Hochschule Offenburg
offenburg.university

Machine Learning VI

Outlier Detection

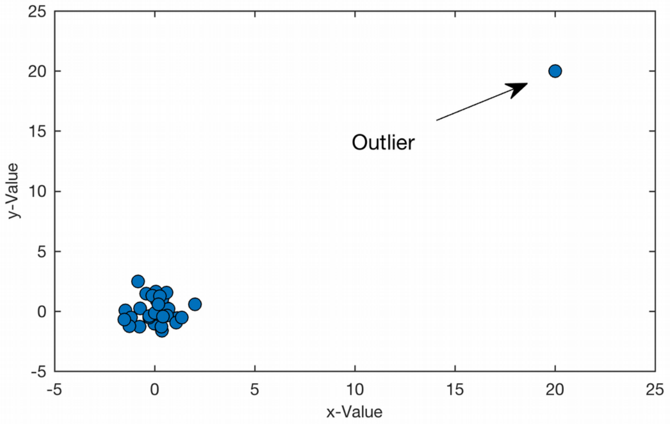
Outlier Detection

Outline

- **Motivation**
- **Traditional ML Methods**
  - DB-Scan Clustering
  - One-Class SVMs
  - Isolation Forests
- **Deep Learning Methods**
  - Auto Encoder

Motivation

Outlier Detection



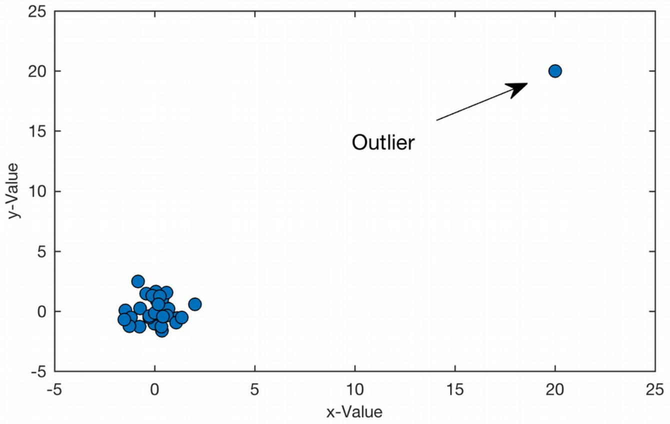
Oversimplification

Machine Learning / Data Mining so far…

- We used supervised and unsupervised methods to learn from data
- Constraint: given training data needs to be a (statistically) good representation of the problem → generalization
- Works well for many applications ...

Motivation

Outlier Detection



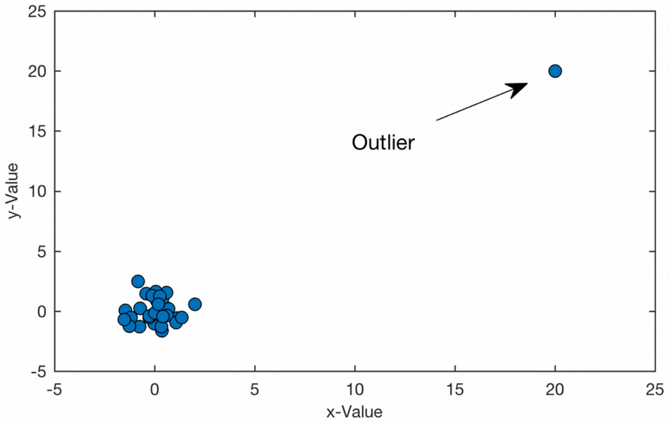
Oversimplification

But, what if we are looking to detect things that we have not seen before ?

- Only “good” examples
- No or only a few “bad” examples
- **Applications:**
  - Find faulty parts (production QC)
  - Find Malware
  - Find Intruders
  - ...

Motivation

Outlier Detection



Oversimplification

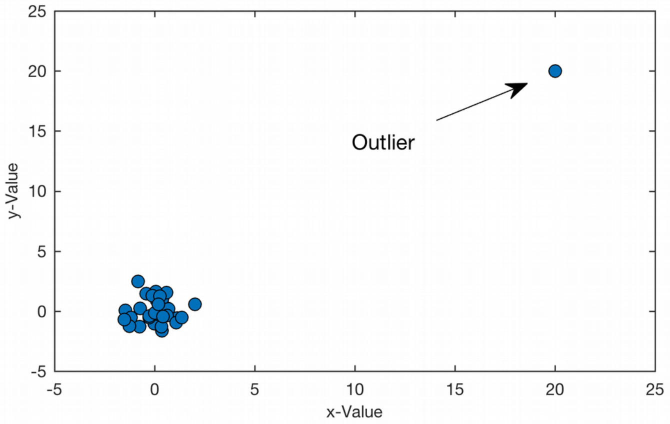
But, what if we are looking to detect things that we have not seen before ?

- Only “good” examples
- No or only a few “bad” examples
- **Applications:**
  - Find faulty parts (production QC)
  - Find Malware
  - Find Intruders
  - ...

→ learn “what is normal” and detect derivations (=outliers)

Motivation

Outlier Detection



Oversimplification

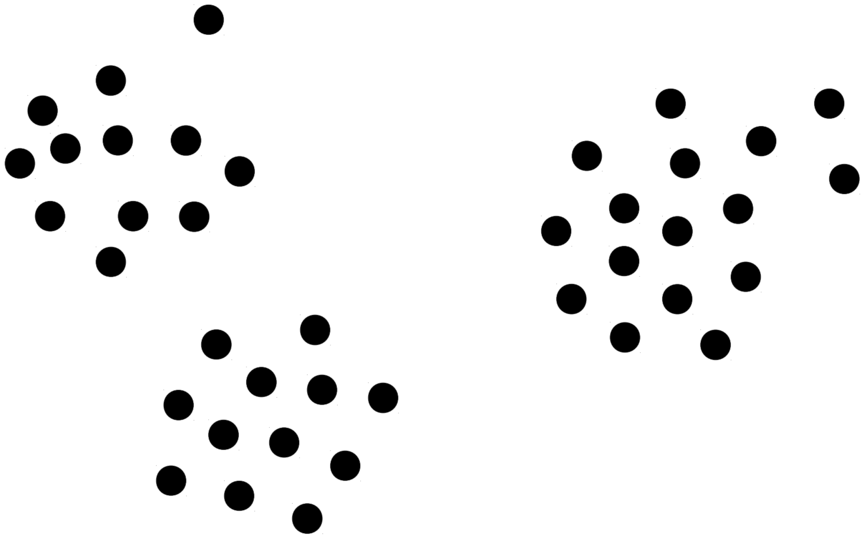
Definition [Wikipedia]

- *“In statistics, an **outlier** is a data point that differs significantly from other observations. An outlier may be due to variability in the measurement or it may indicate experimental error; the latter are sometimes excluded from the data set.”*

- *“In data mining, **anomaly detection** (also outlier detection) is the identification of rare items, events or observations which raise suspicions by differing significantly from the majority of the data.”*

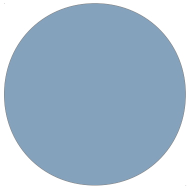
Outliers with Machine Learning

Using DBSCAN clustering [see ML 1 in block 1]



Init:

