Clustering algorithms and autoencoders for anomaly detection

Alessia Saggio

ESRs seminars - part 1

Université catholique de Louvain,
Belgium

16th May 2017

Outline

- Introduction
- Clustering algorithms
- Autoencoders

What is B12?

- **Consulting company** founded in 2012 in Louvain-la-neuve (BE), focused on problem solving, software development, and data mining

- It is one of the **non-academic partners** of the **AMVA4NP network**

- I've been working there for **three months** (Oct-Dec 2016) as foreseen by the AMVA4NP training program

A few words about consulting

- The main idea behind a consulting is to bring **solutions to real world problems**: the projects are several and very different

- One of the main values of a consulting is **confidentiality: I'm not allowed to mention anything regarding data and results**

  But I can still talk about Clustering techniques and Autoencoders since it looks like they are publicly available…

  http://scikit-learn.org/stable/modules/clustering.html
  http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/

The project at B12

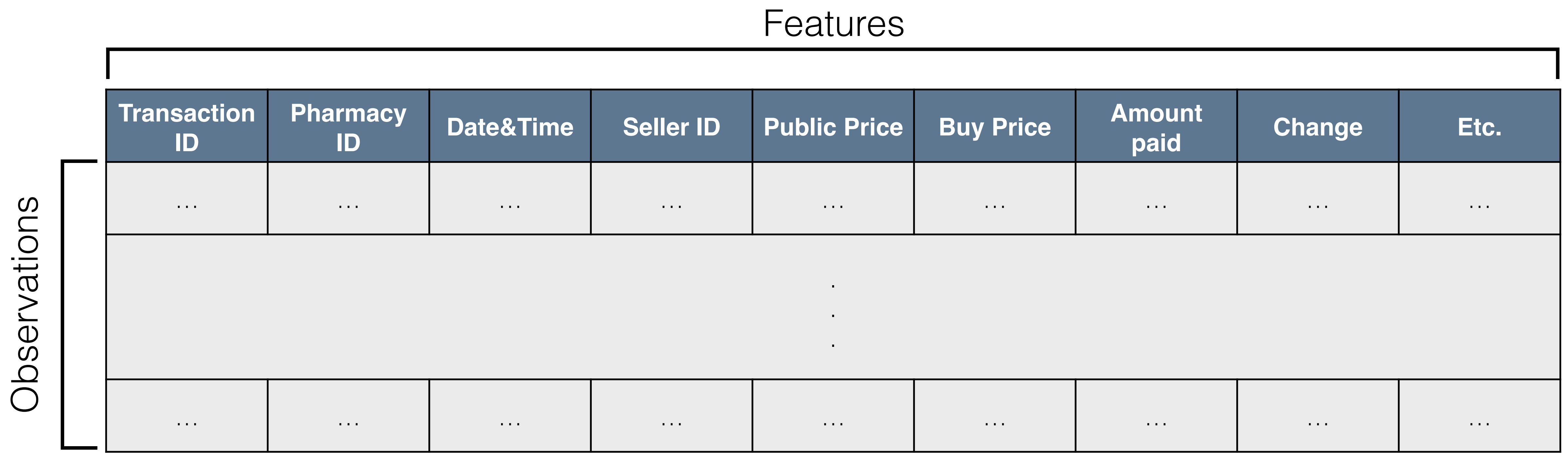
- We were given a huge amount of European pharmacy data (~17 GB)
- The goal was to detect hidden **anomalies** in the data
- Two techniques used: **clustering algorithms** and **machine learning autoencoders**

The project at B12

- We were given a huge amount of European pharmacy data (~17 GB)
- The goal was to detect hidden **anomalies** in the data
- Two techniques used: **clustering algorithms** and **machine learning autoencoders**

Anomaly: data point which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism

An insight on the structure of the data


Features
Observations

Transaction ID	Pharmacy ID	Date&Time	Seller ID	Public Price	Buy Price	Amount paid	Change	Etc.
…	…	…	…	…	…	…	…	…
				⋮				
…	…	…	…	…	…	…	…	…

An insight on the structure of the data

Features

Observations

Transaction ID	Pharmacy ID	Date&Time	Seller ID	Public Price	Buy Price	Amount paid	Change	Etc.
…	…	…	…	…	…	…	…	…
				⋮				
…	…	…	…	…	…	…	…	…

Data can be:

- **Numerical**: the data is described by continuous values with a mathematical meaning (e.g. price, height, weight…)

- **Categorical**: the data is described by discrete values with no mathematical meaning (e.g. day of the week, patient ID etc.)

The two faces of Machine Learning

Machine Learning is generally split into two macro-areas:

- **Supervised learning**: inferring a function from labeled training data that will eventually be used to map a new set of data (used in physics)

- **Unsupervised learning**: inferring a function to describe hidden structures from unlabelled data

I will focus on Unsupervised learning only

Working environment: Anaconda and Python

- **Anaconda**: open-source distribution for Python and R programming languages to create different Python environments on your computer while preserving the dependencies among the different packages
  https://www.continuum.io/downloads

- **Python libraries**:
  - **Pandas**: Data Analysis Library designed to provide easy-to-handle data structures. Useful when working with several and highly-structured datasets.
    http://pandas.pydata.org/pandas-docs/stable/10min.html
  - **Scikit-learn**: Machine Learning Library (classification, regression, clustering…) designed to interoperate with NumPy and SciPy.
    http://scikit-learn.org/stable/

Clustering algorithms

- **Clustering** is an **unsupervised ML technique** (it works with unlabelled data)
- It splits observations into **groups** according to the **similarity** among them
- Three types of algorithms:
    - k-Means (numerical data)
    - k-Modes (categorical data)
    - k-Prototypes (numerical & categorical data)

We expect the clusters to have, on average, the same number of data points. **Anomalies could lie in small clusters** (since they are rare).

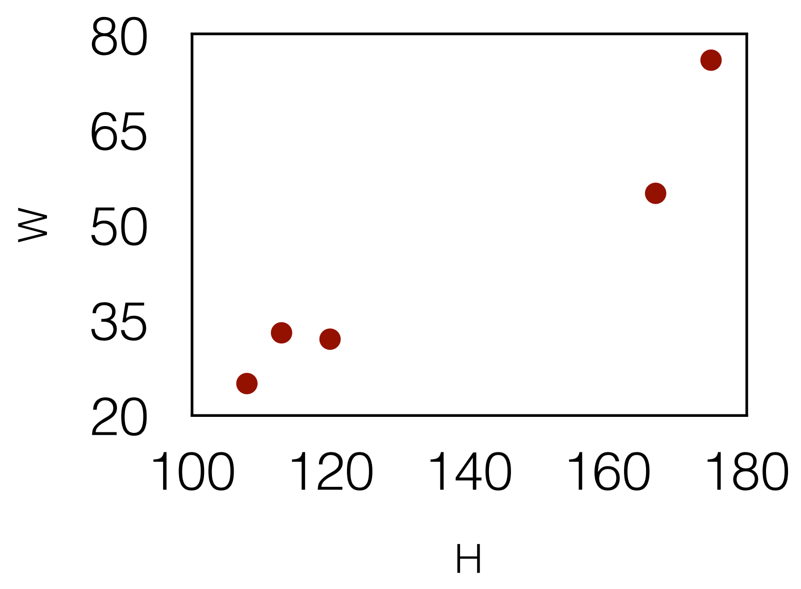
k-Means

- It groups data by means of **distance** between data points (Euclidian distance)
- It partitions the space into **Voronoi cells**, i.e. separates the samples into k groups of equal variance
- Implemented in the *scikit-learn* package in Python
- It consists of three steps:
  1. **Choose k** (number of clusters)
  2. **Assignment**: observations are assigned to the clusters they are more similar to
  3. **Update**: the cluster center is recalculated based on the mean of the previous assignment

Repeat 2. and 3. until convergence is reached

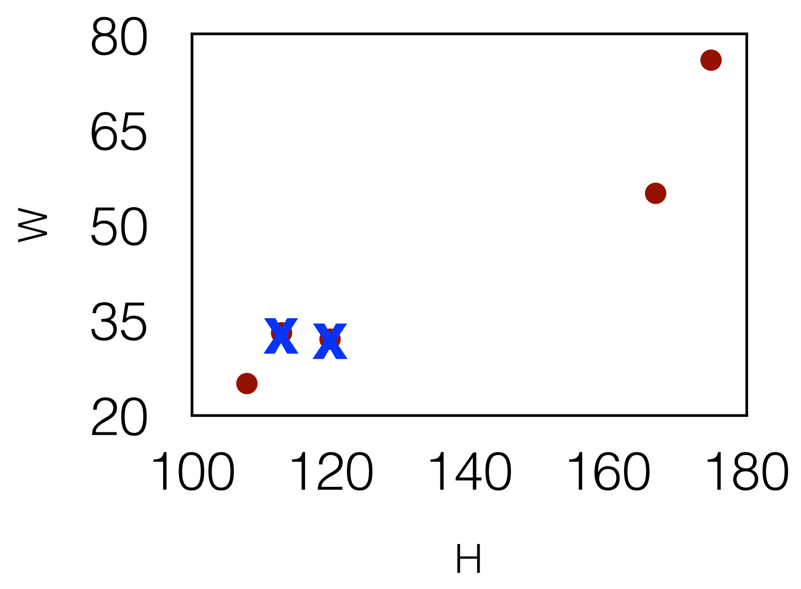
Clustering: a simple example

	Height (H)	Weight (W)
Person 1	167	55
Person 2	120	32
Person 3	113	33
Person 4	175	76
Person 5	108	25



