^\top \boldsymbol{x}_q, \quad \boldsymbol{\theta}_Q \in \mathbb{R}^{d_x \times d_h}, \quad \boldsymbol{q} \in \mathbb{R}^{d_h}$$

Keys and Queries

Now, create a query from some $\boldsymbol{x}_q$

$$\boldsymbol{q} = \boldsymbol{\theta}_Q^\top \boldsymbol{x}_q, \quad \boldsymbol{\theta}_Q \in \mathbb{R}^{d_x \times d_h}, \quad \boldsymbol{q} \in \mathbb{R}^{d_h}$$

A matching key and query will have a large dot product

Keys and Queries

Now, create a query from some $\boldsymbol{x}_q$

$$\boldsymbol{q} = \boldsymbol{\theta}_Q^\top \boldsymbol{x}_q, \quad \boldsymbol{\theta}_Q \in \mathbb{R}^{d_x \times d_h}, \quad \boldsymbol{q} \in \mathbb{R}^{d_h}$$

A matching key and query will have a large dot product

$$\boldsymbol{q}^\top \boldsymbol{k}_i = \left(\boldsymbol{\theta}_Q^\top \boldsymbol{x}_q\right)^\top \left(\boldsymbol{\theta}_K^\top \boldsymbol{x}_i\right)$$

Keys and Queries

Now, create a query from some $\boldsymbol{x}_q$

$$\boldsymbol{q} = \boldsymbol{\theta}_Q^\top \boldsymbol{x}_q, \quad \boldsymbol{\theta}_Q \in \mathbb{R}^{d_x \times d_h}, \quad \boldsymbol{q} \in \mathbb{R}^{d_h}$$

A matching key and query will have a large dot product

$$\boldsymbol{q}^\top \boldsymbol{k}_i = \left(\boldsymbol{\theta}_Q^\top \boldsymbol{x}_q\right)^\top \left(\boldsymbol{\theta}_K^\top \boldsymbol{x}_i\right)$$

Question: What is the shape of $\boldsymbol{q}^\top \boldsymbol{k}_i$?

Keys and Queries

Now, create a query from some $\boldsymbol{x}_q$

$$\boldsymbol{q} = \boldsymbol{\theta}_Q^\top \boldsymbol{x}_q, \quad \boldsymbol{\theta}_Q \in \mathbb{R}^{d_x \times d_h}, \quad \boldsymbol{q} \in \mathbb{R}^{d_h}$$

A matching key and query will have a large dot product

$$\boldsymbol{q}^\top \boldsymbol{k}_i = \left(\boldsymbol{\theta}_Q^\top \boldsymbol{x}_q\right)^\top \left(\boldsymbol{\theta}_K^\top \boldsymbol{x}_i\right)$$

Question: What is the shape of $\boldsymbol{q}^\top \boldsymbol{k}_i$?

Answer: $(1, d_h) \times (d_h, 1) = 1$, the output is a scalar

Keys and Queries

Now, create a query from some $\boldsymbol{x}_q$

$$\boldsymbol{q} = \boldsymbol{\theta}_Q^\top \boldsymbol{x}_q, \quad \boldsymbol{\theta}_Q \in \mathbb{R}^{d_x \times d_h}, \quad \boldsymbol{q} \in \mathbb{R}^{d_h}$$

A matching key and query will have a large dot product

$$\boldsymbol{q}^\top \boldsymbol{k}_i = \left(\boldsymbol{\theta}_Q^\top \boldsymbol{x}_q\right)^\top \left(\boldsymbol{\theta}_K^\top \boldsymbol{x}_i\right)$$

Question: What is the shape of $\boldsymbol{q}^\top \boldsymbol{k}_i$?

