COMPUTATIONAL INTELLIGENCE

Data Preprocessing and Autoencoders for Feature Extraction





Adrian Horzyk

horzyk@agh.edu.pl

AGH

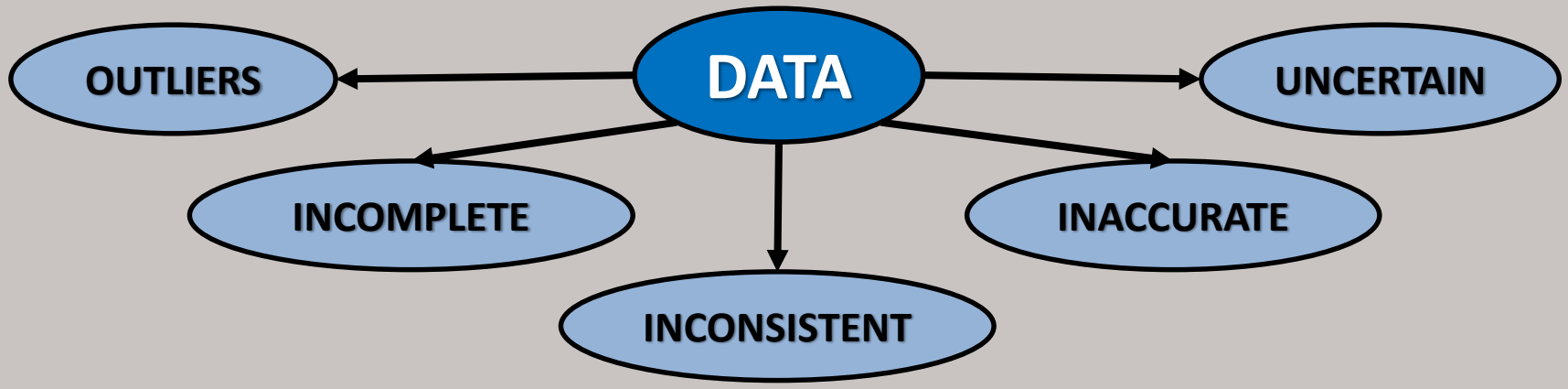
AGH University of Science and Technology
Krakow, Poland



PROBLEM OF DATA QUALITY

Data may be incomplete, uncertain, inaccurate, outliers or inconsistent:

- **Uncertain data - data whose accuracy is uncertain and difficult to verify.**
- **Incomplete data - data that has no value for at least one attribute or element of a sequence or other structure.**
- **Inaccurate data - data with limited precision or expressed in a symbolic or fuzzy way.**
- **Inconsistent data - data that assigns one object more than one value for at least one attribute, i.e. different values are associated with the same objects.**
- **Data outliers - data that are significantly different from others, which may indicate that they are incorrect or exceptional.**



