From Principal Component Analysis to Deep Learning with De-Noising Variational Auto-Encoders

Dr. Daniel Guterding
daniel.guterding@gmail.com

February 5th, 2020

Lecture topic

Data sets with a large number of dimensions (variables) are a challenge for machine learning algorithms with respect to computational effort and memory usage.

The lecture will answer three closely related questions in this context:

1. How to reduce the number of dimensions in a sensible way?
2. How to distinguish relevant from irrelevant dimensions?
3. How to suppress noise in high-dimensional data?

Discussed algorithms

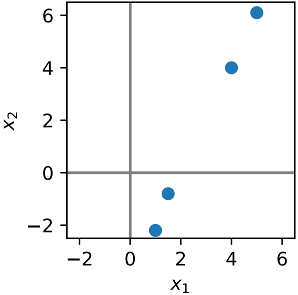
1. Principal Component Analysis (PCA)
2. De-Noising Variational Auto-Encoder (DVAE)

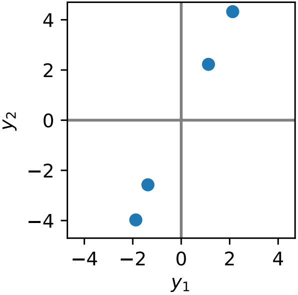
Principal Component Analysis - Foundations

- take $\mathbf{N}$ **data points** with $\mathbf{D}$ **dimensions**
- write single data point as column vector with $\underline{x_i}$ $(D \times 1)$
- matrix of all data points is $\underline{\underline{X}} = (\underline{x_1}, \underline{x_2}, \ldots, \underline{x_N})$ with $(D \times N)$
- center of all data points is $\underline{\mu} = \frac{1}{N} \sum_{i=1}^{N} \underline{x_i}$
- **centered data points** are $\underline{\underline{Y}} = (\underline{y_1}, \underline{y_2}, \ldots, \underline{y_N})$ with $\underline{y_i} = \underline{x_i} - \underline{\mu}$

Example

$$\underline{x_1} = \begin{pmatrix} 4.0 \\ 4.0 \end{pmatrix}, \underline{x_2} = \begin{pmatrix} 5.0 \\ 6.1 \end{pmatrix}, \underline{x_3} = \begin{pmatrix} 1.5 \\ -0.8 \end{pmatrix}, \underline{x_4} = \begin{pmatrix} 1.0 \\ -2.2 \end{pmatrix}, \underline{\mu} = \begin{pmatrix} 2.875 \\ 1.775 \end{pmatrix}$$





Principal Component Analysis - Foundations

- calculate **covariance matrix** $\underline{\underline{\Sigma}} = \underline{\underline{Y}}\,\underline{\underline{Y}}^{\mathsf{T}}$, real and symm. with $(D \times D)$
- solve eigenvalue equation $\underline{\underline{\Sigma}} = \underline{\underline{V}}\,\underline{\underline{\Lambda}}\,\underline{\underline{V}}^{\mathsf{T}}$, real eigvecs and eigvals
- **diagonal variance matrix** $\underline{\underline{\Lambda}} = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_D)$
- eigenvectors $\underline{\underline{V}} = (\underline{v_1}, \underline{v_2}, \ldots, \underline{v_D})$ are **orthogonal**
- $\underline{v_i}$ is i-th principal component with variance λ_i
- **principal components can be sorted by variance**

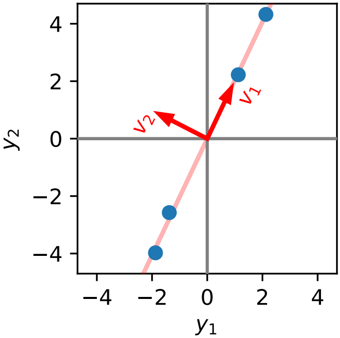
Example

$$\underline{\underline{Y}} = \begin{pmatrix} 1.125 & 2.125 & -1.375 & -1.875 \\ 2.225 & 4.325 & -2.575 & -3.975 \end{pmatrix}$$

$$\underline{\underline{\Sigma}} = \underline{\underline{Y}}\,\underline{\underline{Y}}^{\mathsf{T}} = \begin{pmatrix} 11.1875 & 22.6875 \\ 22.6875 & 46.0875 \end{pmatrix}$$

$$\underline{\underline{V}} = (\underline{v_1}, \underline{v_2}) = \begin{pmatrix} 0.442 & -0.897 \\ 0.897 & 0.442 \end{pmatrix}$$

$$\underline{\underline{\Lambda}} = \mathrm{diag}\big(\lambda_1, \lambda_2\big) = \mathrm{diag}\big(57.3, 0.02\big)$$