Answer: $(1, d_h) \times (d_h, 1) = 1$, the output is a scalar

Keys and Queries

Example:

$$\boldsymbol{k}_i = \boldsymbol{\theta}_K^\top$$ 

Keys and Queries

Example:

$$\boldsymbol{k}_i = \boldsymbol{\theta}_K^\top$$ 

$$\boldsymbol{q} = \boldsymbol{\theta}_Q^\top \text{ Musician}$$

Keys and Queries

Example:

$\boldsymbol{k}_i = \boldsymbol{\theta}_K^\top$ 

$\boldsymbol{q} = \boldsymbol{\theta}_Q^\top \text{ Musician}$

$\boldsymbol{q}^\top \boldsymbol{k}_i = \left(\boldsymbol{\theta}_Q^\top \text{ Musician}\right)^\top \Big(\boldsymbol{\theta}_K^\top$  $\Big) = 100$

Keys and Queries

Example:

$$\boldsymbol{k}_i = \boldsymbol{\theta}_K^\top$$ 

$$\boldsymbol{q} = \boldsymbol{\theta}_Q^\top \text{ Musician}$$

$$\boldsymbol{q}^\top \boldsymbol{k}_i = \left(\boldsymbol{\theta}_Q^\top \text{ Musician}\right)^\top \left(\boldsymbol{\theta}_K^\top\right.$$  $$\left.\right) = 100$$

Large attention!

Keys and Queries

Example:

$$\boldsymbol{k}_i = \boldsymbol{\theta}_K^\top$$ 

Keys and Queries

Example:

$$\boldsymbol{k}_i = \boldsymbol{\theta}_K^\top$$ 

$$\boldsymbol{q} = \boldsymbol{\theta}_Q^\top \text{ Mathematician}$$

Example:

$$\boldsymbol{k}_i = \boldsymbol{\theta}_K^\top$$ 

$$\boldsymbol{q} = \boldsymbol{\theta}_Q^\top \text{ Mathematician}$$

$$\boldsymbol{q}^\top \boldsymbol{k}_i = \left(\boldsymbol{\theta}_Q^\top \text{ Mathematician}\right)^\top \left(\boldsymbol{\theta}_K^\top\right.$$  $$\left.\right) = -50$$

Keys and Queries

Example:

$$\boldsymbol{k}_i = \boldsymbol{\theta}_K^\top$$


$$\boldsymbol{q} = \boldsymbol{\theta}_Q^\top \text{ Mathematician}$$

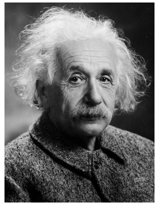
$$\boldsymbol{q}^\top \boldsymbol{k}_i = \left(\boldsymbol{\theta}_Q^\top \text{ Mathematician}\right)^\top \left(\boldsymbol{\theta}_K^\top\right.$$

$$\left.\right) = -50$$

Small attention!

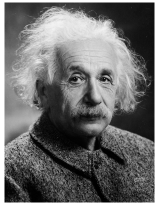
Keys and Queries

Example:

$$\boldsymbol{k}_i = \boldsymbol{\theta}_K^\top$$ 

Keys and Queries

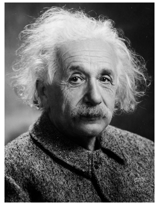
Example:

$$\boldsymbol{k}_i = \boldsymbol{\theta}_K^\top$$ 

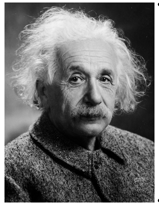
$$\boldsymbol{q} = \boldsymbol{\theta}_Q^\top \text{ Mathematician}$$

Keys and Queries

Example:

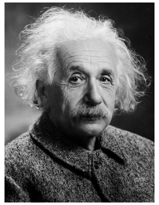
$$\boldsymbol{k}_i = \boldsymbol{\theta}_K^\top$$ 

$$\boldsymbol{q} = \boldsymbol{\theta}_Q^\top \text{ Mathematician}$$

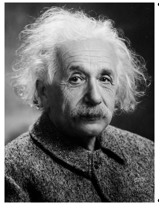
$$\boldsymbol{q}^\top \boldsymbol{k}_i = \left(\boldsymbol{\theta}_Q^\top \text{ Mathematician}\right)^\top \left(\boldsymbol{\theta}_K^\top \right.$$  $$\left. \right) = 90$$