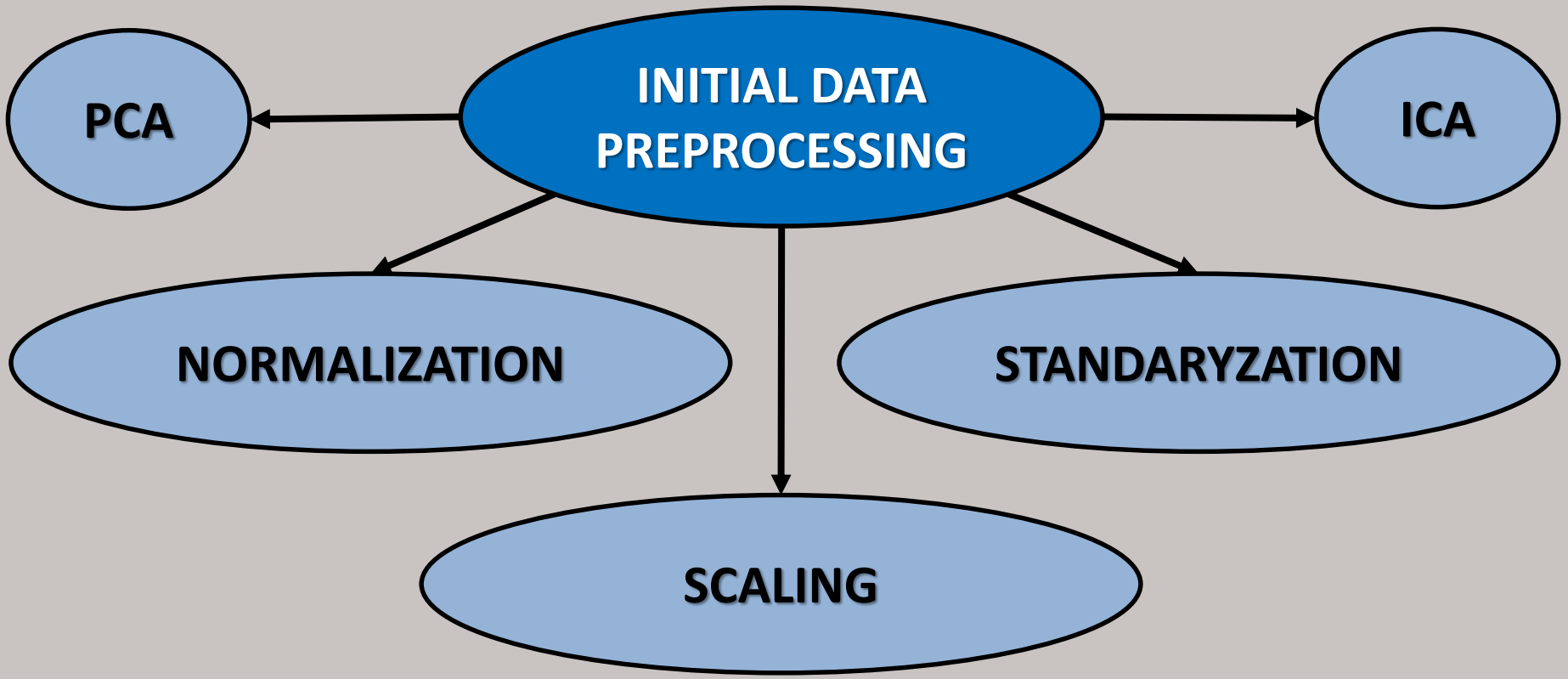
This causes various difficulties in their processing in accordance with the slogan:

"Garbage at the entrance - garbage at the exit."



INITIAL DATA PREPROCESSING

Initial data preprocessing are various operations on data that transform, scale, normalize, or standardize to simplify them, make them easier discriminable and the task faster solvable:





DATA STANDARIZATION

Standardization is an operation commonly used in statistics, which consists in rescaling data of each element of the set against the mean values and standard deviation in accordance with the formula:

$$y_i = \frac{x_i - m}{\sigma}$$

$x = [x_1, x_2, \ldots, x_N]$ – is the N-element vector of the source data,

$y = [y_1, y_2, \ldots, y_N]$ – is the N-element data vector after standardization,

m – is the average value determined from these data,

σ – is the standard deviation.

As a result of standardization, we get a vector of features which average value is zero, while the standard deviation is equal to one.

It should not be used for data about standard deviation close to zero!



DATA NORMALIZATION

Normalization is the data scaling with respect to extreme values (min and max) of a given data vector, usually to the range [0, 1] (sometimes to [-1, 1]) according to the following formula:

$$y_i = \frac{x_i - x_{min}}{x_{max} - x_{min}}$$

$x = [x_1, x_2, \ldots, x_N]$ – is the N-element vector of the source data,

$y = [y_1, y_2, \ldots, y_N]$ – is the N-element data vector after normalization.

Normalization is sensitive to outliers and large scatter because then the right data will be squeezed in a narrow range, which can significantly hamper their discrimination!

Normalization is sometimes necessary to use a method that requires input or output data to fall within a certain range, e.g. using sigmoidal functions or hyperbolic tangent.



PROBLEM OF DATA OUTLIERS

- Outliers are data that do not match the data model represented by other data.
- Outliers often fall out outside the range of variation of other data for one or more attributes.
- Sometimes outliers are unusual combinations of common data, which are within the limits of the variation of individual attributes, but this variation is so strange that it is not compatible with the other combinations, e.g. for classification problems.
- Outliers may arise as a result of errors, anomalies (e.g. in measurement), or specific (sometimes interesting) phenomena.
- There is no strict mathematical definition of outliers, as they usually depend on the nature of the data and the subjective assessment.
- Outliers are usually removed or replaced with zero, average, median, ...
- The median is quite robust to data outliers, but the average is not.
- It uses a Vinsor's average in which selected extreme observations are replaced by the minimum and maximum values from the remaining data, respectively.