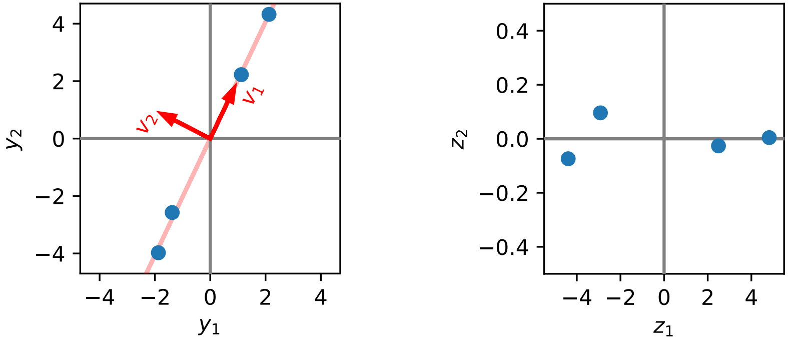


Principal Component Analysis - Foundations

- principal components form **new coordinate system**
- matrix of linear transformation is unitary $\underline{\underline{V}}^{\mathsf{T}} = \underline{\underline{V}}^{-1}$
- new data matrix $\underline{\underline{Z}} = \underline{\underline{V}}^{\mathsf{T}}\underline{\underline{Y}}$ with $(\mathrm{D}\times\mathrm{N}) = (\mathrm{D}\times\mathrm{D})\cdot(\mathrm{D}\times\mathrm{N})$
- variance is **diagonal**: $\underline{\underline{Z}}\,\underline{\underline{Z}}^{\mathsf{T}} = \underline{\underline{V}}^{\mathsf{T}}\underline{\underline{Y}}\,\underline{\underline{Y}}^{\mathsf{T}}\underline{\underline{V}} = \underline{\underline{V}}^{\mathsf{T}}\underline{\underline{\Sigma}}\,\underline{\underline{V}} = \underline{\underline{V}}^{\mathsf{T}}\underline{\underline{V}}\,\underline{\underline{\Lambda}}\,\underline{\underline{V}}^{\mathsf{T}}\underline{\underline{V}} = \underline{\underline{\Lambda}}$

Example

$$\underline{\underline{Z}} = \begin{pmatrix} 2.493 & 4.819 & -2.918 & -4.394 \\ -0.026 & 0.004 & 0.096 & -0.074 \end{pmatrix}$$



Principal Component Analysis - Recap

- principal component analysis finds **new coordinate system**
- **transformation** between old and new coordinates is **linear**
- **new basis vectors are called principal components**
- covariance matrix is diagonal in these coordinates
- principal components can be ordered by their contribution to the data set's total variance
- for data that almost lie on a line, the **variance can be concentrated within a few principal components**
- this allows for **dimensionality reduction by discarding principal components** with low associated variance

Principal Component Analysis - Dimensionality Reduction

- discarding principal component $\underline{v_i}$ discards variance λ_i
- easily done by erasing the i-th column in $\underline{\underline{V}}$
- usually $\underline{\boldsymbol{v_i}}$ **with smallest** $\boldsymbol{\lambda_i}$ **discarded first**
- new transformation $\underline{\underline{\tilde{V}}} = (\underline{v_1}, \underline{v_2}, \ldots, \underline{v_{D-1}})$ with $(D \times (D-1))$
- **reduced data** $\underline{\underline{\tilde{Z}}} = \underline{\underline{\tilde{V}}}^T \underline{\underline{Y}}$ with $((D-1) \times N)$
- discarding $\underline{v_i}$ erases i-th row in $\underline{\underline{Z}}$

Example

$$\underline{\underline{V}} = (\underline{v_1}, \underline{v_2}) = \begin{pmatrix} 0.442 & -0.897 \\ 0.897 & 0.442 \end{pmatrix}, \underline{\underline{\Lambda}} = \text{diag}\big(\lambda_1, \lambda_2\big) = \text{diag}\big(57.3, 0.02\big)$$

