Introduction to Density Estimation and Anomaly Detection

Tom Dietterich

Oregon State University

Outline

- Definition and Motivations
- Density Estimation
  - Parametric Density Estimation
  - Mixture Models
  - Kernel Density Estimation
  - Neural Density Estimation
- Anomaly Detection
  - Distance-based methods
  - Isolation Forest and LODA
  - RPAD theory

What is Density Estimation?

- Given a data set $\{x_1, \ldots, x_N\}$ where $x_i \in \mathbb{R}^d$
- We assume the data have been drawn iid from an unknown probability density: $x_i \sim P(x_i)$
- Goal: Estimate P

- Requirements
  - $P(x) \geq 0 \; \forall x \in \mathbb{R}^d$
  - $\int_{x \in \mathbb{R}^d} P(x) dx = 1$

How to Evaluate a Density Estimator

- Suppose I have computed a density estimator $\hat{P}$ for P. How can I evaluate it?
- A good density estimator should assign high density where P is large and low density where P is low
- Standard metric: the Log Likelihood

$$\sum_i \log \hat{P}(x_i)$$

Important: Holdout Likelihood

- If we use our training data to construct $\hat{P}$, we cannot use that same data to evaluate $\hat{P}$.
- Solution:
  - Given our initial data $S = x_1, \ldots, x_N$
  - Randomly split into S_{train} and S_{test}
  - Compute $\hat{P}$ using S_{train}
  - Evaluate $\hat{P}$ using S_{test}

$$\sum_{x_i \in S_{test}} \log \hat{P}(x_i)$$

Reminder: Densities, Probabilities, Events

- A density μ is a "measure" over some space $\mathcal{X}$
- A density can be > 1 but must integrate to 1
- An "event" is a subspace (region) $E \subseteq \mathcal{X}$
- The probability of an event is obtained by integration

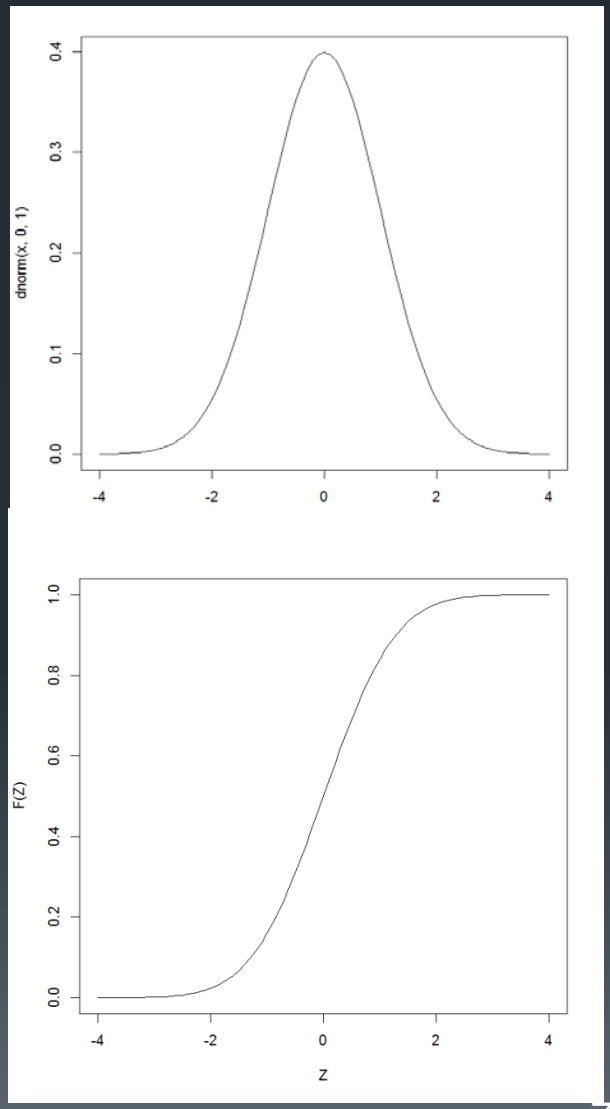
$$P(E) = \int_{x \in E} \mu(x) dx$$

Example from the ipython notebook

- Normal probability density

$$P(x;\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma}\exp -\frac{1}{2}\left[\frac{x-\mu}{\sigma}\right]^2$$

- Normal cumulative distribution function
  - $F(z;\mu,\sigma)$ = probability of the event $[-\infty, z]$
  - $F(z;\mu,\sigma) = \int_{\infty}^{z} P(x;\mu,\sigma)dx$



Why Estimate Densities?

