Dimension reduction and autoencoders

Hung-Hsuan Chen

Dimension reduction

- Let $\boldsymbol{X} \in R^{n \times d}$
- We would like to find a new representation $\boldsymbol{Z} \in R^{n \times k}$, where $k < d$
- We want $\boldsymbol{Z}$ can still well represent the original $\boldsymbol{X}$

Why dimension reduction?

- Compress data and preserve useful information
- Data visualization

Toy example 1

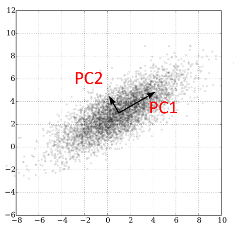
- Consider the following 3d points
  - (1,2,3), (2,4,6), (3,6,9), (4,8,12), (5,10,15), (6,12,18)
  - If each integer requires 1 byte, we need 1*3*6=18 bytes
- However, we may also store the first point (1,2,3) as the base, and store the multiplier of each point
  - One point (3 bytes) + multipliers (6 bytes)
- Reduced 50% of the storage

Toy example (con't)

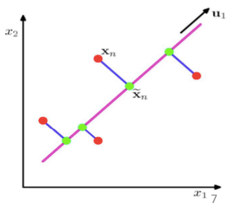
- Consider the following 3d points
  - (1,2,3), (2,4,6), (3,6,8), (4,8,12), (5,10,15), (6,12,19)
  - If each integer requires 1 byte, we need 1*3*6=18 bytes
- If we store the first point (1,2,3) as the base, and store the multiplier of each point
  - One point (3 bytes) + multipliers (6 bytes)
- Reduced 50% of the storage
- However, we have some small loss

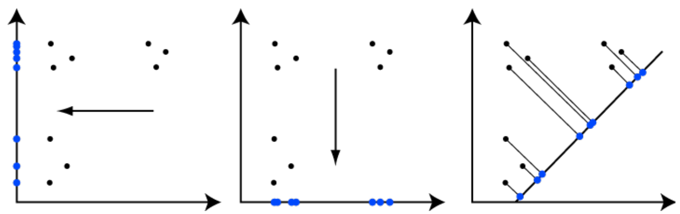
Principal component analysis (PCA)

PCA



- Goal: find base direction(s) that preserves "important" aspects of data
- PCA
  - Allow only linear transforms from the original data point to the new data point
  - Define the goodness by
    - Maximizing the variance of projected data (purple line)
    - Minimize mean squared distance between data points and projections (blue lines)





Preprocessing steps

1. Let $\mu = \frac{1}{n}\sum x_i$
2. Replace each x_i with $x_i - \mu$
3. Let $\sigma_j^2 = \frac{1}{n}\sum_i x_{ij}^2$
4. Replace each x_{ij} with x_{ij}/σ_j

- Step 1 & 2 zero out the mean
- Step 3 & 4 rescale each coordinate to have unit variance

1/11/21 9

9

Finding $\boldsymbol{u}_1$

- Given an unit vector $\boldsymbol{u}_1$ and a point $\boldsymbol{x}$, the length of the projection of $\boldsymbol{x}$ onto $\boldsymbol{u}_1$ is given by $\boldsymbol{x} \cdot \boldsymbol{u}_1 = \boldsymbol{x}^T\boldsymbol{u}_1$
- Our task becomes to select a unit-length $\boldsymbol{u}_1$ to maximize

$$\frac{1}{n}\sum\left(\boldsymbol{x}_i^T\boldsymbol{u}_1\right)^2 = \frac{1}{n}\sum\left(\boldsymbol{u}_1{}^T\boldsymbol{x}_i\boldsymbol{x}_i^T\boldsymbol{u}_1\right) = \boldsymbol{u}_1{}^T\left(\frac{1}{n}\sum\boldsymbol{x}_i\boldsymbol{x}_i^T\right)\boldsymbol{u}_1$$
$$= \boldsymbol{u}_1{}^T\boldsymbol{\Sigma}\boldsymbol{u}_1$$