Keys and Queries

Example:

$\boldsymbol{k}_i = \boldsymbol{\theta}_K^\top$ 

$\boldsymbol{q} = \boldsymbol{\theta}_Q^\top \text{ Mathematician}$

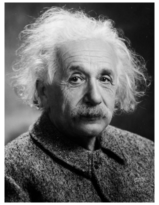
$\boldsymbol{q}^\top \boldsymbol{k}_i = (\boldsymbol{\theta}_Q^\top \text{ Mathematician})^\top (\boldsymbol{\theta}_K^\top$  $) = 90$

Large attention!

Keys and Queries

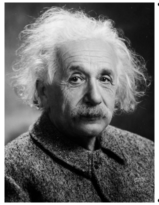
Example:

$$\boldsymbol{k}_i = \boldsymbol{\theta}_K^\top$$



$$\boldsymbol{q} = \boldsymbol{\theta}_Q^\top \text{ Mathematician}$$

$$\boldsymbol{q}^\top \boldsymbol{k}_i = \left(\boldsymbol{\theta}_Q^\top \text{ Mathematician}\right)^\top \Bigg(\boldsymbol{\theta}_K^\top$$



$$\Bigg) = 90$$

Large attention!

Remember, there are multiple inputs to pay attention to

Keys and Queries

We compute attention for each input

Keys and Queries

We compute attention for each input

$$\boldsymbol{q}^\top \boldsymbol{K} = \boldsymbol{q}^\top \begin{bmatrix} \boldsymbol{k}_1 \\ \boldsymbol{k}_2 \\ \vdots \\ \boldsymbol{k}_T \end{bmatrix} = \begin{bmatrix} \left(\boldsymbol{\theta}_Q^\top \boldsymbol{x}_q\right)^\top \left(\boldsymbol{\theta}_K^\top \boldsymbol{x}_1\right) \\ \left(\boldsymbol{\theta}_Q^\top \boldsymbol{x}_q\right)^\top \left(\boldsymbol{\theta}_K^\top \boldsymbol{x}_2\right) \\ \vdots \\ \left(\boldsymbol{\theta}_Q^\top \boldsymbol{x}_q\right)^\top \left(\boldsymbol{\theta}_K^\top \boldsymbol{x}_T\right) \end{bmatrix}$$

Keys and Queries

We compute attention for each input

$$\boldsymbol{q}^\top \boldsymbol{K} = \boldsymbol{q}^\top \begin{bmatrix} \boldsymbol{k}_1 \\ \boldsymbol{k}_2 \\ \vdots \\ \boldsymbol{k}_T \end{bmatrix} = \begin{bmatrix} \left(\boldsymbol{\theta}_Q^\top \boldsymbol{x}_q\right)^\top \left(\boldsymbol{\theta}_K^\top \boldsymbol{x}_1\right) \\ \left(\boldsymbol{\theta}_Q^\top \boldsymbol{x}_q\right)^\top \left(\boldsymbol{\theta}_K^\top \boldsymbol{x}_2\right) \\ \vdots \\ \left(\boldsymbol{\theta}_Q^\top \boldsymbol{x}_q\right)^\top \left(\boldsymbol{\theta}_K^\top \boldsymbol{x}_T\right) \end{bmatrix}$$

Question: Anything missing from before?