- Anomaly Detection
- Classification

- If we can learn high-dimensional densities, then all machine learning problems can be solved using probabilistic methods

Anomaly Detection

- Anomaly: A data point generated by a different process than the process that generates the normal data points
  - Example: Fraud Detection
    - Normal points: Legitimate financial transactions
    - Anomaly points: Fraudulent transactions
  - Example: Sensor Data
    - Normal points: Correct data values
    - Anomaly points: Bad values (broken sensors)

Anomaly Score using Surprise

- In information theory, the surprise of an observation x, $S(x)$ is defined as

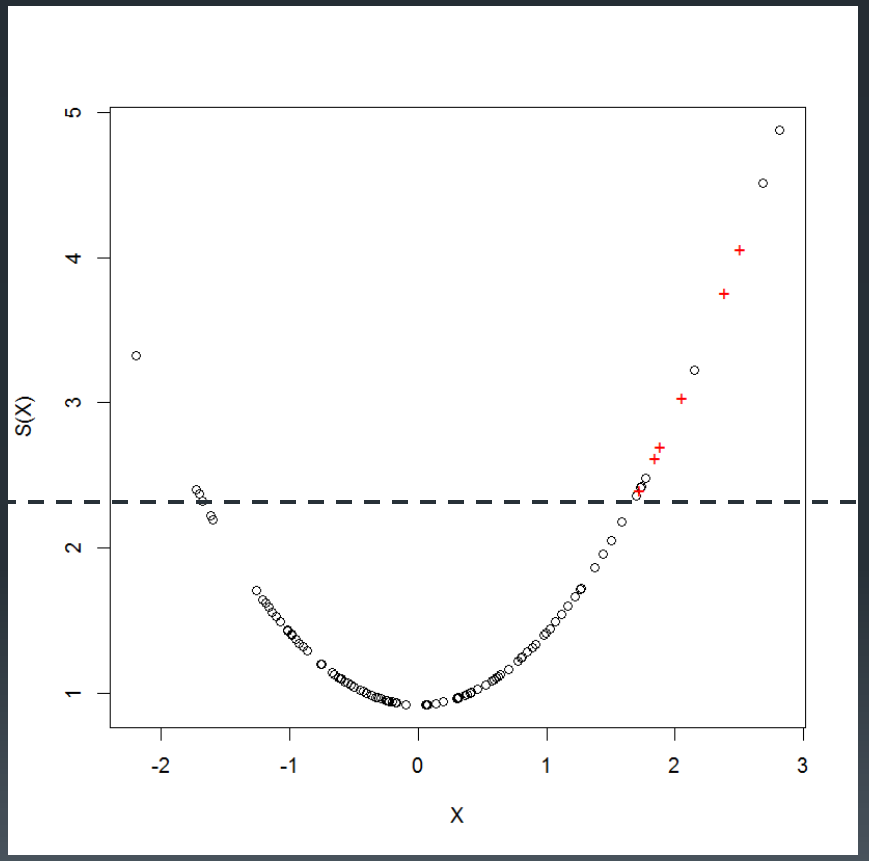
$$S(x) = -\log P(x)$$

- Properties:
  - If $P(x) = 0$, then $S(x) = +\infty$
  - If $P(x) = 1$, then $S(x) = 0$

- Surprise is only defined for events, but we often use it for densities when we are interested on small values of $P(x)$

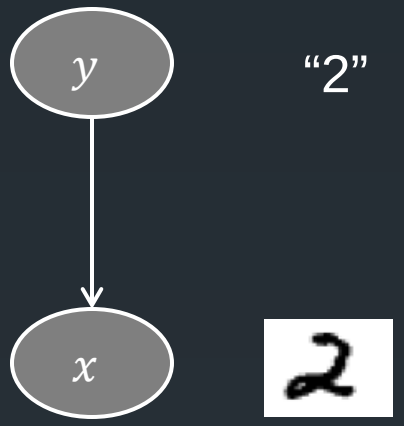
Example

- Nominal distribution: $\mathrm{Normal}(0,1)$
- Anomaly distribution: $\mathrm{Normal}(3,1)$
- Generate 100 nominals and 10 anomalies
- Plot anomaly score as a function of x
- Setting a threshold at 2.39 will detect all anomalies and also have 8 false alarms



Classification

- Class-Conditional Models
- Goal: Predict y from x
- Model the process that creates the data:
  - $y \sim P(y)$ discrete
  - $x \sim P(x|y) = N(x|\mu_y, \Sigma_y)$ Gaussian
- "Conditional Density Estimation"
- Classification via probabilistic inference
  - $P(y|x) = \frac{P(y)N(x|\mu_y,\Sigma_y)}{\sum_{y'} P(y')N(x|\mu_{y'},\Sigma_{y'})}$
  - Which class y best explains the observed x?
- Challenge: $P(x|y)$ is usually very complex and difficult to model.