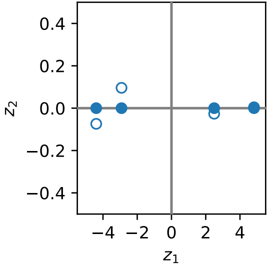
$$\underline{\underline{Z}} = \underline{\underline{V}}^T \underline{\underline{Y}} = \begin{pmatrix} 2.493 & 4.819 & -2.918 & -4.394 \\ -0.026 & 0.004 & 0.096 & -0.074 \end{pmatrix}$$

$$\underline{\underline{\tilde{V}}} = (\underline{v_1}) = \begin{pmatrix} 0.442 \\ 0.897 \end{pmatrix}$$

$$\underline{\underline{\tilde{Z}}} = \underline{\underline{\tilde{V}}}^T \underline{\underline{Y}} = \begin{pmatrix} 2.493 & 4.819 & -2.918 & -4.394 \end{pmatrix}$$

Principal Component Analysis - Dimensionality Reduction

- **transformation back to original coordinates** with $\underline{\underline{\tilde{Y}}} = \underline{\underline{\tilde{V}}}\,\underline{\underline{\tilde{Z}}} = \underline{\underline{\tilde{V}}}\,\underline{\underline{\tilde{V}}}^{\mathsf{T}}\underline{\underline{Y}}$
- although $\underline{\underline{V}}\,\underline{\underline{V}}^{\mathsf{T}} = \underline{\underline{1}}$, we have $\underline{\underline{\tilde{V}}}\,\underline{\underline{\tilde{V}}}^{\mathsf{T}} \neq \underline{\underline{1}}$
- **information loss** due to reduced dimension
- back-transformed data lie in subspace of lower dimension



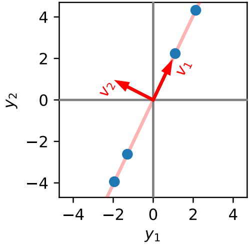
Example

$$\underline{\underline{Y}} = \begin{pmatrix} 1.125 & 2.125 & -1.375 & -1.875 \\ 2.225 & 4.325 & -2.575 & -3.975 \end{pmatrix}$$

$$\underline{\underline{\tilde{V}}} = (\underline{v_1}) = \begin{pmatrix} 0.442 \\ 0.897 \end{pmatrix}$$

$$\underline{\underline{\tilde{Z}}} = \underline{\underline{\tilde{V}}}^{\mathsf{T}}\underline{\underline{Y}} = \begin{pmatrix} 2.493 & 4.819 & -2.918 & -4.394 \end{pmatrix}$$

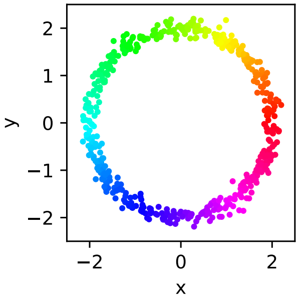
$$\underline{\underline{\tilde{Y}}} = \underline{\underline{\tilde{V}}}\,\underline{\underline{Z}} = \underline{\underline{\tilde{V}}}\,\underline{\underline{\tilde{V}}}^{\mathsf{T}}\underline{\underline{Y}} = \begin{pmatrix} 1.101 & 2.129 & -1.289 & -1.941 \\ 2.237 & 4.323 & -2.617 & -3.942 \end{pmatrix}$$



Principal Component Analysis - Problems

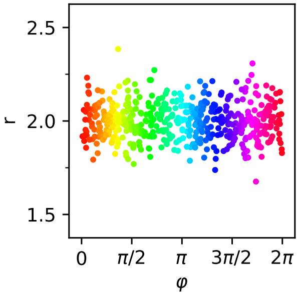
Non-Linear Data

- principal component analysis only works well for **linear data**
- introduction of non-linear variables would further increase dimensionality
- kernel methods are solution (kernel PCA)