ε



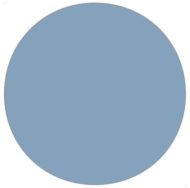
n_ε = 3

Outliers with Machine Learning

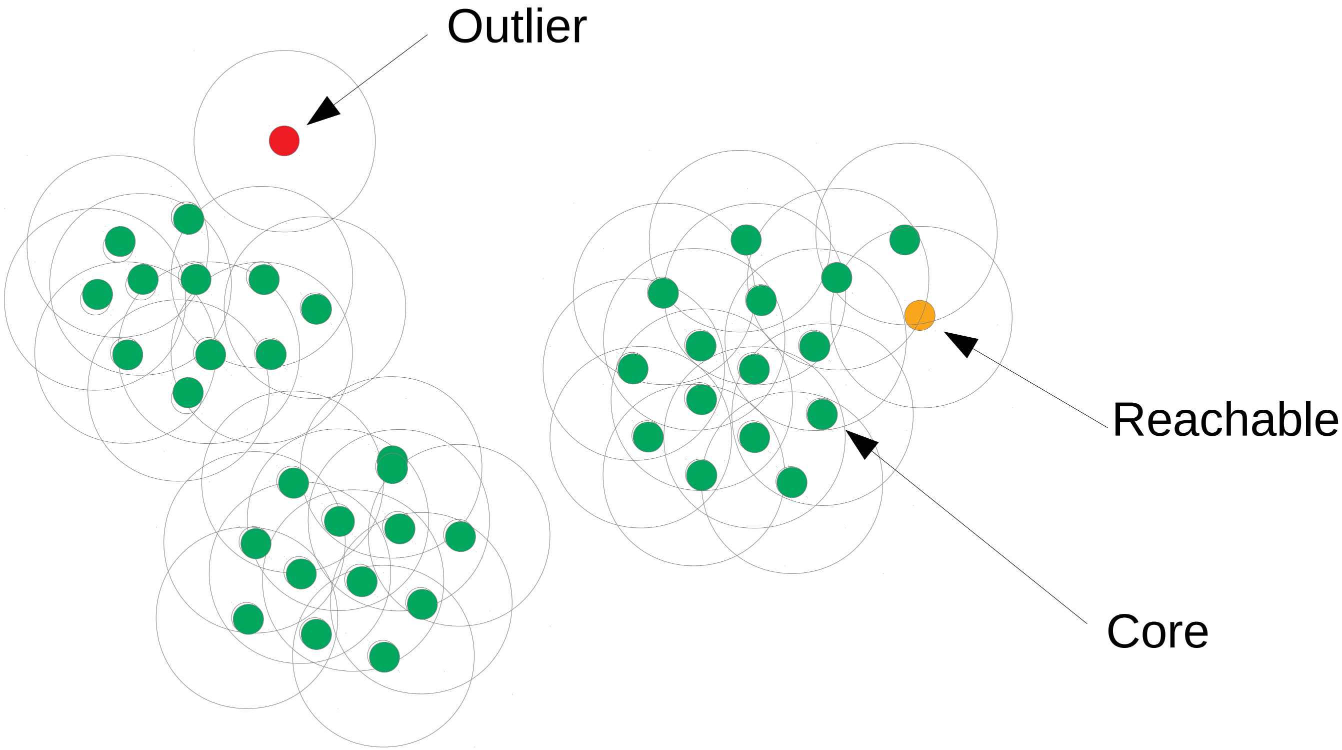
Using DBSCAN clustering

Init:

ε



n_ε = 3



Outliers with Machine Learning

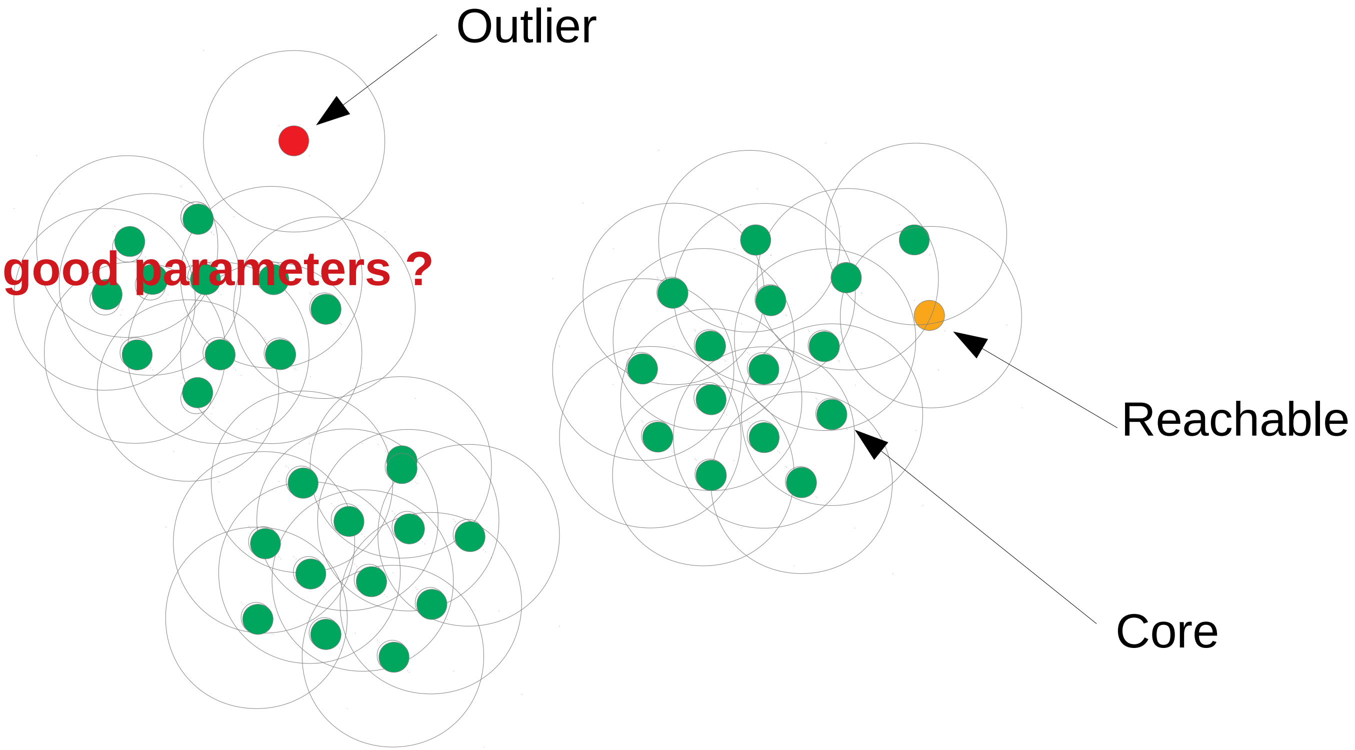
Using DBSCAN clustering

Problem: what are good parameters ?