Subject to $\|\boldsymbol{u}_1\|_2 = 1$

- $\boldsymbol{\Sigma} = \frac{1}{n}\sum\boldsymbol{x}_i\boldsymbol{x}_i^T$ is the covariance matrix
  - Remember that we replaced each x_i with $x_i - \mu$

1/11/21 10

10

Finding $\boldsymbol{u}_1$ (cont')

- By Lagrange multiplier, we have

$$\mathcal{L} = \boldsymbol{u}_1{}^T\boldsymbol{\Sigma}\boldsymbol{u}_1 + \lambda(1 - \boldsymbol{u}_1^T\boldsymbol{u}_1)$$

- Take derivative and set to 0

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{u}_1} = 2\boldsymbol{\Sigma}\boldsymbol{u}_1 - 2\lambda\boldsymbol{u}_1 = 0 \Rightarrow \boldsymbol{\Sigma}\boldsymbol{u}_1 = \lambda\boldsymbol{u}_1$$

So, $\boldsymbol{u}_1$ is an eigenvector of $\boldsymbol{\Sigma}$ with eigenvalue λ

- Since we want to maximize $\boldsymbol{u}_1{}^T\boldsymbol{\Sigma}\boldsymbol{u}_1$, $\boldsymbol{u}_1$ must be the eigenvector with maximum eigenvalue of $\boldsymbol{\Sigma}$

1/11/21 11

11

Finding $\boldsymbol{u}_2$

- Select a unit-length $\boldsymbol{u}_2$ to maximize

$$\boldsymbol{u}_2{}^T\boldsymbol{\Sigma}\boldsymbol{u}_2$$

Subject to $\|\boldsymbol{u}_2\|_2 = 1$ and $\boldsymbol{u}_2^T\boldsymbol{u}_1 = 0$

- Lagrange form

$$\mathcal{L} = \boldsymbol{u}_2{}^T\boldsymbol{\Sigma}\boldsymbol{u}_2 + \lambda_1(1 - \boldsymbol{u}_2^T\boldsymbol{u}_2) + \lambda_2\boldsymbol{u}_2^T\boldsymbol{u}_1$$

- Taking derivative and set to 0

$$\frac{\partial\mathcal{L}}{\partial\boldsymbol{u}_2} = 2\boldsymbol{\Sigma}\boldsymbol{u}_2 - 2\lambda_1\boldsymbol{u}_2 + \lambda_2\boldsymbol{u}_1 = 0$$
$$\Rightarrow 2\boldsymbol{u}_1^T\boldsymbol{\Sigma}\boldsymbol{u}_2 - 2\lambda_1\boldsymbol{u}_1^T\boldsymbol{u}_2 + \lambda_2\boldsymbol{u}_1^T\boldsymbol{u}_1 = 0$$
$$\Rightarrow 0 - 0 + \lambda_2 = 0$$
$$\Rightarrow \lambda_2 = 0$$

1/11/21 12

12

Finding $\boldsymbol{u}_2$ (cont')

- Taking derivative and set to 0

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{u}_2} = 2\boldsymbol{\Sigma}\boldsymbol{u}_2 - 2\lambda_1 \boldsymbol{u}_2 = 0$$

$$\Rightarrow \boldsymbol{\Sigma}\boldsymbol{u}_2 = \lambda_1 \boldsymbol{u}_2$$

So, $\boldsymbol{u}_2$ is an eigenvector of $\boldsymbol{\Sigma}$ with eigenvalue λ_1

- Since we want to maximize ${\boldsymbol{u}_2}^T\boldsymbol{\Sigma}\boldsymbol{u}_2$ and $\boldsymbol{u}_2 \neq \boldsymbol{u}_1$ ($\because \boldsymbol{u}_2^T\boldsymbol{u}_1 = 0$), $\boldsymbol{u}_2$ must be the eigenvector with second largest eigenvalue of $\boldsymbol{\Sigma}$

13

PCA in general

- Perform PCA by computing the eigenvectors of the *k* largest eigenvalues of the covariance matrix

