# IntroML - Lecture Notes Week 9

Ruben Schenk, ruben.schenk@inf.ethz.ch

July 17, 2022

## 1 Unsupervised Learning: Dimension Reduction

### 1.1 Introduction

The basic challenge is posed as follows: Given a data set $D = \{x_1, ..., x_n\}$ with $x_i \in \mathbb{R}^d$, obtain an **embedding** (low-dimensional representation) $z_1, ..., z_n \in \mathbb{R}^k$ where typically $k << d$.

One might want to do this for several reasons:

- Visualization ($k = 1, 2, 3$)
- Regularization (model selection)
- Unsupervised feature discovery (i.e. determining features from data)
- etc.

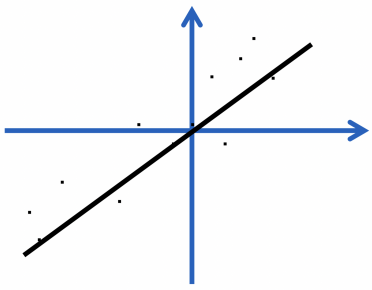
**Note:** Our focus is on model-based approaches, i.e.:

- Given: Data $D = \{x_1, ..., x_n\} \subseteq \mathbb{R}^d$
- Goal: Obtain a mapping $f : \mathbb{R}^d \to \mathbb{R}^k$ where usually $k << d$
- We can distinguish:
  - Linear dimension reduction $f(x) = Ax$
  - Nonlinear dimension reduction (parametric or non-parametric)

### 1.2 Dimension Reduction

**Linear dimension redcution** can be seen as compression. The motivation behind this process is that low-dimensional representation should allow to compress the original data and allow for a accurate reconstruction.

Let us consider a simple example for $k = 1$. Given is a data set $D = \{x_1, ..., x_n\} \subseteq \mathbb{R}^d$, assumed to be centered, i.e. $\mu = \frac{1}{n}\sum_i x_i = 0$. We want to represent the data as points on a line $x_i \approx z_i w$ with coefficients $w \in \mathbb{R}^d$.



In other words, we want $x_i \approx z_i w$ minimizing $||z_i w - x_i||_2^2$. To ensure the uniqueness of the solution, we normalize $w$, i.e. $||w||_2 = 1$. We want to optimize jointly over $w$, $z_1$, $z_2$, ...:

$$(w^*, z^*) = \arg \min_{||w||_2=1,\, z} \sum_{i=1}^{n} ||z_i w - x_i||_2^2.$$

In our $k = 1$ case, the optimal $z$ is given by:

$$z_i^* = w^T x_i$$

Thus, we effectively solve a *regression* problem, interpreting $x$ as features and $z$ as labels. Since for any fixed $||w||_2 = 1$, it holds that $z_i^* = w^T x_i$. Therefore, we only need:

$$w^* = \arg \min_{||w||_2=1} \sum_{i=1}^{n} ||w w^T x_i - x_i||_2^2,$$

which is equivalent to

$$w^* = \arg \min_{||w||_2=1} \sum_{i=1}^{n} (w^T x_i)^2,$$

which is furthermore equivalent to

$$w^* = \arg \max_{||w||_2=1} w^T \Sigma w,$$

where $\Sigma = \frac{1}{n} \sum_{i=1}^{n} x_i x_i^T$ is the *empirical covariance,* assuming the data is centered (i.e. $\mu = \frac{1}{n} \sum_i x_i = 0$). Finally, the optimal solution to $w^* = \arg\max_{||w||_2=1} w^T \Sigma w$ is given by the **principal eigenvector** of $\Sigma$, i.e. $w = v_1$ where, for $\lambda_1 \geq \cdots \geq \lambda_d \geq 0$,

$$\Sigma = \sum_{i=1}^{d} \lambda_i v_i v_i^T.$$

But what if $k > 1$? Suppose we wish to project more than one dimension. Thus we want:

$$(W, z_1, ..., z_n) = \arg \min_{W^T W = I_k,\, z} \sum_{i=1}^{n} ||W z_i - x_i||_2^2,$$

where $W \in \mathbb{R}^{d \times k}$ is *orthogonal,* and $z_1, ..., z_n \in \mathbb{R}^k$. This is called the *principal component analysis* problem and its solution can be obtained in closed form even for $k > 1$.