Clustering: a simple example

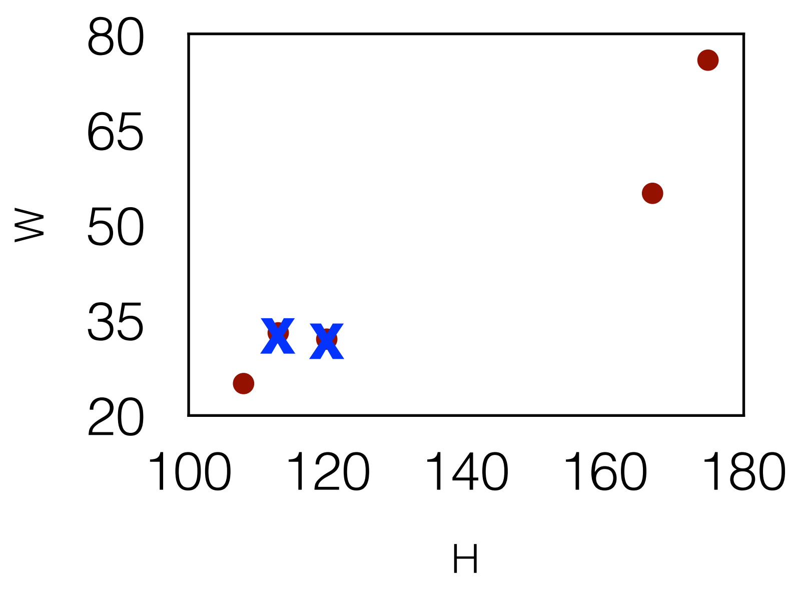
	Height (H)	Weight (W)
Person 1	167	55
Person 2	120	32
Person 3	113	33
Person 4	175	76
Person 5	108	25



1. Choose k=2 and assign random centroids, e.g. c1=(120,32) and c2=(113,33)

Clustering: a simple example

	Height (H)	Weight (W)
Person 1	167	55
Person 2	120	32
Person 3	113	33
Person 4	175	76
Person 5	108	25



1. Choose k=2 and assign random centroids, e.g. c1=(120,32) and c2=(113,33)

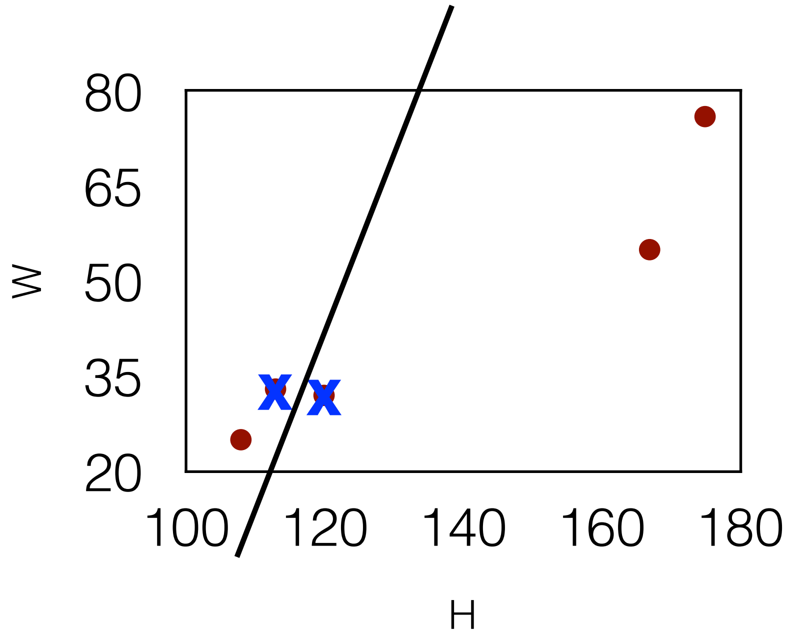
2. Calculate the euclidian distance of each observation from each centroid.

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_1)^2}$$

	Distance of object from c1	Distance of object from c2
Person 1	52.3	58.3
Person 2	0	7.1
Person 3	7.1	0
Person 4	70.4	75.4
Person 5	13.9	9.4

Clustering: a simple example

	Height (H)	Weight (W)
Person 1	167	55
Person 2	120	32
Person 3	113	33
Person 4	175	76
Person 5	108	25



1. Choose k=2 and assign random centroids, e.g. c1=(120,32) and c2=(113,33)

2. Calculate the euclidian distance of each observation from each centroid.

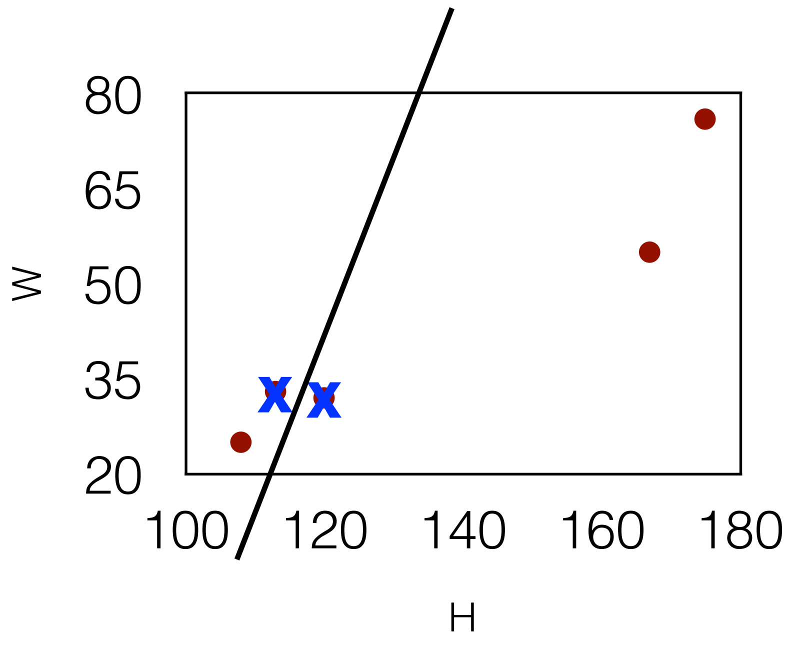
$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_1)^2}$$

Minimize it and make assignment

	Distance of object from c1	Distance of object from c2
Person 1	52.3	58.3
Person 2	0	7.1
Person 3	7.1	0
Person 4	70.4	75.4
Person 5	13.9	9.4

Clustering: a simple example

	Height (H)	Weight (W)
Person 1	167	55
Person 2	120	32
Person 3	113	33
Person 4	175	76
Person 5	108	25