Init:

ε

n_ε = 3



Outliers with Machine Learning

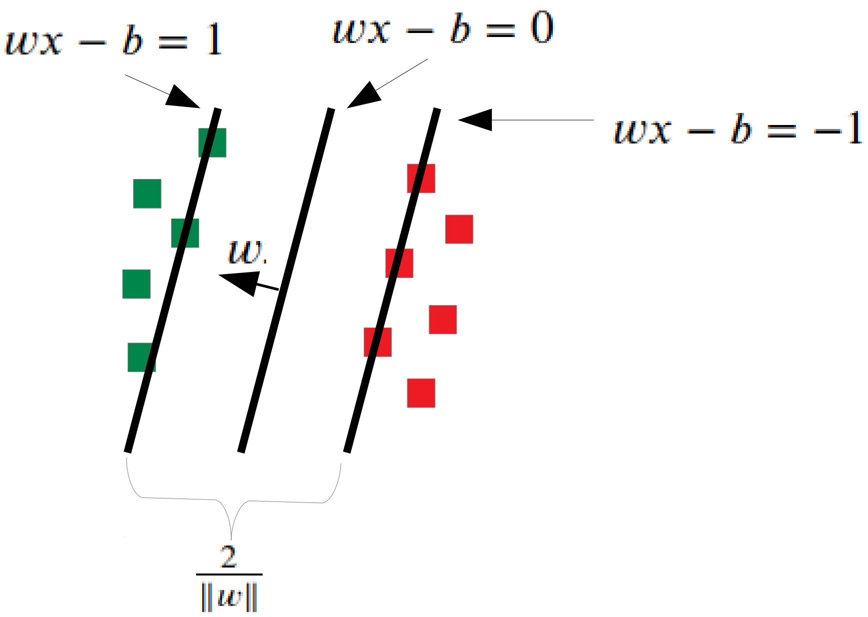
Using One-Class SVMs [recall SVMs from ML IV block 2]

SVM: binary supervised classification
→ New optimization problem
→ maximize “Margin”:
→ equals minimizing the uncertainty

$$\arg\min_{w} \sum_{i=0}^{N} \xi_i + \lambda \|w_i\|^2$$

subject to $y_i(w \cdot x_i - b) \geq 1 - \zeta_i$ and $\zeta_i \geq 0$, for all i.

$$\zeta_i = \max\left(0, 1 - y_i(w \cdot x_i - b)\right)$$



Outliers with Machine Learning

Using One-Class SVMs (following Schöllkopf)

Support Vector Method for Novelty Detection

Bernhard Schölkopf*, Robert Williamson§,
Alex Smola§, John Shawe-Taylor†, John Platt*
* Microsoft Research Ltd., 1 Guildhall Street, Cambridge, UK
§ Department of Engineering, Australian National University, Canberra 0200
† Royal Holloway, University of London, Egham, UK
* Microsoft, 1 Microsoft Way, Redmond, WA, USA
bsc/jplatt@microsoft.com, Bob.Williamson/Alex.Smola@anu.edu.au, john@dcs.rhbnc.ac.uk

Abstract

Suppose you are given some dataset drawn from an underlying probability distribution P and you want to estimate a "simple" subset S of input space such that the probability that a test point drawn from P lies outside of S equals some a priori specified ν between 0 and 1.

We propose a method to approach this problem by trying to estimate a function f which is positive on S and negative on the complement. The functional form of f is given by a kernel expansion in terms of a potentially small subset of the training data; it is regularized by controlling the length of the weight vector in an associated feature space. We provide a theoretical analysis of the statistical performance of our algorithm.

The algorithm is a natural extension of the support vector algorithm to the case of unlabelled data.

1 INTRODUCTION

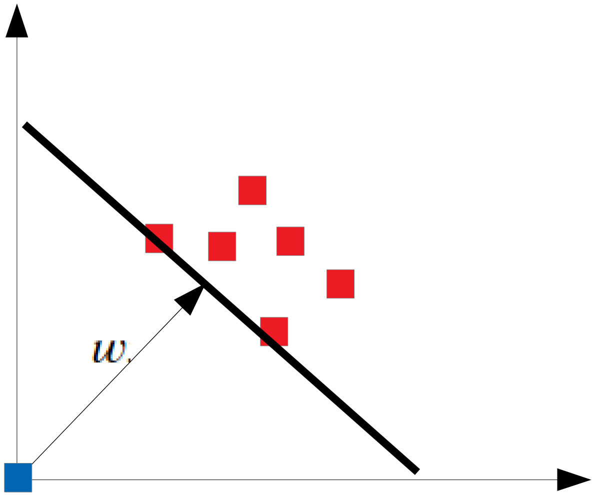
During recent years, a new set of kernel techniques for supervised learning has been developed [8]. Specifically, support vector (SV) algorithms for pattern recognition, regression estimation and solution of inverse problems have received considerable attention. There have been a few attempts to transfer the idea of using kernels to compute inner products in feature spaces to the domain of unsupervised learning. The problems in that domain are, however, less precisely specified. Generally, they can be characterized as estimating

Now: unitary supervised classification

→ New optimization problem

→ maximize "Margin" between origin and data

→ equals minimizing the uncertainty



Outliers with Machine Learning

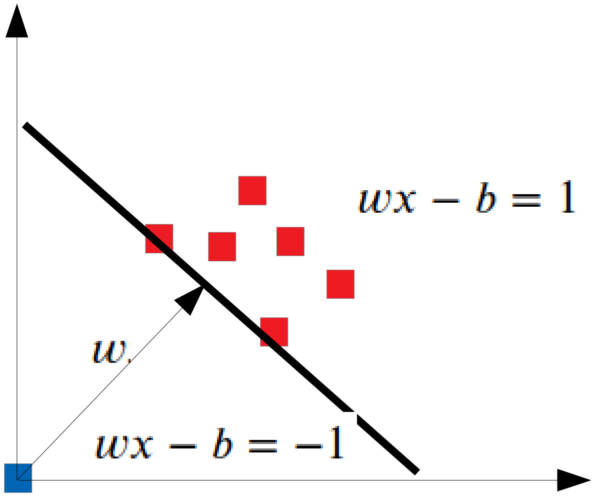
Using One-Class SVMs (following Schöllkopf)

Now: unitary supervised classification
→ New optimization problem
→ maximize "Margin" between origin and data
→ equals minimizing the uncertainty

$$\min_{w,\,\xi_i,\,\rho} \frac{1}{2}||w||^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \rho$$

subject to:

$$(w \cdot \phi(x_i)) \geq \rho - \xi_i \quad \text{for all } i = 1, \ldots, n$$
$$\xi_i \geq 0 \quad \text{for all } i = 1, \ldots, n$$



Outliers with Machine Learning

Hochschule Offenburg
offenburg.university

Using One-Class SVMs (following Schöllkopf)

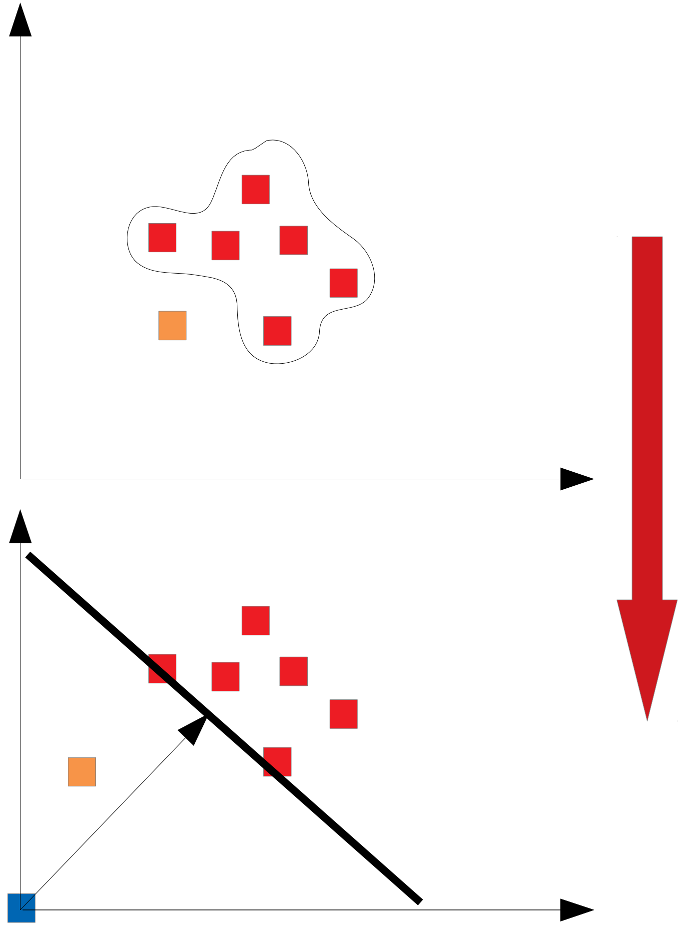
Now: unitary supervised classification
→ New optimization problem
→ maximize “Margin” between origin and data
→ equals minimizing the uncertainty

