pip install yano

Simon Kluettermann

ls9 tu Dortmund

24. Mai 2022

Simon Kluettermann

Problem

- Paper with Benedikt
- require multiple very specific datasets
  - many but not to many features
  - at least some samples (for the NN)
  - Only numerical attributes best
  - specific quality
  - unrelated datasets
- Requires you to search for many datasets and filter them

Students

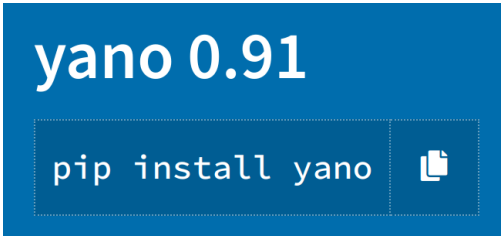
- Not clear what you can use
- Many different formats
- train/test splits
- So for Students I just do this work and send them archives directly
- $\Rightarrow$Not a good solution

yano

- So I have been packaging all my scripts
- I had surprisingly much fun doing this
  - More than just standard functions
  - A couple of weird decisions
  - And this will likely grow further
- ⇒So I would like to discuss some parts with you and maybe you even have more features you might want

yano

- Simply install it over pip
- Contains 187 real-World Datasets
- ⇒biggest library of datasets explicitely for anomaly detection
- not yet happy with this
- especially only mostly contains numerical and nominal attributes
- ⇒few categorical and no time-series attributes



selector

```
import yano
from yano.symbols import *
condition= (number_of_features>5) &
           (number_of_features<100) &
           (number_of_samples>100) &
           (number_of_samples<10000) &
           (number_of_samples>2*number_of_features) &
           ~index
print(len(condition), "Datasets found")
```

⇒33 Datasets found

selectors

- Lots of symbols like this
  - name
  - number_of_features
  - number_of_samples
  - index (correlated datasets)
- Feature types
  - numeric
  - nominal
  - categorical
  - (textual)
- Count based
  - number_anomalies
  - number_normals
  - fraction_anomalies
- Specific ones
  - image_based
  - (linearly_seperable)

iterating

```
for dataset in condition:
    print(condition)
```

- *annthyroid*
- *breastw*
- *cardio*
- ...
- *Housing_low*

iterating

```
for dataset in condition:
    x=dataset.getx()
    y=dataset.gety()
```

pipeline

```
from yano.iter import *
for dataset, x,tx,ty in pipeline(condition,
                                 split,
                                 shuffle,
                                 normalize("minmax")):
    ...
```

pipeline

- Again there are a couple modifiers possible
  - nonconst⇒remove constant features
  - shuffle
  - normalize('zscore'/'minmax')
  - cut(10)⇒at most 10 datasets
  - split⇒train test split, all anomalies in test set
  - crossval(5)⇒similar to split, but do multiple times (crossvalidation)
- modifiers interact with each other
- For example: normalize('minmax'), split
- ⇒train set always below 1, but no guarantees for the test set

CrossValidation

- Learned from DMC: Crossvalidation is important
- Rarely found in Anomaly Detection, why?
- A bit more complicated (not all samples are equal), but no reason why not
- ⇒So I implemented it into yano
  - folding only on normal data
  - How to handle anomalies?
  - If not folding them, cross-validation less useful
  - if folding them, often rare anomalies even more rare
  - ⇒test set always 50% anomalous
  - ⇒Also improves simple evaluation metrics (accuracy)
- Do you know a reason why Cross Validation is not common in AD?
- Are there Problems with the way I fold my Anomalies?

Logging

```
from yano.logging import Logger
from pyod.models.iforest import IForest
from extended_iforest import train_extended_ifor
l=Logger({"IFor":IForest(n_estimators=100),
          "eIFor":train_extended_ifor})
for dataset, folds in pipeline(condition,
                               crossval(5),
                               normalize("minmax"),
                               shuffle):
    l.run_cross(dataset, folds)
latex=l.to_latex()
```

Seeding

- If you dont do anything, everything is seeded.
- Makes rerunning a Model until the performance is good quite obvious.
- But as every Run is seeded itself, this might induce bias.
- Do you think this is worth it?
- Are there any Problems with this?