CORRELATIONS AND COVARIANCES

Pearson correlation - is calculated as the ratio of the covariance of x and y vectors to the product of standard deviations:

$$p_{xy} = \frac{cov(x, y)}{std(x) \cdot std(y)}$$

Spearman's rank correlation also uses the rank vector of the original set of observations x or y :

$$ps_{xy} = \frac{cov(r(x), r(y))}{std(r(x)) \cdot std(r(y))}$$

Example:

If the x vector consists of the following values:

$$x_1 = 2.2; x_2 = 1.3; x_3 = 1.7; x_4 = 2.2; x_5 = 4.2; x_6 = 3.8$$

As a result of sorting, we get:

$$x_2 = 1.3; x_3 = 1.7; x_1 = 2.2; x_4 = 2.2; x_6 = 3.8; x_5 = 4.2$$

Assigning to particular observations (given) the rankings resulting from their order:

$$r_2 = 1; r_3 = 2; r_1 = 3.5; r_4 = 3.5; r_6 = 5; r_5 = 6$$

for the same values, the rank value is the average of their order (r_1 and r_4).

So we get the following set of ranks assigned to the data:

$$r_1 = 3.5; r_2 = 1; r_3 = 2; r_4 = 3.5; r_5 = 6; r_6 = 5$$



PROCESSING DATA OF LIMITED QUALITY

Processing of incomplete data:

- **after bypassing incomplete records (objects, tuples),**
- **after removing attributes (columns) introducing incompleteness to records, if the incompleteness is caused by a small number of attributes,**
- **after replacing the missing data with default data, average, median (middle value), fashion (the most common value) for a given attribute,**
- **after replacing the missing data with the most probable values, based on the most similar objects, e.g. using the KNN method,**
- **after building the model for complete data, an attempt is made to assign the missing records to one of the groups/classes based on the model built.**



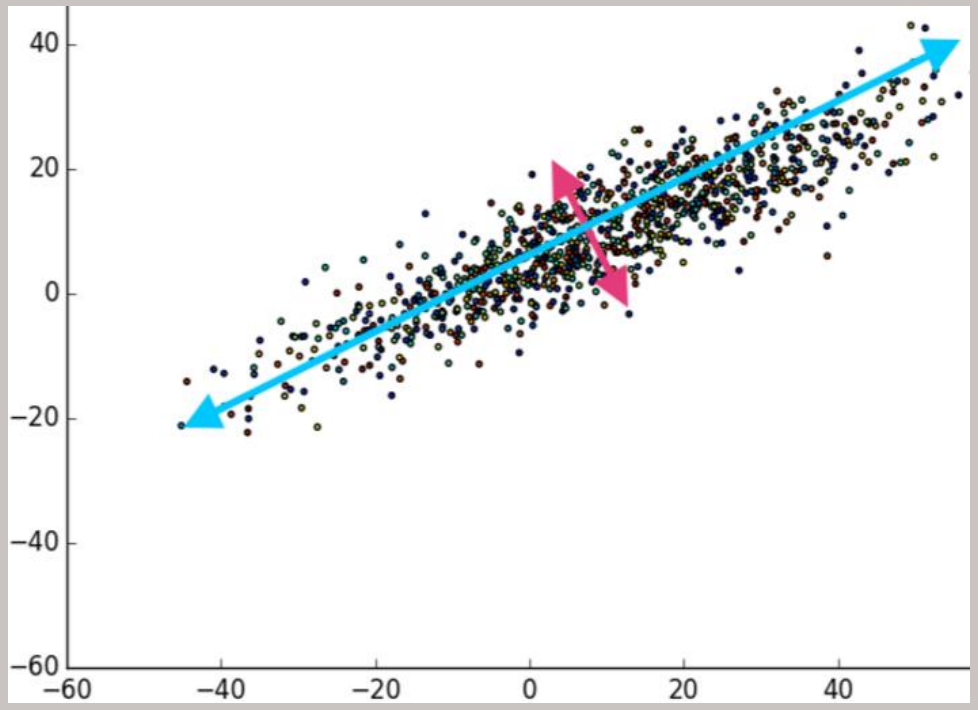
PCA – Principal Component Analysis

PCA is a method of data preprocessing based on such rotation of an orthogonal coordinate system, so as to maximize the variance for subsequent coordinates: 1, 2, and

On the basis of the covariance matrix, we construct a new space of data observation, in which initial factors (first designated coordinates) are the most volatile and characterized by these initial factor.