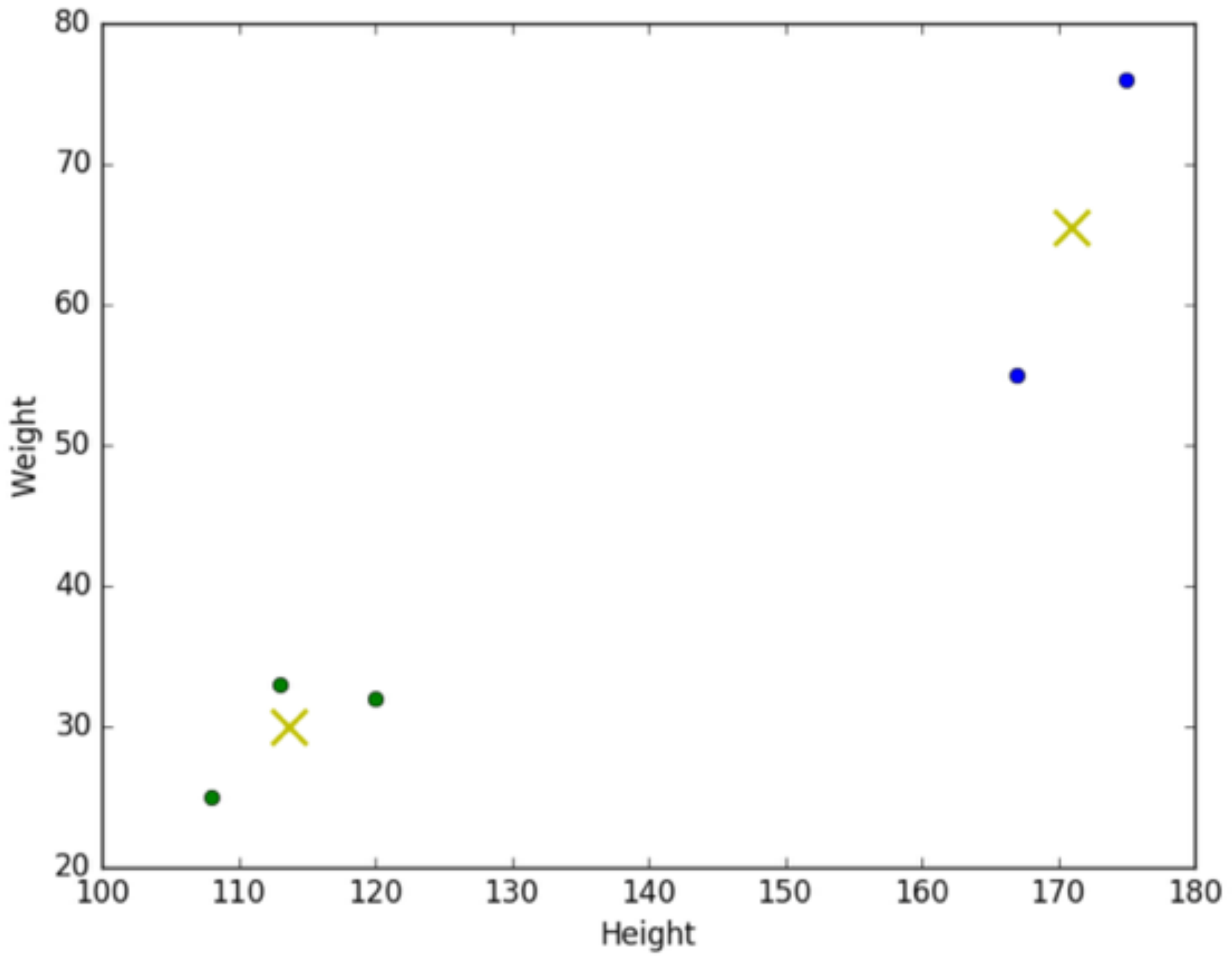
1. Choose k=2 and assign random centroids, e.g. c1=(120,32) and c2=(113,33)

2. Calculate the euclidian distance of each observation from each centroid.
$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_1)^2}$$
Minimize it and make assignment

	Distance of object from c1	Distance of object from c2
Person 1	52.3	58.3
Person 2	0	7.1
Person 3	7.1	0
Person 4	70.4	75.4
Person 5	13.9	9.4

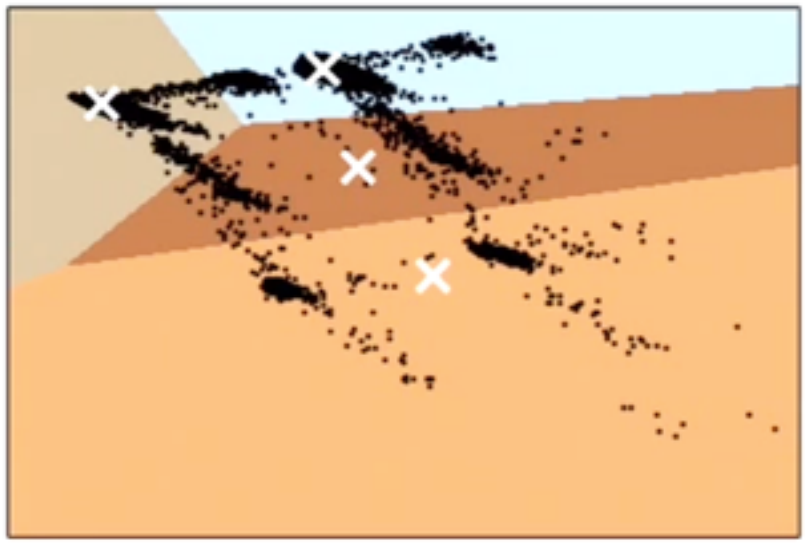
3. Update the centroids: c1'=mean of the members of cluster 1
   c2'=mean of the members of cluster 2

Repeat 2. and 3. until the observations no longer move around:

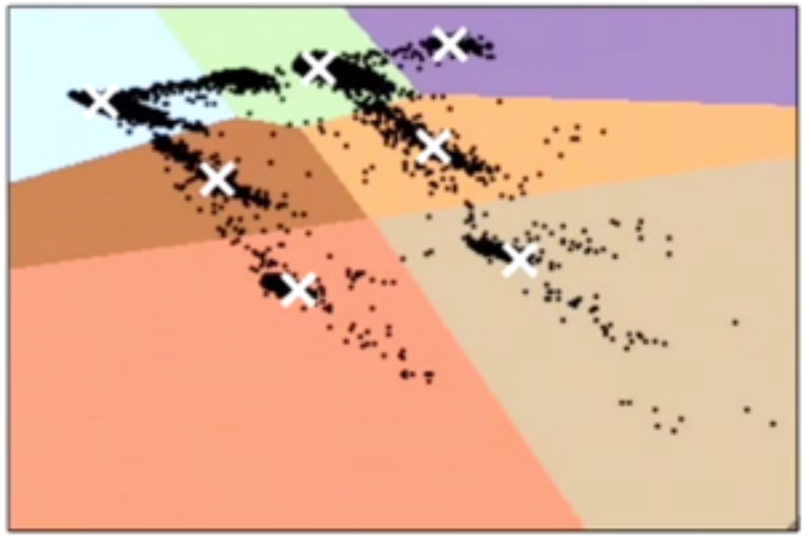


But there are cases where choosing k is not that easy…

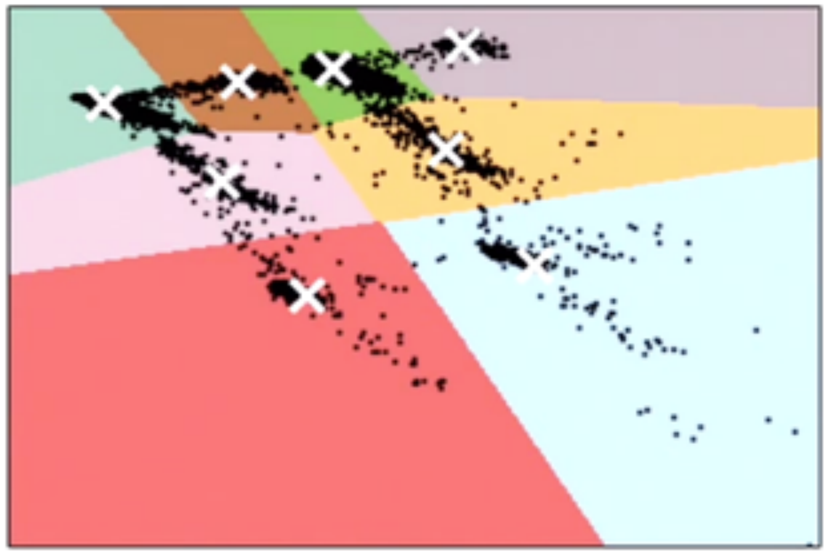
k=4



k=7



k=8

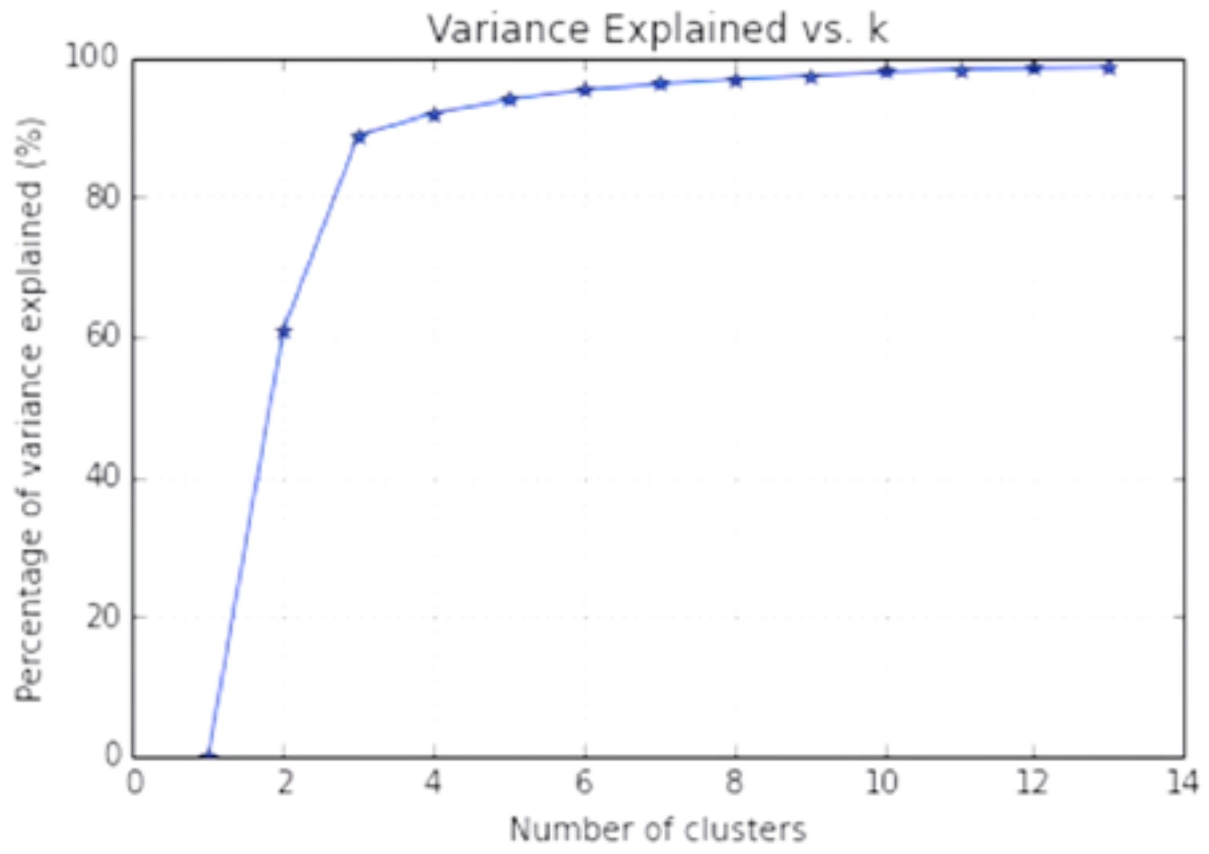


Plots by Sarah Guido, senior data scientist at Mashable, New York

How to choose k

Graph the percentage of **variance explained** VS **k**

$$Variance\,explained = \frac{Between - cluster\,sum\,of\,squares}{Total\,sum\,of\,squares}$$



Choose the k with the highest variance

Plot by Sarah Guido, senior data scientist at Mashable, New York

What if our data is categorical?

- By definition, **k-Means works only with numerical values**

⚠ If the observations are non-numerical, transform them into numerical? Beware! Really distant object which have been assigned two close numbers could end up in the same cluster

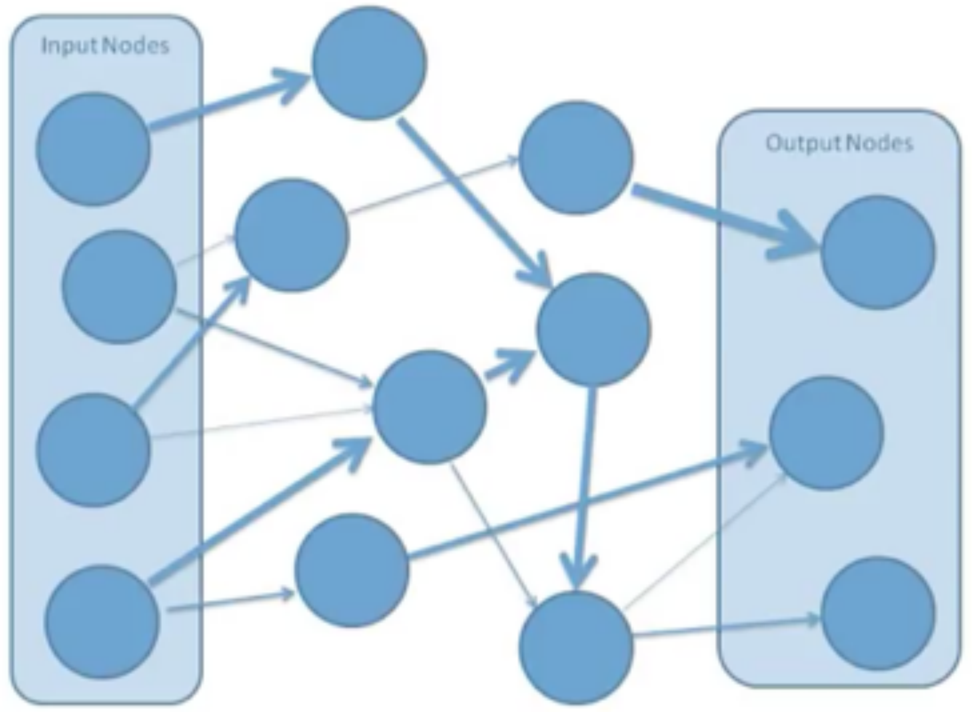
- **k-Modes is the solution**. Same "philosophy" as k-Means but:
    - Instead of Euclidian distances —> **dissimilarities** (total mismatches of a given object for a given feature)
    - Instead of means —> **modes** (vectors that best represent the frequency of objects in the data)
    - Instead of minimizing distances between the objects and the mean, it minimizes dissimilarities between the objects and the modes
- For both numerical and categorical data: **k-Prototypes**
- Only a Python implementation of k-Modes and k-Prototypes exists on Github

Disadvantages of clustering algorithms for anomaly detection

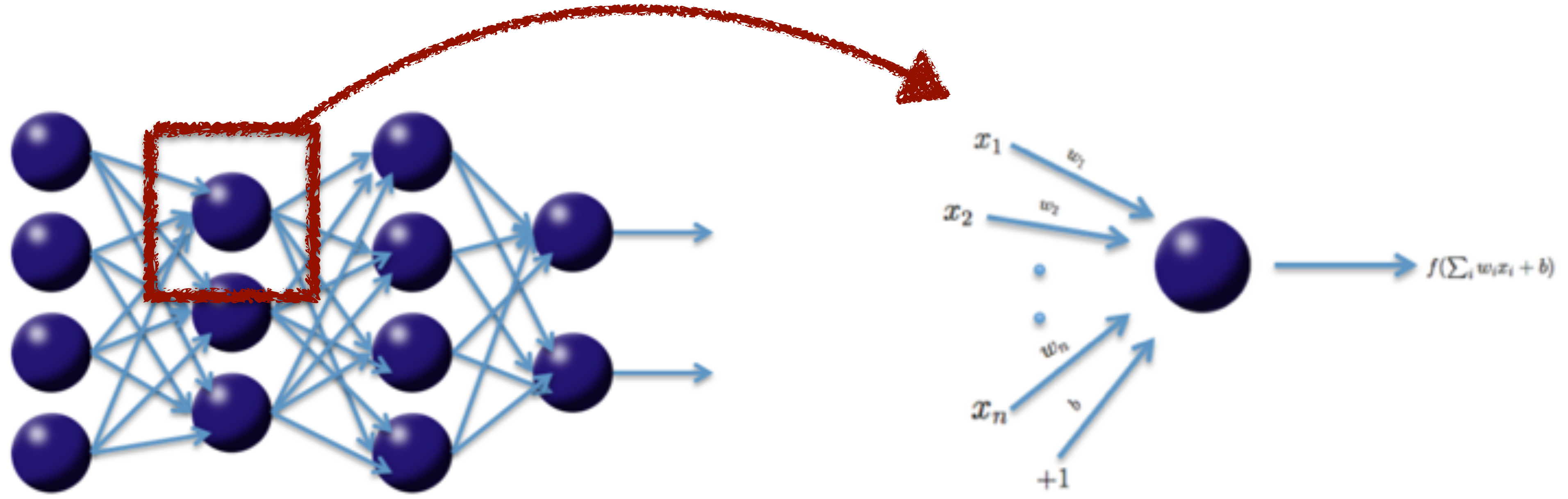
- **k-Means** is a promising way for numerical data, but **not applicable** to our data
- **k-Prototypes** implementation is incredibly **slow**! It doesn't scale well
- k-Modes could be used by converting numerical values into categorical, but still slow
- But overall: **they don't quantify the anomaly!**

From clustering to neural networks

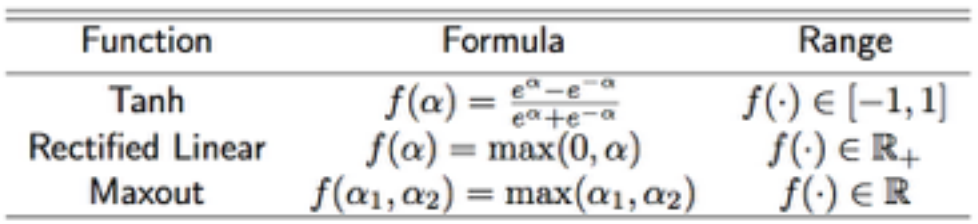
- **Neural network**: machine learning algorithms inspired to human brain
- It consists of **input** nodes, **output** nodes and some **intermediate** nodes that elaborate the original information
- The nodes are connected by **connection weights**



Neural networks overview



- **f**: **activation function** (e.g. Tanh). It's the value transmitted by the connected neuron
- **w**: **weights** connections
- **b**: **bias** (neuron's activation threshold), included in each non-output layer



Function	Formula	Range
Tanh	$f(\alpha) = \frac{e^{\alpha}-e^{-\alpha}}{e^{\alpha}+e^{-\alpha}}$	$f(\cdot) \in [-1, 1]$
Rectified Linear	$f(\alpha) = \max(0, \alpha)$	$f(\cdot) \in \mathbb{R}_+$
Maxout	$f(\alpha_1, \alpha_2) = \max(\alpha_1, \alpha_2)$	$f(\cdot) \in \mathbb{R}$

