Dimensionality Reduction

Lecture series „Machine Learning“

Niels Landwehr

Research Group „Data Science“
Institute of Computer Science
University of Hildesheim

Agenda

- Linear dimensionality reduction: PCA
- Nonlinear dimensionality reduction: Autoencoders

Agenda

- Linear dimensionality reduction: PCA
- Nonlinear dimensionality reduction: Autoencoders

Motivation: Dimensionality Reduction

- What is dimensionality reduction?

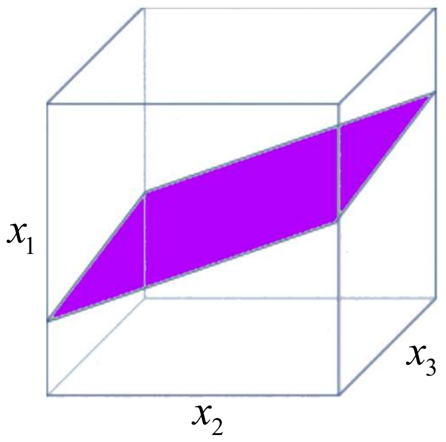
- Assume we are given a data set $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ with $\mathbf{x}_n \in \mathcal{X}$, where for this lecture we assume that instances are real-valued feature vectors, that is, $\mathcal{X} = \mathbb{R}^M$
- The dimensionality *M* of the input space can be quite high, for example
  - Images: *M* is the number of RGB pixel values, so for an RGB image of size 256x256 pixel we have $M = 256 \cdot 256 \cdot 3 = 196,608$
  - Texts: in a bag-of-word representation, every word that could appear in a text is a feature. Depending on the size of the vocabulary considered, we often have $M > 100,000$

- Idea: For many domains, the data will approximately lie on a manifold (subspace) that has a much lower dimensionality *K*
- That is, the actual degrees of freedom to choose a data point are much fewer than *M*
- *K* is also called the intrinsic dimensionality of the data
- The intrinsic dimensionality depends on the data set/generating process

Visualization: Lower-Dimensional Subspace

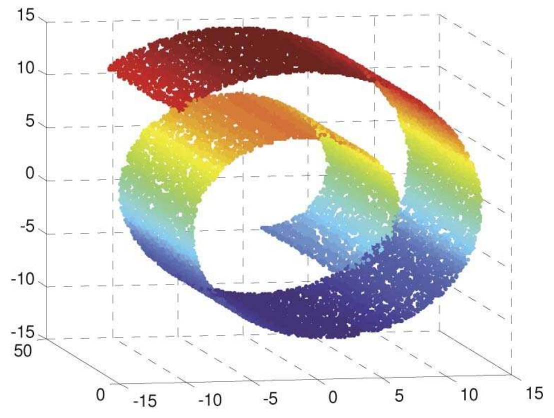
- Toy examples for lower-dimensional subspaces: data points in a 3D-space can approximately lie on a 2D-manifold

Dimensionality of the original space is *M=3*



The plane is a linear subspace with dimension *K=2*

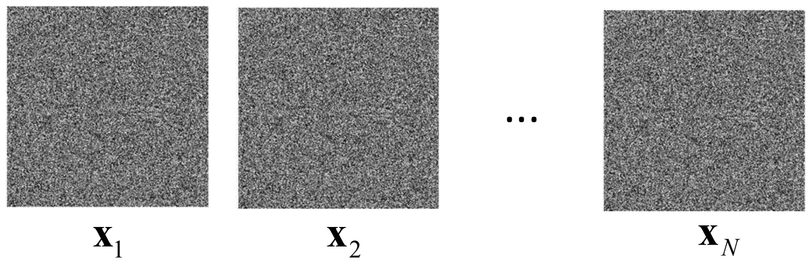
Dimensionality of the original space is *M=3*



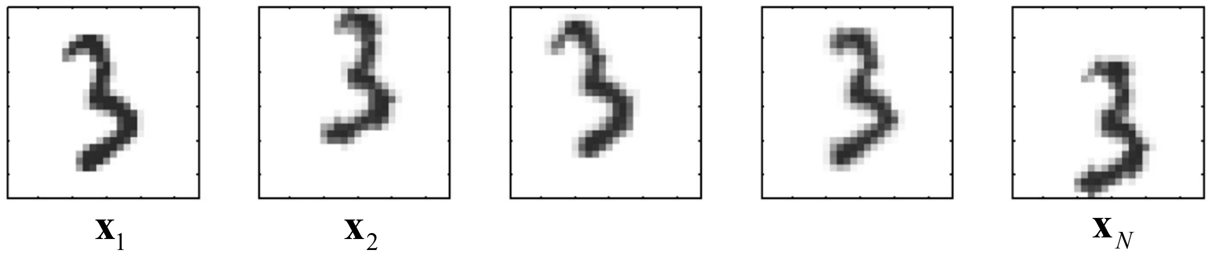
The "spiral" is a non-linear subspace with dimension *K=2*

Motivation: Dimensionality Reduction

- As an example with more dimensions, let‘s look at grayscale image data
- First, assume a data set of 256 x 256 images that all show random noise:



- The intrinsic dimensionality of this data set is *M*=65536 (given large enough *N*). There is no low-dimensional subspace
- In contrast, assume now a data set of images that show a fixed character „3“, embedded into an image at a random position and with a random rotation:

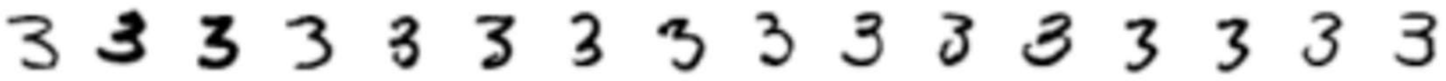


- Even though the dimensionality of the input space is high, the intrinsic dimensionality of the data is only three because there are three degrees of freedom

Figures: C. Bishop, Pattern Recognition and Machine Learning, 2006

Motivation: Dimensionality Reduction

- As a third example, consider a data set of general handwritten digits „3“



- The intrinsic dimensionality of this data will be higher than three, because there is variation in position, rotation, scaling, and writing style, but still much less than the formal dimensionality of the input space

- In the handwritten digit example, the manifold or subspace the data lies on is clearly nonlinear in the original input space
  - For example, looking at the degree of freedom „translate to the left“: if we translate a digit from left to right, pixel values can change from white to black and then back to white again, which cannot be a linear function of the translation
  - Therefore the function mapping from degrees of freedom to the full input space cannot be linear

