Probability density learning for heterogeneous tree structured data

Václav Šmídl[1], Tomáš Pevný[1], Milan Papež[1], et. al.

[1] AI Center, FEL, CTU, Prague,
[2] RICE, FEL, UWB, Pilsen,

February 10, 2023

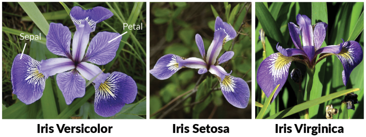
Motivation: tree structured data

Classical machine learning methods:
Consider data in the form of d-dimensional vectors $x \in \mathbb{R}^d$.

Classifiers:

- Random Forest
- SVM
- Neural networks...

Classify Iris flowers:



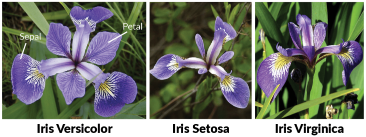
Motivation: tree structured data

Classical machine learning methods: Consider data in the form of d-dimensional vectors $x \in \mathbb{R}^d$.

Classifiers:

- Random Forest
- SVM
- Neural networks...

Classify Iris flowers:



Feature engineering 4d vector:

1. Sepal length
2. Sepal width
3. Petal length
4. Petal width

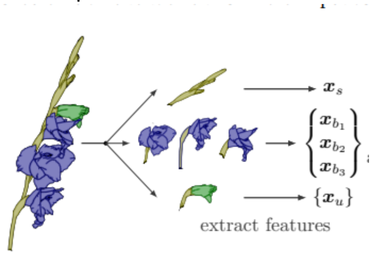
Motivation: tree structured data

Classical machine learning methods: Consider data in the form of d-dimensional vectors $x \in \mathbb{R}^d$.

Classifiers:

- Random Forest
- SVM
- Neural networks...

Natural parametrization:



Classify Iris flowers:

Iris Versicolor Iris Setosa Iris Virginica

Feature engineering 4d vector:

1. Sepal length
2. Sepal width
3. Petal length
4. Petal width

- Which leaf to choose?
- Or average them?
- Does their count matter?

Tree-structured data: examples

Hierarchical tree structured data = special case of graph data composed of three types of nodes:

1. Leafs: Scalar/Vector/Tensor
2. Dicts: key-value pairs
3. Lists: arbitrary length

Mutagenesis

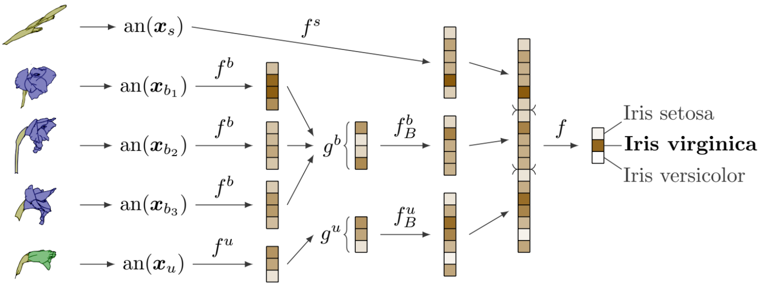
data set, description of molecules in json-like format.

```
[Dict]
  ├── lumo: [Scalar - Float64]
  ├── inda: [Scalar - Int64]
  ├── logp: [Scalar – Float64, Int64]
  ├── ind1: [Scalar - Int64]
  └── atoms: [List]
              └── [Dict]
                    ├── element: [Scalar - String]
                    ├── bonds: [List]
                    │           └── [Dict]
                    │                 ├── element: [Scalar - String]
                    │                 ├── bond_type: [Scalar - Int64]
                    │                 ├── charge: [Scalar - Float64]
                    │                 └── atom_type: [Scalar - Int64]
                    ├── charge: [Scalar - Float64]
                    └── atom_type: [Scalar - Int64]
```

Discriminative learning: HMIL

Hierarchical Multi-instance Learning[1] – no message passing

- hierarchical application of NN projection (Leafs,Lists), aggregation (Lists), and concatenation (Dict)



- Highly automated to handle various data types, missing data, etc.[2]
- Clustering? We do not have a metric neither probability distribution (likelihood).

[1] Pevny, T. and Somol, P., 2016, October. Discriminative models for multi-instance problems with tree structure. In Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security (pp. 83-91).

[2] Mandlík, Š., Račinský, M., Lisý, V. and Pevný, T., 2022. JsonGrinder. jl: automated differentiable neural architecture for embedding arbitrary JSON data. Journal of Machine Learning Research, 23(298), pp.1-5.

Global goal: density learning

Tools of probability for JSON structure:

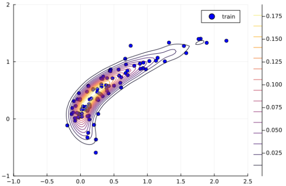
- Leaf: probability density of vector data, $p(x)$
- Dict: joint probability density, $p(a, b, c)$
- List: random set theory, $p(X)$, $X = \{x_1, x_2 \ldots, x_n\}$

Challenges:

1. How to represent **Leafs**?
   - many types! compact representation
2. Dependent or independent **Dict**?
   - incomplete data, discrete data
3. Proper treatment of cardinality in **Lists**?