The weights linking neurons and biases with other neurons fully determine the output of the entire network. **Learning occurs when these weights are adapted to minimize the error** on the labeled training data, or more specifically a **loss function**

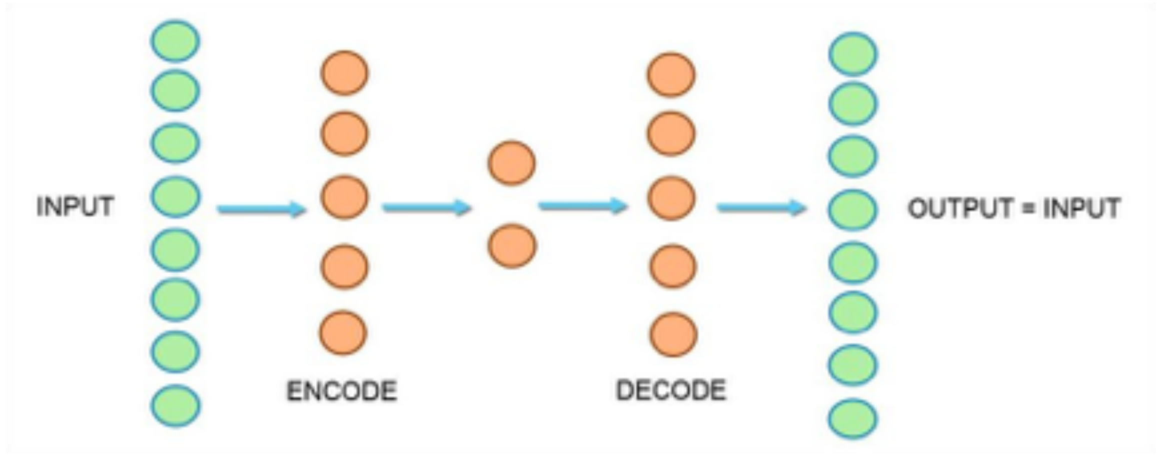
Neural networks overview

Thus, the neural network algorithm consists of two steps:

1. **Training**: the algorithm adjusts the weights to obtain the desired output (we know what the output has to look like), iteratively, on a training dataset

2. **Testing**: the trained n.n. is applied to a testing dataset to extract information (e.g. to make classification or regression)

A particular type of n.n.: autoencoders



- Structure of a n.n. with **autoencoder**:

  1. **Input data** (features)

  2. **Hidden layer(s)**: compress the input data in a lower dimensional space —> obtain the true nature of data without uninteresting features and noise

  3. **Output nodes**: from the hidden layer(s) reconstruct the original space

- In simple words, **the autoencoder learns the pattern of the data** by learning an **identity function** h(x)≈x in the training process

H2O: an open-source software for big data analysis

H_2O.ai

Scalable Machine Learning
For Smarter Applications

- To apply the autoencoder: **H2O**, a machine learning platform
- Advantages: fast parsing, free, automatic treatment of missing values (mean imputation) and of categorical values (binary encoding)
- Lot of documentation in R and recently in Python as well
- http://docs.h2o.ai/h2o/latest-stable/index.html

How to detect anomalies with autoencoders

- **Train the autoencoder on data with no anomalies**

- The training process outputs a MSE = $\frac{1}{2n}\Sigma_i||t_i - o_i||^2$ as indicator of the difference between the original data and its low-dimensional reconstruction

- **Apply the autoencoder to the test dataset**

- Each row of the output for the test dataset comes out with a reconstruction error, to be compared with the MSE coming from the training

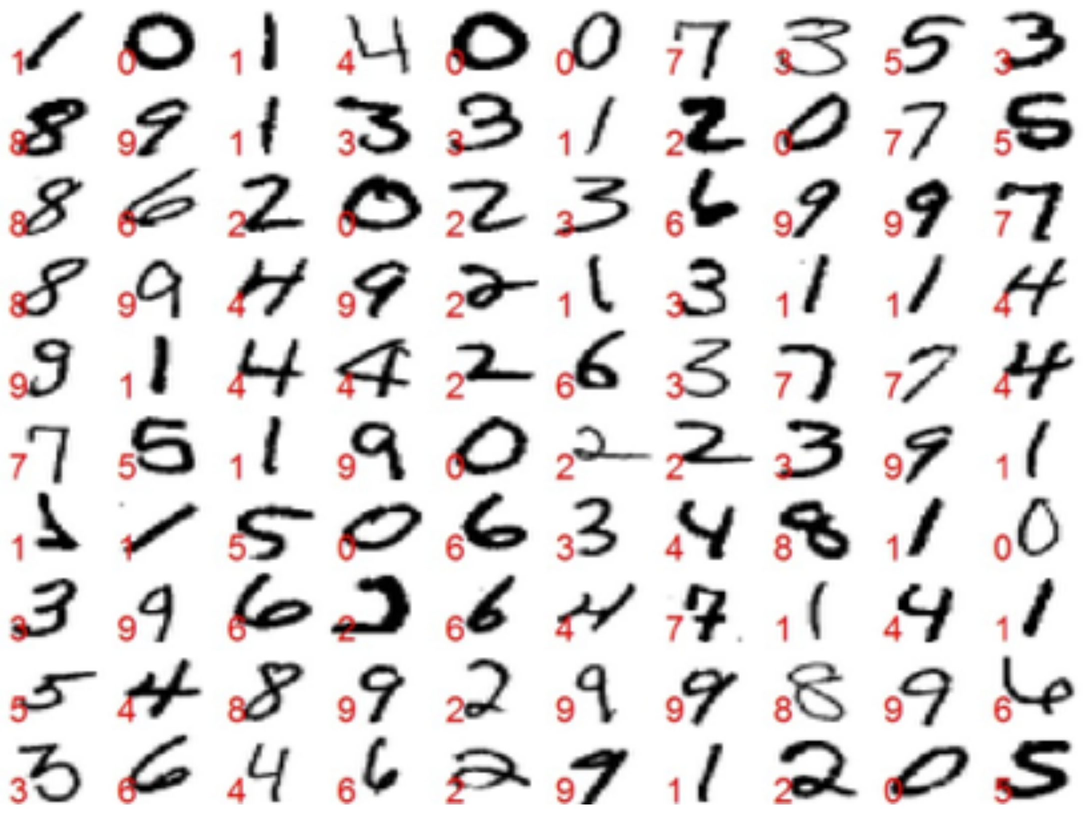
- **Rows with high reconstruction error w.r.t. MSE coming from the training are likely to be outliers**

Anomaly detection in the MNIST dataset

- MNIST dataset: 42.000 handwritten digits, each one made up of 28x28=784 pixels

  42.000 rows: each row is a digit

  785 columns: pixel intensity (784 columns) plus one additional label column.



Applying an autoencoder to MNIST

Goal: let's find the ugly digits (outliers because rare)

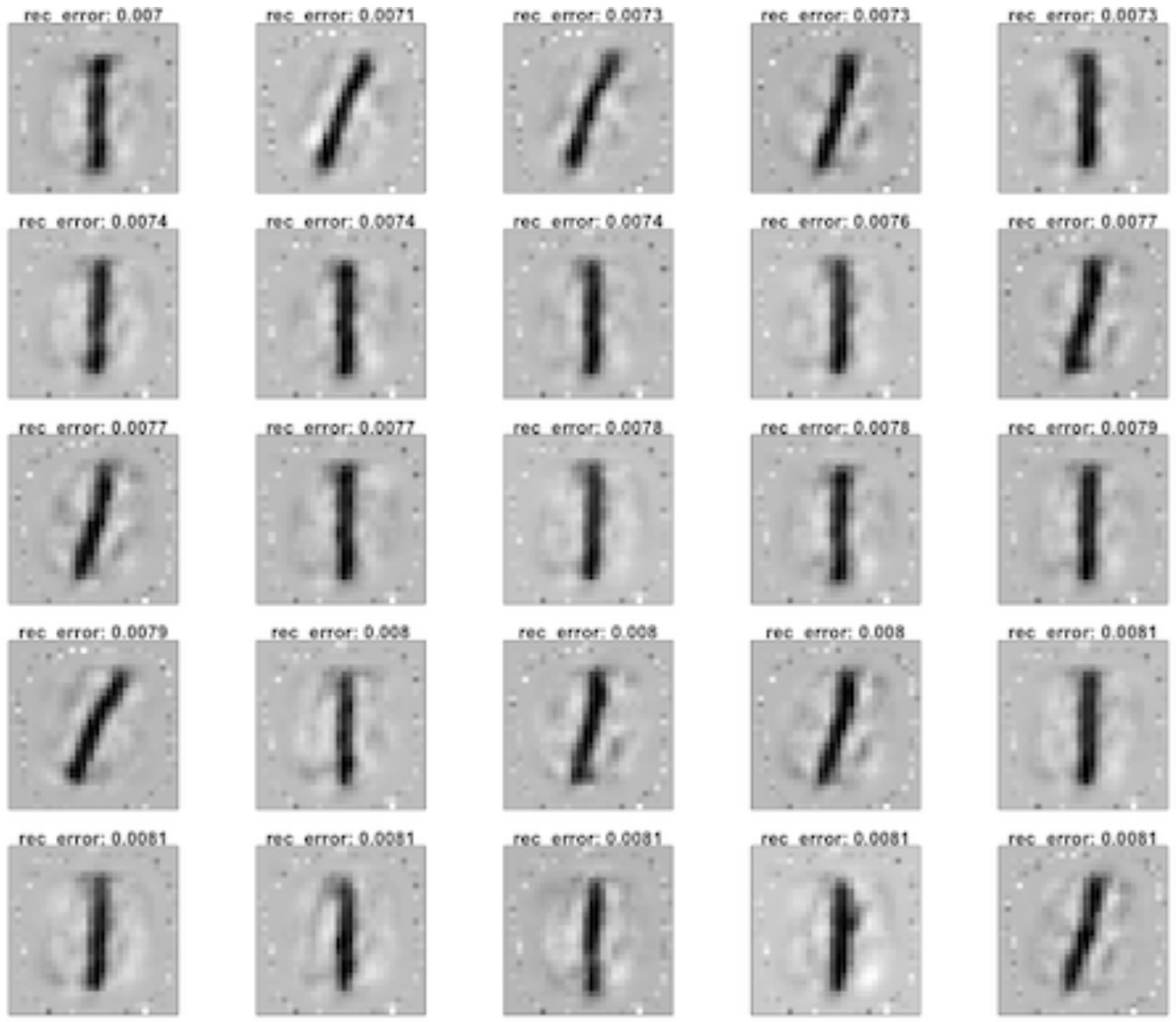
1. **Train the autoencoder to learn the nice digits** on a training dataset (for the MNIST dataset 50 or 100 compressed features are enough to extract the pattern)