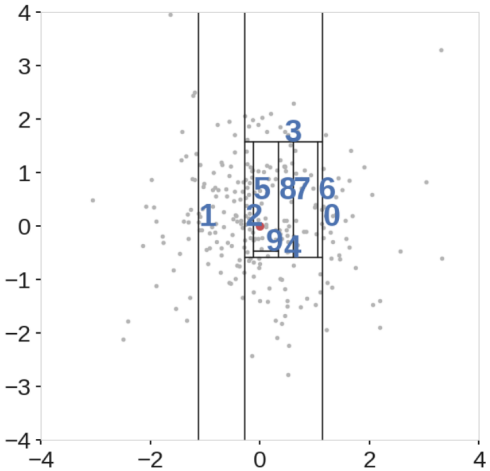
Dataset	eIFor	IFor
pc3	**0.7231** $\pm$ 0.0153	**0.7223** $\pm$ 0.0178
pima	**0.7405** $\pm$ 0.0110	**0.7347** $\pm$ 0.0126
Diabetes_present	**0.7414** $\pm$ 0.0195	**0.7344** $\pm$ 0.0242
waveform $-$ 5000	**0.7687** $\pm$ 0.0123	**0.7592** $\pm$ 0.0206
vowels	**0.7843** $\pm$ 0.0298	**0.7753** $\pm$ 0.0334
Vowel_0	**0.8425** $\pm$ 0.0698	0.7193 $\pm$ 0.0817
Abalone_1_8	**0.8525** $\pm$ 0.0263	0.8452 $\pm$ 0.0257
annthyroid	0.8399 $\pm$ 0.0135	**0.9087** $\pm$ 0.0090
Vehicle_van	**0.8792** $\pm$ 0.0265	**0.8697** $\pm$ 0.0383
ionosphere	**0.9320** $\pm$ 0.0069	0.9086 $\pm$ 0.0142
breastw	**0.9948** $\pm$ 0.0031	**0.9952** $\pm$ 0.0033
segment	**1.0**	**0.9993** $\pm$ 0.0015
Average	**0.8005**	**0.7957**

statistics

- Friedman test to see if there is a difference between models
- Nemenyi test to see which models are equal, mark those equal to the maximum
- For 2 models, Friedman not defined $\Rightarrow$ use Wilcoxon test
- Does this match your expectation from the table?
- Two models are 'equal' if their probability of being from the same distribution is $p_b \leq p$, what value should $p_b = 0.1$ have?
- Do I need to correct for p hacking (n experiments, so increase the difficulty for each, or is that clear from the table)

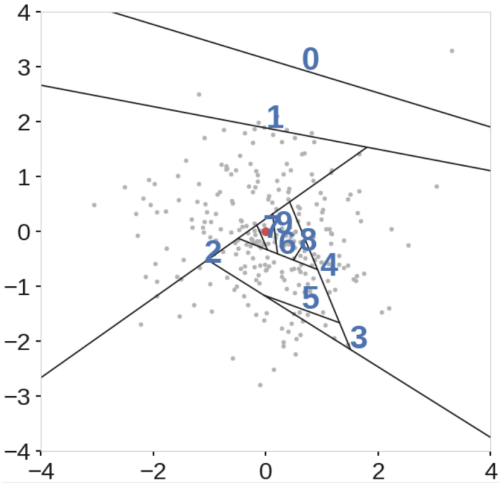
Extended Isolation Forests

- Isolation Forests are one algorithm for AD
- Tries to isolate abnormal (rare) points instead of modelling normal ones
- Creative approach⇒fairly successful (3000 Citations)
- Many follow up papers
- Extended Isolation Forest (Hariri et. al. 2018, 140 Citations)
- Remove bias from the Isolation Forests
- Also claim to improve their anomaly detection quality



Extended Isolation Forests

- Isolation Forests are one algorithm for AD
- Tries to isolate abnormal (rare) points instead of modelling normal ones
- Creative approach⇒fairly successful (3000 Citations)
- Many follow up papers
- Extended Isolation Forest (Hariri et. al. 2018, 140 Citations)
- Remove bias from the Isolation Forests
- Also claim to improve their anomaly detection quality