Roadmap: probability learning

Vector data



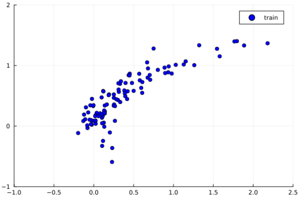
Set data

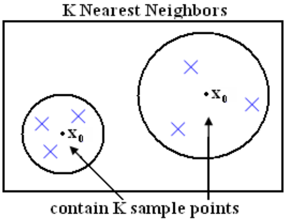
Full Hierarchy

```
[Dict]
├── lumo: [Scalar - Float64]
├── inda: [Scalar - Int64]
├── logp: [Scalar – Float64, Int64]
├── ind1: [Scalar - Int64]
└── atoms: [List]
           └── [Dict]
                 ├── element: [Scalar - String]
                 ├── bonds: [List]
                 │         └── [Dict]
                 │               ├── element: [Scalar - String]
                 │               ├── bond_type: [Scalar - Int64]
                 │               ├── charge: [Scalar - Float64]
                 │               └── atom_type: [Scalar - Int64]
                 ├── charge: [Scalar - Float64]
                 └── atom_type: [Scalar - Int64]
```

Comparing vector data density models

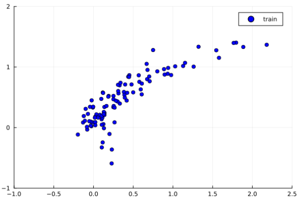
1. Classical GMM, $p(x) = \sum_{i=1}^{k} w_i \mathcal{N}(\mu, \Sigma)$

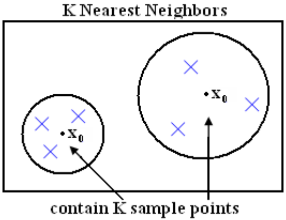




Comparing vector data density models

1. Classical GMM, $p(x) = \sum_{i=1}^{k} w_i \mathcal{N}(\mu, \Sigma)$
2. Kernel methods (kNN, OC-SVM)



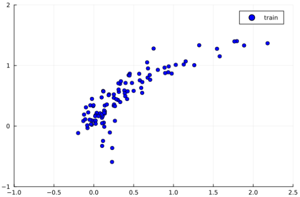


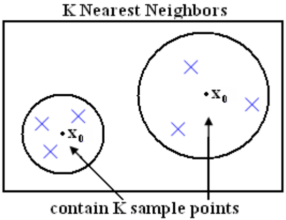
Comparing vector data density models

1. Classical GMM, $p(x) = \sum_{i=1}^{k} w_i \mathcal{N}(\mu, \Sigma)$
2. Kernel methods (kNN, OC-SVM)
3. Flow models, $x = f(z)$, from known $p_z(z)$, via

$$p(x) = p_z(z)|\det J(z)|, \;\; z = f^{-1}(x)$$

   for invertible f. (Special purpose NN: MAF, RNVP).





Comparing vector data density models

1. Classical GMM, $p(x) = \sum_{i=1}^{k} w_i \mathcal{N}(\mu, \Sigma)$
2. Kernel methods (kNN, OC-SVM)
3. Flow models, $x = f(z)$, from known $p_z(z)$, via

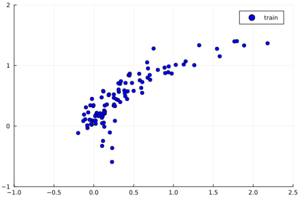
$$p(x) = p_z(z)|\det J(z)|, \;\; z = f^{-1}(x)$$

   for invertible f. (Special purpose NN: MAF, RNVP).

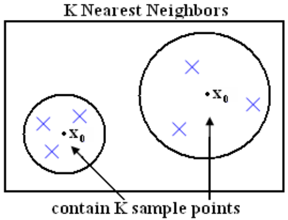
4. Autoencoder-based models, $x = f(z) + e$,

$$p(x) = \int p(x|z)p(z)dz$$

   with inherent dimensionality reduction, $\dim(z) < \dim(x)$. VAE or WAE.



$$x = [z^2, z] + e,$$
$$z \sim \mathcal{N}(0.5, 0.15)$$



Comparing vector data density models

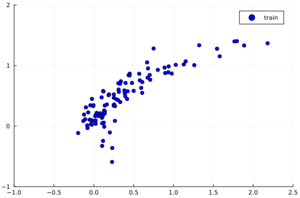
1. Classical GMM, $p(x) = \sum_{i=1}^{k} w_i \mathcal{N}(\mu, \Sigma)$
2. Kernel methods (kNN, OC-SVM)
3. Flow models, $x = f(z)$, from known $p_z(z)$, via

   $$p(x) = p_z(z)|\det J(z)|, \;\; z = f^{-1}(x)$$

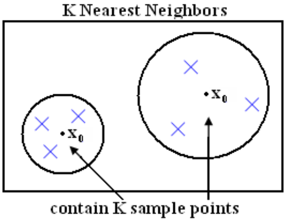
   for invertible f. (Special purpose NN: MAF, RNVP).
4. Autoencoder-based models, $x = f(z) + e$,

   $$p(x) = \int p(x|z)p(z)dz$$

   with inherent dimensionality reduction, $\dim(z) < \dim(x)$. VAE or WAE.
5. Others: two-stage models, GANs,



$$x = [z^2, z] + e,$$
$$z \sim \mathcal{N}(0.5, 0.15)$$



Comparing vector data density models

1. Classical GMM, $p(x) = \sum_{i=1}^{k} w_i \mathcal{N}(\mu, \Sigma)$
2. Kernel methods (kNN, OC-SVM)
3. Flow models, $x = f(z)$, from known $p_z(z)$, via

