DSC 140B

Representation Learning

Lecture 26 | Part 1

Autoencoders

Representation Learning

- At a high level, representation learning finds an **encoding function** $\text{encode}(\vec{x}) : \mathbb{R}^d \to \mathbb{R}^k$.

- Ideally, this function captures useful aspects of the data distribution.

Decoding

- Encoding can decrease dimensionality.
- Intuitively, we may want to preserve as much "information" about $\vec{x}$ as possible.
- We should be able to **decode** the encoding and **reconstruct** the original point, approximately.

$$\vec{x} \approx \text{decode}(\text{encode}(\vec{x}))$$

Representation Learning

- **Goal:** find an encoder (and decoder) such that

$$\text{encode}(\text{decode}(\vec{x})) \approx \vec{x}$$

Reconstruction Error

- In general, decode(encode($\vec{x}$)) will not be exactly equal to $\vec{x}$.

- One way of quantifying the difference w.r.t. data is the (ℓ_2) **reconstruction error**:

$$\sum_{i=1}^{n} \|\vec{x}^{(i)} - \text{decode}(\text{encode}(\vec{x}^{(i)}))\|^2$$

Note

- Of course, it is trivial to find an encoder/decoder with zero reconstruction error:

$$\text{encode}(\vec{x}) = \vec{x} = \text{decode}(\vec{x})$$

- Such an encoder is not useful.

- Instead, we constrain the form of the encoder so that it cannot simply copy the input.

Example: PCA

- Assume encode($\vec{x}$) = $U\vec{x}$, for some matrix U whose $k \leq d$ columns are orthonormal.
  - That is, the encoding is an orthogonal projection.

- **Goal:** find U to minimize reconstruction error on a dataset $\vec{x}^{(1)}, \ldots, \vec{x}^{(d)}$.

- **Solution:** pick columns of U to be top k eigenvectors of data covariance matrix.

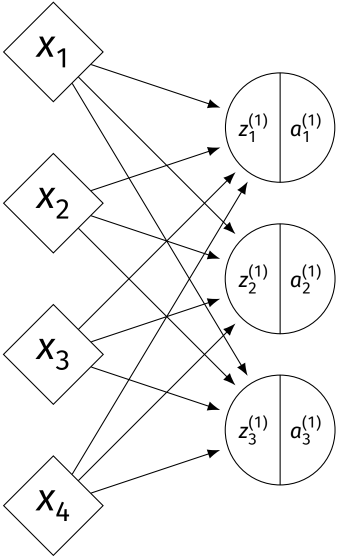
Now

- $\text{encode}(\vec{x}) = U\vec{x}$ is a linear encoding function.
- What if we let encode be nonlinear?
- That is, let's generalize PCA.

Encoder as a Neural Network

- Assume encode($\vec{x}$) is a (deep) **neural network**.
- Output is not a single number, but k numbers.
  - I.e., a vector in $\mathbb{R}^k$
- Can use nonlinear activations, have more than one layer.

Encoder as a Neural Network



Encoder as a Neural Network

- The output of the encoder is the new representation.

- To train the encoder, we'll need a **decoder**.

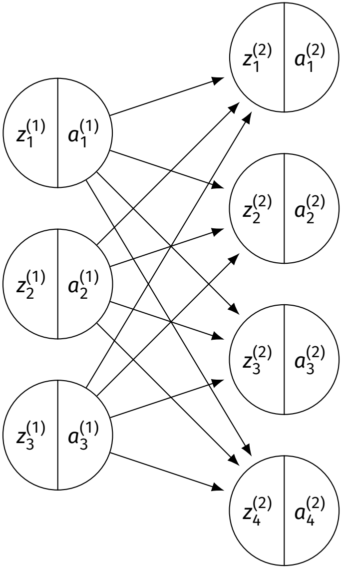
Decoder as a Neural Network

- Assume decode($\vec{z}$) is a (deep) **neural network**.

- Output is not a single number, but d numbers.
  - Same dimensionality as original input, $\vec{x}$.
  - I.e., a vector in $\mathbb{R}^d$

- Can use nonlinear activations, have more than one layer.