Keys and Queries

We compute attention for each input

$$\boldsymbol{q}^\top \boldsymbol{K} = \boldsymbol{q}^\top \begin{bmatrix} \boldsymbol{k}_1 \\ \boldsymbol{k}_2 \\ \vdots \\ \boldsymbol{k}_T \end{bmatrix} = \begin{bmatrix} \left(\boldsymbol{\theta}_Q^\top \boldsymbol{x}_q\right)^\top \left(\boldsymbol{\theta}_K^\top \boldsymbol{x}_1\right) \\ \left(\boldsymbol{\theta}_Q^\top \boldsymbol{x}_q\right)^\top \left(\boldsymbol{\theta}_K^\top \boldsymbol{x}_2\right) \\ \vdots \\ \left(\boldsymbol{\theta}_Q^\top \boldsymbol{x}_q\right)^\top \left(\boldsymbol{\theta}_K^\top \boldsymbol{x}_T\right) \end{bmatrix}$$

Question: Anything missing from before?

Answer: Normalize attention to sum to one!

Keys and Queries

Normalize, only pay attention to important matches

Keys and Queries

Normalize, only pay attention to important matches

$$\operatorname{softmax}(\boldsymbol{q}^\top \boldsymbol{K}) = \operatorname{softmax}\left(\boldsymbol{q}^\top \begin{bmatrix} \boldsymbol{k}_1 \\ \boldsymbol{k}_2 \\ \vdots \\ \boldsymbol{k}_T \end{bmatrix} \right) = \operatorname{softmax}\left(\begin{bmatrix} \left(\boldsymbol{\theta}_Q^\top \boldsymbol{x}_q\right)^\top \left(\boldsymbol{\theta}_K^\top \boldsymbol{x}_1\right) \\ \left(\boldsymbol{\theta}_Q^\top \boldsymbol{x}_q\right)^\top \left(\boldsymbol{\theta}_K^\top \boldsymbol{x}_2\right) \\ \vdots \\ \left(\boldsymbol{\theta}_Q^\top \boldsymbol{x}_q\right)^\top \left(\boldsymbol{\theta}_K^\top \boldsymbol{x}_T\right) \end{bmatrix} \right)$$

Keys and Queries

Normalize, only pay attention to important matches

$$\text{softmax}(\boldsymbol{q}^\top \boldsymbol{K}) = \text{softmax}\left(\boldsymbol{q}^\top \begin{bmatrix} \boldsymbol{k}_1 \\ \boldsymbol{k}_2 \\ \vdots \\ \boldsymbol{k}_T \end{bmatrix} \right) = \text{softmax}\left(\begin{bmatrix} \left(\boldsymbol{\theta}_Q^\top \boldsymbol{x}_q\right)^\top \left(\boldsymbol{\theta}_K^\top \boldsymbol{x}_1\right) \\ \left(\boldsymbol{\theta}_Q^\top \boldsymbol{x}_q\right)^\top \left(\boldsymbol{\theta}_K^\top \boldsymbol{x}_2\right) \\ \vdots \\ \left(\boldsymbol{\theta}_Q^\top \boldsymbol{x}_q\right)^\top \left(\boldsymbol{\theta}_K^\top \boldsymbol{x}_T\right) \end{bmatrix} \right)$$

We call this **dot-product attention**

Keys and Queries

Query: Which person will help me on my exam?

Keys and Queries

Query: Which person will help me on my exam?

$\boldsymbol{q}^\top \boldsymbol{k}_1$ $\boldsymbol{q}^\top \boldsymbol{k}_2$ $\boldsymbol{q}^\top \boldsymbol{k}_3$ $\boldsymbol{q}^\top \boldsymbol{k}_4$ $\boldsymbol{q}^\top \boldsymbol{k}_5$

Keys and Queries

Query: Which person will help me on my exam?

$\boldsymbol{q}^\top \boldsymbol{k}_1$	$\boldsymbol{q}^\top \boldsymbol{k}_2$	$\boldsymbol{q}^\top \boldsymbol{k}_3$	$\boldsymbol{q}^\top \boldsymbol{k}_4$	$\boldsymbol{q}^\top \boldsymbol{k}_5$
-1.71	0.60	-1.01	-0.61	2.73

softmax

Keys and Queries

Query: Which person will help me on my exam?