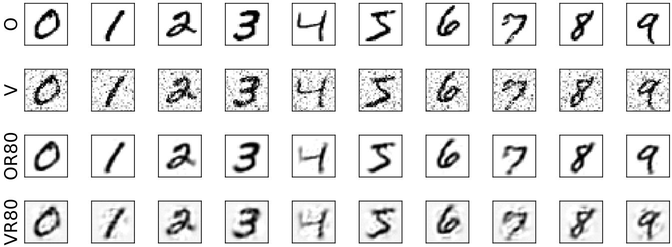


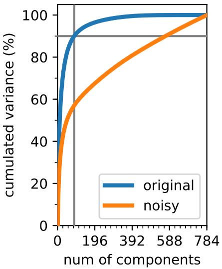
Algorithm

- diagonalization of covariance matrix does not scale well with dimensionality D
- implementations usually use **singular value decomposition**
- same result, but relation to variance harder to understand



Principal Component Analysis - De-Noising of Images

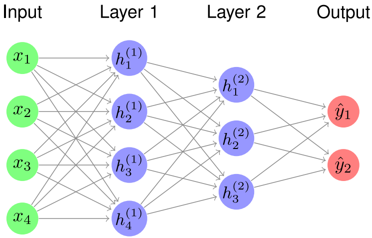


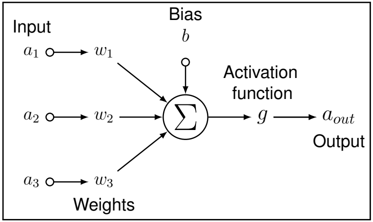


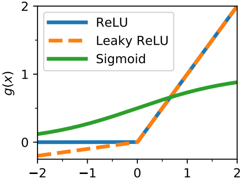
- MNIST Numbers contains 70000 hand-written numbers from 0 to 9
- $28 \times 28 = 784$ pixel per image, values from 0 (white) to 255 (black)
- **transform** every **image to column vector** with dimension (784×1)
- 80 (about 10%) of principal components contain 90% of variance
- we introduce random noise into a derived data set
- contained variance converges slower than for original data
- noise leads to small reconstruction errors and lowered contrast

Artificial Neural Networks - Foundations

- inspired by the structure of the brain
- layers of **simple units** with **complex connections**
- allows for **non-linear transformations**
- complexity of network can be controlled
- unit performs **weighted sum** and applies **activation function**
- output of unit is $a_{\text{out}} = g(b + \sum_i a_i w_i)$
- activation function ReLU $g(x) = \max(0, x)$ popular
- **weights w_i must be learned**
- high computational effort, training best done on GPUs

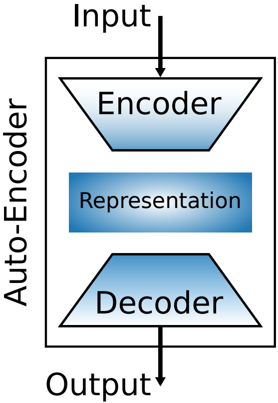






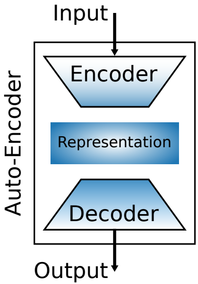
Auto-Encoder - Foundations

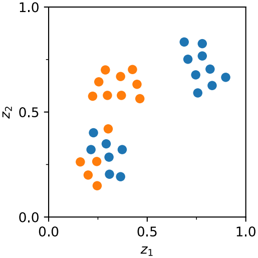
- encoder compresses high-dim. input data into low-dim. representation vector $\underline{z}$
- decoder decompresses $\underline{z}$ into high-dim. output data
- compression is **lossy**, depends on structure of encoder/decoder and dimension of $\underline{z}$
- principal component analysis is primitive Auto-Encoder; $\underline{\underline{\tilde{V}}}^{\mathsf{T}}$ as encoder, $\underline{\underline{\tilde{V}}}$ as decoder
- **real Auto-Encoder uses neural network as encoder/decoder**
- allows for better compression and reconstruction because of more complex structure