2. **Apply the n.n. to the test dataset** and **get the reconstructed error** for each row

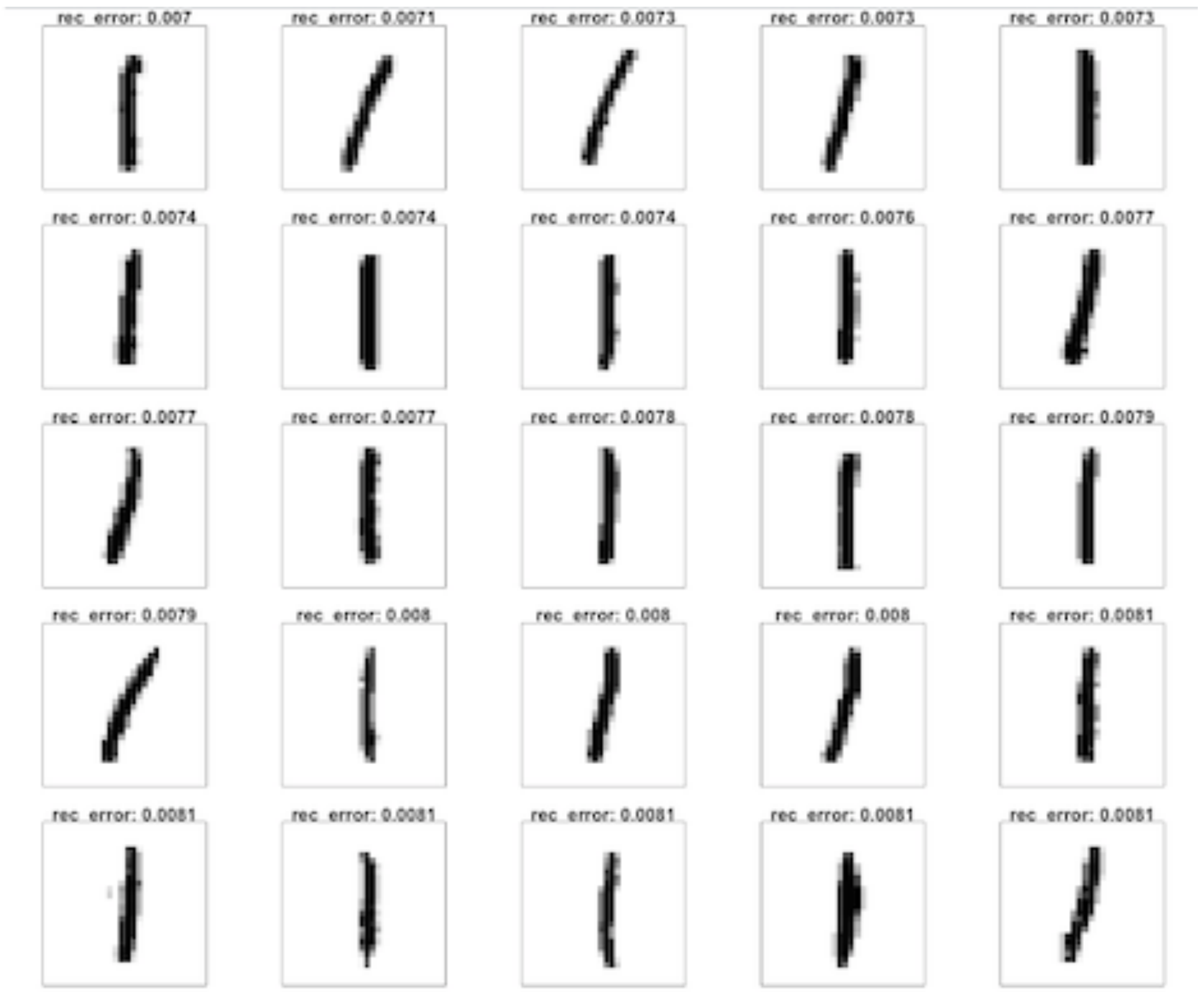
3. **Visualize the good, the bad and the ugly digits**, according to the reconstructed error