Higher variance/variability enable classification methods to achieve better discrimination.

In addition, PCA allows simplifying data on these factors/coordinates which are characterized by the least variability.



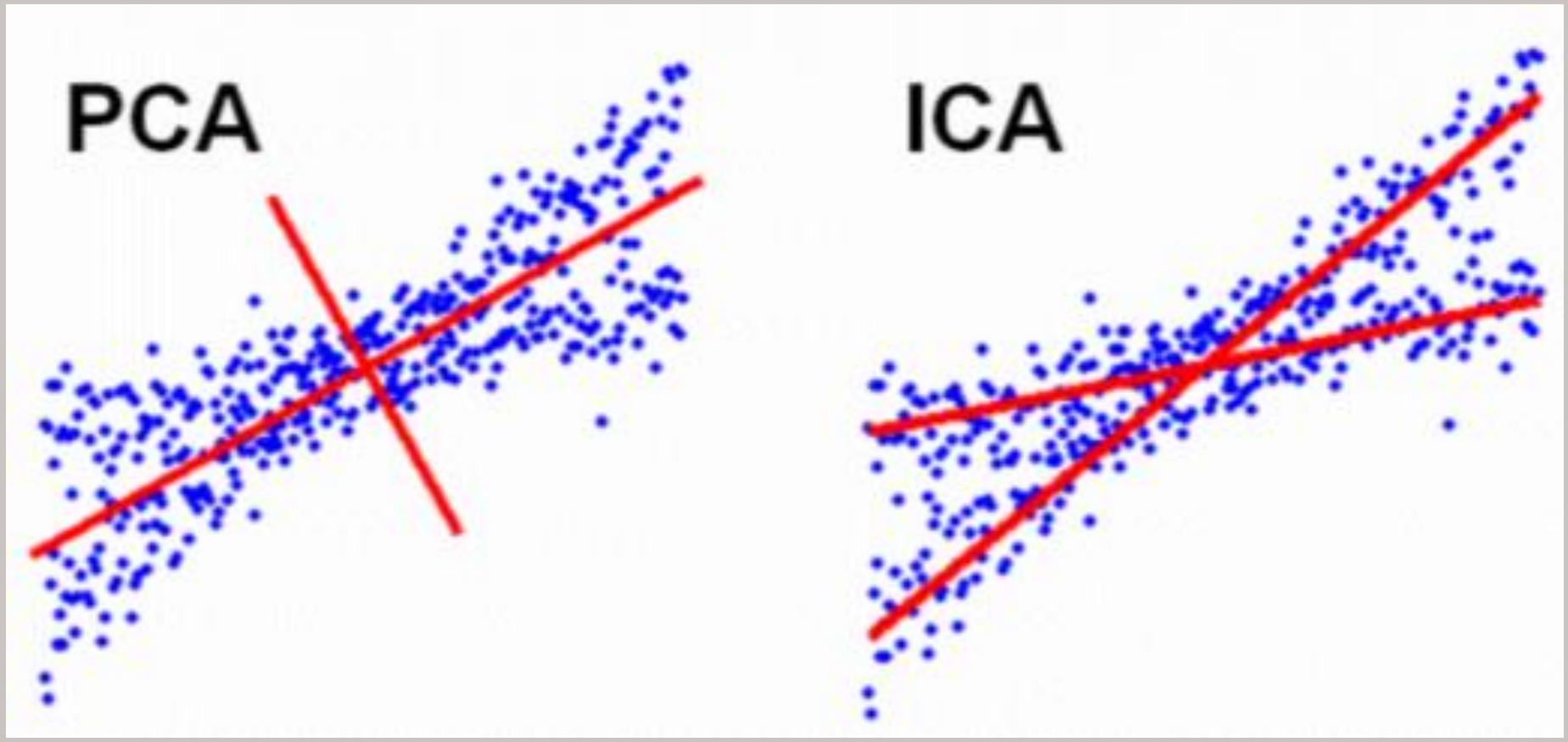


ICA – Independent Component Analysis

ICA is a statistical method similar to PCA, which task is to find independent coordinates describing the highest variability (variance) of data.

ICA also enables the reduction of the data dimension.

It usually gives better results than PCA.