Outline

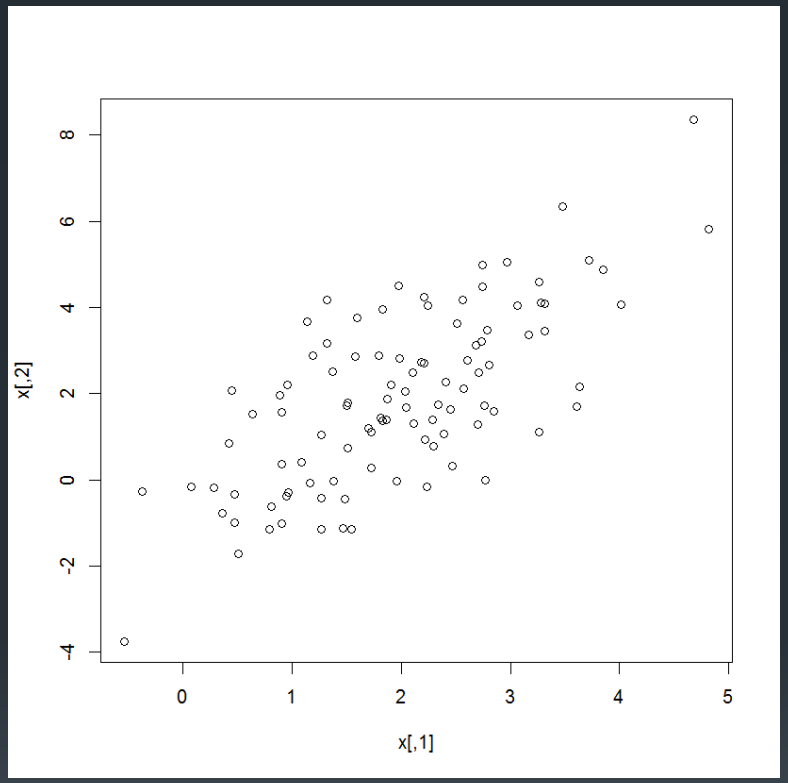
- Definition and Motivations
- Density Estimation
  - Parametric Density Estimation
  - Mixture Models
  - Kernel Density Estimation
  - Neural Density Estimation
- Anomaly Detection
  - Distance-based methods
  - Isolation Forest and LODA
  - RPAD theory

Parametric Density Estimation

- Assume $P(x) = \text{Normal}(x|\mu, \Sigma)$ is the multivariate Gaussian distribution
- $P(x) = \frac{1}{\sqrt{(2\pi)^d \det(\Sigma)}} \exp -\frac{1}{2}(x-\mu)^\top \Sigma^{-1}(x-\mu)$
- Fit by computing moments:
  - $\hat{\mu} = \frac{1}{N}\sum_i x_i$
  - $\hat{\Sigma} = \frac{1}{N}\sum_i (x_i - \hat{\mu})(x_i - \hat{\mu})^\top$

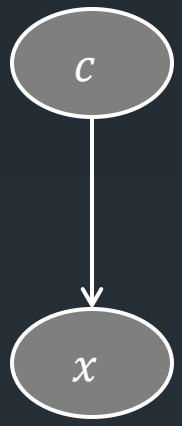
Example

- Sample 100 points from multivariate Gaussian with $\mu = (2,2)$ and $\Sigma = \begin{bmatrix} 1 & 1.5 \\ 1.5 & 4 \end{bmatrix}$
- Estimates:
  - $\hat{\mu} = (1.968731, 1.894511)$
  - $\hat{\Sigma} = \begin{bmatrix} 1.081423 & 1.462467 \\ 1.462467 & 4.000821 \end{bmatrix}$