Smallest reconstruction errors

Reconstructed digits



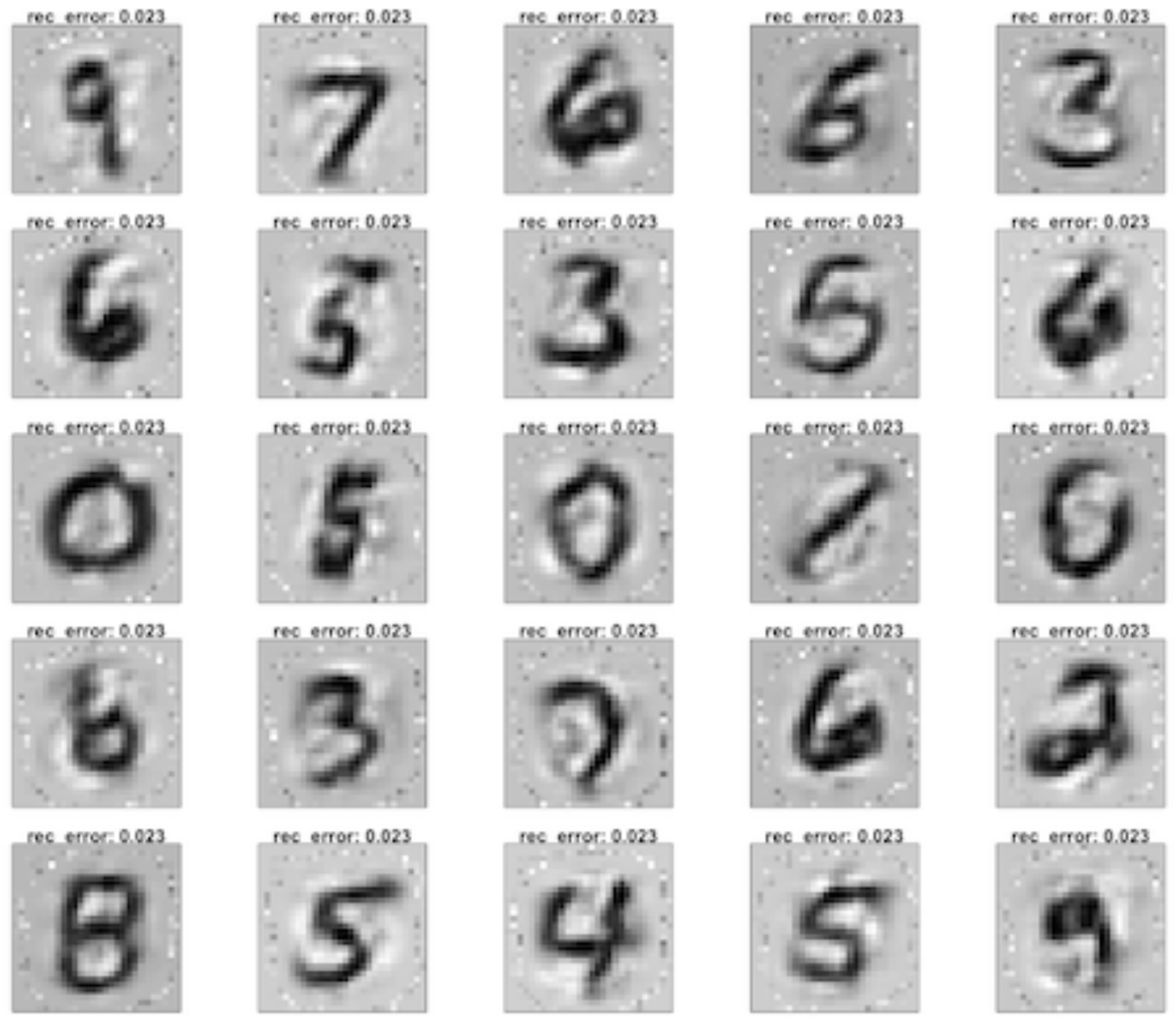
Test digits



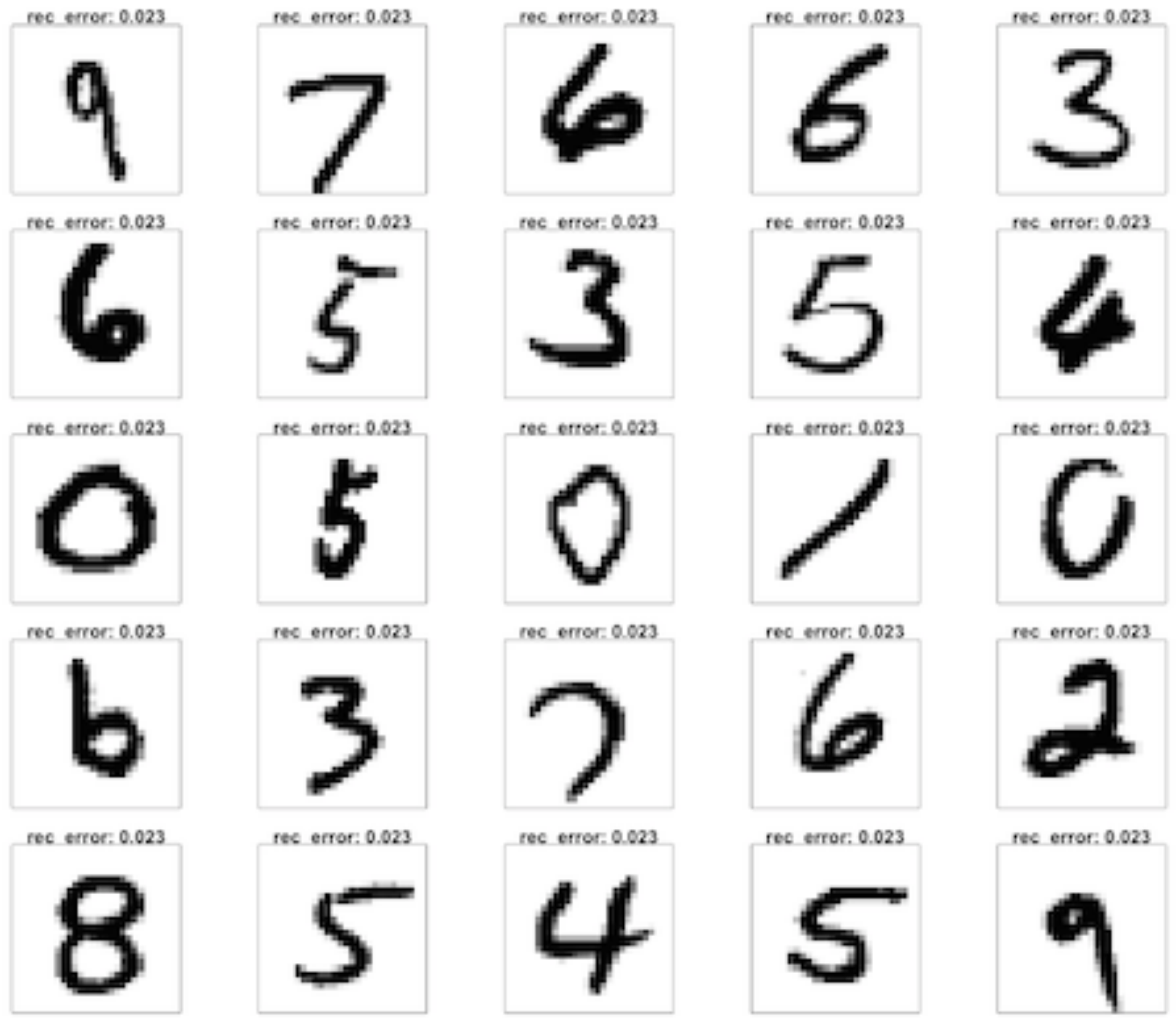
Not surprisingly, the best reconstructed digits are 1… there's no much you can do wrong with a straight line!