ICA Algorithm

Fast ICA algorithm using the concept of negentropy:

1. Center / Move x data so that their mean is equal to zero:

$$x = x - x_m \qquad x_m = E\{x\}$$

2. Clear x to maximize non-gaussian characteristics (PCA with filtration):

$$z = V \Lambda^{-1/2} V^T x \qquad V \Lambda V^T = E\{x\, x^T\}$$

3. Take a random initial vector w, so as to ||w|| =1.

4. Update in (maximally in the non-Gaussian direction)

$$w = E\{z * g(w^T z)\} - E\{g'(w^T z)\} \text{ in}$$

$$g(y) = \tanh(a_1 y) \text{ OR } g(y) = y*\exp(-y^2/2) \qquad \text{where } 1< a_1 <2$$

$$w = w / \|w\|$$

5. If it is not convergent, go back to point 4.

6. Get an independent coordinate s

7. $s = [\, w_1\ w_2 \ldots w_n \,]\, x$

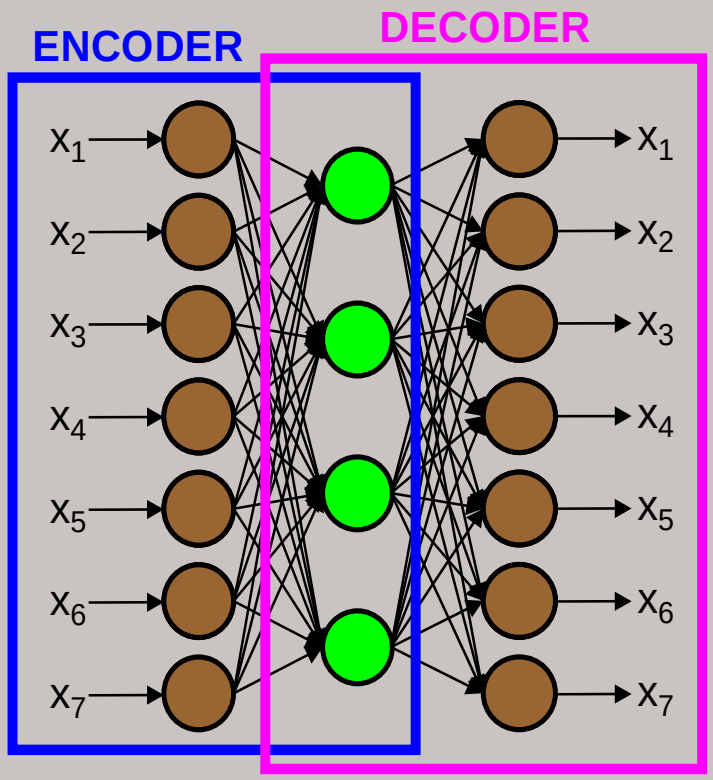


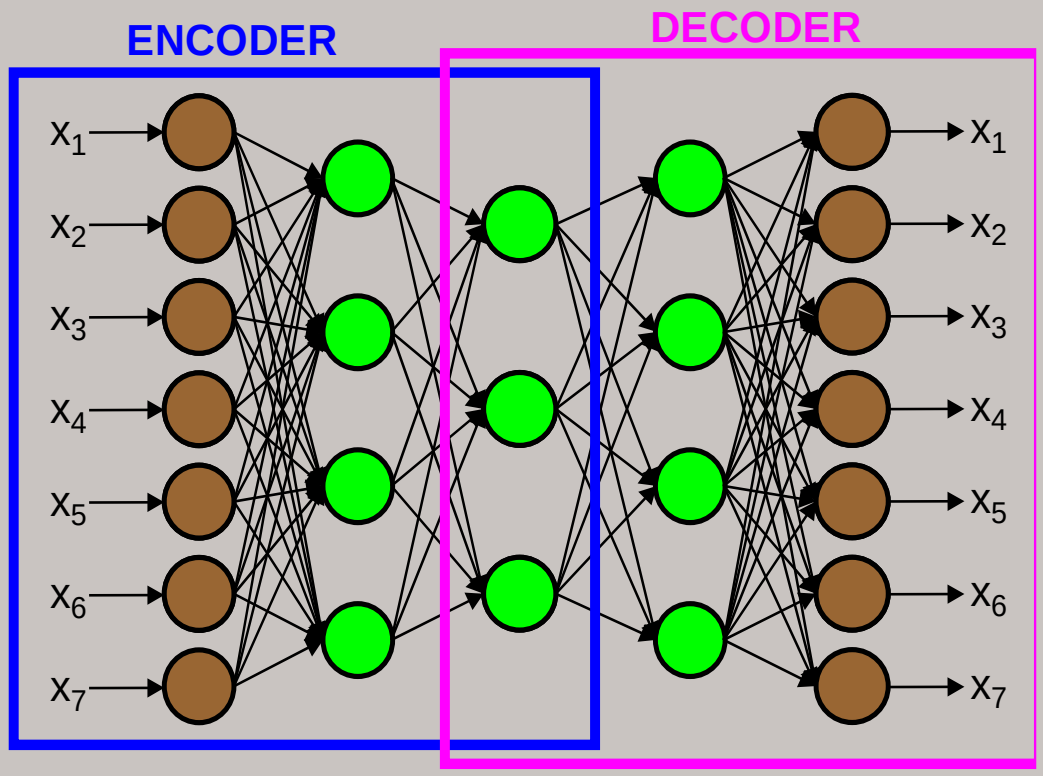
AUTOENCODERS

Autoencoder is a kind of artificial neural networks which is trained to represent a set of training data in an unsupervised manner using a reduced dimensionality and gets the same output data as input ones.

The reduced dimensionality is used to find out frequent combinations which constitute complex data features which can be used in various classifiers.

Autoencoders consist of **encoders** and **decoders**:







TYPES OF AUTOENCODERS

- **Undercomplete Autoencoders** – are defined to represent data in undercomplete way, i.e. the outputs do not reproduce inputs precisely in order to allow for generalization, feature extraction, data distribution, and correction of outliers. Training of such autoencoders aims to minimize the loss function defining the differences between outputs and inputs. When the autoencoders are linear, they work similarly to PCA (Principal Components Analysis), so they can replace such kind of preprocessing algorithms (PCA or ICA).
- **Autoencoders with Regularization** – use the complexity of the modeled distribution of the data to select an adequate dimension and capacity of encoders and decoders. They use a loss function to be resistant to noise and missing data and learn correct data distribution. These autoencoders can be non-linear and overcomplete as well.
- **Sparse Autoencoders** – are autoencoders which are used for other computational tasks, e.g. for classification, where we need to represent frequent features more than find a perfect identity function. In this approach, the representation of rare features is penalized. This leads to a sparse representation of inputs and useful feature extraction as a preparation phase for classification.
- **Anomaly Detection Autoencoders** – are autoencoders which are used to detect rare features that stand for various anomalies in data and can identify outliers.
- **Denoising Autoencoders (DAE)** – try to find a function which returns correct output for noised, corrupted or incomplete inputs. They have to recover the original undistorted inputs on their outputs.



TRAINING OF AUTOENCODERS

Autoencoders are trained in an unsupervised way using the algorithm typically used for supervised learning, e.g. backpropagation. This is because we use the outputs which are the same as the inputs:

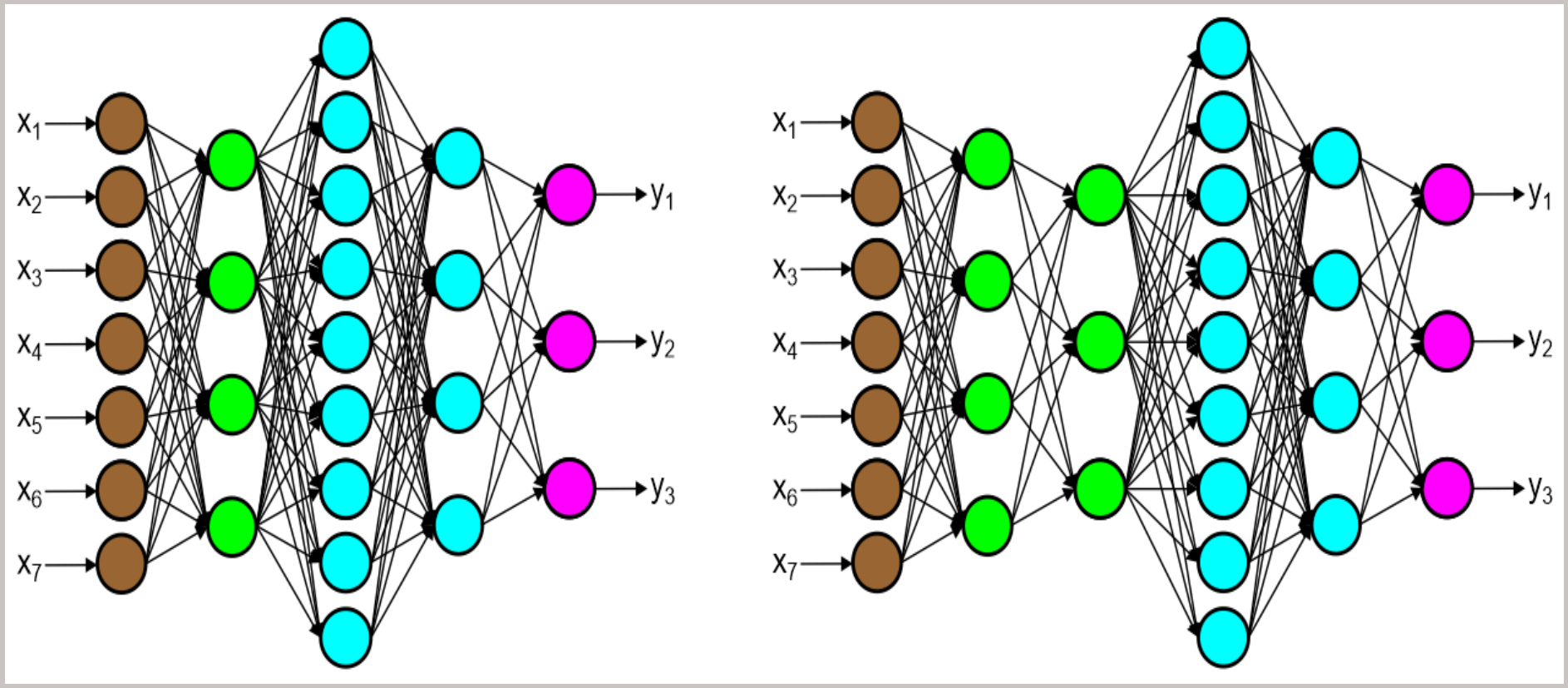
- ✓ **Assume that we have a set of unlabeled training examples $\{x_1, x_2, x_3, \ldots\}$, where $x_i \in R^n$.**
- ✓ **An autoencoder uses outputs defined as $y_i = x_i$ where y_i is an expected output value.**
- ✓ **Autoencoders can learn to extract features similarly as Convolutional Neural Networks (CNN) do.**
- ✓ **The training capabilities of autoencoders are associated with the number of encoding and decoding layers.**
  When autoencoders have more than single encoding and decoding layers, we call them deep autoencoders.
  Deep autoencoders usually have a better compression ratio than flat autoencoders.
- ✓ **Deep autoencoders can be constructed from flat autoencoders trained subsequently and separately.**
- ✓ **Autoencoders are usually trained using the backpropagation algorithm, however, we can also use other algorithms, e.g. the recirculation algorithm.**



DATA PREPROCESSING

Autoencoders are typically used in deep architectures as layers that do not demand supervised training but they are trained in an unsupervised ways in order to find out and represent the most important features which can be used for next classifications:

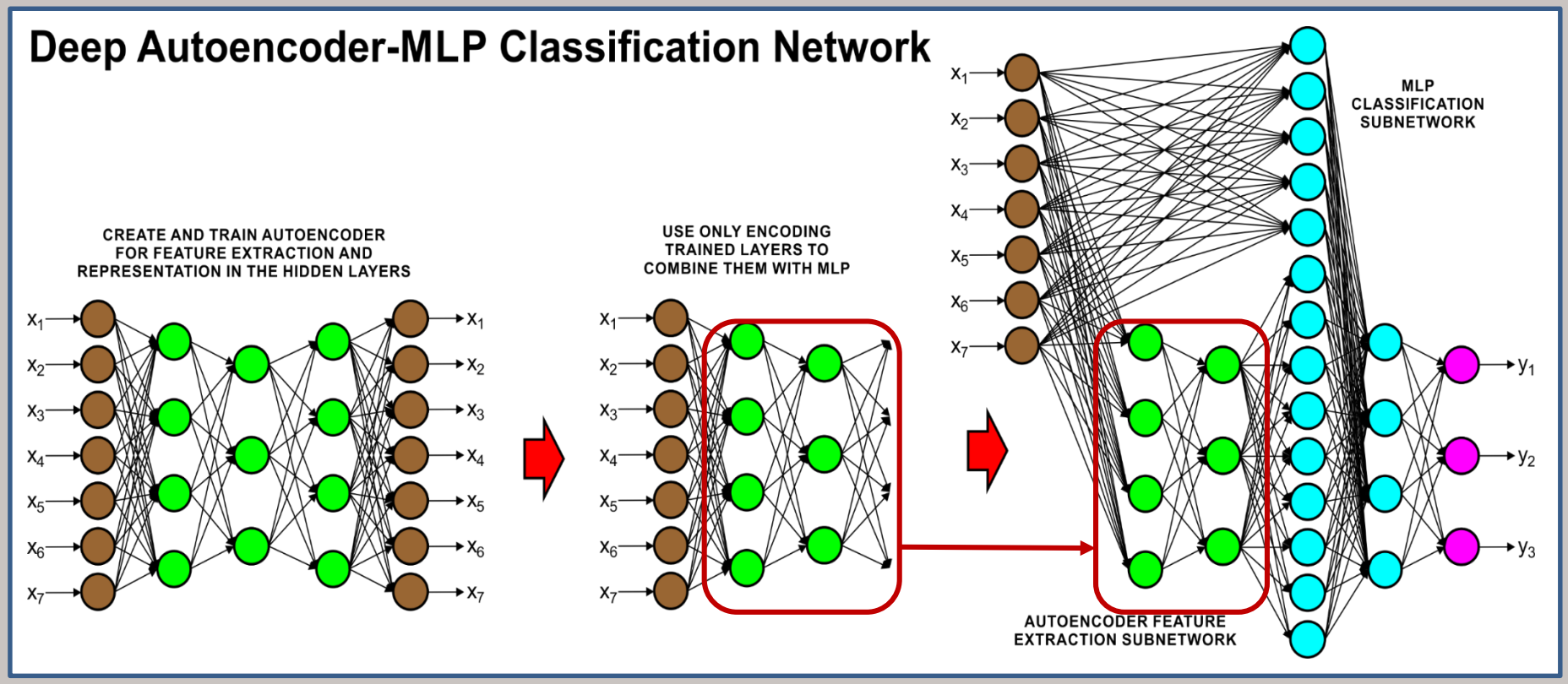
Autoencoders are trained in the unsupervised manner in the first preliminary stage of the whole adaptation process, and then the supervised training of the remaining part of the neural network is proceeded to fine-tune the outputs for e.g. classification:



COMBINING AUTOENCODERS WITH MLP



Sparse Autoencoders are often trained to be combined with other types of artificial neural networks, e.g. MLPs. After training of the autoencoder, we usually freeze the weights of it after combining it with an MLP network, train for some time and unfreeze these weights when the MLP network is already quite well-trained in order not to spoil weights trained by this autoencoder.
This is because they can preprocess raw input data and extract useful features for other networks:



One of our goals during laboratory classes will be to implement such a combination of an autoencoder and a MLP network to find out what benefits we can have from such cooperation of the networks in producing the results.

Bibliography and Literature

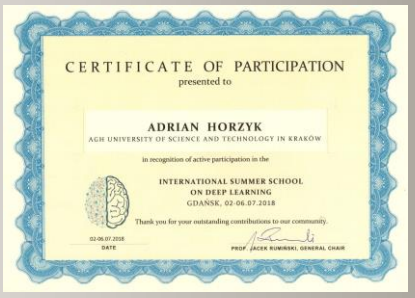
1. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, MIT Press, 2016, ISBN 978-1-59327-741-3 or PWN 2018.
2. Stanford University Tutorial of Unsupervised Learning used to Autoencoders: http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/
3. Shyam M. Guthikonda, Kohonen Self-Organizing Maps, 2005
4. Mat Buckland, http://www.ai-junkie.com/ann/som/som1.html
5. ET-Map – http://ai.eller.arizona.edu/research/dl/etspace.htm
6. Holk Cruse, Neural Networks as Cybernetic Systems, 2nd and revised edition
7. R. Rojas, Neural Networks, Springer-Verlag, Berlin, 1996.
8. Convolutional Neural Network (Stanford)
9. Visualizing and Understanding Convolutional Networks, Zeiler, Fergus, ECCV 2014
10. IBM: https://www.ibm.com/developerworks/library/ba-data-becomes-knowledge-1/index.html



Adrian Horzyk

horzyk@agh.edu.pl

Google: Horzyk



University of Science and Technology in Krakow, Poland