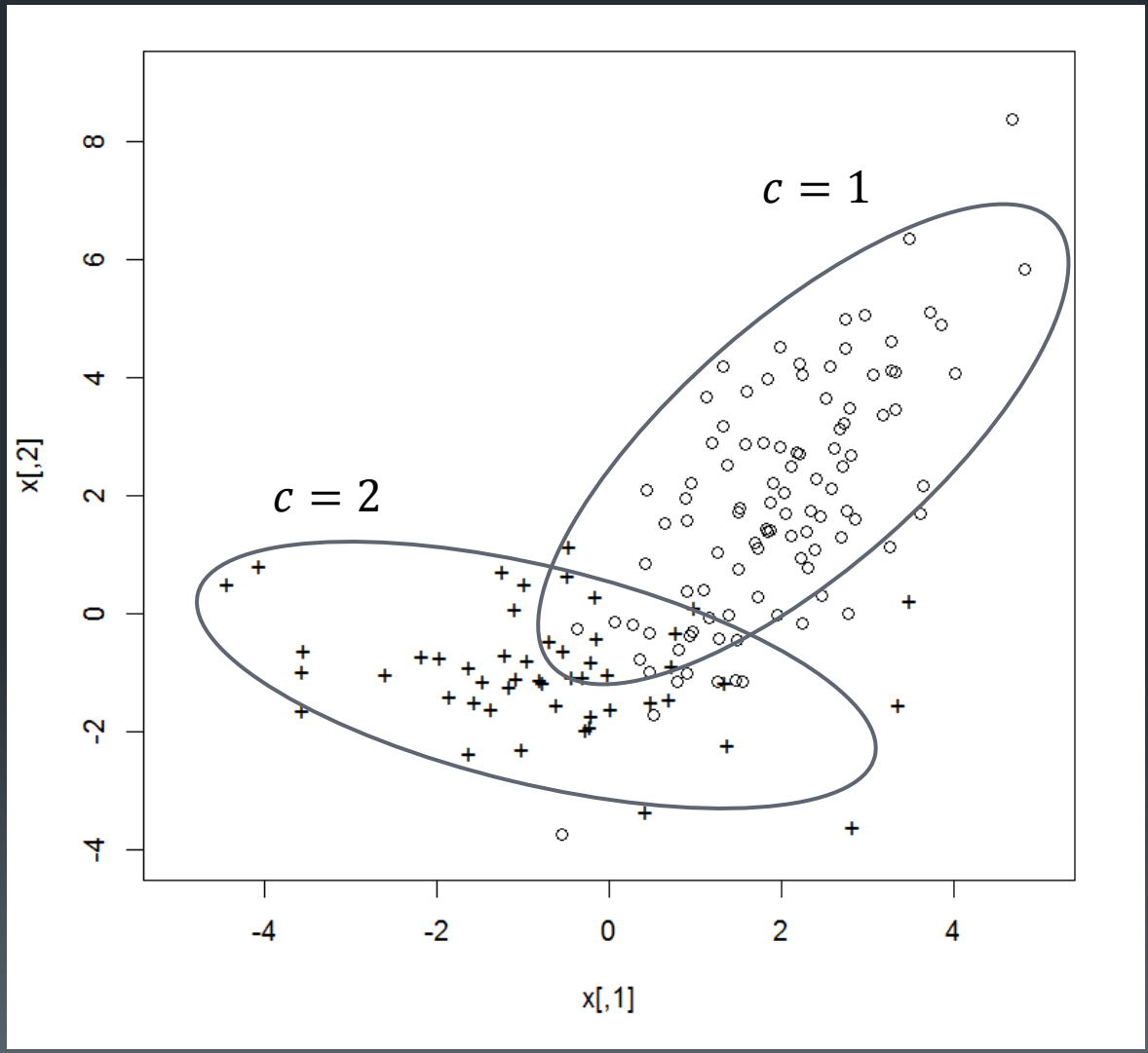
Mixture Models

- $P(x) = \sum_c P(c)P(x|c)$
- c indexes the mixture component
- Each mixture component has its own conditional density $P(x|c)$



Mixture Models

- Example
  - $P(c = 1) = 2/3$
  - $P(c = 2) = 1/3$
  - $P(x|c = 1)$ “o”
  - $P(x|c = 2)$ “+”



Fitting Mixture Models: Expectation Maximization

- Choose the number of mixture components K
- Create a matrix R of dimension $K \times N$. This will represent $P(c_i = k|x_i) = R[k, i]$
  - This is called the "membership probability". The probability that data point i was generated by component k
- Randomly initialize $R[k, i] \in (0,1)$ subject to the constraint that $\sum_k R[k, i] = 1$