“Median” reconstruction errors

Reconstructed digits



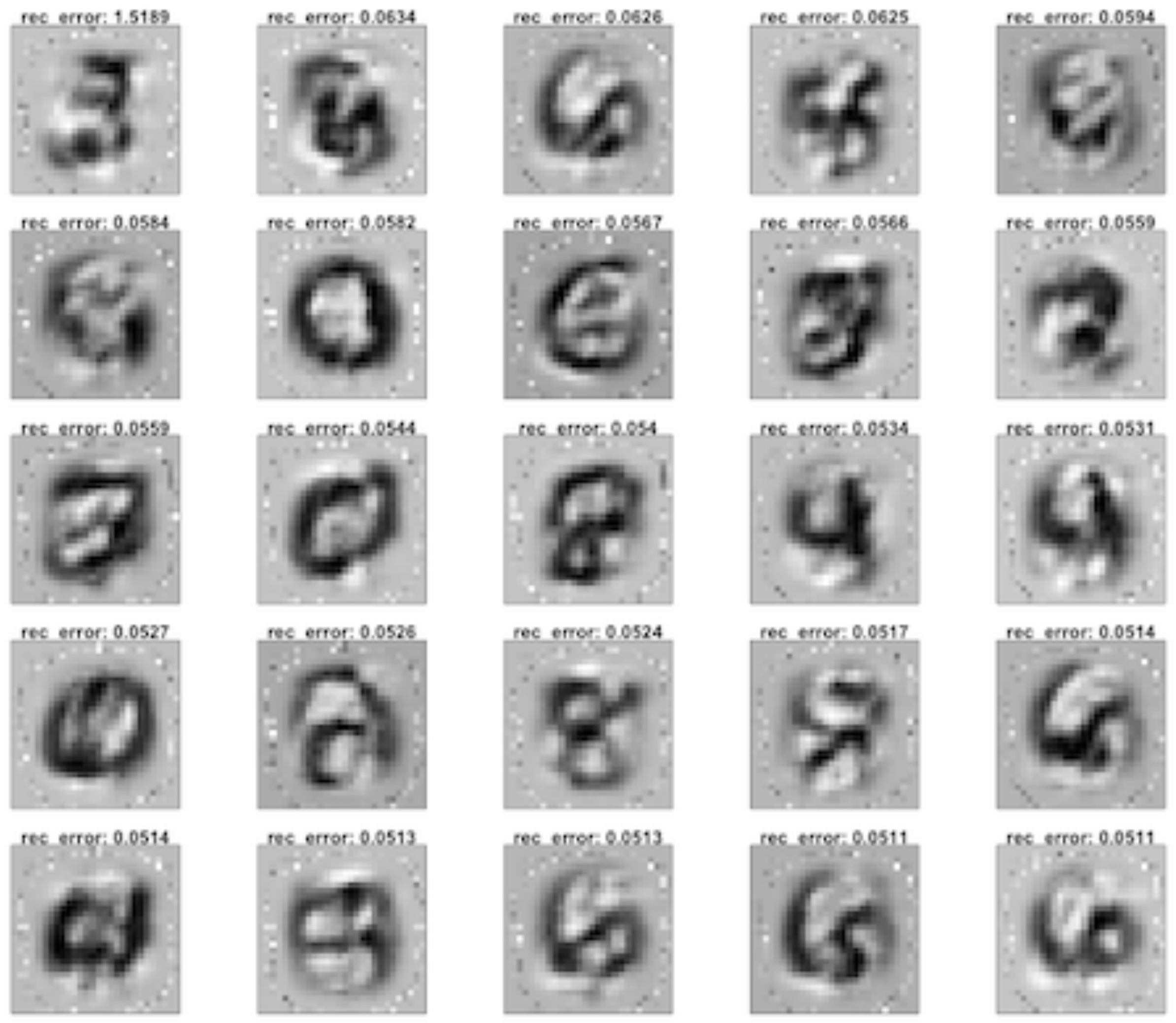
Test digits



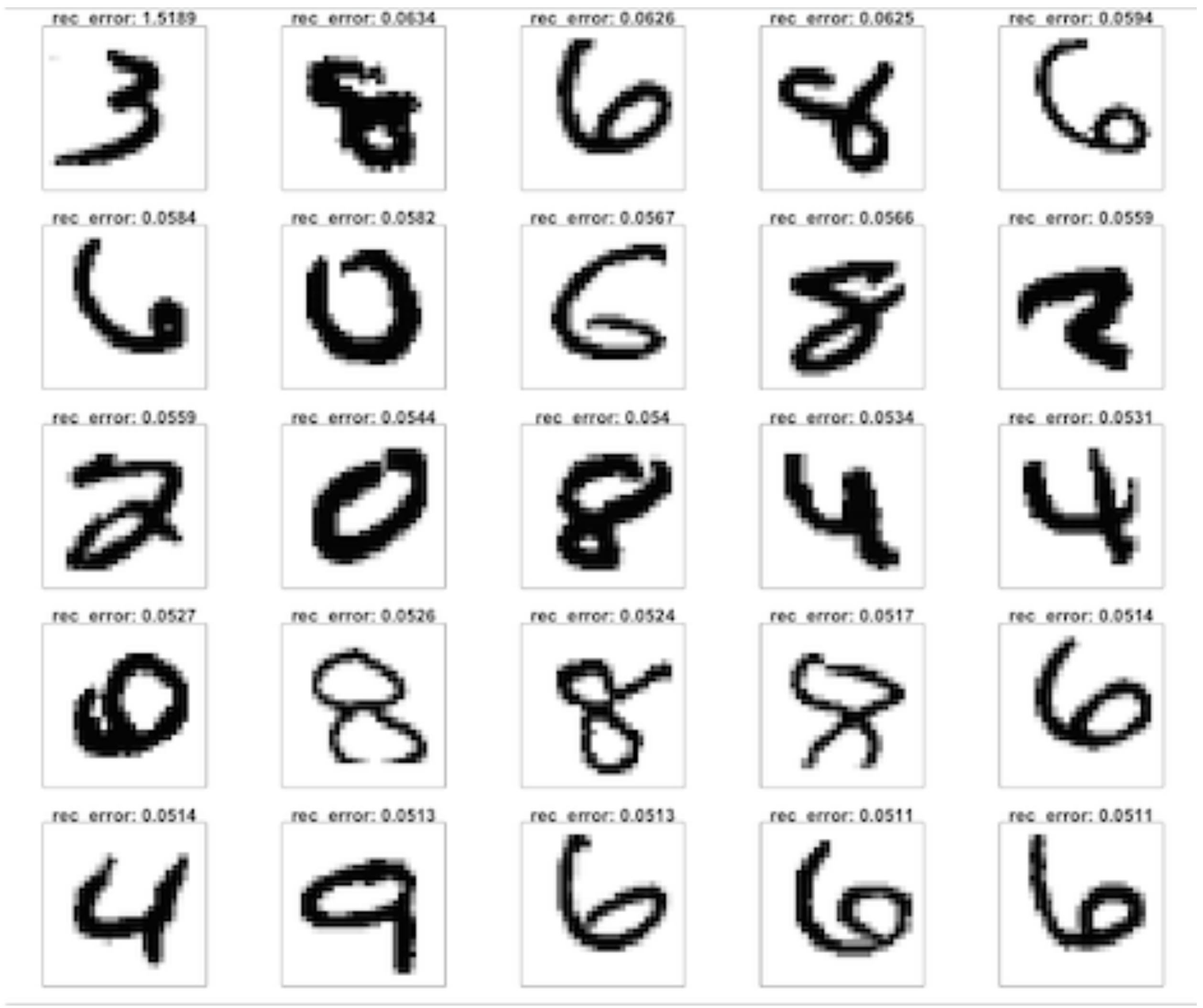
They look "normal" - it is plausible that they resemble digits from the training data to a large extent, but they do have some particularities that cause some reconstruction error.

Highest reconstruction errors

Reconstructed digits



Test digits



Pretty ugly digits that are plausibly not commonly found in the training data - some are even hard to classify by humans.

Anomaly detection on the pharmacy data

- **Training dataset**: 1 year of data, all pharmacies but one (supposing the potential anomalies are diluted)
- **Test dataset**: data with the excluded pharmacy

Parameters used:

# features: 10
# hidden layers: 1
# neurons in the hidden layer: 5
# epochs: 20

Very interesting results found!

Conclusions

- **Clustering algorithms** are **unsupervised ML techniques** that group data with the same characteristics
- But **not ideal to detect anomalies**, especially with categorical values: slow and anomalies not quantified
- The other way around: train a **(deep) learning autoencoder** to learn the pattern of the data and possibly discover outliers
- H2O was a promising way to proceed - it is fast, scalable and with a nice documentation

Backup slides

Silhouette coefficient

For each datum *i*:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

where:

a(*i*): average dissimilarity of *i* with all the other data within the same cluster;
b(*i*): lowest average dissimilarity of *i* to any other cluster, of which *i* is not a member.

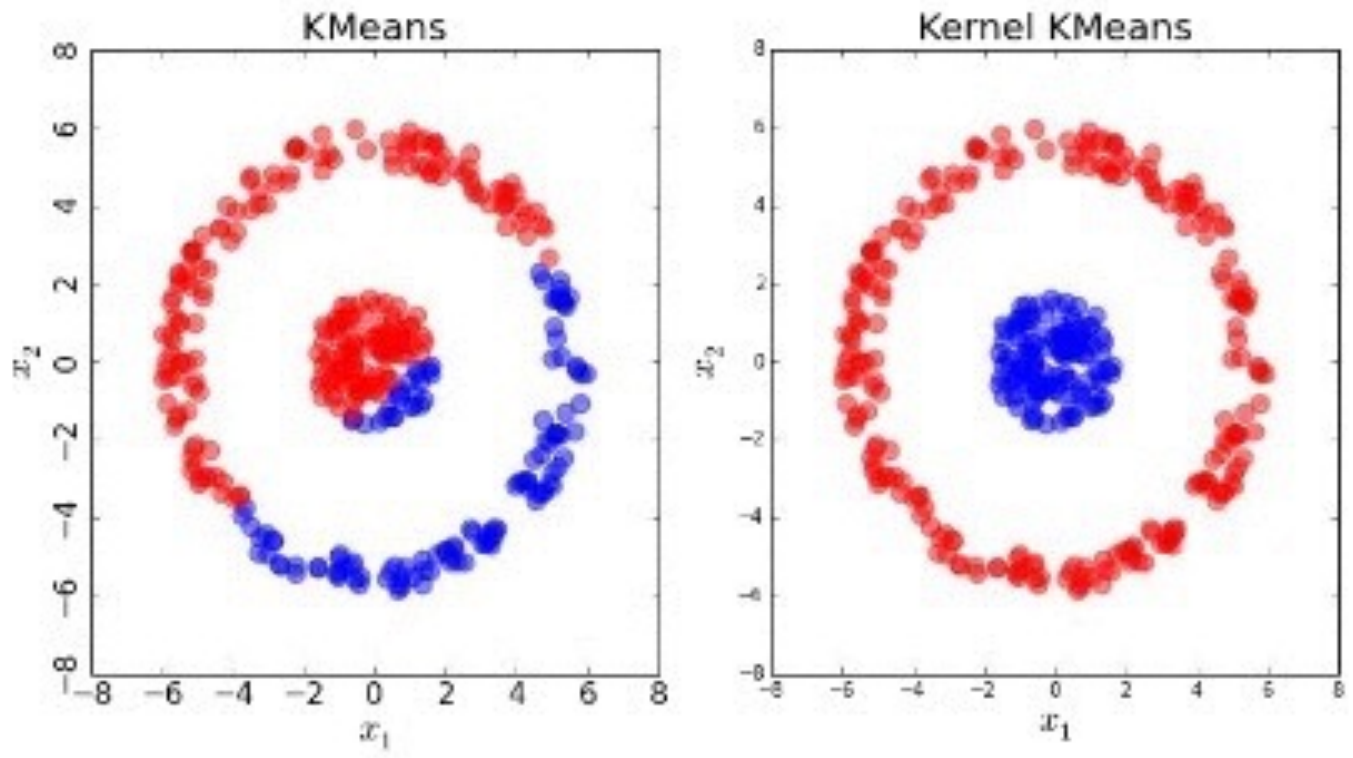
$$s(i) = \begin{cases} 1 - a(i)/b(i), & \text{if } a(i) < b(i) \\ 0, & \text{if } a(i) = b(i) \\ b(i)/a(i) - 1, & \text{if } a(i) > b(i) \end{cases}$$

Choose the configuration with the highest s

Missing values

- Occasionally, not all the observations are complete: some values missing
- Ways to treat missing values:
    - **Casewise deletion**: discard observations with incomplete measurements (not doable in our case since many features and almost always with at least one missing value)
    - **Mean imputation**: replace every missing value with a constant i.e. the mean of each feature over the nonmissing data.
- For categorical variables, consider the value "missing" as just another categorical value (if it's reasonable to consider two objects as being similar if they both have missing values on the same variables).

Kernel K-means vs. K-means



Pyclust: Open Source Data Clustering Pckage

11