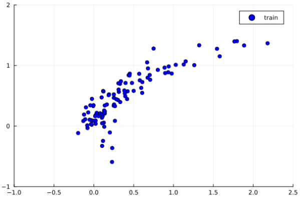
   $$p(x) = p_z(z)|\det J(z)|, \quad z = f^{-1}(x)$$

   for invertible f. (Special purpose NN: MAF, RNVP).
4. Autoencoder-based models, $x = f(z) + e$,

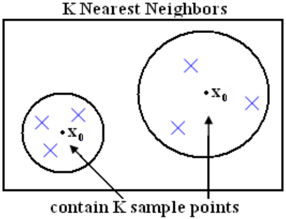
   $$p(x) = \int p(x|z)p(z)dz$$

   with inherent dimensionality reduction, $\dim(z) < \dim(x)$. VAE or WAE.
5. Others: two-stage models, GANs,

Evaluation metric: anomaly detection (out of distribution detection).

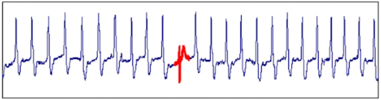


$$x = [z^2, z] + e,$$
$$z \sim \mathcal{N}(0.5, 0.15)$$



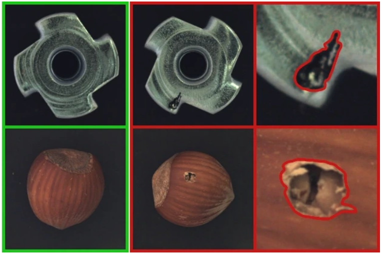
Anomaly / out-of-distribution / novelty detection

Anomaly is sample different from others that it raises suspicion that it was generated by a different process than normal samples.



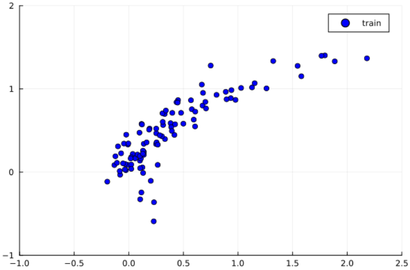
The anomaly is either

- far away from normal samples
  - what is the right distance?
- less likely than normal samples
  - likelihood function?



Anomaly detection process

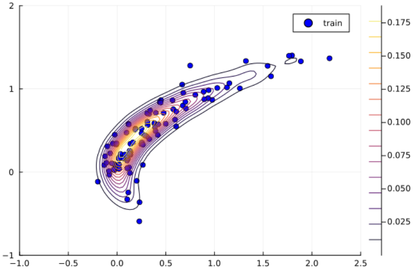
1. Training is done of normal data (no contamination)



Anomaly detection process

1. Training is done of normal data (no contamination)
2. Method should provide an anomaly score, typically negative log-likelihood

$$s(x) = -\log p(x)$$

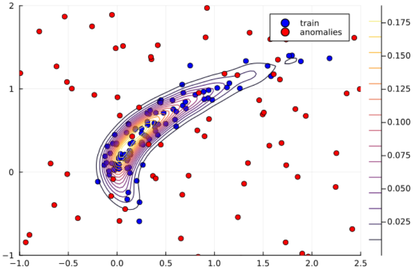


Anomaly detection process

1. Training is done of normal data (no contamination)
2. Method should provide an anomaly score, typically negative log-likelihood

$$s(x) = -\log p(x)$$

3. Test on both normal and anomalous. Score them all.

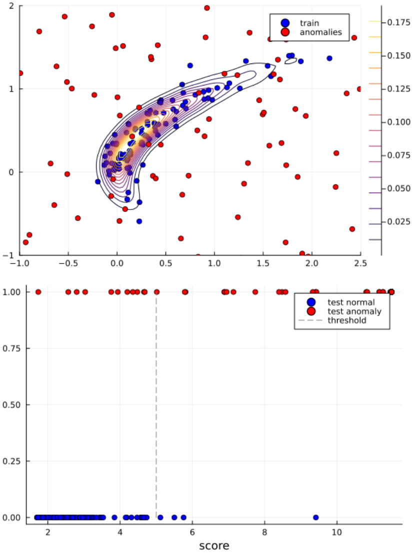


Anomaly detection process

1. Training is done of normal data (no contamination)
2. Method should provide an anomaly score, typically negative log-likelihood

$$s(x) = -\log p(x)$$

3. Test on both normal and anomalous. Score them all.
4. The results are evaluated as binary classification with threshold, AUC

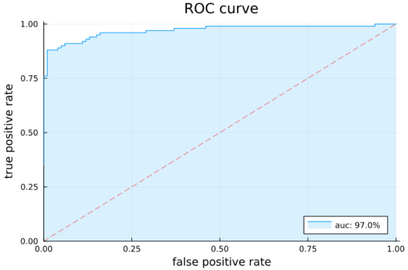


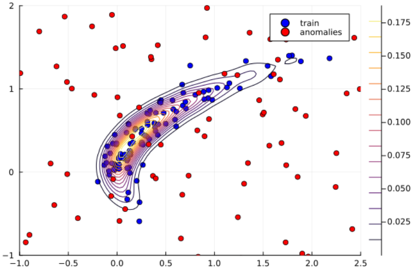
Anomaly detection process

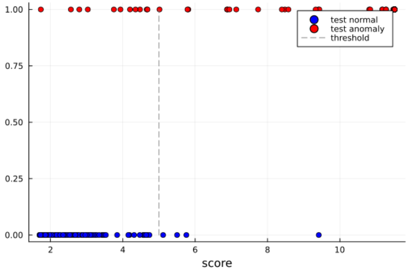
1. Training is done of normal data (no contamination)
2. Method should provide an anomaly score, typically negative log-likelihood

$$s(x) = -\log p(x)$$

3. Test on both normal and anomalous. Score them all.
4. The results are evaluated as binary classification with threshold, AUC