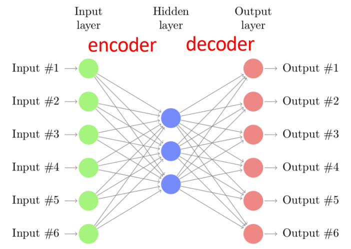
14

Autoencoder

15

Autoencoder

- An autoencoder is a neural network whose outputs are its own inputs
  - Unsupervised (or self-supervised) learning
- Objective: minimize reconstruction error



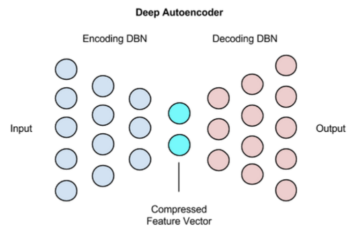
16

Autoencoder

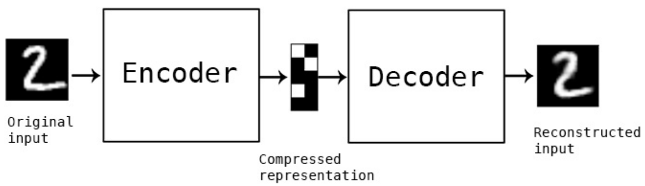
- Define
$$\boldsymbol{a}_i = g(\boldsymbol{W}\boldsymbol{x}_i),$$
$$\widetilde{\boldsymbol{x}}_i = \boldsymbol{V}\boldsymbol{a}_i = \boldsymbol{V}g(\boldsymbol{W}\boldsymbol{x}_i)$$
- Target: minimize $\frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i - \widetilde{\boldsymbol{x}}_i)^2$
- If g is linear, then:
$$\widetilde{\boldsymbol{x}}_i = \boldsymbol{V}\boldsymbol{W}\boldsymbol{x}_i$$
- Target: minimize
$$\frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i - \boldsymbol{V}\boldsymbol{W}\boldsymbol{x}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{x}_i - \boldsymbol{U}\boldsymbol{x}_i)^2$$
  – Optimal solution is PCA!

Autoencoder as non-linear PCA

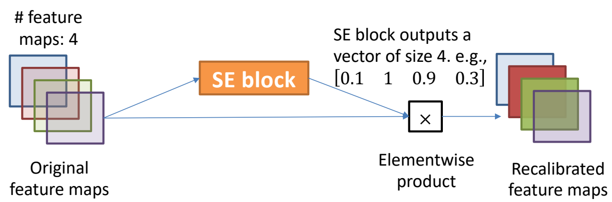
- If g is non-linear, then we have non-linear dimension reduction
- We may further use deep autoencoder to perform non-linear dimension reduction



Autoencoder example



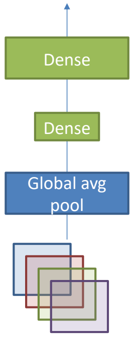
Remember the SE network?



- SE block outputs a vector $\boldsymbol{v} = [v_1, \dots, v_d]$
  - d = # feature maps
- Recalibrate the feature map $M_{\text{original}}^{(i)}$ by
  - $M_{\text{recalibrated}}^{(i)} = v_i \times M_{\text{original}}^{(i)}$