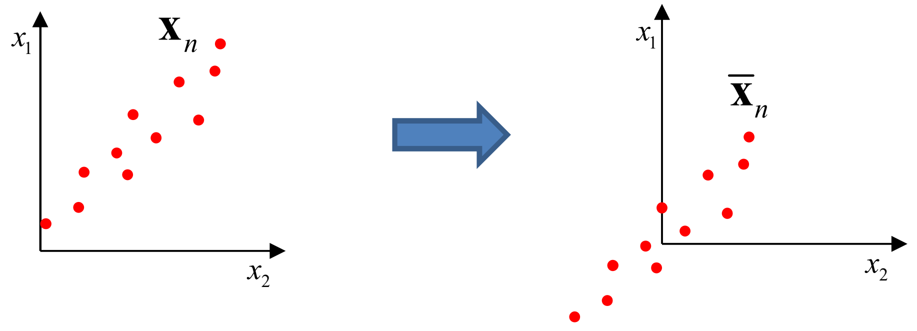
Empirical Mean Zero

- In the first part of the lecture, we will however look at linear subspaces
- For mathematical convenience, we will assume that the data is normalized to an empirical mean of zero:

$$\frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n = \mathbf{0} \in \mathbb{R}^M$$

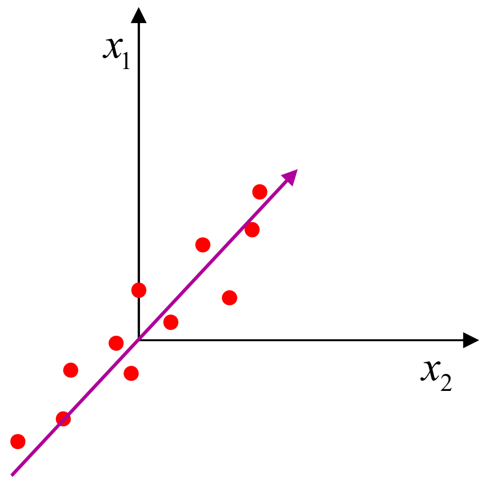
- Any data set can easily be normalized to empirical mean zero by substracting the empirical mean vector

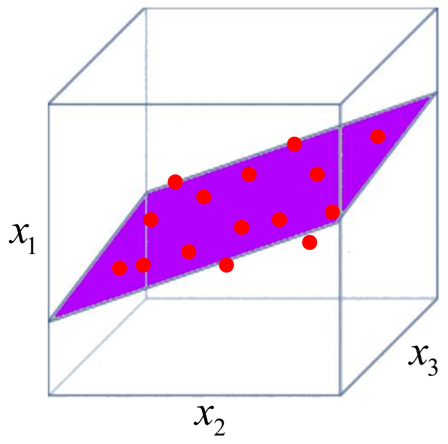
$$\overline{\mathbf{x}}_n = \mathbf{x}_n - \frac{1}{N}\sum_{n'=1}^{N}\mathbf{x}_{n'}$$



Linear Subspaces

- As a toy examples, consider the following 2D and 3D data:





- Here, data points lie close to a linear subspace of dimension *M-1* in the original *M*-dimensional space (*M=2* left, *M=3* right)
  - Instances do not necessarily lie directly in the linear subspace, but can be very well approximated by projecting them into the subspace
  - In dimensionality reduction, we would basically consider the differences between the instances and their projection on the subspace as „noise“

Principal Component Analysis

- **Principal component analysis** (or PCA) is a widely used technique of dimensionality reduction
- It can be used to identify optimal linear subspaces of a given dimension $K < M$ that data (approximately) lies on
- Data can then be projected onto the linear subspace, thereby reducing its dimension

- What is an „optimal“ subspace?
  - For PCA, optimality can be defined in two different ways: can require that the projected data have maximum variance in the subspace, or that the difference between the original and projected data is minimal („minimal reconstruction error“)
  - It turns out that these two criteria lead to exactly the same solution

- We will start by looking at the first criterion (maximum variance) to derive PCA, and then give a formal statement about the minimum reconstruction error

PCA: Projection on Single Dimension

- To derive the optimal subspace according to the maximum variance criterion, let‘s start with the simplified case of *K*=1
- In this case, the one-dimensional subspace is represented by a vector $\mathbf{u}_1 \in \mathbb{R}^M$, such that the subspace is given by the set of points $\mathcal{U} = \{\alpha \mathbf{u}_1 \mid \alpha \in \mathbb{R}\} \subset \mathbb{R}^M$
- Note that the subspace is invariant under rescalings of the vector $\mathbf{u}_1$. To make the representation more canonical, we require that $\| \mathbf{u}_1 \| = 1$

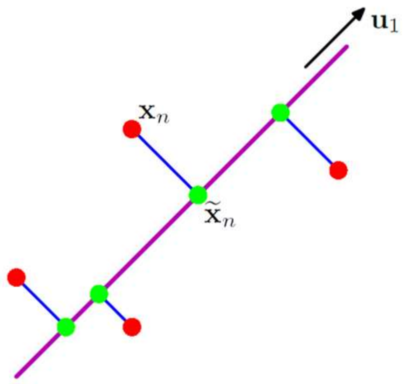
- A point $\mathbf{x}_n \in \mathbb{R}^M$ can be projected onto the subspace as follows:

In the original space $\mathbb{R}^M$, the projection $\tilde{\mathbf{x}}_n$ of a point $\mathbf{x}_n$ is given by

$$\tilde{\mathbf{x}}_n = \underbrace{\mathbf{u}_1^{\mathrm{T}} \mathbf{x}_n}_{\substack{\text{coordinate} \\ \text{in space } \mathcal{U}}} \mathbf{u}_1 \in \mathbb{R}^M$$

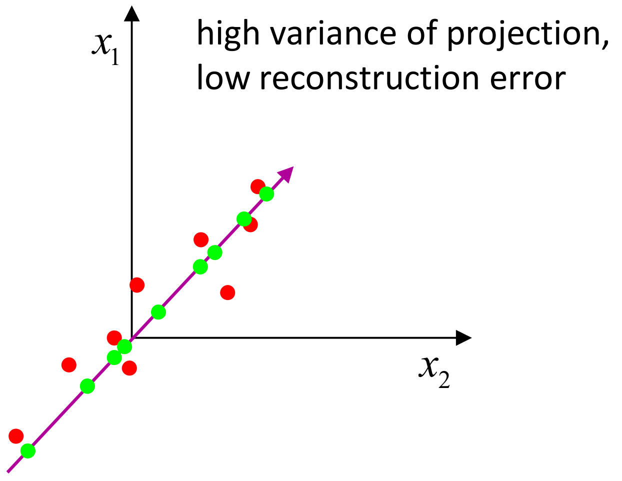
The reconstruction error of the projection is $\| \mathbf{x}_n - \tilde{\mathbf{x}}_n \|$