Extended Isolation Forests

- Isolation Forests are one algorithm for AD
- Tries to isolate abnormal (rare) points instead of modelling normal ones
- Creative approach⇒fairly successful (3000 Citations)
- Many follow up papers
- Extended Isolation Forest (Hariri et. al. 2018, 140 Citations)
- Remove bias from the Isolation Forests
- Also claim to improve their anomaly detection quality

Table 3: AUC values for both ROC and PRC for benchmark datasets using standard Isolation Forest and Extended Isolation Forest

	AUC ROC		AUC PRC	
Data	iForest	EIF	iForest	EIF
Cardio	0.888	0.915	0.466	0.483
ForestCover	0.809	0.924	0.430	0.504
Ionosphere	0.85	0.913	0.877	0.893
Mammography	0.859	0.862	0.4198	0.4271
Satellite	0.714	0.778	0.783	0.808

Dataset	eIFor	IFor
Delft_pump_5x3_noisy	**0.3893** $\pm$ 0.0345	**0.4272** $\pm$ 0.0680
vertebral	**0.4260** $\pm$ 0.0111	**0.4554** $\pm$ 0.0416
Liver_1	0.5367 $\pm$ 0.0508	**0.5474** $\pm$ 0.0541
Sonar_mines	**0.6882** $\pm$ 0.1264	0.6189 $\pm$ 0.1301
letter	**0.6756** $\pm$ 0.0119	0.6471 $\pm$ 0.0111
Glass_building_float	**0.6480** $\pm$ 0.1012	**0.6755** $\pm$ 0.1117
pc3	**0.7231** $\pm$ 0.0153	**0.7223** $\pm$ 0.0178
pima	**0.7405** $\pm$ 0.0110	**0.7347** $\pm$ 0.0126
Diabetes_present	**0.7414** $\pm$ 0.0195	**0.7344** $\pm$ 0.0242
waveform $-$ 5000	**0.7687** $\pm$ 0.0123	**0.7592** $\pm$ 0.0206
steel $-$ *plates* $-$ *fault*	**0.7735** $\pm$ 0.0351	**0.7682** $\pm$ 0.0402
vowels	**0.7843** $\pm$ 0.0298	**0.7753** $\pm$ 0.0334

Dataset	eIFor	IFor
Vowel_0	**0.8425** $\pm$ 0.0698	0.7193 $\pm$ 0.0817
Housing_low	**0.7807** $\pm$ 0.0333	**0.7862** $\pm$ 0.0336
ozone − level − 8hr	**0.7904** $\pm$ 0.0207	**0.7768** $\pm$ 0.0118
Spectf_0	**0.8155** $\pm$ 0.0255	0.7535 $\pm$ 0.0239
HeartC	0.7795 $\pm$ 0.0258	**0.8079** $\pm$ 0.0255
satellite	**0.8125** $\pm$ 0.0170	**0.8103** $\pm$ 0.0061
optdigits	**0.8099** $\pm$ 0.0310	**0.8142** $\pm$ 0.0267
spambase	**0.8085** $\pm$ 0.0110	**0.8202** $\pm$ 0.0042
Abalone_1_8	**0.8525** $\pm$ 0.0263	0.8452 $\pm$ 0.0257
qsar − biodeg	**0.8584** $\pm$ 0.0119	**0.8628** $\pm$ 0.0135
annthyroid	0.8399 $\pm$ 0.0135	**0.9087** $\pm$ 0.0090
Vehicle_van	**0.8792** $\pm$ 0.0265	**0.8697** $\pm$ 0.0383

Dataset	eIFor	IFor
ionosphere	**0.9320** $\pm$ 0.0069	0.9086 $\pm$ 0.0142
page − blocks	0.9189 $\pm$ 0.0061	**0.9299** $\pm$ 0.0016
Ecoli	**0.9418** $\pm$ 0.0292	0.9192 $\pm$ 0.0332
cardio	**0.9564** $\pm$ 0.0043	**0.9535** $\pm$ 0.0036
wbc	**0.9611** $\pm$ 0.0121	**0.9607** $\pm$ 0.0107
pendigits	**0.9641** $\pm$ 0.0097	**0.9652** $\pm$ 0.0076
thyroid	0.9818 $\pm$ 0.0024	**0.9871** $\pm$ 0.0025
breastw	**0.9948** $\pm$ 0.0031	**0.9952** $\pm$ 0.0033
segment	**1.0**	**0.9993** $\pm$ 0.0015
Average	**0.8005** $\pm$ 0.1458	**0.7957** $\pm$ 0.1431

highdim

- **High Dimensional Data**: one of the main limitation to standard, distance-based methods is their inefficiency in dealing with high dimensional datasets:.[10] The main reason for that is, in a high dimensional space every point is equally sparse, so using a distance-based measure of separation is pretty ineffective. Unfortunately, high-dimensional data also affects the detection performance of iForest, but the performance can be vastly improved by adding a features selection test like Kurtosis to reduce the dimensionality of the sample space.[1][5]