$\boldsymbol{q}^\top \boldsymbol{k}_1$	$\boldsymbol{q}^\top \boldsymbol{k}_2$	$\boldsymbol{q}^\top \boldsymbol{k}_3$	$\boldsymbol{q}^\top \boldsymbol{k}_4$	$\boldsymbol{q}^\top \boldsymbol{k}_5$
-1.71	0.60	-1.01	-0.61	2.73

$$\text{softmax}$$

0.01	0.10	0.02	0.03	0.84

Keys and Queries

Query: Which person will help me on my exam?

$\boldsymbol{q}^\top \boldsymbol{k}_1$	$\boldsymbol{q}^\top \boldsymbol{k}_2$	$\boldsymbol{q}^\top \boldsymbol{k}_3$	$\boldsymbol{q}^\top \boldsymbol{k}_4$	$\boldsymbol{q}^\top \boldsymbol{k}_5$
-1.71	0.60	-1.01	-0.61	2.73

$$\text{softmax}$$

0.01	0.10	0.02	0.03	0.84

Keys and Queries

Query: Which person will help me on my exam?

$\boldsymbol{q}^\top \boldsymbol{k}_1$	$\boldsymbol{q}^\top \boldsymbol{k}_2$	$\boldsymbol{q}^\top \boldsymbol{k}_3$	$\boldsymbol{q}^\top \boldsymbol{k}_4$	$\boldsymbol{q}^\top \boldsymbol{k}_5$
-1.71	0.60	-1.01	-0.61	2.73

$$\mathrm{softmax}$$

0.01	0.10	0.02	0.03	0.84

Keys and Queries

Put dot product attention into our composite model

Keys and Queries

Put dot product attention into our composite model

$$\lambda\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}_\lambda\right)_i = \text{softmax}\left(\boldsymbol{q}^\top \begin{bmatrix} \boldsymbol{k}_1 \\ \boldsymbol{k}_2 \\ \vdots \\ \boldsymbol{k}_T \end{bmatrix}\right)_i$$

Keys and Queries

Put dot product attention into our composite model

$$\lambda\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}_\lambda\right)_i = \operatorname{softmax}\left(\boldsymbol{q}^\top \begin{bmatrix} \boldsymbol{k}_1 \\ \boldsymbol{k}_2 \\ \vdots \\ \boldsymbol{k}_T \end{bmatrix}\right)_i$$

Then, write our composite memory model with forgetting

Keys and Queries

Put dot product attention into our composite model

$$\lambda\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}_\lambda \right)_i = \text{softmax}\left(\boldsymbol{q}^\top \begin{bmatrix} \boldsymbol{k}_1 \\ \boldsymbol{k}_2 \\ \vdots \\ \boldsymbol{k}_T \end{bmatrix} \right)_i$$

Then, write our composite memory model with forgetting

$$f\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta} \right) = \sum_{i=1}^{T} \boldsymbol{\theta}^\top \boldsymbol{x}_i \cdot \lambda\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}_\lambda \right)_i$$

Keys and Queries

Put dot product attention into our composite model

$$\lambda\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}_\lambda\right)_i = \text{softmax}\left(\boldsymbol{q}^\top \begin{bmatrix} \boldsymbol{k}_1 \\ \boldsymbol{k}_2 \\ \vdots \\ \boldsymbol{k}_T \end{bmatrix}\right)_i$$

Then, write our composite memory model with forgetting

$$f\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}\right) = \sum_{i=1}^{T} \boldsymbol{\theta}^\top \boldsymbol{x}_i \cdot \lambda\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}_\lambda\right)_i$$

Keys and Queries

$$f\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}\right) = \sum_{i=1}^{T} \boldsymbol{\theta}^\top \boldsymbol{x}_i \cdot \lambda\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}_\lambda\right)_i$$

Keys and Queries

$$f\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}\right) = \sum_{i=1}^{T} \boldsymbol{\theta}^\top \boldsymbol{x}_i \cdot \lambda\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}_\lambda\right)_i$$

$$f\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}\right) = \sum_{i=1}^{T} \boldsymbol{\theta}_{V}^\top \boldsymbol{x}_i \cdot \lambda\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}_\lambda\right)_i$$