Large scale study

BASIC STATISTICS OF IMAGE DATASETS DESIGNED FOR ANOMALY DETECTION (ABOVE SPLIT) AND MULTICLASS DATASETS (BELOW SPLIT)

dataset	alias	dim	anom	normal
MNIST-C	mnistc	28x28x1	70000	70000
MVTec-AD - wood	wood	1024x1024x3	60	266
MVTec-AD - grid	grid	1024x1024x3	57	285
MVTec-AD - transistor	transistor	1024x1024x3	40	273
CIFAR10	cifar10	32x32x3	54000	6000
FashionMNIST	fmnist	28x28x1	63000	7000
MNIST	mnist	28x28x1	63686	6312
SVHN2	svhn2	32x32x3	80327	18960

OVERVIEW OF THE MAIN CLASSES OF COMPARED METHODS AND THE ACRONYMS USED IN THE TEXT

class	model	acronym	class	model	acronym
flows	MAF	maf	two-stage	DAGMM	dgmm
	RealNVP	rnvp		DeepSVDD	dsvd
	SPTN	sptn		REPEN	rpn
				VAE-kNN	vaek
autoencoders	AAE	aae		VAE-OC-SVM	vaeo
	adVAE	avae			
	GANomaly	gano	classical	ABOD	abod
	skipGANomaly	skip		HBOS	hbos
	VAE	vae		IsolationForest	if
	WAE	wae		kNN	knn
				LODA	loda
gans	fAnoGAN	fano		LOF	lof
	fmGAN	fmgn		OC-SVM	osvm
	GAN	gan		PidForest	pidf
	MOGAAL	mgal			

dataset	alias	dim	anom	normal
ANNthyroid	ann	21	534	6665
Arrhythmia	arr	275	206	245
HAR	har	561	1944	8355
HTRU2	htr	8	1638	16257
KDD99 (10%)	kdd	118	396742	97276
Mammography	mam	6	260	10921
Seismic	sei	24	170	2412
Spambase	spm	57	1812	2786
Abalone	aba	10	50	2151
Blood Transfusion	blt	4	16	382
Breast Cancer Wisconsin	bcw	30	206	356
Breast Tissue	bts	9	22	65
Cardiotocography	crd	27	228	1830
Ecoli	eco	7	108	205
Glass	gls	10	94	112
Haberman	hab	3	14	225
Ionosphere	ion	33	122	225
Iris	irs	4	46	100
Isolet	iso	617	3300	4496
Letter Recognition	ltr	617	3600	4196
Libras	lbr	90	142	215
Magic Telescope	mgc	10	3882	12331
Miniboone	mnb	50	23922	93565
Multiple Features	mlt	649	800	1200
PageBlocks	pgb	10	384	4911
Parkinsons	prk	22	44	146
Pendigits	pen	16	5384	5537
Pima Indians	pim	8	176	500
Sonar	snr	60	96	110
Spect Heart	sph	44	52	211
Statlog Satimage	sat	36	2630	3592
Statlog Segment	seg	18	938	1320
Statlog Shuttle	sht	8	28	57767
Statlog Vehicle	vhc	18	132	627
Synthetic Control Chart	scc	60	200	400

Lessons learned:

- Hyperparameters important for all
- How many anomalies available for hyper-parameter selection

validation	tabular data	image stat	image semantic
no anomalies	KNN (Flow)	VAE	DSVD
many anomalies	OCSVM (VAE)	VAE	fmGAN (VAE)

- Score in VAE treated a hyperparameter
- Poor performance of complex methods[3]

[3]Škvára, V., Francu, J., Zorek, M., Pevný, T. and Šmídl, V., 2021. Comparison of anomaly detectors: context matters. IEEE Transactions on Neural Networks and Learning Systems, 33(6), pp.2494–2507.

Lessons learned:

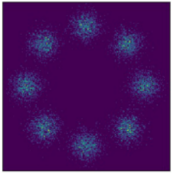
- Hyperparameters important for all
- How many anomalies available for hyper-parameter selection

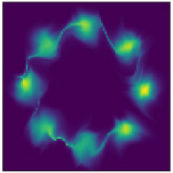
validation	tabular data	image stat	image semantic
no anomalies	KNN (Flow)	VAE	DSVD
many anomalies	OCSVM (VAE)	VAE	fmGAN (VAE)

- Score in VAE treated a hyperparameter
- Poor performance of complex methods[3]

Open issues:

- Discrete data (overfitting on some)
- Issues with multi-modal distributions
  - Huge Flows
  - Mixtures of simple Flows?



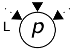


Space for a new model?

[3] Škvára, V., Francu, J., Zorek, M., Pevný, T. and Šmídl, V., 2021. Comparison of anomaly detectors: context matters. IEEE Transactions on Neural Networks and Learning Systems, 33(6), pp.2494-2507.