New Condition

```
condition= (number_of_samples>200) &
           (number_of_samples<10000) &
           (number_of_features>50) &
           (number_of_features<500) &
           ~index
print(len(condition),"Datasets found")
```

⇒13 Datasets found

New Models

```
from pyod.models.iforest import IForest
from pyod.models.knn import KNN
from pyod.models.lof import LOF
l=Logger({"IFor":Iforest(n_estimators=100),
          "Lof":LOF(),
          "Knn": KNN()}, addfeat=True)
```

Dataset	Knn	Lof	IFor
Delft_pump_5x3_noisy(64)	0.3800 ± 0.0475	0.3462 ± 0.0327	**0.4272** ± 0.0680
hill − valley(100)	0.4744 ± 0.0269	**0.5060** ± 0.0327	0.4720 ± 0.0288
speech(400)	0.4903 ± 0.0103	**0.5104** ± 0.0115	0.4872 ± 0.0184
Sonar_mines(60)	**0.7284** ± 0.0939	0.6769 ± 0.0933	0.6189 ± 0.1301
ozone − level − 8hr(72)	**0.8051** ± 0.0288	0.7738 ± 0.0292	**0.7768** ± 0.0118
spambase(57)	0.8038 ± 0.0125	0.7712 ± 0.0055	**0.8202** ± 0.0042
arrhythmia(274)	**0.8137** ± 0.0185	0.8042 ± 0.0186	**0.8086** ± 0.0099
mnist(100)	0.9345 ± 0.0039	**0.9548** ± 0.0037	0.8732 ± 0.0069
Concordia3_32(256)	0.9246 ± 0.0107	**0.9486** ± 0.0099	**0.9322** ± 0.0178
optdigits(64)	0.9966 ± 0.0012	**0.9975** ± 0.0012	0.8142 ± 0.0267
gas − drift(128)	**0.9790** ± 0.0018	0.9585 ± 0.0055	0.8764 ± 0.0166
Delft_pump_AR(160)	**0.9965**	**0.9953** ± 0.0019	0.9665 ± 0.0096
musk(166)	**1.0**	**1.0**	0.9808 ± 0.0117
Average	**0.7944**	**0.7879**	0.7580

- Hypothesis: Isolation Forests are better when there are numerical and nominal attributes
- Easy to test

```
condition=condition & (numeric & nominal)
```

Dataset	Knn	IFor	Lof
ozone − *level* − 8*hr*(72)	**0.8051** ± 0.0288	**0.7768** ± 0.0118	0.7738 ± 0.0292
spambase(57)	0.8038 ± 0.0125	**0.8202** ± 0.0042	0.7712 ± 0.0055
arrhythmia(274)	**0.8137** ± 0.0185	**0.8086** ± 0.0099	0.8042 ± 0.0186
musk(166)	**1.0**	0.9808 ± 0.0117	**1.0**
Average	**0.8556**	**0.8466**	**0.8373**

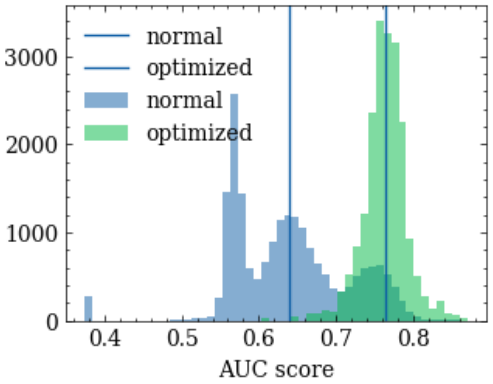
- Only 4 datasets, so not clear at all
- ⇒More datasets