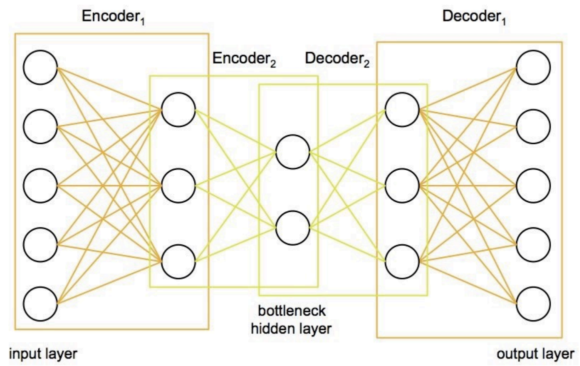
SE block architecture

- Global avg pool: mean for each feature map
  - If d feature maps, get d avg pools
    - E.g., 256 maps, avg pools $\boldsymbol{a} = [a_1, \dots, a_{256}]^T$
- Consider SE block as an autoencoder
  - Learn $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ such that $\boldsymbol{W}_2 \mathrm{Relu}(\boldsymbol{W}_1 \boldsymbol{a}) \approx \boldsymbol{a}$
    - $\boldsymbol{W}_1 \in R^{c \times d}, \boldsymbol{W}_2 \in R^{d \times c}, c < d$
      - E.g., $\boldsymbol{W}_1 \in R^{16 \times 256}, \boldsymbol{W}_2 \in R^{256 \times 16}$
  - The small vector, $\boldsymbol{m} = \mathrm{Relu}(\boldsymbol{W}_1 \boldsymbol{a})$, is a compact representation of global avg pool $\boldsymbol{a}$
    - So relationship among a_is are captured by $\boldsymbol{m}$
- The output of SE block is $\sigma\left(\boldsymbol{W}_2 \mathrm{Relu}(\boldsymbol{W}_1 \boldsymbol{a})\right)$



21

Stacked autoencoder



22

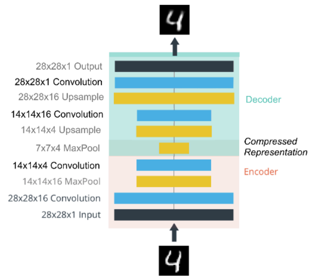
Convolutional autoencoder

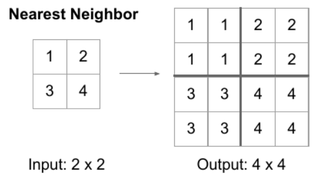
- Two main structure of a CNN:
  - Convolution
  - Pooling
- Convolutional autoencoder
  - Encoder
    - Consists of convolutional layers and pooling layers
    - Downscale spatial dimensionality (i.e., height and weight) but increase depth (i.e., # feature maps)
  - Decoder:
    - Upscale spatial size and reduce depth back to original dimensions

23

Convolutional autoencoder

- Up-sampling

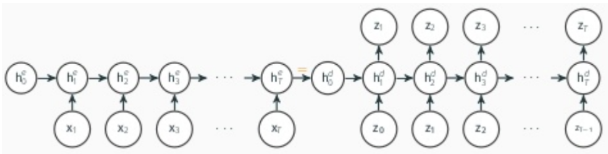




24

Recurrent autoencoder

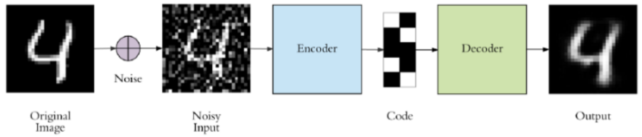
- Similar to the encoder-decoder model
- Encoder is a sequence-to-vector RNN
- Decoder is a vector-to-sequence RNN



25

Denoising autoencoder

- Add noise to input
- Network tries to recover the original (noise-free) input



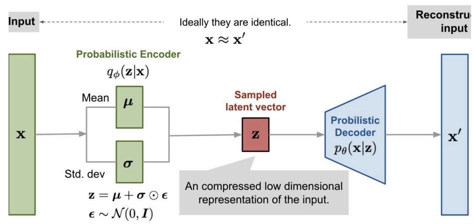
26

Variational autoencoder

- Variational autoencoder is unique because
  - It is a generative model
  - The output is probabilistic

27

Variational autoencoder



- Encoder generates mean coding $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$
- Actual coding $\boldsymbol{z} \sim N(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$
- The decoder works as normal
- VAE is a generative model because we can “sample” new $\boldsymbol{z}$s to generate new $\boldsymbol{x}'$

28

Summary

- PCA linearly projects data points into low dimension
  - Good interpretability
  - Can project new data points
- Autoencoder projects the data points non-linearly
  - Interpretability?
  - AE vs Word2Vec
- VAE can generate new instances

1/11/21 29

29