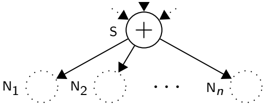
Representation of probability functions proposed by Poon and Domingos[4] as a computational graph. Combines two types of "nodes" operating on selected elements $\overline{x}$ of vector x:

Leaf node



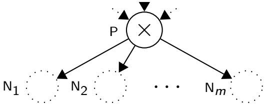
$$p_L(\overline{x}) = \begin{cases} \mathcal{N}(\mu, \sigma) \\ Po(\lambda) \\ \dots \end{cases}$$

Sum node



$$p_S(\overline{x}) = \sum_{N \in \mathrm{Ch}(S)} w_N \cdot p_N(\overline{x})$$

Product node



$$p_P(\vec{x}) = \prod_{N \in \mathrm{Ch}(P)} p_N(\vec{x}_N)$$

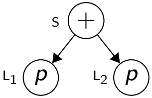
Only independent products!

- Allow for tractable marginals
- Dependence due to sum nodes

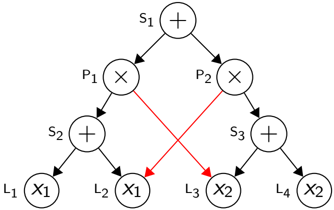
[4] Poon, H. and Domingos, P., 2011, November. Sum-product networks: A new deep architecture. In 2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops) (pp. 689-690). IEEE.

Examples

Mixture model in 1d: $p(x_1)$



Mixture model in 2d: $[x_1, x_2]$



$$p(x_1) = wp_1(x_1) + (1 - w)p_2(x_1)$$

$$\begin{aligned} p(x_1, x_2) = & w\left(up_1(x_1) + (1-u)p_2(x_1)\right) p_3(x_2) \\ & (1-w)p_2(x_1)\left(vp_3(x_2) + (1-u)p_4(x_2)\right) \end{aligned}$$

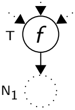
- Sharing parametrization
- Structure estimation
- Combining different distributions!
  - categorical
  - continuous
- Advantages in higher-dimensions

Sum-product-transform networks

Original SPN focus mostly on cathegorical data, less attention to continuous.

We[5] proposed to combine SPN with Flow models:

Transformation node



$$p(x) = p_z(z)|\det J(z)|,$$
$$z = f^{-1}(x)$$

Lightweight flow:
Dense layer with SDV weight matrix

$$y = \sigma(Ax + b) = \sigma(UDVx + b)$$

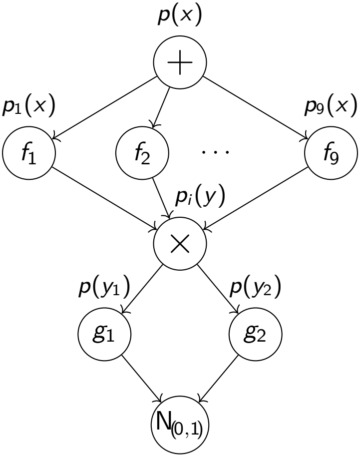
where U, D, V can be learned by GD.
Tractable Jacobian

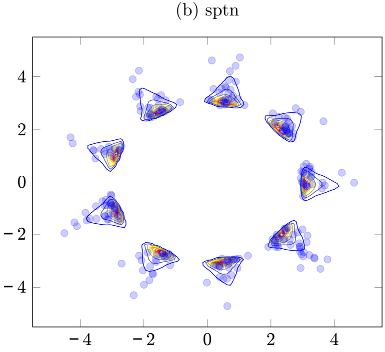
$$\log \det J(z) = \sum_{i=1}^{d} \log d_{ii} + \sum_{i=1}^{d} \log \frac{\partial \sigma}{\partial z}$$

- For continuous data, $\sigma =$ identity, and leafs $N(0, 1)$, the model becomes a fancy mixture of Gaussians (block covariance matrices)
- On the anomaly detection task, the model is comparable to other flow models.

[5] Pevný, T., Smídl, V., Trapp, M., Poláček, O. and Oberhuber, T., 2020, February. Sum-product-transform networks: Exploiting symmetries using invertible transformations. In International Conference on Probabilistic Graphical Models (pp. 341-352). PMLR.

Example





- SPTN on anomaly detection benchmark comparable to flow models.
- Slow inference.
  - progress using Metropolis-Hastings MC[6]

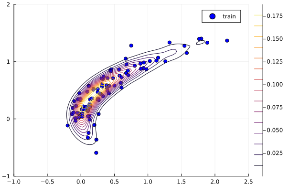
[6]Papez, M., Pevný, T. and Smidl, V., Reducing the Cost of Fitting Mixture Models via Stochastic Sampling. In The 5th Workshop on Tractable Probabilistic Modeling. UAI 2022

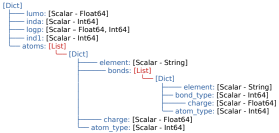
Roadmap: probability learning

Vector data

Set data

Full Hierarchy





Probability models of sets of data

By a set of data, we understand an unordered set of feature vectors, $X = \{x_1, \dots x_n\}$ for arbitrary $n \in \mathbb{N}$.

IID cluster all vectors in the set are realizations from the same distribution

$$x_i \sim p_x(x), \qquad n \sim p_c(\lambda),$$

The set can be perceived as an empirical distribution. Formally simple.