Unsupervised Optimization

- There are analysis that are only possible with many datasets
- Here: unsupervised optimization
- Given multiple AD models, find which is best:
- Use AUC score? Requires Anomalies⇒Overfitting
- Can you find an unsupervised Method?
- In general very complicated, so here only focus on very small differences in the model.
- So each model is an autoencoder, trained on the same dataset, where the difference is only in the initialisation

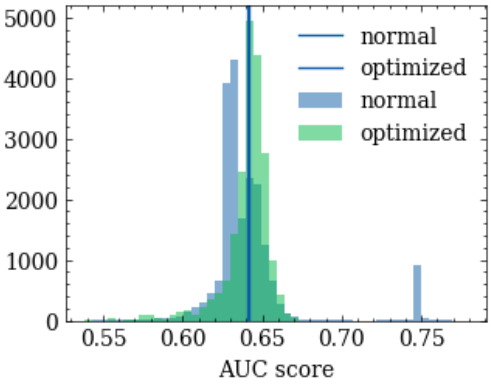
Loss Optimization

- First guess Loss of the Model on the training Data
- How to evaluate this?
- Train many models, look at the average AUC score.
- For the alternative, take groups of 20 models, and look at the AUC score of the best model.
- Is there a meaningfull difference between results? Give result as z_score $\left(\frac{m_1 - m_2}{\sqrt{s_1^2 + s_2^2}}\right)$
- This difference depends a lot on the dataset
- $\Rightarrow$even $30 \leq z$ does not mean much



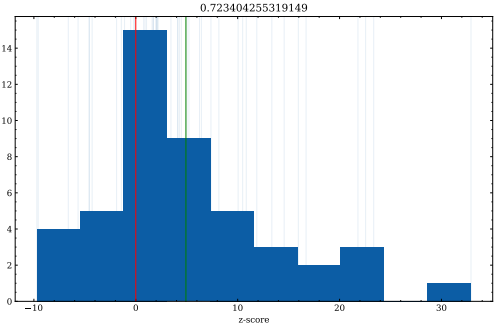
Loss Optimization

- First guess Loss of the Model on the training Data
- How to evaluate this?
- Train many models, look at the average AUC score.
- For the alternative, take groups of 20 models, and look at the AUC score of the best model.
- Is there a meaningfull difference between results? Give result as z_score $\left(\frac{m_1 - m_2}{\sqrt{s_1^2 + s_2^2}}\right)$
- This difference depends a lot on the dataset
- $\Rightarrow$even $30 \leq z$ does not mean much



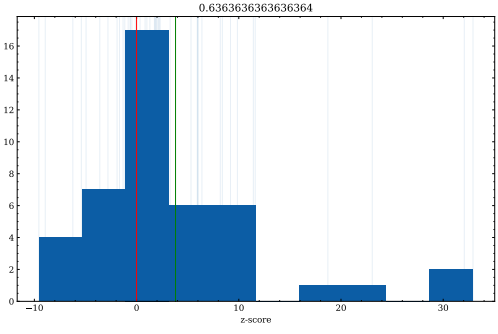
loss

- Pick the Model with the lowest l2loss



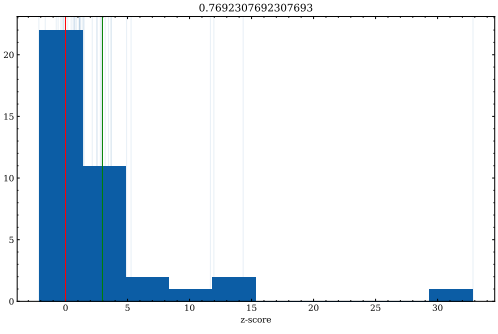
Robustness

- Pick points with 1% width difference in input space around each point.
- for each point, find the maximum difference in output space.
- average this difference



Distance Correlation

- Pick random points in the input space.
- measure the distance in input and output space
- a low correlation is a good model



Other

- Things I still want to add:
  - Ensemble Methods
  - Visualisation options
  - Alternative Evaluations
  - Hyperparameter optimisation (with crossvalidation)
  - Parallelisation
  - Contamination
  - Dokumentation

Feedback

- What do you think about this?
- Is there something I should also add?
- What would you need for you to actually use this?