$$\min_{w,\,\xi_i,\,\rho} \frac{1}{2}\|w\|^2 + \frac{1}{\nu n}\sum_{i=1}^{n}\xi_i - \rho$$

subject to:

$$(w \cdot \phi(x_i)) \geq \rho - \xi_i \qquad \text{for all } i = 1, \dots, n$$
$$\xi_i \geq 0 \qquad \text{for all } i = 1, \dots, n$$

Non-linear transformation (a.k.a “kernel trick”)



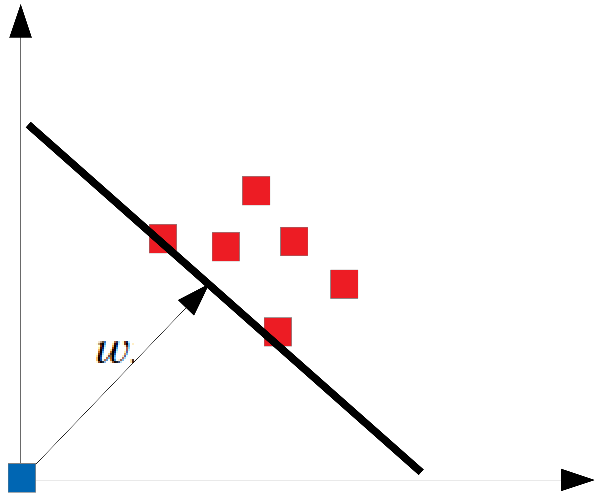
Outliers with Machine Learning

Using One-Class SVMs (following Schöllkopf)

In Scikit-Learn:

```
>>> from sklearn.svm import OneClassSVM
>>> X = [[0], [0.44], [0.45], [0.46], [1]]
>>> clf = OneClassSVM(gamma='auto').fit(X)
>>> clf.predict(X)
array([-1,  1,  1,  1, -1])
>>> clf.score_samples(X)  # doctest: +ELLIPSIS
array([1.7798..., 2.0547..., 2.0556..., 2.0561..., 1.7332...])
```

https://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html

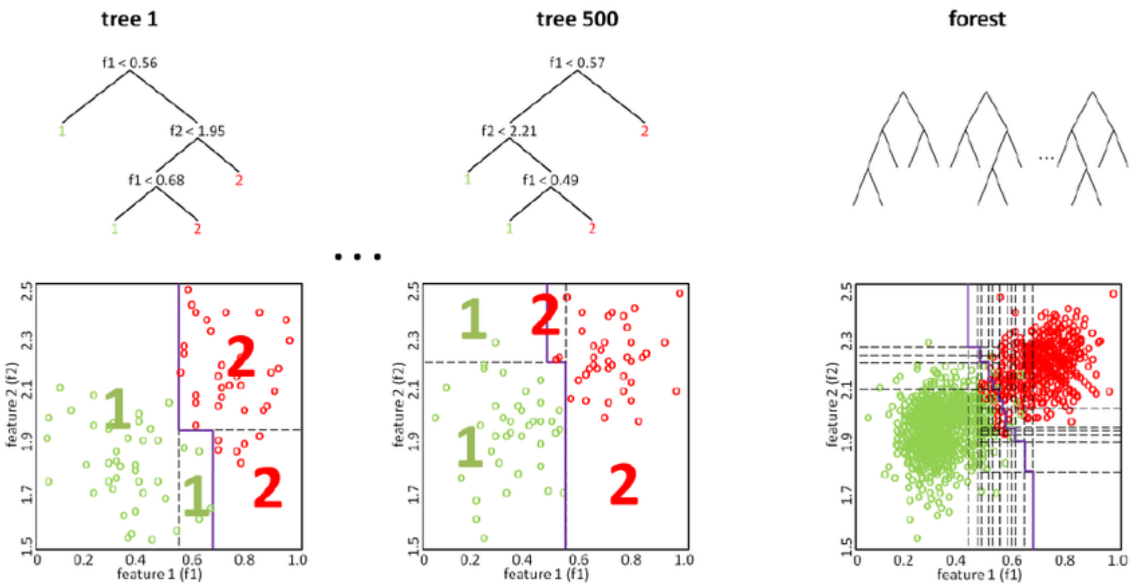


Outliers with Machine Learning

Using Isolation Forests [Recall Random Forests from ML II in Block 1]

Random Forests:

- Ensemble of simple decision trees
- Classification as voting
- Combination of piecewise linear models

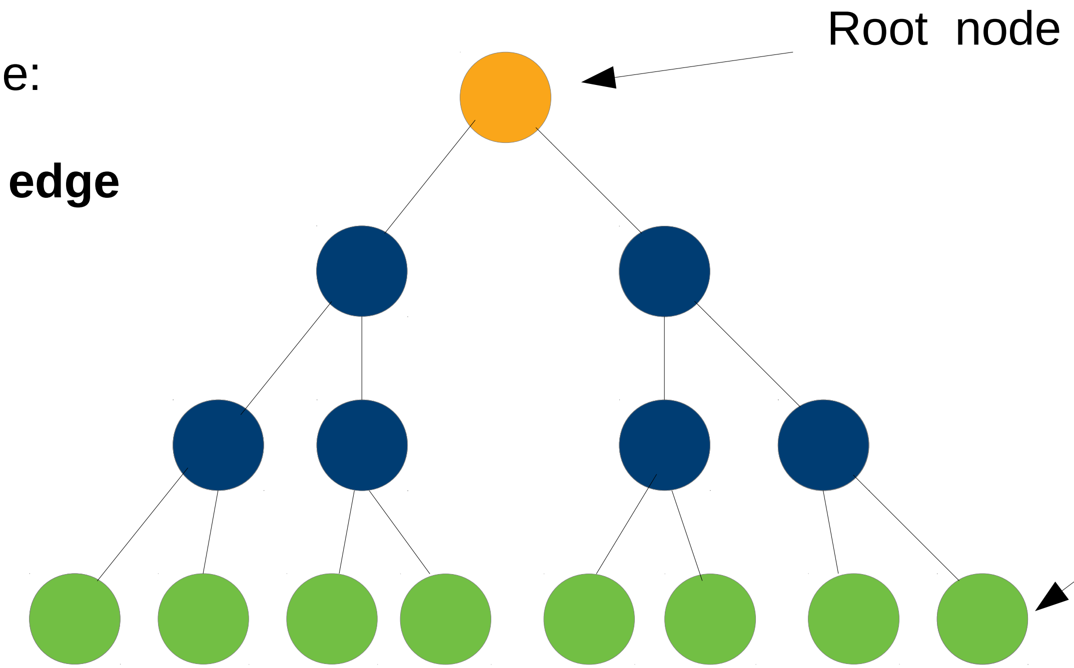


Random Forests

Recall Decision Trees: the base classifier for Random/Isolation Forests

Typically Binary Tree:

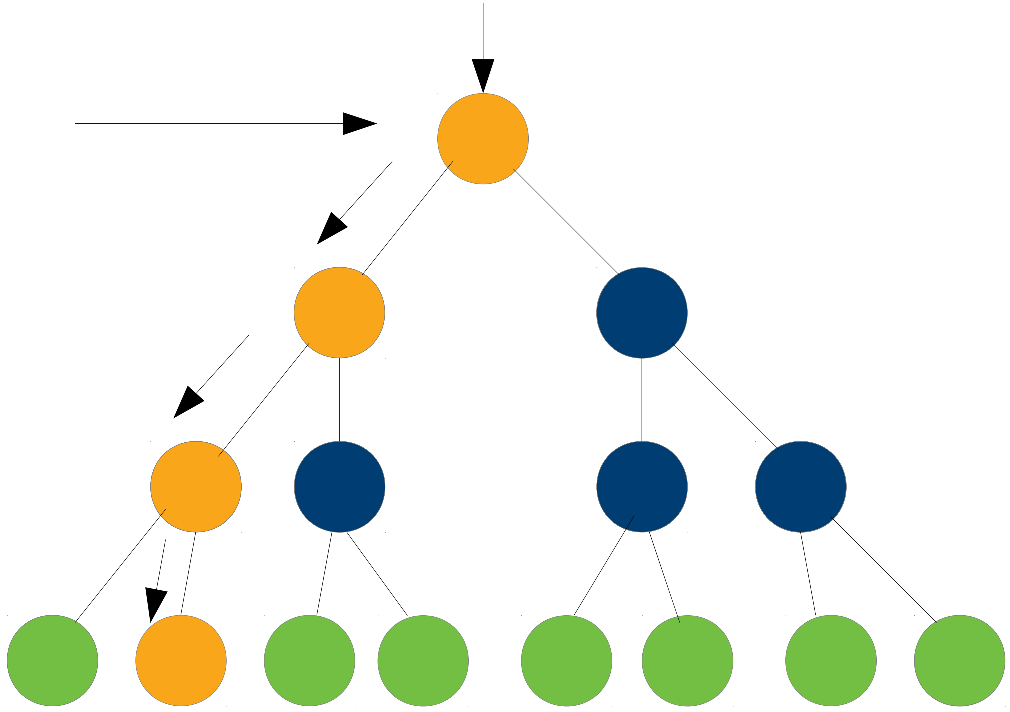
Vertex (Node) and edge



Random Forests

Isolation Forest, just like RF, BUT

Split: random
feature at random
value [min,max]



Split till leaf
contains single
sample

Isolation Forest

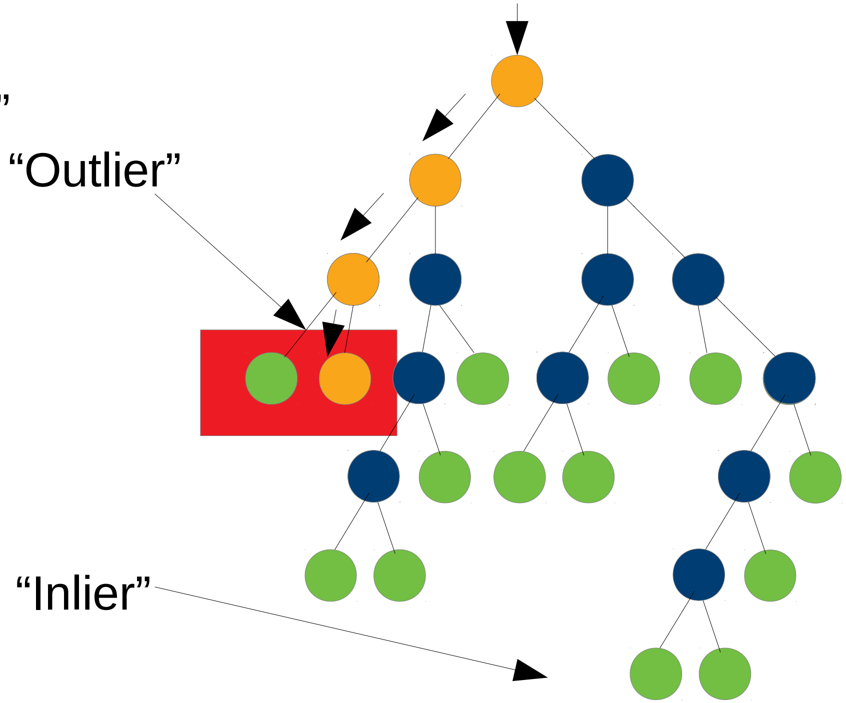
Effect: trees become much deeper

Depth of as sample as “of out of distribution” Measure:

$$s(x,n) = 2^{-\frac{E(h(x))}{c(n)}}$$

where h(x) is the path length of observation x, c(n) is the average path length of unsuccessful search in a Binary Search Tree and n is the number of external nodes.

→ 1 outlier, 0 inlier

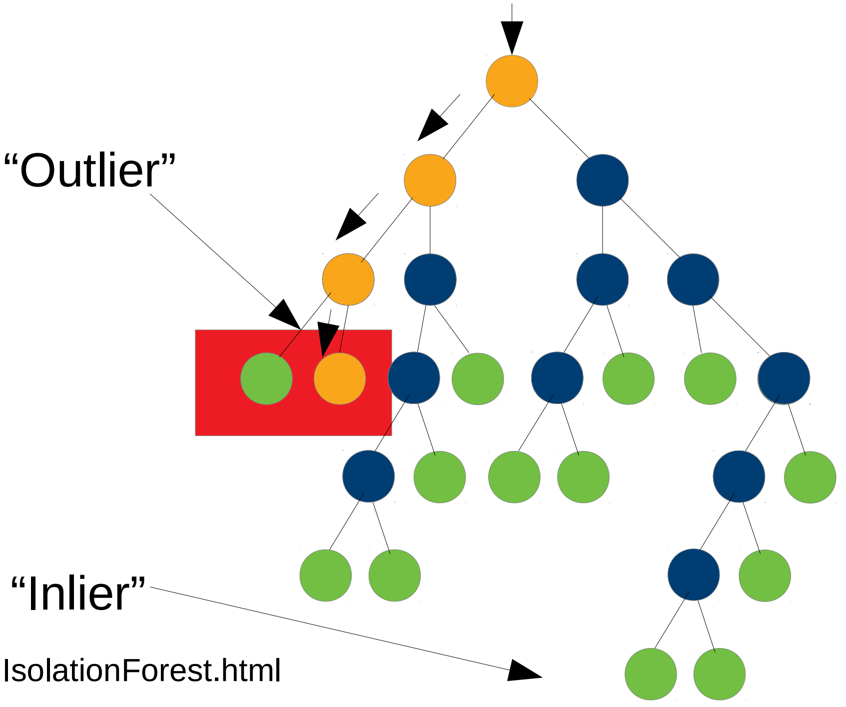


Random Forests

Isolation Forest

In Scikit-Learn:

```
>>> from sklearn.ensemble import IsolationForest
>>> X = [[-1.1], [0.3], [0.5], [100]]
>>> clf = IsolationForest(random_state=0).fit(X)
>>> clf.predict([[0.1], [0], [90]])
array([ 1,  1, -1])
```