1. Kernel methods with statistical divergence[7] or Chamfer distance

$$D_{CH}(X, X') = \frac{1}{n} \sum_i \min_j \|x_i - x_j'\|_2^2 + \frac{1}{n'} \sum_j \min_i \|x_i - x_j'\|_2^2$$

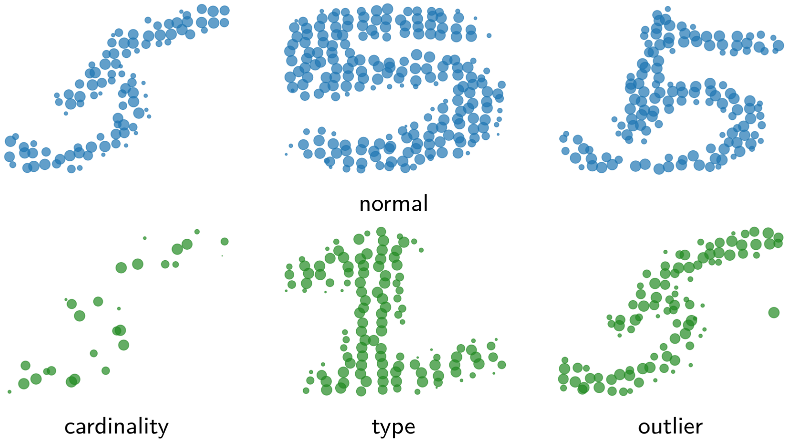
2. Likelihood of a random set

$$p(X) = p_c(n) U^n n! \prod_{i=1}^{n} p_x(x_i),$$

with MLE estimation from union of all feature vectors. "Just" choose p_x, p_c.

[7] Muandet, K., Fukumizu, K., Dinuzzo, F. and Schölkopf, B., 2012. Learning from distributions via support measure machines. Advances in neural information processing systems, 25.

Comparison on Set Anomaly Detection



Compared methods

Methods

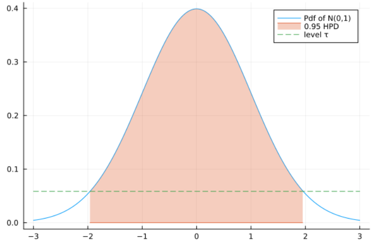
- **OCSVM** on chamfer or mmd distance
- **IVAE** learn regular VAE on features points, $p(x)$
- **Pool** aggregate features to "embedding", generate from it
- **NeuralStat** combination of IVAE and Pool
- **SetVAE** transformer-based model of sets

Data sets:

- MNIST point cloud
- MVTech – SIFT features
- MI problems
- LHC challenge

Lessons learned

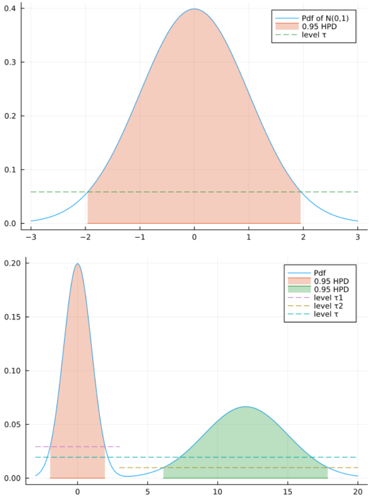
1. Likelihood is **useless** as an anomaly measure!
   - HPD region of $N(0, 1)$



[a] Vo, B.N., Dam, N., Phung, D., Tran, Q.N. and Vo, B.T., 2018. Model-based learning for point pattern data. Pattern Recognition, 84, pp.136-151.

Lessons learned

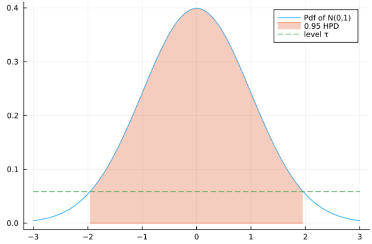
1. Likelihood is **useless** as an anomaly measure!
   - HPD region of $N(0,1)$
   - HPD region of $\frac{1}{2}N(0,1) + \frac{1}{2}N(12,3)$

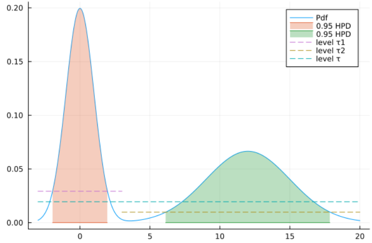


[a]Vo, B.N., Dam, N., Phung, D., Tran, Q.N. and Vo, B.T., 2018. Model-based learning for point pattern data. Pattern Recognition, 84, pp.136-151.

Lessons learned

1. Likelihood is **useless** as an anomaly measure!
   - HPD region of $N(0,1)$
   - HPD region of $\frac{1}{2}N(0,1) + \frac{1}{2}N(12,3)$
   - Random finite set likelihood is a mixture of components in increasing dimensions
   - Likelihood is either rejecting high or low cardinalities





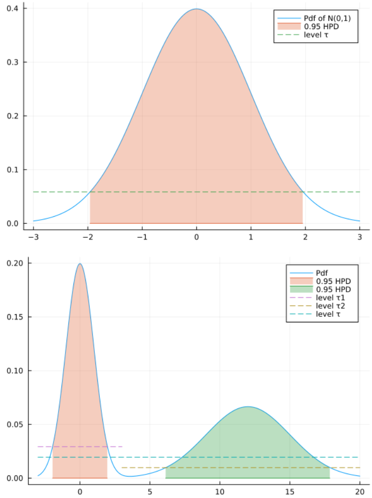
[a]Vo, B.N., Dam, N., Phung, D., Tran, Q.N. and Vo, B.T., 2018. Model-based learning for point pattern data. Pattern Recognition, 84, pp.136-151.