Auto-Encoder - Foundations

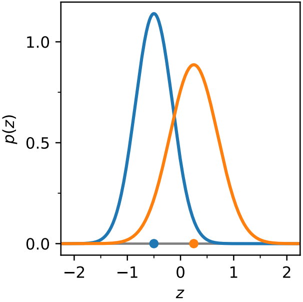
- **training** of encoder/decoder **minimizes quadratic deviation** between input and output
- representation vector $\underline{z}$ lies in so-called **latent space**
- possible to generate and decode new $\underline{z}$, so-called **generative model**
- **transformation is point-wise**, points that are close in latent space not necessarily related
- no guarantee that similar $\underline{z}$ lead to similar outputs
- **insufficient reconstruction between training data**
- Variational Auto-Encoder solves these problems

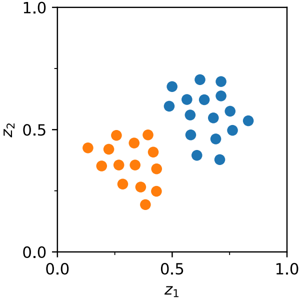




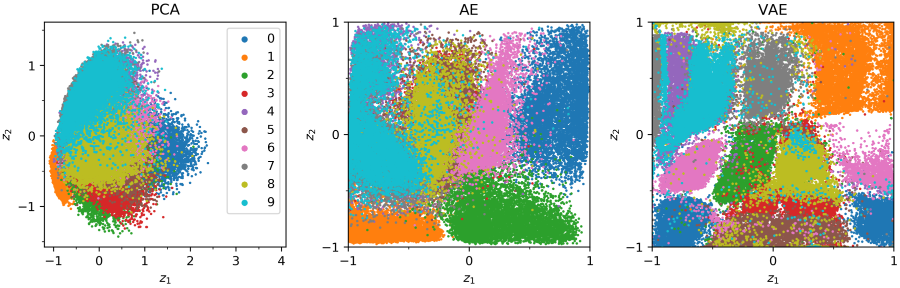
Variational Auto-Encoder - Foundations

- **learns probability distribution** $\mathrm{p}(\underline{z})$
- input data are mapped to normal distribution with $\underline{\mu}$ and $\ln \underline{\underline{\Sigma}}$, no correlation
- $\underline{z} = \underline{\mu} + \underline{\underline{\sigma}}\,\underline{\varepsilon}$, $\underline{\underline{\sigma}} = \exp\left(\frac{1}{2} \ln \underline{\underline{\Sigma}}\right)$, $\underline{\varepsilon} \sim \mathcal{N}(0,1)$
- **Kullback-Leibler divergence** measures deviation of distribution from $\mathcal{N}(0,1)$
- $D_{KL} = \frac{1}{2} \sum_i \left(-1 - \ln(\sigma_i^2) + \mu_i^2 + \sigma_i^2 \right) \geqslant 0$
- **KLD is added to quadratic error**, regularizes $\underline{\mu}$ and $\ln \underline{\underline{\Sigma}}$
- KLD centers $\underline{z}$ of training data around $\underline{0}$
- **similar input data, similar** $\underline{z}$
- algorithm seems ad-hoc, but mathematically well-established