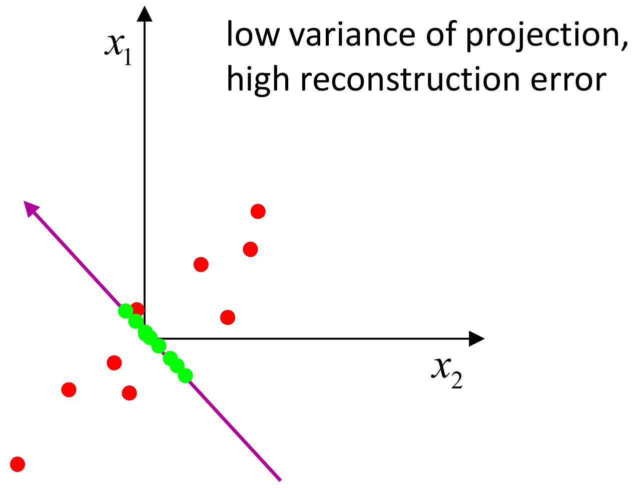
In the coordinate system of the space $\mathcal{U}$, the projected point is given by $\mathbf{u}_1^{\mathrm{T}} \mathbf{x}_n \in \mathbb{R}$



Intuition: Variance Maximization

- The criterion for choosing the subspace $\mathcal{U}$ will be to maximize the variance of the projected data
- Why is this a good idea?
- Variance is highest if the subspace $\mathcal{U}$ is aligned well with the „main direction“ in the data. This also minimizes the reconstruction error as stated later





Mean and Variance of Projection

- More formally: Let $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, and $\mathcal{U} = \{\alpha \mathbf{u}_1 \mid \alpha \in \mathbb{R}\}$

- Within the space $\mathcal{U}$, the mean of the projected data set $\mathbf{X}$ is

$$\tilde{\mu} = \frac{1}{N} \sum_{n=1}^{N} \underbrace{\mathbf{u}_1^{\mathrm{T}} \mathbf{x}_n}_{\substack{\text{coordinate} \\ \text{in space } \mathcal{U}}} = \mathbf{u}_1^{\mathrm{T}} \boldsymbol{\mu}$$

Note these are coordinates in $\mathcal{U}$, therefore one-dimensional

where $\boldsymbol{\mu} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n$ is the empirical mean of the original data

- The variance of the data set $\mathbf{X}$ projected onto the subspace $\mathcal{U}$ is

$$\tilde{\sigma}^2 = \frac{1}{N} \sum_{n=1}^{N} \left(\mathbf{u}_1^{\mathrm{T}} \mathbf{x}_n - \tilde{\mu} \right)^2$$

- If we assume that the data has empirical mean zero (see above), it follows that $\tilde{\mu} = 0$ and the variance simplifies to

$$\tilde{\sigma}^2 = \frac{1}{N} \sum_{n=1}^{N} \left(\mathbf{u}_1^{\mathrm{T}} \mathbf{x}_n \right)^2$$

Mean and Variance of Projection

- We can express the variance of the projected data in $\mathcal{U}$ as

$$\begin{aligned}\tilde{\sigma}^2 &= \frac{1}{N}\sum_{n=1}^{N}\left(\mathbf{u}_1^{\mathrm{T}}\mathbf{x}_n\right)^2 \\ &= \frac{1}{N}\sum_{n=1}^{N}\left(\mathbf{u}_1^{\mathrm{T}}\mathbf{x}_n\right)\left(\mathbf{x}_n^{\mathrm{T}}\mathbf{u}_1\right) \\ &= \mathbf{u}_1^{\mathrm{T}}\left(\frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n\left(\mathbf{x}_n^{\mathrm{T}}\mathbf{u}_1\right)\right) \\ &= \mathbf{u}_1^{\mathrm{T}}\left(\frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n\mathbf{x}_n^{\mathrm{T}}\right)\mathbf{u}_1 \\ &= \mathbf{u}_1^{\mathrm{T}}\mathbf{S}\mathbf{u}_1\end{aligned}$$

where $\mathbf{S} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n\mathbf{x}_n^{\mathrm{T}}$ is the empirical covariance matrix of the original data $\mathbf{X}$

Maximizing Variance in Projection Space

- We now maximize the variance $\mathbf{u}_1{}^{\mathrm{T}}\mathbf{S}\mathbf{u}_1$ in the subspace direction $\mathbf{u}_1$

- To prevent the trivial solution of $\| \mathbf{u}_1 \| \rightarrow \infty$, we need to use the assumption (see above) that $\| \mathbf{u}_1 \| = 1$ (or equivalently $\mathbf{u}_1{}^T\mathbf{u}_1 = 1$):

$$\arg\max_{\mathbf{u}_1} \mathbf{u}_1{}^T\mathbf{S}\mathbf{u}_1$$
$$\text{subject to } \mathbf{u}_1{}^T\mathbf{u}_1 = 1$$

- This is a constrained optimization problem with an equality constraint, which we will solve using a Lagrange multiplier λ

Lagrange Multipliers for Equality Constraints

- In general, if we have functions $f:\mathbb{R}^M \to \mathbb{R}$ and $g:\mathbb{R}^M \to \mathbb{R}$ and a constant $c \in \mathbb{R}$, and we want to find

$$\begin{aligned}&\arg\max_{\mathbf{x}\in\mathbb{R}^M} f(\mathbf{x})\\&\text{subject to } g(\mathbf{x}) = c\end{aligned}$$

  we can instead find the stationary point (point where all partial derivatives are zero) of the unconstrained function $Z:\mathbb{R}^{M+1} \to \mathbb{R}$ given by:

$$Z(\mathbf{x},\lambda) = f(\mathbf{x}) + \lambda(c - g(\mathbf{x}))$$

Maximum Variance With Lagrange Multipliers

- The subspace maximizing the variance of the projected data can therefore be found by finding the stationary point of the function

$$\mathbf{u}_1^T \mathbf{S} \mathbf{u}_1 + \lambda(1 - \mathbf{u}_1^{\mathrm{T}} \mathbf{u}_1)$$

- The derivative with respect to the vector $\mathbf{u}_1$ is given by

$$\frac{\partial}{\partial \mathbf{u}_1} \mathbf{u}_1^T \mathbf{S} \mathbf{u}_1 + \lambda(1 - \mathbf{u}_1^{\mathrm{T}} \mathbf{u}_1) = 2\mathbf{S}\mathbf{u}_1 - 2\lambda \mathbf{u}_1$$

In general, for a constant matrix $\mathbf{A}$ and vector $\mathbf{x}$ it holds that

$$\frac{\partial}{\partial \mathbf{x}} \mathbf{x}^{\mathrm{T}} \mathbf{A} \mathbf{x} = 2\mathbf{A}\mathbf{x}$$

- The derivative with respect to λ is $1 - \mathbf{u}_1^{\mathrm{T}} \mathbf{u}_1$

- Setting the derivative w.r.t $\mathbf{u}_1$ to zero yields the following condition for the optimal $\mathbf{u}_1$:

$$\mathbf{S}\mathbf{u}_1 = \lambda \mathbf{u}_1$$

- If we left-multiply with $\mathbf{u}_1^{\mathrm{T}}$ and use the condition $\mathbf{u}_1^{\mathrm{T}} \mathbf{u}_1 = 1$ we additionally obtain

$$\tilde{\sigma}^2 = \mathbf{u}_1^{\mathrm{T}} \mathbf{S} \mathbf{u}_1 = \lambda$$