Keys and Queries

$$f\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}\right) = \sum_{i=1}^{T} \boldsymbol{\theta}^\top \boldsymbol{x}_i \cdot \lambda\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}_\lambda\right)_i$$

$$f\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}\right) = \sum_{i=1}^{T} \boldsymbol{\theta}_{\color{red}{V}}^\top \boldsymbol{x}_i \cdot \lambda\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}_\lambda\right)_i$$

In dot-product attention, we call $\boldsymbol{\theta}_V^\top \boldsymbol{x}_i$ the **value**

Keys and Queries

$$\boldsymbol{q}^\top \boldsymbol{k}_1 \quad \boldsymbol{q}^\top \boldsymbol{k}_2 \quad \boldsymbol{q}^\top \boldsymbol{k}_3 \quad \boldsymbol{q}^\top \boldsymbol{k}_4 \quad \boldsymbol{q}^\top \boldsymbol{k}_5$$

$$\cdot\, \boldsymbol{\theta}_V^\top \boldsymbol{x}_1 \quad \cdot\, \boldsymbol{\theta}_V^\top \boldsymbol{x}_2 \quad \cdot\, \boldsymbol{\theta}_V^\top \boldsymbol{x}_3 \quad \cdot\, \boldsymbol{\theta}_V^\top \boldsymbol{x}_4 \quad \cdot\, \boldsymbol{\theta}_V^\top \boldsymbol{x}_5$$

Keys and Queries

$$\boldsymbol{q}^\top \boldsymbol{k}_1 \quad \boldsymbol{q}^\top \boldsymbol{k}_2 \quad \boldsymbol{q}^\top \boldsymbol{k}_3 \quad \boldsymbol{q}^\top \boldsymbol{k}_4 \quad \boldsymbol{q}^\top \boldsymbol{k}_5$$

$$\cdot\, \boldsymbol{\theta}_V^\top \boldsymbol{x}_1 \quad \cdot\, \boldsymbol{\theta}_V^\top \boldsymbol{x}_2 \quad \cdot\, \boldsymbol{\theta}_V^\top \boldsymbol{x}_3 \quad \cdot\, \boldsymbol{\theta}_V^\top \boldsymbol{x}_4 \quad \cdot\, \boldsymbol{\theta}_V^\top \boldsymbol{x}_5$$

Agenda

1. GNN Review
2. VAE Review and Coding
3. Attention
4. **Keys and Queries**
5. Transformer
6. Positional Encoding
7. Coding

Agenda

1. GNN Review
2. VAE Review and Coding
3. Attention
4. Keys and Queries
5. **Transformer**
6. Positional Encoding
7. Coding

Transformer

Previously, we chose our own $\boldsymbol{x}_q = \text{Musician}$

Transformer

Previously, we chose our own $\boldsymbol{x}_q = \text{Musician}$

We can also create queries from all the inputs

Transformer

Previously, we chose our own $\boldsymbol{x}_q = \text{Musician}$

We can also create queries from all the inputs

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{q}_1 \\ \boldsymbol{q}_2 \\ \vdots \\ \boldsymbol{q}_T \end{bmatrix} = \begin{bmatrix} \boldsymbol{\theta}_Q^\top \boldsymbol{x}_1 \\ \boldsymbol{\theta}_Q^\top \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{\theta}_Q^\top \boldsymbol{x}_T \end{bmatrix}, \quad \boldsymbol{Q} \in \mathbb{R}^{T \times d_h}$$

Transformer

Previously, we chose our own $\boldsymbol{x}_q = \text{Musician}$

We can also create queries from all the inputs

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{q}_1 \\ \boldsymbol{q}_2 \\ \vdots \\ \boldsymbol{q}_T \end{bmatrix} = \begin{bmatrix} \boldsymbol{\theta}_Q^\top \boldsymbol{x}_1 \\ \boldsymbol{\theta}_Q^\top \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{\theta}_Q^\top \boldsymbol{x}_T \end{bmatrix}, \quad \boldsymbol{Q} \in \mathbb{R}^{T \times d_h}$$