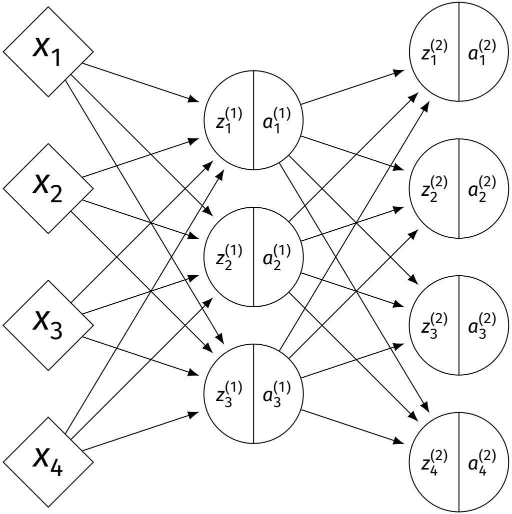
Decoder as a Neural Network



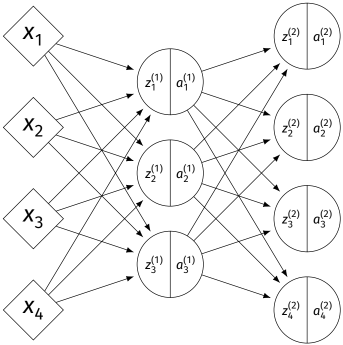
decode(encode($\vec{x}$)) **as a NN**

- Together, decode(encode($\vec{x}$)) is a neural network $H(\vec{x}) : \mathbb{R}^d \to \mathbb{R}^d$.

decode(encode($\vec{x}$)) **as a NN**



Training



- We want $H(\vec{x}) \approx \vec{x}$
- One approach: train network to minimize reconstruction error.

$$\sum_{i=1}^{n} \|\vec{x}^{(i)} - H(\vec{x}^{(i)})\|^2 = \sum_{i=1}^{n} \sum_{j=1}^{d} (\vec{x}_j^{(i)} - (H(\vec{x}^{(i)}))_j)^2$$

$$= \sum_{i=1}^{n} \sum_{j=1}^{d} (\vec{x}_j^{(i)} - a_j^{(2)}(\vec{x}^{(i)}))^2$$

Training

- The network can be trained using gradient-based methods.
  - E.g., stochastic gradient descent.

- **Note:** this is an **unsupervised** learning problem.

Autoencoders

- When the encoder/decoder are NNs, $H(\vec{x}) = \text{decode}(\text{encode}(\vec{x}))$ is an **autoencoder**.

Generalizing PCA

- We can view autoencoders as generalizations of PCA.

- Consider again the encoder that performs an orthogonal projection:

$$\text{encode}(\vec{x}) = U^T \vec{x}$$

$$\text{decode}(\vec{z}) = U \vec{z}$$

- encode/decode are neural networks (with linear activations).

DSC 140B
Representation Learning

Lecture 26 | Part 2

Conclusion of DSC 140B

Recap

- DSC 140B was about **representation learning**.
- We saw PCA, Laplacian Eigenmaps, RBF Networks, neural networks and deep learning
- Learned ML methods, but also theoretical tools for understanding why other ML methods work

More Deep Learning

- We have only scratched the surface of deep learning.
  - LSTMs, transformer models, graph neural networks, deep RL, GANs, etc.
- In this class, we focused on the fundamental principles behind NNs.
- You might consider taking CSE 151B.

More Deep Learning

- Latest progresses: e.g., Sora, GAIA-1

Sora

GAIA-1 for auto-driving

Prompted with a couple of seconds of the same starting context. Then it can unroll multiple possible futures.

[Hu, Russell, Yeo, et al., 2023]

GAIA-1 for auto-driving

Inject a natural language prompt "**It's night, and we have turned on our headlights**." after three seconds.

[Hu, Russell, Yeo, et al., 2023]

Limitations in Large Models

Limitations in Large Models

Explain why this is funny

GPT-4V

… The final panel reveals the punchline: the robot has merely produced a pile of crumpled paper, just like the human did, suggesting that the robot also suffers from writer's block … highlighting a situation where the human and the AI are equally challenged

Limitations in Large Models

How to solve the limitations?

- Better model architectures
- Better learning algorithms
- Better inference algorithms

You may consider taking

"DSC 291: Machine Learning with Few Labels"

Diverse machine learning algorithms

maximum likelihood estimation
reinforcement learning as inference
data re-weighting
inverse RL
active learning
policy optimization
data augmentation
reward-augmented maximum likelihood
label smoothing
imitation learning
softmax policy gradient
actor-critic
adversarial domain adaptation
GANs
posterior regularization
knowledge distillation
intrinsic reward
constraint-driven learning
prediction minimization
generalized expectation
regularized Bayes
learning from measurements
energy-based GANs
weak/distant supervision

Thanks!