Subspace as First Principal Component

- The condition $\mathbf{S}\mathbf{u}_1 = \lambda \mathbf{u}_1$ implies that $\mathbf{u}_1$ has to be an eigenvector of the empirical covariance matrix $\mathbf{S}$

- **Definition** (eigenvectors/eigenvalues): given a matrix $\mathbf{A} \in \mathbb{R}^{M \times M}$, a vector $\mathbf{x} \in \mathbb{R}^M$ is called an **eigenvector** of the matrix $\mathbf{A}$ with eigenvalue $\lambda \in \mathbb{R}$ if

$$\mathbf{A}\mathbf{x} = \lambda \mathbf{x}$$

- Eigenvectors can always be rescaled: $\mathbf{A}\mathbf{x} = \lambda \mathbf{x}$ implies $\mathbf{A}(\alpha \mathbf{x}) = \lambda(\alpha \mathbf{x})$. But $\mathbf{u}_1$ is normalized by $\mathbf{u}_1{}^T \mathbf{u}_1 = 1$, so scaling is clear
- In general, a matrix can have different eigenvectors (ignoring rescaling), that in general also have different eigenvalues
- The condition

$$\tilde{\sigma}^2 = \mathbf{u}_1{}^{\mathrm{T}} \mathbf{S} \mathbf{u}_1 = \lambda$$

  shows that the variance is the eigenvalue. Thus, a subspace maximizing the variance is given by the eigenvector with the largest eigenvalue
- The eigenvector with largest eigenvalue is also called the **first principle component**

PCA with General Subspaces: Maximal Variance

- The analysis so far has only covered the case of a one-dimensional subspace $\mathcal{U} = \{\alpha \mathbf{u}_1 \mid \alpha \in \mathbb{R}\}$
- The general principal component analysis problem can be stated as follows:
- **Given**:
  - Data $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ with $\mathbf{x}_n \in \mathbb{R}^M$ and $N^{-1}\sum_{n=1}^{N} \mathbf{x}_n = \mathbf{0}$
  - A subspace dimensionality $K \leq M$
- **Find**:
  - Vectors $\mathbf{u}_1, \ldots, \mathbf{u}_K \in \mathbb{R}^M$ with $\mathbf{u}_k{}^T \mathbf{u}_k = 1$ and $\mathbf{u}_k{}^\mathrm{T} \mathbf{u}_i = 0$ for $k \neq i$
  - Such that the variance of the data projected to the subspace

$$\mathcal{U} = \{\alpha_1 \mathbf{u}_1 + \ldots + \alpha_K \mathbf{u}_K \mid (\alpha_1, \ldots, \alpha_K) \in \mathbb{R}^K\} \subseteq \mathbb{R}^M$$

  where the data has coordinates $(\mathbf{x}_n^\mathrm{T} \mathbf{u}_1, \ldots, \mathbf{x}_n^\mathrm{T} \mathbf{u}_K) \in \mathbb{R}^K$, is maximal. The variance is given by $\sum_{k=1}^{K} \mathbf{u}_k{}^\mathrm{T} \mathbf{S} \mathbf{u}_k$

- **Solution**: $\mathbf{u}_1, \ldots, \mathbf{u}_K \in \mathbb{R}^M$ are the *K* eigenvectors of the covariance matrix $\mathbf{S} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n{}^\mathrm{T}$ with largest eigenvalues

PCA with General Subspaces: Minimal Error

- Principle component analyis can also be derived from the criterion of minimal reconstruction error as follows
- **Given**:
  - Data $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ with $\mathbf{x}_n \in \mathbb{R}^M$ and $N^{-1}\sum_{n=1}^{N} \mathbf{x}_n = \mathbf{0}$
  - A subspace dimensionality $K \le M$
- **Find**:
  - Vectors $\mathbf{u}_1, \ldots, \mathbf{u}_K \in \mathbb{R}^M$ with $\mathbf{u}_k{}^T \mathbf{u}_k = 1$ and $\mathbf{u}_k{}^\mathrm{T} \mathbf{u}_i = 0$ for $k \neq i$
  - Such that the data projected to the subspace

$$\mathcal{U} = \{\alpha_1 \mathbf{u}_1 + \ldots + \alpha_K \mathbf{u}_K \mid (\alpha_1, \ldots, \alpha_K) \in \mathbb{R}^K\}$$

which is given by $\{\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_N\}$ with $\tilde{\mathbf{x}}_n = \sum_{k=1}^{K} \mathbf{x}_n^\mathrm{T} \mathbf{u}_k \mathbf{u}_k \in \mathbb{R}^M$ has minimal reconstruction error

$$J = \sum_{n=1}^{N} \| \tilde{\mathbf{x}}_n - \mathbf{x}_n \|^2$$

- **Solution**: $\mathbf{u}_1, \ldots, \mathbf{u}_K \in \mathbb{R}^M$ are the *K* eigenvectors of the covariance matrix $\mathbf{S} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n \mathbf{x}_n{}^\mathrm{T}$ with largest eigenvalues

PCA as Eigenvector Problem

- To compute the principal components $\mathbf{u}_1,\ldots,\mathbf{u}_K \in \mathbb{R}^M$ for a subspace dimensionality $K < M$, we need to find the *K* eigenvectors of the matrix $\mathbf{S}$ corresponding to the largest *K* eigenvalues

- Without proof: For a symmetric matrix $\mathbf{S}$ it holds that we can always find eigenvectors that are orthonormal, that is,

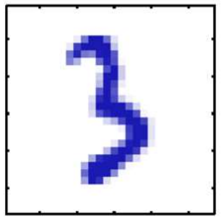
$$\mathbf{u}_k{}^{\mathrm{T}}\mathbf{u}_i = \begin{cases} 1 : k = i \\ 0 : k \neq i \end{cases}$$

  as required for the principal components

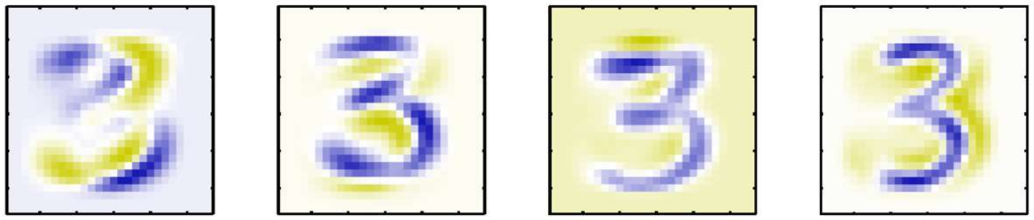
- There are different algorithms available for computing such eigenvalues, widely used ones are for example the QR-algorithm or the so-called power method (no details here)
- The runtime of these methods are generally $O(M^3)$ or $O(KM^2)$