Lessons learned

1. Likelihood is **useless** as an anomaly measure!
   - HPD region of $N(0,1)$
   - HPD region of $\frac{1}{2}N(0,1) + \frac{1}{2}N(12,3)$
   - Random finite set likelihood is a mixture of components in increasing dimensions
   - Likelihood is either rejecting high or low cardinalities
2. Known issue[a] with proposed fix

$$s(X) = -\log p(X) - n \log U$$

where $U = \int p(x)^2 dx$.



[a] Vo, B.N., Dam, N., Phung, D., Tran, Q.N. and Vo, B.T., 2018. Model-based learning for point pattern data. Pattern Recognition, 84, pp.136-151.

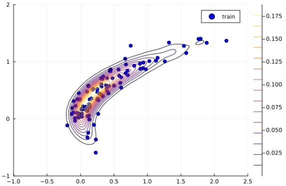
Results: ranks of anomaly detectors

collection	# a.	IVAE	VB	IVAE-CH	NS	PoolModel	SetVAE	FN-VAE	HMIL
MNIST	0	**5.800**	5.400	3.200	5.700	**1.100**	4.800	2.000	—
(class	5	3.800	5.700	4.600	4.600	**1.500**	4.300	3.600	**7.900**
anomalies)	10	4.900	5.700	4.100	4.600	**1.200**	4.300	3.300	**7.900**
	all	5.600	**6.700**	5.100	5.500	2.400	5.300	4.400	**1.000**
MIL	0	**2.111**	**5.667**	4.222	2.222	3.056	5.333	5.389	—
datasets	5	**2.167**	4.722	4.944	2.278	3.611	**6.278**	6.111	5.889
(mixed	10	**1.778**	4.500	5.444	2.000	3.778	**6.667**	6.333	5.500
anomalies)	all	**2.111**	4.889	5.889	2.167	4.500	6.722	**6.778**	2.889
MV-TEC	0	**2.250**	5.000	**5.750**	2.500	3.750	4.500	4.250	—
(instance	5	2.250	5.250	5.250	**1.250**	4.750	5.000	4.750	**7.500**
anomalies)	10	2.500	5.000	4.250	**1.500**	4.750	5.000	5.250	**7.750**
	all	2.750	5.750	5.500	2.250	5.750	6.250	**6.500**	**1.250**

- Pooling important on point-clouds
- Complicated methods not improving
- VAE works well, NS just a slight differences for VAE
- Score:
  - Vo's fix not working
  - estimation of $\log U$ much better
  - $\mathrm{mean}(p(x_i))$ almost the best
- Missing theory!

Roadmap: probability learning

Vector data



Set data

Full Hierarchy

```
[Dict]
├── lumo: [Scalar - Float64]
├── inda: [Scalar - Int64]
├── logp: [Scalar – Float64, Int64]
├── ind1: [Scalar - Int64]
└── atoms: [List]
      └── [Dict]
            ├── element: [Scalar - String]
            ├── bonds: [List]
            │     └── [Dict]
            │           ├── element: [Scalar - String]
            │           ├── bond_type: [Scalar - Int64]
            │           ├── charge: [Scalar - Float64]
            │           └── atom_type: [Scalar - Int64]
            ├── charge: [Scalar - Float64]
            └── atom_type: [Scalar - Int64]
```

Sum-Product-Set Networks

Tools of probability for JSON structure:

```
[Dict]
├── lumo: [Scalar - Float64]
├── inda: [Scalar - Int64]
├── logp: [Scalar – Float64, Int64]
├── ind1: [Scalar - Int64]
└── atoms: [List]
           └── [Dict]
                ├── element: [Scalar - String]
                ├── bonds: [List]
                │          └── [Dict]
                │               ├── element: [Scalar - String]
                │               ├── bond_type: [Scalar - Int64]
                │               ├── charge: [Scalar - Float64]
                │               └── atom_type: [Scalar - Int64]
                ├── charge: [Scalar - Float64]
                └── atom_type: [Scalar - Int64]
```

Leaf: probability density of vector data = SPN

Dict: joint probability density = SPN

Sum-Product-Set Networks

Tools of probability for JSON structure:

```
[Dict]
├── lumo: [Scalar - Float64]
├── inda: [Scalar - Int64]
├── logp: [Scalar – Float64, Int64]
├── ind1: [Scalar - Int64]
└── atoms: [List]
        └── [Dict]
              ├── element: [Scalar - String]
              ├── bonds: [List]
              │        └── [Dict]
              │              ├── element: [Scalar - String]
              │              ├── bond_type: [Scalar - Int64]
              │              ├── charge: [Scalar - Float64]
              │              └── atom_type: [Scalar - Int64]
              ├── charge: [Scalar - Float64]
              └── atom_type: [Scalar - Int64]
```

Leaf: probability density of vector data = SPN

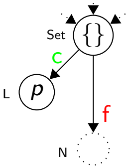
Dict: joint probability density = SPN

List: random set theory, $p(X)$,
$X = \{x_1, x_2 \ldots, x_n\}$

Sum-Product-Set Networks

Tools of probability for JSON structure:

```
[Dict]
├── lumo: [Scalar - Float64]
├── inda: [Scalar - Int64]
├── logp: [Scalar – Float64, Int64]
├── ind1: [Scalar - Int64]
└── atoms: [List]
      └── [Dict]
            ├── element: [Scalar - String]
            ├── bonds: [List]
            │     └── [Dict]
            │           ├── element: [Scalar - String]
            │           ├── bond_type: [Scalar - Int64]
            │           ├── charge: [Scalar - Float64]
            │           └── atom_type: [Scalar - Int64]
            ├── charge: [Scalar - Float64]
            └── atom_type: [Scalar - Int64]
```



$$p_{\text{Set}}(X) = p_c(n)\,|n|! \prod_{i=1}^{n} p_f(x_i)\,.$$

Acts as any node (can be nested) with constraints.