We call this dot-product **self** attention

Transformer

Writing dot-product self attention in matrix form is easier

Transformer

Writing dot-product self attention in matrix form is easier

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{q}_1 \\ \boldsymbol{q}_2 \\ \vdots \\ \boldsymbol{q}_T \end{bmatrix} = \begin{bmatrix} \boldsymbol{\theta}_Q^\top \boldsymbol{x}_1 \\ \boldsymbol{\theta}_Q^\top \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{\theta}_Q^\top \boldsymbol{x}_T \end{bmatrix} \quad \boldsymbol{K} = \begin{bmatrix} \boldsymbol{k}_1 \\ \boldsymbol{k}_2 \\ \vdots \\ \boldsymbol{k}_T \end{bmatrix} = \begin{bmatrix} \boldsymbol{\theta}_K^\top \boldsymbol{x}_1 \\ \boldsymbol{\theta}_K^\top \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{\theta}_K^\top \boldsymbol{x}_T \end{bmatrix} \quad \boldsymbol{V} = \begin{bmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \\ \vdots \\ \boldsymbol{v}_T \end{bmatrix} = \begin{bmatrix} \boldsymbol{\theta}_V^\top \boldsymbol{x}_1 \\ \boldsymbol{\theta}_V^\top \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{\theta}_V^\top \boldsymbol{x}_T \end{bmatrix}$$

Transformer

Writing dot-product self attention in matrix form is easier

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{q}_1 \\ \boldsymbol{q}_2 \\ \vdots \\ \boldsymbol{q}_T \end{bmatrix} = \begin{bmatrix} \boldsymbol{\theta}_Q^\top \boldsymbol{x}_1 \\ \boldsymbol{\theta}_Q^\top \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{\theta}_Q^\top \boldsymbol{x}_T \end{bmatrix} \quad \boldsymbol{K} = \begin{bmatrix} \boldsymbol{k}_1 \\ \boldsymbol{k}_2 \\ \vdots \\ \boldsymbol{k}_T \end{bmatrix} = \begin{bmatrix} \boldsymbol{\theta}_K^\top \boldsymbol{x}_1 \\ \boldsymbol{\theta}_K^\top \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{\theta}_K^\top \boldsymbol{x}_T \end{bmatrix} \quad \boldsymbol{V} = \begin{bmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \\ \vdots \\ \boldsymbol{v}_T \end{bmatrix} = \begin{bmatrix} \boldsymbol{\theta}_V^\top \boldsymbol{x}_1 \\ \boldsymbol{\theta}_V^\top \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{\theta}_V^\top \boldsymbol{x}_T \end{bmatrix}$$

$$\text{attn}\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta} \right) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_h}} \right) \boldsymbol{V}$$

Transformer

Writing dot-product self attention in matrix form is easier

$$\boldsymbol{Q} = \begin{bmatrix} \boldsymbol{q}_1 \\ \boldsymbol{q}_2 \\ \vdots \\ \boldsymbol{q}_T \end{bmatrix} = \begin{bmatrix} \boldsymbol{\theta}_Q^\top \boldsymbol{x}_1 \\ \boldsymbol{\theta}_Q^\top \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{\theta}_Q^\top \boldsymbol{x}_T \end{bmatrix} \quad \boldsymbol{K} = \begin{bmatrix} \boldsymbol{k}_1 \\ \boldsymbol{k}_2 \\ \vdots \\ \boldsymbol{k}_T \end{bmatrix} = \begin{bmatrix} \boldsymbol{\theta}_K^\top \boldsymbol{x}_1 \\ \boldsymbol{\theta}_K^\top \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{\theta}_K^\top \boldsymbol{x}_T \end{bmatrix} \quad \boldsymbol{V} = \begin{bmatrix} \boldsymbol{v}_1 \\ \boldsymbol{v}_2 \\ \vdots \\ \boldsymbol{v}_T \end{bmatrix} = \begin{bmatrix} \boldsymbol{\theta}_V^\top \boldsymbol{x}_1 \\ \boldsymbol{\theta}_V^\top \boldsymbol{x}_2 \\ \vdots \\ \boldsymbol{\theta}_V^\top \boldsymbol{x}_T \end{bmatrix}$$