Example: PCA on Image Data

- As an example for PCA on high-dimensional data, we consider the data set consisting of scans of the hand-written digit „3“



- The original dimension of this data is 28 x 28 = 784 (grayscale pixel data)
- We can now compute the first four eigenvectors $\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3, \mathbf{u}_4 \in \mathbb{R}^{784}$ and visualize them as images (with coordinate values indicated by a color map from green through white to blue):

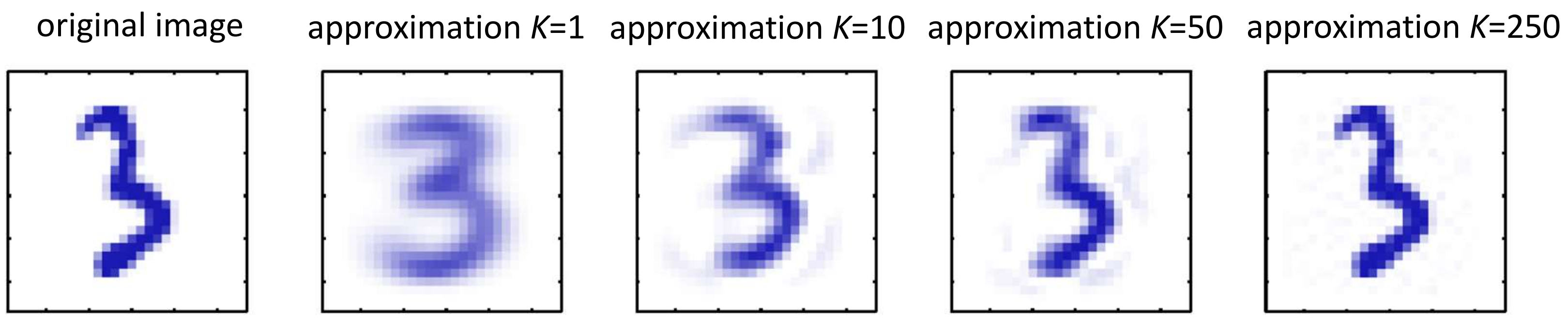


- Intution: most images of handwritten digits „3“ can be approximated reasonably well by a linear combination of these images

Figures: C. Bishop, Pattern Recognition and Machine Learning, 2006

Approximation Error

- Any image of a handwritten digit „3“ can be (optimally) approximated by a linear combination of the first *K* eigenvectors
- For a concrete image of a handwritten digit, we can see how the reconstruction error decreases as a function of the subspace dimensionality *K:*

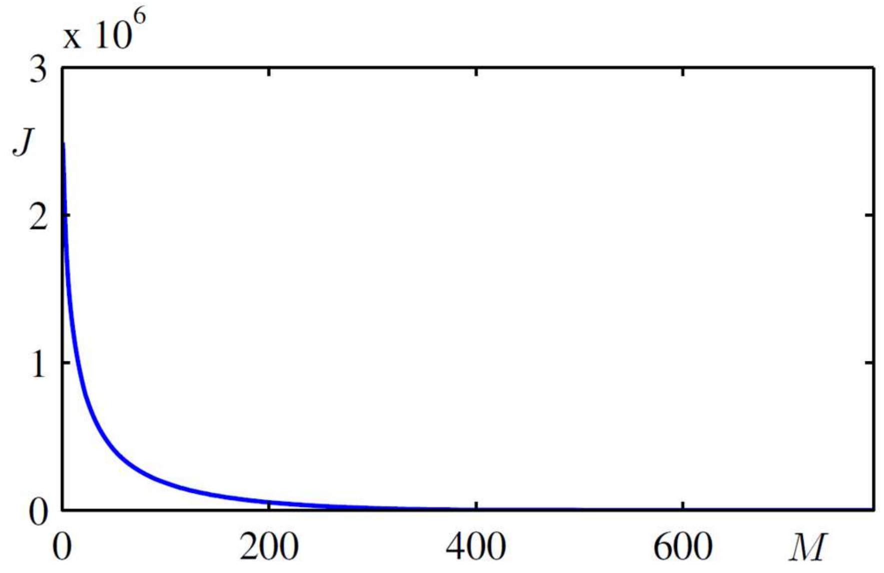


- Principal component analysis can in this way be used for data compression:
  - find the first *K* principal components (eigenvectors) with PCA
  - store the eigenvectors and for each image in the data set the coordinates of that image in the projection
  - images can then be reconstructed from the eigenvectors and the coordinates

Figures: C. Bishop, Pattern Recognition and Machine Learning, 2006

Projection Error as a Function of *K*

- We can also plot the reconstruction error *J* as a function of the subspace dimensionality, again for the digit „3“ data set:



- It can be observed that the reconstruction error decreases quickly for the first approximately 200 components and then is almost zero

Figures: C. Bishop, Pattern Recognition and Machine Learning, 2006

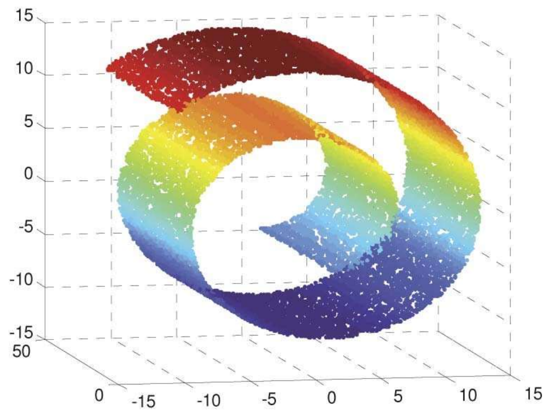
Agenda

- Linear dimensionality reduction: PCA
- Nonlinear dimensionality reduction: Autoencoders

Nonlinear Dimensionality Reduction?

- PCA is a linear operation: it finds a linear subspace in the data (spanned by the eigenvectors of the empirical covariance matrix) and projects data onto this subspace
- This linearity is mathematically convenient, but can limit the efficiency of PCA if the data lie on a low-dimensional manifold that is not linear

2D manifold within a 3D space



Translated and rotated digits (intrinsic dimensionality 3)

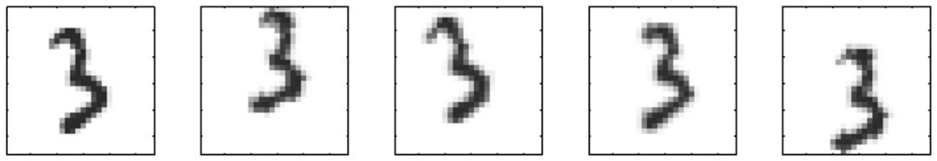


Figure: Ho & Huang, Fractal-Based Intrinsic Dimension Estimation and Its Application in Dimensionality Reduction, 2012

Autoencoders

- An alternative to the linear projection carried out by PCA is to construct a nonlinear projection using a neural network