Leaf: probability density of vector data = SPN

Dict: joint probability density = SPN

List: random set theory, $p(X)$,
$X = \{x_1, x_2 \ldots, x_n\}$
New node: **SetNode**

Automatic probabilistic model for JSON

HMIL discriminative learner

```
[Dict]
  ├── lumo: [Scalar - Float64]
  ├── inda: [Scalar - Int64]
  ├── logp: [Scalar – Float64, Int64]
  ├── ind1: [Scalar - Int64]
  └── atoms: [List]
               └── [Dict]
                     ├── element: [Scalar - String]
                     ├── bonds: [List]
                     │            └── [Dict]
                     │                  ├── element: [Scalar - String]
                     │                  ├── bond_type: [Scalar - Int64]
                     │                  ├── charge: [Scalar - Float64]
                     │                  └── atom_type: [Scalar - Int64]
                     ├── charge: [Scalar - Float64]
                     └── atom_type: [Scalar - Int64]
```

SPSN probabilistic model

```
ProductNode
  ├── lumo: Categorical
  ├── inda: Categorical
  ├── logp: Categorical
  ├── ind1: Categorical
  └── atoms: SetNode
               ├── c: Poisson
               └── f: ProductNode
                        ├── element: Categorical
                        ├── bonds: SetNode
                        │            ├── c: Poisson
                        │            └── f: ProductNode
                        │                     ├── element: Categorical
                        │                     ├── bond_type: Categorical
                        │                     ├── charge: SumNode
                        │                     │             ├── Normal
                        │                     │             └── Normal
                        │                     └── atom_type: Categorical
                        ├── charge: SumNode
                        │             ├── Normal
                        │             └── Normal
                        └── atom_type: Categorical
```

- Cardinality is Poisson distributed
- Continuous variables represented by a 2component GMM.

Automatic probabilistic model for JSON

HMIL discriminative learner

```
[Dict]
├── lumo: [Scalar - Float64]
├── inda: [Scalar - Int64]
├── logp: [Scalar – Float64, Int64]
├── ind1: [Scalar - Int64]
└── atoms: [List]
      └── [Dict]
            ├── element: [Scalar - String]
            ├── bonds: [List]
            │     └── [Dict]
            │           ├── element: [Scalar - String]
            │           ├── bond_type: [Scalar - Int64]
            │           ├── charge: [Scalar - Float64]
            │           └── atom_type: [Scalar - Int64]
            ├── charge: [Scalar - Float64]
            └── atom_type: [Scalar - Int64]
```

SPSN probabilistic model

```
ProductNode
├── lumo: Categorical
├── inda: Categorical
├── logp: Categorical
├── ind1: Categorical
└── atoms: SetNode
      ├── c: Poisson
      └── f: ProductNode
            ├── element: Categorical
            ├── bonds: SetNode
            │     ├── c: Poisson
            │     └── f: ProductNode
            │           ├── element: Categorical
            │           ├── bond_type: Categorical
            │           ├── charge: SumNode
            │           │     ├── Normal
            │           │     └── Normal
            │           └── atom_type: Categorical
            ├── charge: SumNode
            │     ├── Normal
            │     └── Normal
            └── atom_type: Categorical
```

- Cardinality is Poisson distributed
- Continuous variables represented by a 2component GMM.

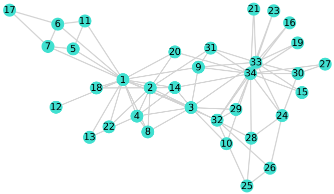
	Accuracy	# parameters
HMill classifier	**0.886**	4172
SPSN classifier (likelihood ratio)	0.818	**516**

Results on MI problems

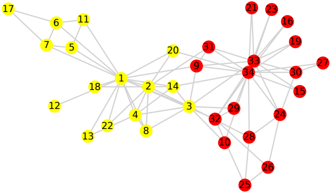
dataset/model	HMill classifier	SPSN classifier
brown_creeper	**0.936**	0.921
corel_african	**0.948**	**0.948**
corel_beach	0.968	**0.982**
elephant	0.785	**0.79**
fox	**0.565**	0.555
musk_1	**0.800**	0.667
musk_2	0.779	**0.86**
mutagenesis_1	**0.833**	0.728
mutagenesis_2	**0.825**	0.675
protein	**0.947**	0.863
tiger	**0.810**	0.715
ucsb_breast_cancer	**0.780**	0.64
winter_wren	**0.991**	0.908

Model-based clustering of graphs

Graph dataset (karate)



Results of 2component SPSN



Conclusion: density learning state of the art

1. Density learning on vector data
    1.1 classical methods still valueable
    1.2 deep models suitable for image data
    1.3 space for new models on heterogenous data
    1.4 challenges for complex problems (semantic data)
2. Density learning on set data
    2.1 poor results of kernel methods
    2.2 space for smart combination of set-embedding and feature-embedding
    2.3 how to properly treat cardinality in anomaly detection?
3. Sum-product-set networks
    3.1 elementary blocks are ready
    3.2 computational speed
    3.3 structure selection
    3.4 anomaly score?