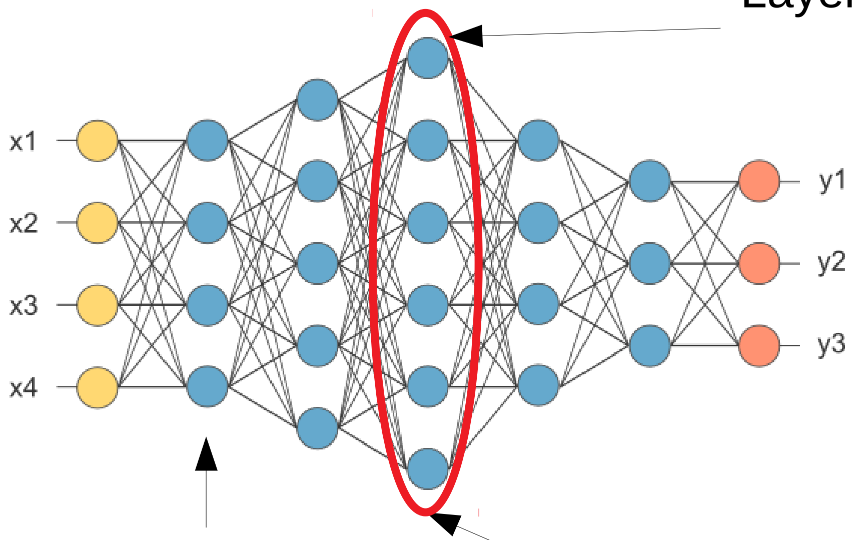


https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html

Outliers with Neural Networks

Using Auto Encoders

“Layer of the Network



$\phi_1(W_1)$

$\phi_3(W_3)$

Training via SGD optimization of

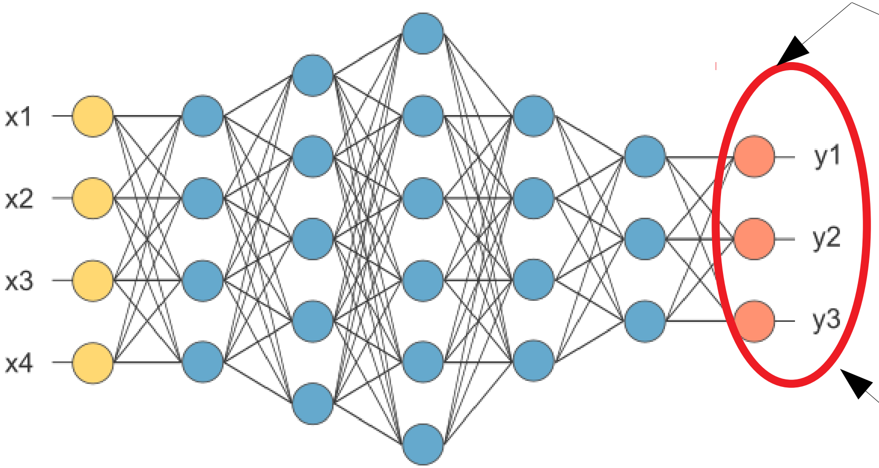
$$\arg\min_{w} \sum_{i=0}^{N} L(y_i, f(x)) + \lambda \sum_j w_j^2$$

$$f(x) = \phi_3(w_3 \phi_2(W_2 \phi_1(W_1 x)))$$

Outliers with Neural Networks

Using Auto Encoders [recall Neural Networks from ML IV in Block 2]

Classification via SOFTMAX over given labels



Training via SGD optimization of

$$\arg\min_{w} \sum_{i=0}^{N} L(y_i, f(x)) + \lambda \sum_j w_j^2$$

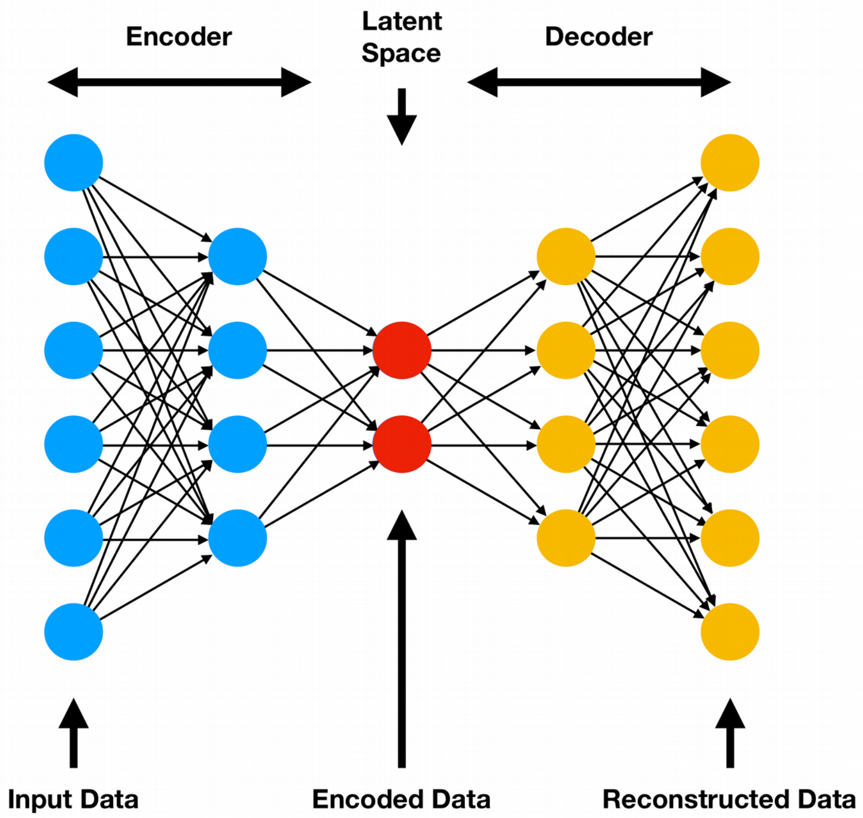
$$\text{softmax}(y)_i = \frac{\exp(y_i)}{\sum_j \exp(y_j)}$$

Outliers with Neural Networks

Auto Encoders Networks

Basic Version:

- MLP (fully connected or matrix multiplication network)