- Review: fully connected neural networks (or multilayer perceptrons) map input instances $\mathbf{x} \in \mathbb{R}^M$ to an output $f_{\boldsymbol{\theta}}(\mathbf{x}) \in \mathbb{R}^K$, which in the case of classification problems is interpreted as class probabilities
- The mapping is obtained by pushing the input through several „layers" of linear models each followed by a nonlinear activation function

Example: $M = 3, \; T = 2, \; k_1 = 4, \; D = 1$

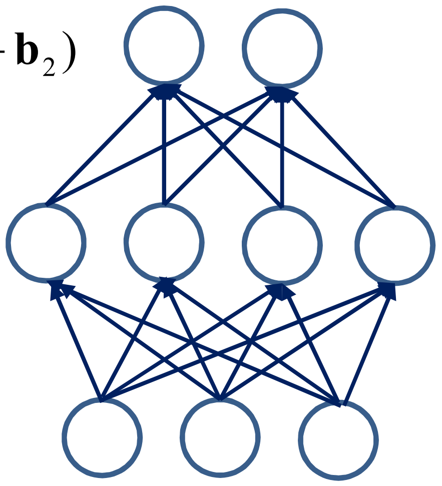
$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \text{softmax}(\mathbf{W}_2 \mathbf{z}_1 + \mathbf{b}_2)$$

$$\mathbf{W}_2 \in \mathbb{R}^{T \times k_1}, \quad \mathbf{b}_2 \in \mathbb{R}^T$$

$$\mathbf{z}_1 = \sigma(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1)$$

$$\mathbf{W}_1 \in \mathbb{R}^{k_1 \times M}, \quad \mathbf{b}_1 \in \mathbb{R}^{k_1}$$

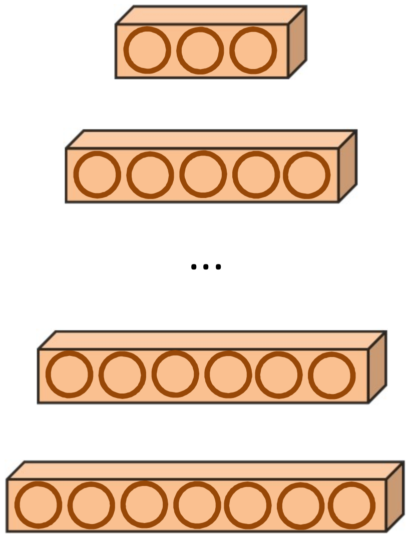
$$\mathbf{x} \in \mathbb{R}^M$$



Neural Network Mapping to Low-Dimensional Space

- Let‘s start by defining a neural network that implements a mapping from the original space $\mathbb{R}^M$ to a space of lower dimension $\mathbb{R}^K$
- The lower-dimensional space into which instances are projected is also called the **embedding** defined by the model

Unlike in classification networks, there is no softmax at the final layer, because embedding is not a distribution



$$\mathbf{z}_D = \sigma(\mathbf{W}_D \mathbf{z}_{D-1} + \mathbf{b}_D) \in \mathbb{R}^K$$

$$\mathbf{z}_{D-1} = \sigma(\mathbf{W}_{D-1} \mathbf{z}_{D-2} + \mathbf{b}_{D-1}) \in \mathbb{R}^{M_{D-1}}$$

...

$$\mathbf{z}_1 = \sigma(\mathbf{W}_1 \mathbf{z}_0 + \mathbf{b}_1) \in \mathbb{R}^{M_1}$$

$$\mathbf{z}_0 = \mathbf{x} \in \mathbb{R}^M \quad \text{(Input)}$$

Dimensionality of feature representation is typically reduced through the layers:

$$K \le M_{D-1} \le \ldots \le M_1 \le M$$

$$\mathbf{W}_1 \in \mathbb{R}^{M_1 \times M}, \mathbf{W}_2 \in \mathbb{R}^{M_2 \times M_1}, \ldots, \mathbf{W}_D \in \mathbb{R}^{K \times M_{D-1}} \qquad \mathbf{b}_1 \in \mathbb{R}^{M_1}, \ldots, \mathbf{b}_D \in \mathbb{R}^{M_{L-1}}$$

How to Learn Embedding in Neural Network?

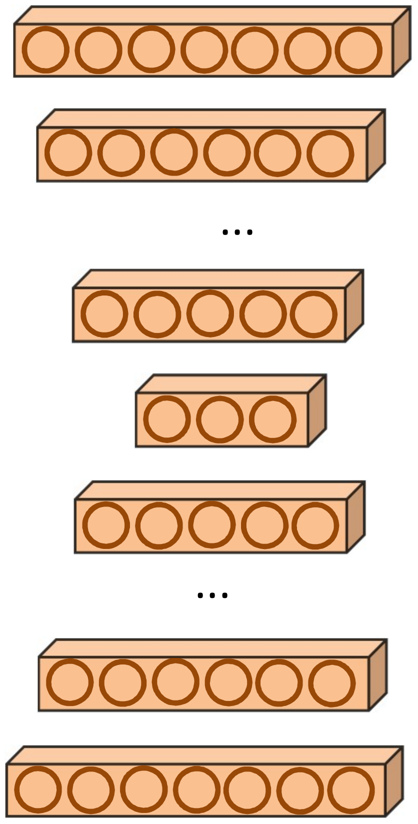
- The network from last slide defines a **nonlinear** mapping from the original input space $\mathbb{R}^M$ into a lower-dimensional space $\mathbb{R}^K$
- The mapping is parameterized by the model parameters $\boldsymbol{\theta}$ (all weight matrices and biases)
- Given a data set $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ with $\mathbf{x}_n \in \mathbb{R}^M$, how do we choose a „good“ mapping?

- **Assumption**: the data implicitly have only a few degrees of freedom (that is, lie on a low-dimensional manifold)
  – We would like a mapping that projects instances to a lower-dimensional space that captures the real degrees of freedom in instances well
  – If the mapping has this property, it will not loose much information about an instance: the original instance can be (almost) fully recovered from low-dimensional image

- We will extend the model by an inverse mapping from the low-dimensional embedding space back to the original space, and then train it to minimize reconstruction error

Mapping to Low-Dimensional Space and Back

- Let‘s extend the model defined above to include a mapping back from the low-dimensional embedding space to the original space
- Overall, this model implements a function $f_{\boldsymbol{\theta}} : \mathbb{R}^M \to \mathbb{R}^M$



$$f_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbf{z}_{2D} = \sigma(\mathbf{W}_{2D}\mathbf{z}_{2D-1} + \mathbf{b}_{2D}) \in \mathbb{R}^{M}$$

$$\mathbf{z}_{2D-1} = \sigma(\mathbf{W}_{2D-1}\mathbf{z}_{2D-2} + \mathbf{b}_{2D-1}) \in \mathbb{R}^{M_1}$$