## 1.3 Principle Component Analysis (PCA)

The **Principal Component Analysis (PCA)** problem is as follows:

Given centered data $D = \{x_1, ..., x_n\} \subseteq \mathbb{R}^d$ with $\mu = \frac{1}{n} \sum_i x_i = 0$ and $\Sigma = \frac{1}{n} \sum_{i=1}^{n} x_i x_i^T$, the solution to the PCA problem

$$(W, z_1, ..., z_n) = \arg \min_{W^T W = I_k,\, z} \sum_{i=1}^{n} ||W z_i - x_i||_2^2,$$

where $1 \leq k \leq d$, $W \in \mathbb{R}^{d \times k}$ is orthogonal and $z_1, ..., z_n \in \mathbb{R}^k$, is given by

$$W = (v_1 \mid \cdots \mid v_k) \text{ and } z_i = W^T x_i.$$

Hereby: $\Sigma = \sum_{i=1}^{d} \lambda_i v_i v_i^T$ and $\lambda_1 \geq \cdots \geq \lambda_d$.

The linear mapping $f(x) = W^T x$ obtained from PCA *projects* vectors $x \in \mathbb{R}^d$ into a $k$-dimensional subspace. This projection is chosen to *minimize the reconstruction error* (measured in the Euclidean norm).

One might remember that we can obtain PCA through the *Singular-Value Decomposition (SVD).* We recall that any $X \in \mathbb{R}^{n \times d}$ can be represented as

$$X = USV^T,$$

where $U \in \mathbb{R}^{n \times n}$ and $V \in \mathbb{R}^{d \times d}$ are orthogonal, and $S \in \mathbb{R}^{n \times d}$ is diagonal (w.l.o.g. in decreasing order). Its entries are called *singular values.* The top $k$ principal components are exactly the first $k$ columns of $V$:

$$n\Sigma = X^T X = VS^T U^T USV^T = VS^T SV^T = VDV^T.$$

Finally, we can compare PCA and k-means:

**PCA-Problem:**

$$(W, z_1, ..., z_n) = \arg \min_{W^T W = I_k,\, z} \sum_{i=1}^{n} ||Wz_i - x_i||_2^2,$$

where $W \in \mathbb{R}^{d \times k}$ is *orthogonal,* and $z_1, ..., z_n \in \mathbb{R}^k$.

**k-means problem (equivalent formulation):**

$$(W, z_1, ..., z_n) = \arg \min_{W,\, z} \sum_{i=1}^{n} ||Wz_i - x_i||_2^2,$$

where $W \in \mathbb{R}^{d \times k}$ is *arbitrary,* and $z_1, ..., z_n \in E_k$ for $E_k = \{e_1, ..., e_k\}$ are all unit vectors.

In summary:

- We can think of PCA and k-means as options to solve a similar unsupervised learning problem, with different constraints.
- Both aim to compress the data with maximum fidelity under constraints on the model complexity.
- This insight gives rise to a much broader class of techniques.

## 1.4 Kernel PCA

Recall that in supervised learning, kernels allowed us to solve non-linear problems by reducing them to linear ones in high-dimensional (implicitly represented) spaces. We can take the same approach for unsupervised learning!

Recall that the optimal solution to PCA problem solves for

$$w^* = \arg \max_{||w||_2 = 1} w^T X^T X w = \arg \max_{||w||_2} \sum_{i=1}^{n} (w^T x_i)^2.$$