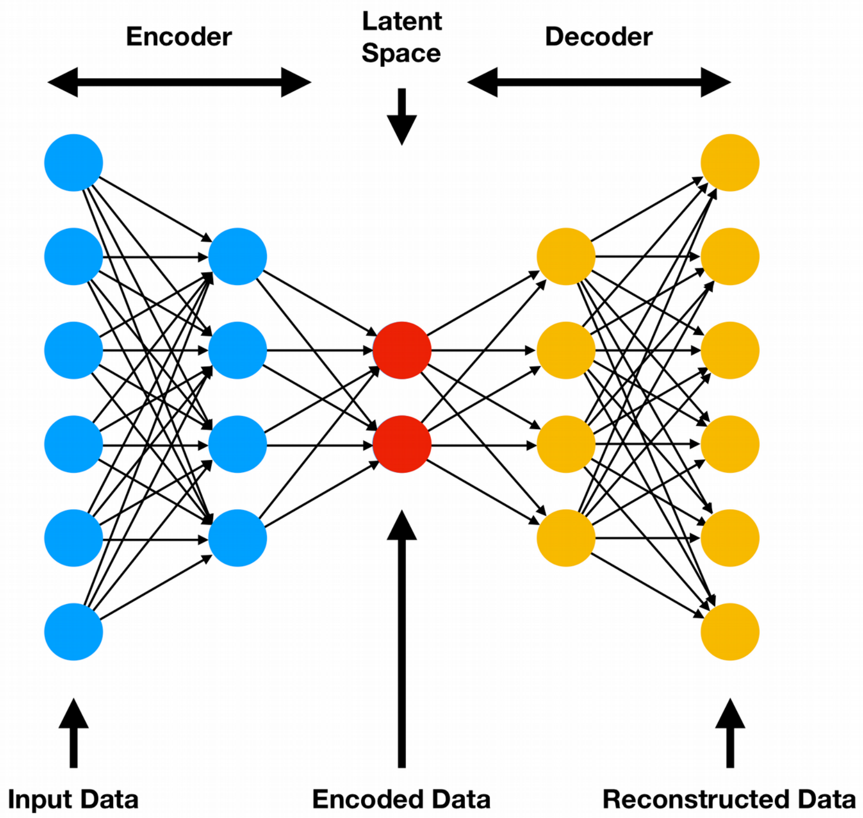


Outliers with Neural Networks

Auto Encoders Networks

Basic Version:

- MLP (fully connected or matrix multiplication network)
- Output Size = Input Size

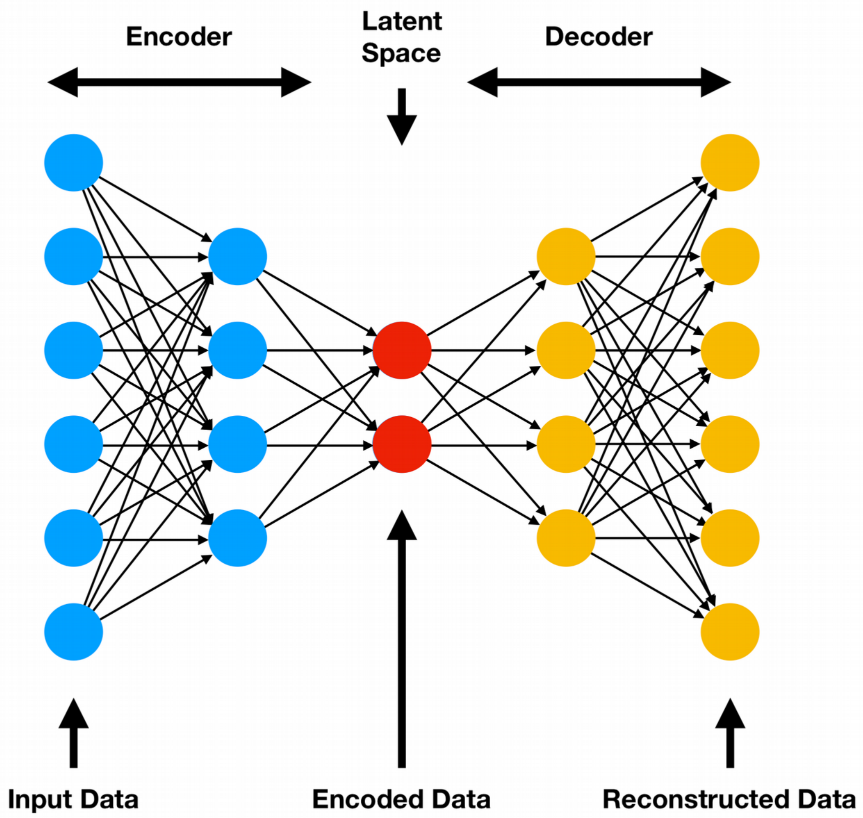


Outliers with Neural Networks

Auto Encoders Networks

Basic Version:

- MLP (fully connected or matrix multiplication network)
- Output Size = Input Size
- Latent Space “Bottleneck”



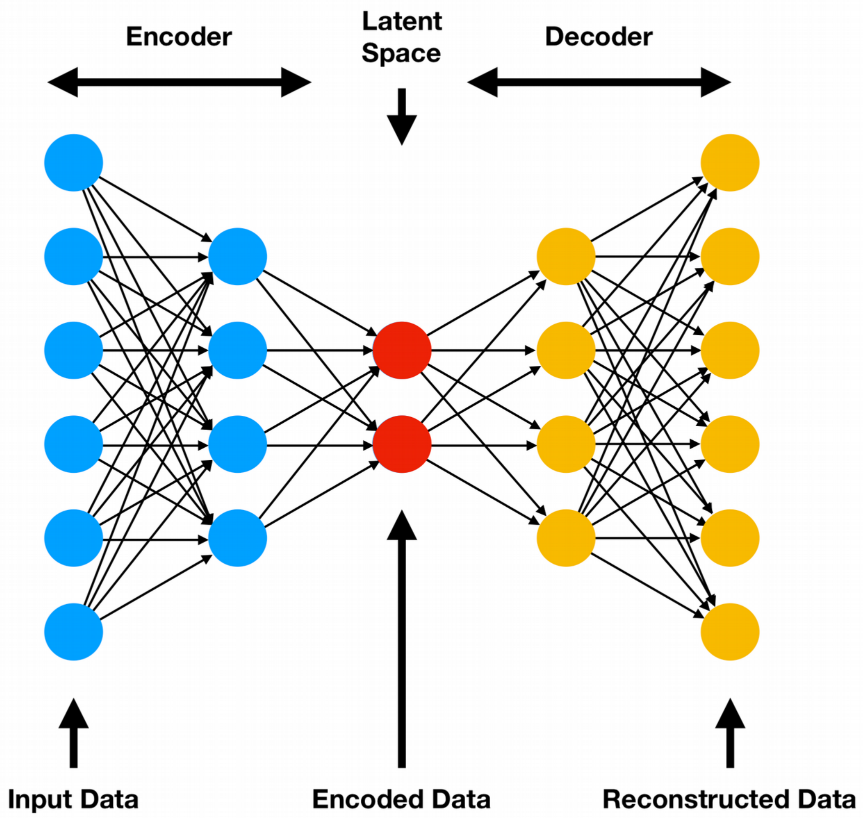
Outliers with Neural Networks

Auto Encoders Networks

Basic Version:

- MLP (fully connected or matrix multiplication network)
- Output Size = Input Size
- Latent Space “Bottleneck”
- Loss: Out put should equal Input simple choice:

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}$$



Outliers with Neural Networks

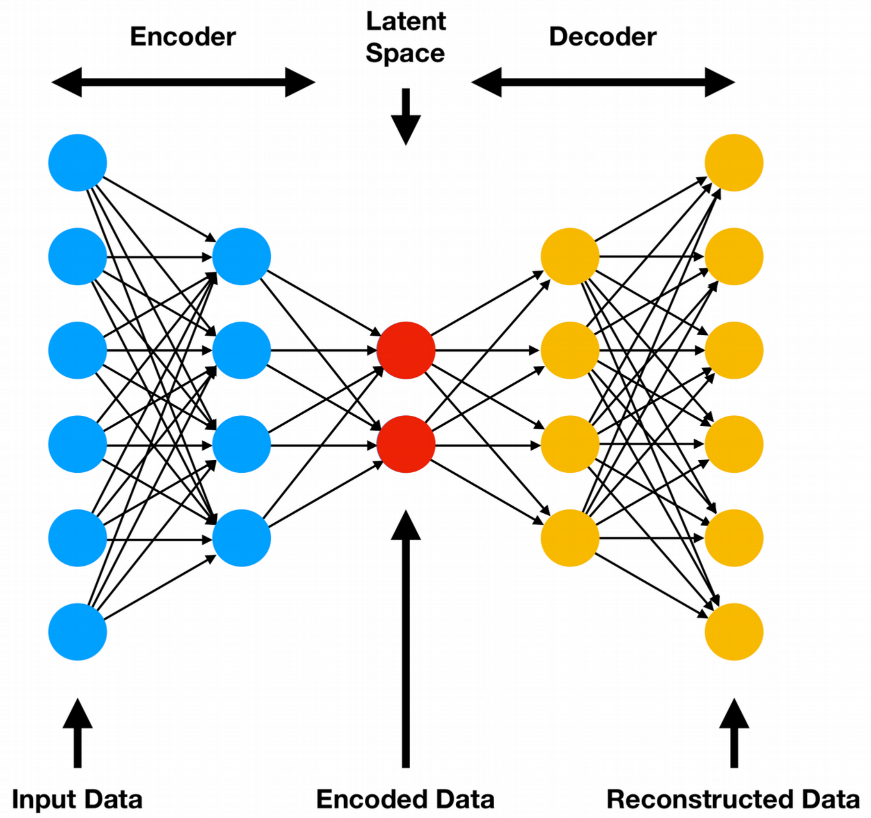
Hochschule Offenburg
offenburg.university

Auto Encoders Networks

Intuitive explanation:

- AE is learning how to “compress” data → compact latent space

Why is this helpful for outlier detection ?



Outliers with Neural Networks

Hochschule Offenburg
offenburg.university

Auto Encoders Networks

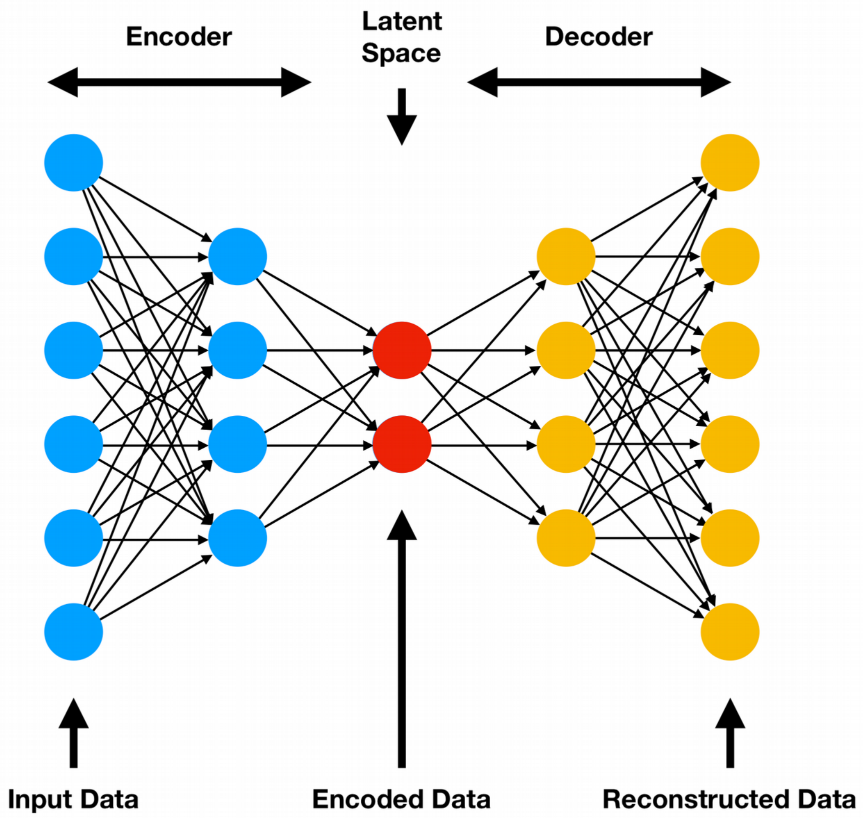
Intuitive explanation:

- AE is learning how to “compress” data → compact latent space

Why is this helpful for outlier detection ?

→ AE can NOT reconstruct out of distribution samples!

→ i.e. used for image de-noising



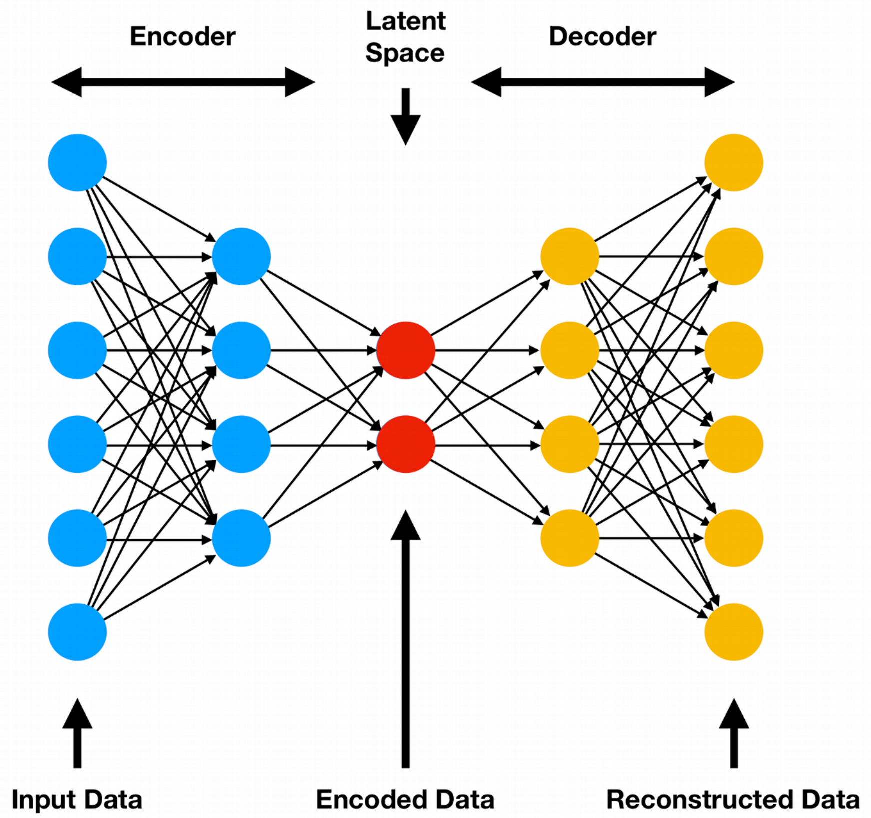
Outliers with Neural Networks

Detect outliers

Train on “normal” data

Feed test data through network
→ compute reconstruction error

Apply Threshold on error → outlier



Discussion