...

$$\mathbf{z}_{D+1} = \sigma(\mathbf{W}_{D+1}\mathbf{z}_{D} + \mathbf{b}_{D+1}) \in \mathbb{R}^{M_{D-1}}$$

$$\mathbf{z}_{D} = \sigma(\mathbf{W}_{D}\mathbf{z}_{D-1} + \mathbf{b}_{D}) \in \mathbb{R}^{K}$$

$$\mathbf{z}_{D-1} = \sigma(\mathbf{W}_{D-1}\mathbf{z}_{D-2} + \mathbf{b}_{D-1}) \in \mathbb{R}^{M_{D-1}}$$

...

$$\mathbf{z}_{1} = \sigma(\mathbf{W}_{1}\mathbf{z}_{0} + \mathbf{b}_{1}) \in \mathbb{R}^{M_1}$$

$$\mathbf{z}_{0} = \mathbf{x} \in \mathbb{R}^{M} \quad \text{(Input)}$$

Dimensionality of feature representation is again increased to original dimension: typically $M \geq M_1 \geq \ldots \geq M_{D-1} \geq K$

Dimensionality of feature representation is reduced through the layers: typically $K \leq M_{D-1} \leq \ldots \leq M_1 \leq M$

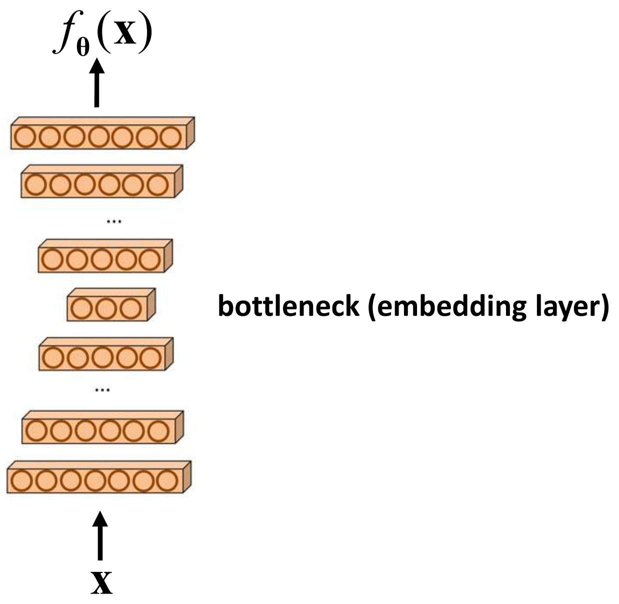
Autoencoder

- We can now train this model by minimizing the reconstruction loss that arises when pushing an instance through the model:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} \sum_{n=1}^{N} \| \mathbf{x}_n - f_{\boldsymbol{\theta}}(\mathbf{x}_n) \|^2$$

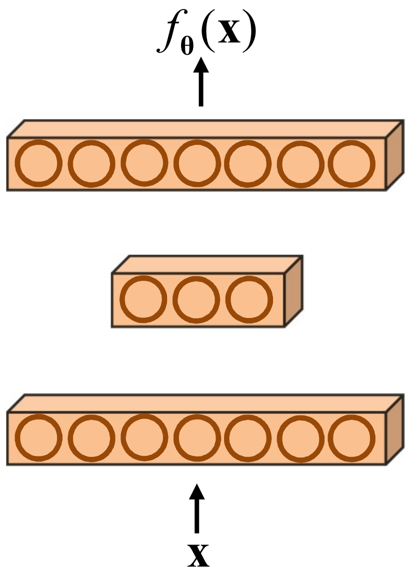
- This means the model is trying to only learn the **identity function**, however, this is nontrivial due to the „bottleneck“ of dimension *K* in the central embedding layer

This model, together with the reconstruction loss function, is known as an **autoencoder**


$f_{\boldsymbol{\theta}}(\mathbf{x})$
...
bottleneck (embedding layer)
...
$\mathbf{x}$

Example: The Linear Autoencoder

- As an example, let‘s look at the most simple case of a linear autoencoder (using no bias, which is ok if we assume that the data is normalized to empirical mean zero)



$\mathbf{z}_2 = \mathbf{W}_2\mathbf{z}_1 \in \mathbb{R}^M$ (Note: no activation function)

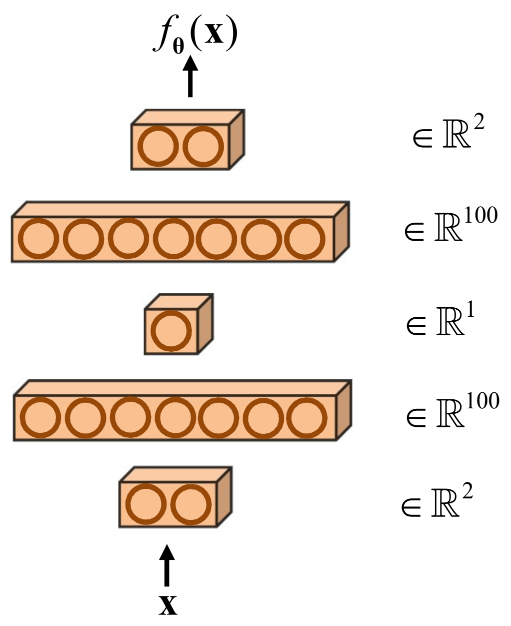
$\mathbf{z}_1 = \mathbf{W}_1\mathbf{z}_0 \in \mathbb{R}^K$ (Note: no activation function)

$\mathbf{z}_0 = \mathbf{x} \in \mathbb{R}^M$ (Input)

- The solution to $\mathbf{W}_1^*, \mathbf{W}_2^* = \arg\min_{\mathbf{W}_1,\mathbf{W}_2} \sum_{n=1}^{N} \| \mathbf{x}_n - f_{\boldsymbol{\theta}}(\mathbf{x}_n) \|^2$ is essentially the same as we get from PCA: it will hold that the subspace $\{\mathbf{W}_1\mathbf{x} \mid \mathbf{x} \in \mathbb{R}^M\}$ is the same subspace as we would get from a PCA on the data $\mathbf{X} = \{\mathbf{x}_1,\dots,\mathbf{x}_N\}$, because this is the subspace that allows reconstruction (with a linear function) with minimal reconstruction loss

Visualization: Autoencoder with 1D Bottleneck

- As another example, consider a nonlinear autoencoder $f_{\boldsymbol{\theta}} : \mathbb{R}^2 \to \mathbb{R}^2$ with a one-dimensional bottleneck layer but with 100-dimensional layers in between
- Note that in this model, we first extend and then reduce the dimensionality so we get a sufficiently expressive nonlinear model
- This model learns a nonlinear subspace in $\mathbb{R}^2$ with intrinsic dimensionality one