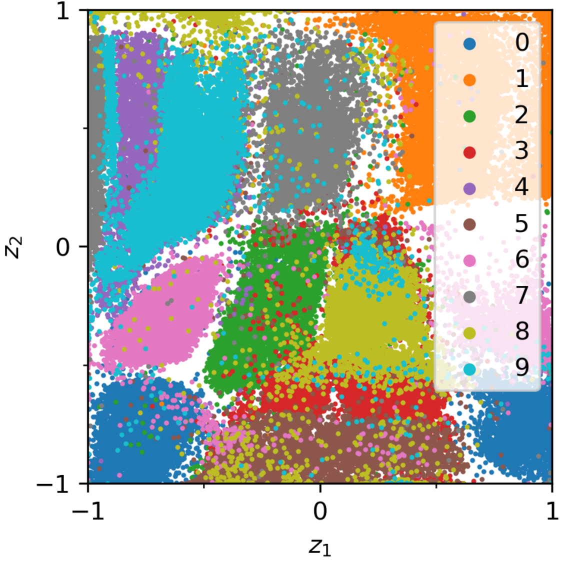


Algorithms Comparison - Latent Space



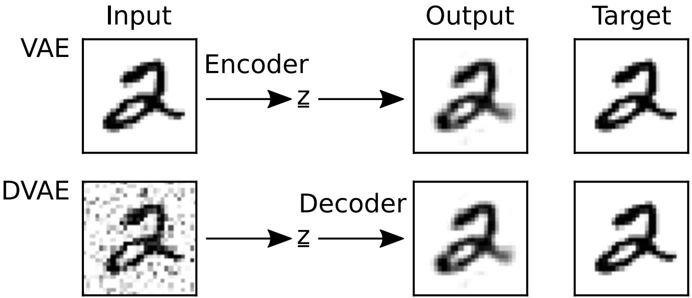
- use MNIST data set
- plots use classification info, model does not
- in all cases two-dimensional representation is learned
- numbers 4, 5, 6 particulary problematic
- AE and VAE also not perfect
- grouping of numbers is done best by VAE

Variational Auto-Encoder - Generative Learning

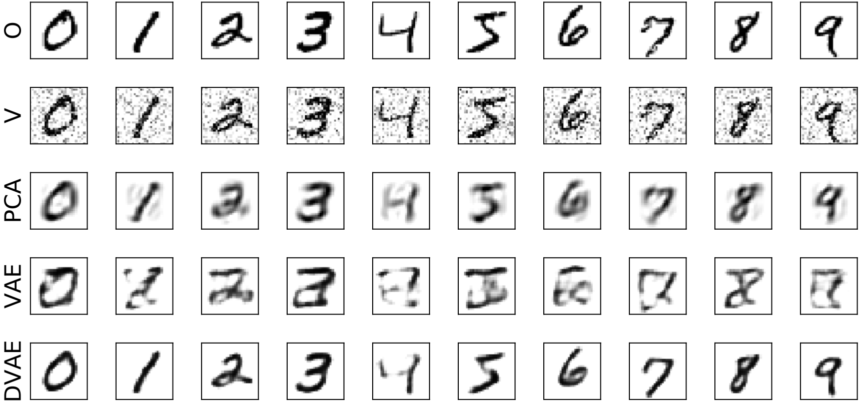


De-Noising Variational Auto-Encoder - Foundations

- train VAE with **artifical noisy input data**
- calculate reconstruction error w.r.t. **original data**
- encoder learns de-noising
- latent space in DVAE similar to VAE
- **decoder unchanged w.r.t. VAE**
- rigorous mathematical justification exists



De-Noising VAE - De-Noising of Images



- 32 dimensions in latent space
- **PCA and VAE trained with original data**
- DVAE shows best image reconstruction
- **VAE very sensitive to noise**

Summary

Principal Component Analysis

- linear transformation diagonalizes variance matrix
- principal comp. can be sorted by variance
- dimensionality reduction by discarding principal components with low variance

De-Noising Variational Auto-Encoder

- combination of neural networks as encoder/decoder
- learns representation of data as low-dim. probability distribution
- trained with artificial noisy data and known clean data as target
- generative model

Literature

- Hastie, Tibshirani, Friedman: The Elements of Statistical Learning
- Geron: Hands-On Machine Learning with Scikit-Learn and TensorFlow
- Goodfellow, Bengio, Courville: Deep Learning
- Foster: Generative Deep Learning
- Kingma, Welling: Auto-Encoding Variational Bayes, arXiv:1312.6114
- Im *et al.*: Denoising Criterion for Variational Auto-Encoding Framework, arXiv:1511.06406
- Doersch: Tutorial on Variational Autoencoders, arXiv:1606.05908
- Rolínek, Zietlow, Martius: Variational Autoencoders pursue PCA directions (by accident), arXiv:1812.06775

Training complexities of ML algorithms

- all listed complexities are provable upper bounds
- more efficient implementations may and do exist
- N: number of training samples
- D: number of features
- M: number of trees
- **Linear regression**: $\mathcal{O}\left(ND^2 + D^3\right)$, because $\underline{\beta} = \left(\underline{\underline{X}}\,\underline{\underline{X}}^{\mathsf{T}}\right)^{-1}\underline{\underline{X}}\,\underline{Y}$
- **Support Vector Machine**: $\mathcal{O}\left(N^2D + N^3\right)$
- **Decision tree**: $\mathcal{O}\left(N^2D\right)$
- **Random forest**: $\mathcal{O}\left(N^2DM\right)$
- **Artificial Neural Network**: no general proof available

MNIST Principal Components