Applying feature maps, using $w = \sum_{j=1}^{n} \alpha_j \phi(x_j)$ and observing that $||w||_2^2 = \alpha^T K \alpha$:

$$\begin{aligned}
\arg \max_{||w||_2 = 1} \sum_{i=1}^{n} &= \arg \max_{\alpha^T K \alpha = 1} \sum_{i=1}^{n} \Big( \sum_{j=1}^{n} \alpha_j \phi(x_j)^T \phi(x_i) \Big)^2 \\
&= \arg \max_{\alpha^T K \alpha = 1} \sum_{i=1}^{n} \Big( \sum_{j=1}^{n} \alpha_j k(x_j, x_i) \Big)^2 \\
&= \arg \max_{\alpha^T K \alpha = 1} \sum_{i=1}^{n} (\alpha^T K_i)^2 \\
&= \arg \max_{\alpha^T K \alpha = 1} \alpha^T K^T K \alpha
\end{aligned}$$

The **Kernel PCA** ($k = 1$) requires solving:

$$\alpha^* = \arg \max_{\alpha^T k \alpha = 1} \alpha^T K^T K \alpha \equiv \arg \max_{\alpha} \frac{\alpha^T K^T K \alpha}{\alpha^T K \alpha}.$$

The optimal solution is obtained in *closed form* from the eigendecomposition of $K$:

$$\alpha^* = \frac{1}{\sqrt{\lambda_1}} v_1 \text{ for } K = \sum_{i=1}^{n} \lambda_i v_i v_i^T \text{ and } \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0.$$

For general $k \geq 1$, the **Kernel Principal Components** are given by $\alpha^{(1)}, ..., \alpha^{(k)} \in \mathbb{R}^n$, where

$$\alpha^{(i)} = \frac{1}{\sqrt{\lambda_i}} v_i$$

is obtained from $K = \sum_{i=1}^{n} \lambda_i v_i v_i^T$ and $\lambda_i \geq \lambda_2 \geq \cdots \geq \lambda_n$. Given this, a new point $x$ is projected as $z \in \mathbb{R}^k$, where

$$z_i = \sum_{j=1}^{n} \alpha_j^{(i)} k(x_j, x).$$

**Note:** Applying k-means or kernel-principal components is soemtimes called *Kernel-k-means* or *Spectral Clustering*.

Some notes on Kernel-PCA:

1. Kernel-PCA corresponds to applying PCA in the feature space induced by the kernel $k$
2. This can be used to discover non-linear feature maps in closed form
3. This can be used as inputs, e.g. to SVMs given "multi-layer support vector machines"
4. We may want to center the kernel, i.e. $K' = K - KE - EK + EKE$ where $E = \frac{1}{n}[1, ..., 1][1, ..., 1]^T$

## 1.5 Neural Network Autoencoders

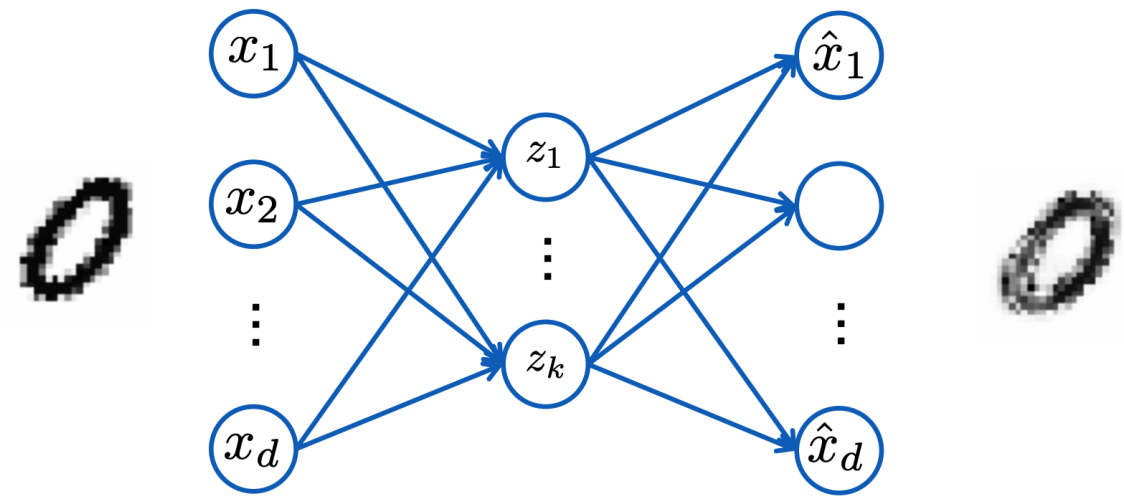
The problem with Kernel-PCA is that it requires data specified as a kernel:

- Complexity grows with the number of data points
- Cannot easily "explicitly" embed high-dimensional data (unless we have an aapropriate kernel)

The key idea of **dimension reduction via Autoencoders** is to try and learn the idenity function $x \approx f(x; \theta)$. But what function $f$ should we pick?

$$f(x; \theta) = f_{dec}(f_{enc}(x; \theta_{enc}); \theta_{dec}),$$

where *enc* stands for "encoder" and *dec* for "decoder". The idea is that the encoder maps the data to the lower-dimensional space and the decoder maps the data back to the original dimension:



**Neural network Autoencoders** are ANNs where:

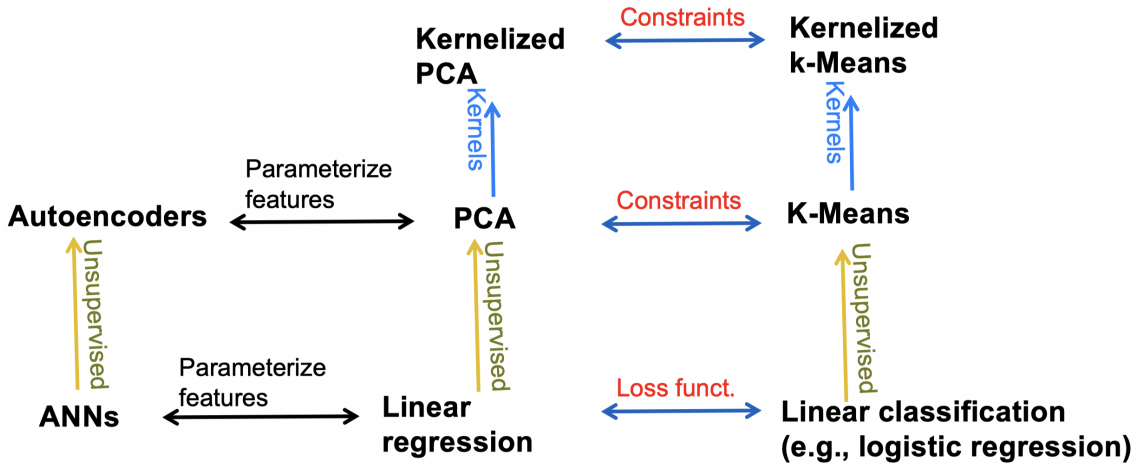
- There is one output unit for each of the $d$ input units
- The number $k$ of hidden units is usually smaller than the number of inputs (the *bottleneck*)

The goal is to optimize the weights such that the *output agrees with the input,* in other words minimizing the squared reconstruction error:

$$\min_W \sum_{i=1}^{n} ||x_i - f(x_i; W)||_2^2.$$

As seen in previous chapters, we can find a local minimum via SGD (backpropagation). If we compare Autoencoders to PCA, we see that the internal representation $z = \Phi(W^{(enc)}x)$ is the "dimensionality reduced" input $x$. If the activation function is the identity, then in fact fitting a NN autoencoder is equivalent to PCA!

## 1.6 Supervised vs. Unsupervised Learning




| | | | |
|---|---|---|---|
| Model / function class: | Linear hypotheses; nonlinear hypotheses with nonlinear feature transforms, kernels, learn nonlinear features via neural nets | | |
| Objective: | Loss-function | + | Regularization/penalty |
| | Squared loss, 0/1 loss, logistic loss, cross-entropy loss, Hinge loss, cost sensitive losses, reconstruction error | | $L^2$ norm, $L^1$ norm, $L^0$ penalty, early stopping, dropout |
| Method: | Exact solution (eigendecomposition for PCA), Gradient Descent, (mini-batch) SGD, Reductions, Lloyd's heuristic | | |
| Evaluation metric: | Mean squared (reconstruction) error, Accuracy, F1 score, AUC, Confusion matrices | | |
| Model selection: | K-fold Cross-Validation, Monte Carlo CV | | |