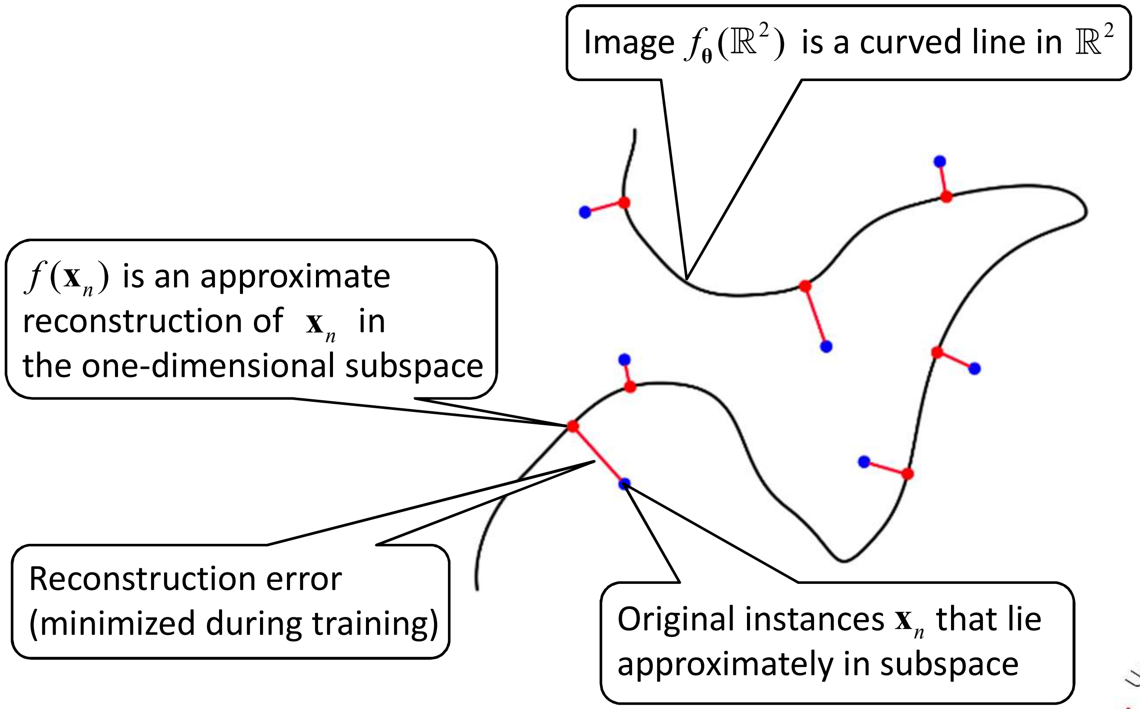


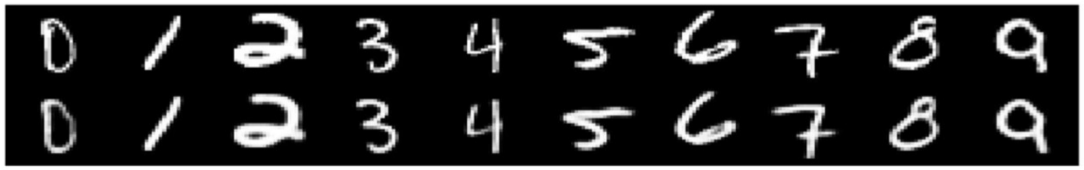
Image $f_{\boldsymbol{\theta}}(\mathbb{R}^2)$ is a curved line in $\mathbb{R}^2$
$f(\mathbf{x}_n)$ is an approximate reconstruction of $\mathbf{x}_n$ in the one-dimensional subspace
Reconstruction error (minimized during training)
Original instances $\mathbf{x}_n$ that lie approximately in subspace

Figure: Roger Grosse, CSC 321 Lec 20

Example: Lower Reconstruction Error of Autoencoder versus PCA

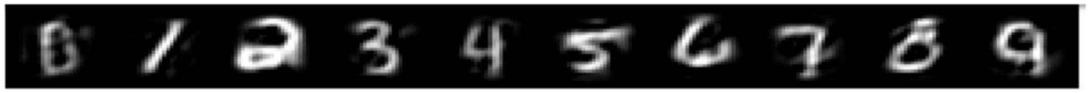
- The non-linear subspace learnt by an autoencoder can have much lower reconstruction error than the linear subspace learned by PCA if the data lies on a nonlinear manifold (subspace)

- For example, for images of handwritten digits:



Original images

Reconstruction from autoencoder with K=30 dimensional subspace



Reconstruction from PCA with K=30 dimensional subspace

Figure: Roger Grosse, CSC 321 Lec 20

Training Autoencoders

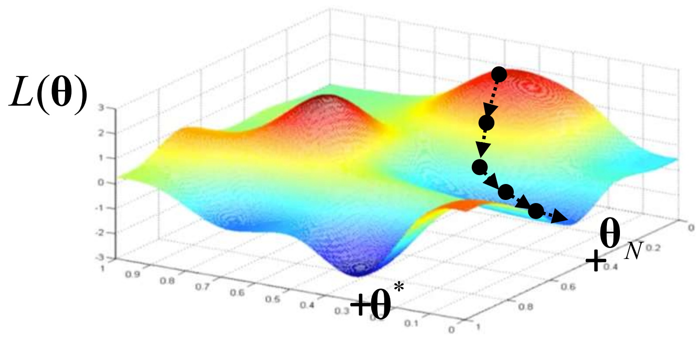
- Autoencoders are trained in the same way as any other neural network
- Given data $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, the loss function to be minimized is

$$L(\boldsymbol{\theta}) = \sum_{n=1}^{N} \| \mathbf{x}_n - f_{\boldsymbol{\theta}}(\mathbf{x}_n) \|^2$$

- Solution (local optimum) is obtained with minibatch stochastic gradient descent

Stochastic gradient descent algorithm

```
1. $\boldsymbol{\theta}_0 = \text{randomInitialization}()$
2. for $i = 0, \dots, i_{max}$:
3.     Select random minibatch $\mathcal{B} \subset \{1, \dots, N\}$
4.     $\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - \eta \nabla L_{\mathcal{B}}(\boldsymbol{\theta}_i)$
```



- Gradient is obtained using automatic differentiation (backpropagation)

Summary: Dimensionality Reduction

- Dimensionality reduction methods can discover structure in data in the sense that they estimate a subspace of lower dimensionality within the original input space in which the data approximate lies

- Principal component analysis does this in a linear setting
  - Objective can be to maximize the variance in the subspace or minimize the reconstruction error, both leading to the same solution
  - Solution is found by finding the principle eigenvectors of the empirical covariance matrix

- Autoencoders perform non-linear dimensionality reduction
  - Define a neural network that first maps instances to a low-dimensional „bottleneck“ layer and then maps them back to the input space
  - Train the model to learn the identity function

Further Reading

- Further reading: Bishop 2006, Chapter 12

- C. Bishop. Pattern Recognition and Machine Learning. Springer, 2006.