$$\operatorname{attn}\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta} \right) = \operatorname{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top}{\sqrt{d_h}} \right) \boldsymbol{V}$$

This operation powers today's biggest models

Transformer

$$\boldsymbol{Q} \in \mathbb{R}^{T \times d_h} \quad \boldsymbol{K} \in \mathbb{R}^{T \times d_h} \quad \boldsymbol{V} \in \mathbb{R}^{T \times d_h}$$

$$\underbrace{\operatorname{attn}\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta}\right)}_{\mathbb{R}^{T \times d_h}} = \operatorname{softmax}\overbrace{\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top,}{\sqrt{d_h}}\right)}^{\mathbb{R}^{T \times T}} \underbrace{\boldsymbol{V}}_{\mathbb{R}^{T \times d_h}}$$

Transformer

$$\boldsymbol{Q} \in \mathbb{R}^{T \times d_h} \quad \boldsymbol{K} \in \mathbb{R}^{T \times d_h} \quad \boldsymbol{V} \in \mathbb{R}^{T \times d_h}$$

$$\underbrace{\operatorname{attn}\left(\begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_T \end{bmatrix}, \boldsymbol{\theta} \right)}_{\mathbb{R}^{T \times d_h}} = \operatorname{softmax} \overbrace{\left(\frac{\boldsymbol{Q}\boldsymbol{K}^\top,}{\sqrt{d_h}} \right)}^{\mathbb{R}^{T \times T}} \underbrace{\boldsymbol{V}}_{\mathbb{R}^{T \times d_h}}$$

Transformer

```
class Attention(nn.Module):
    def __init__(self):
        self.theta_K = nn.Linear(d_x, d_h, bias=False)
        self.theta_Q = nn.Linear(d_x, d_h, bias=False)
        self.theta_V = nn.Linear(d_x, d_h, bias=False)

    def forward(self, x):
        A = softmax(self.theta_Q(x) @ self.theta_K(x) / d_h)
        return A @ self.theta_V(x)
```

Transformer

Transformers

Transformer

If you understand attention, transformers are very simple

Transformer

If you understand attention, transformers are very simple

Each transformer consists of many "transformer blocks"

Transformer

If you understand attention, transformers are very simple

Each transformer consists of many “transformer blocks”

A transformer block is attention and an MLP

Transformer

```python
class TransformerBlock(nn.Module):
    def __init__(self):
        self.attn = Attention()
        self.mlp = Sequential(
            Linear(d_h, d_h), LeakyReLU(), Linear(d_h, d_h))
        self.norm1 = nn.LayerNorm(d_h)
        self.norm2 = nn.LayerNorm(d_h)

    def forward(self, x):
        # Residual connection
        x = self.norm1(self.attn(x) + x)
        x = self.norm2(self.mlp(x) + x)
        return x
```

Transformer

```python
class Transformer(nn.Module):
    def __init__(self):
        self.block1 = TransformerBlock()
        self.block2 = TransformerBlock()
        ...

    def forward(self, x):
        x = self.block1(x)
        x = self.block2(x)
        ...
        return x
```

Transformer

Agenda

1. GNN Review
2. VAE Review and Coding
3. Attention
4. Keys and Queries
5. Transformer
6. **Positional Encoding**
7. Coding

Agenda

1. GNN Review
2. VAE Review and Coding
3. Attention
4. Keys and Queries
5. Transformer
6. Positional Encoding
7. **Coding**

Coding

Transformers are an operation on **sets**