EM Algorithm Main Loop

- M step: Maximum likelihood estimation of the model parameters using the data $x_1, \ldots, x_N$ and R
  - $\hat{P}(c = k) \coloneqq \frac{1}{N} \sum_i P(c_i = k | x_i)$
  - $\hat{\mu}_k \coloneqq \frac{1}{N \hat{P}(c=k)} \sum_i P(c_i = k | x_i) x_i$
  - $\hat{\Sigma}_k \coloneqq \frac{1}{N \hat{P}(c=k)} \sum_i P(c_i = k | x_i) \left[x_i x_i^\top \right]$
- E step: Re-estimate R
  - $R[k, i] \coloneqq \hat{P}(c = k) \text{Normal}\left(x_i \middle| \hat{\mu}_k, \hat{\Sigma}_k \right)$
  - $R[k, i] \coloneqq R[k, i] / \sum_k R[k, i]$

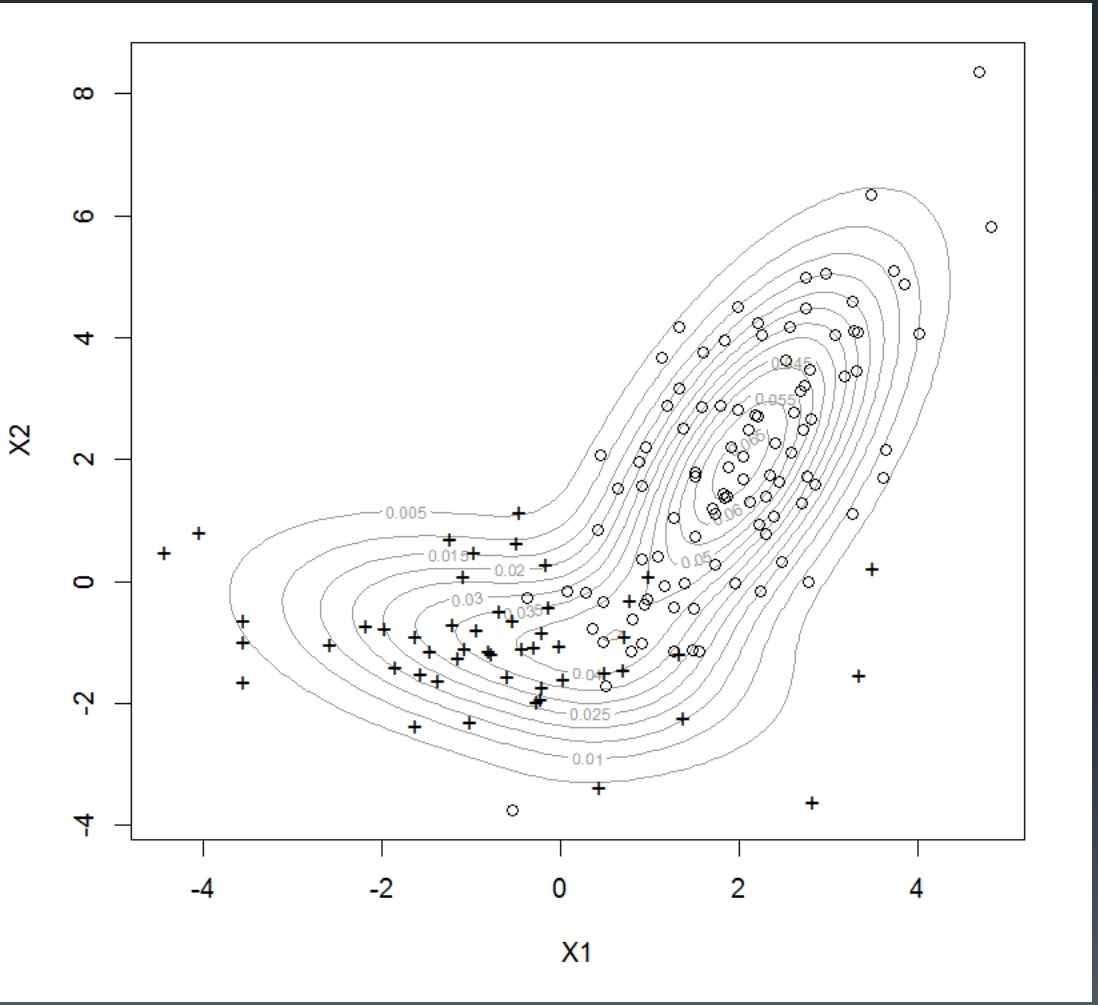
Results on Our Example [R mclust package]

Estimates

- mixing proportions
  - $(0.649, 0.351)$
- means:
  - $(2.014, 1.967)$
  - $(-0.631, -0.957)$

True values

- Mixing proportions:
  - $(0.667, 0.333)$
- Means:
  - $(2.000, 2.000)$
  - $(-1.000, -1.000)$



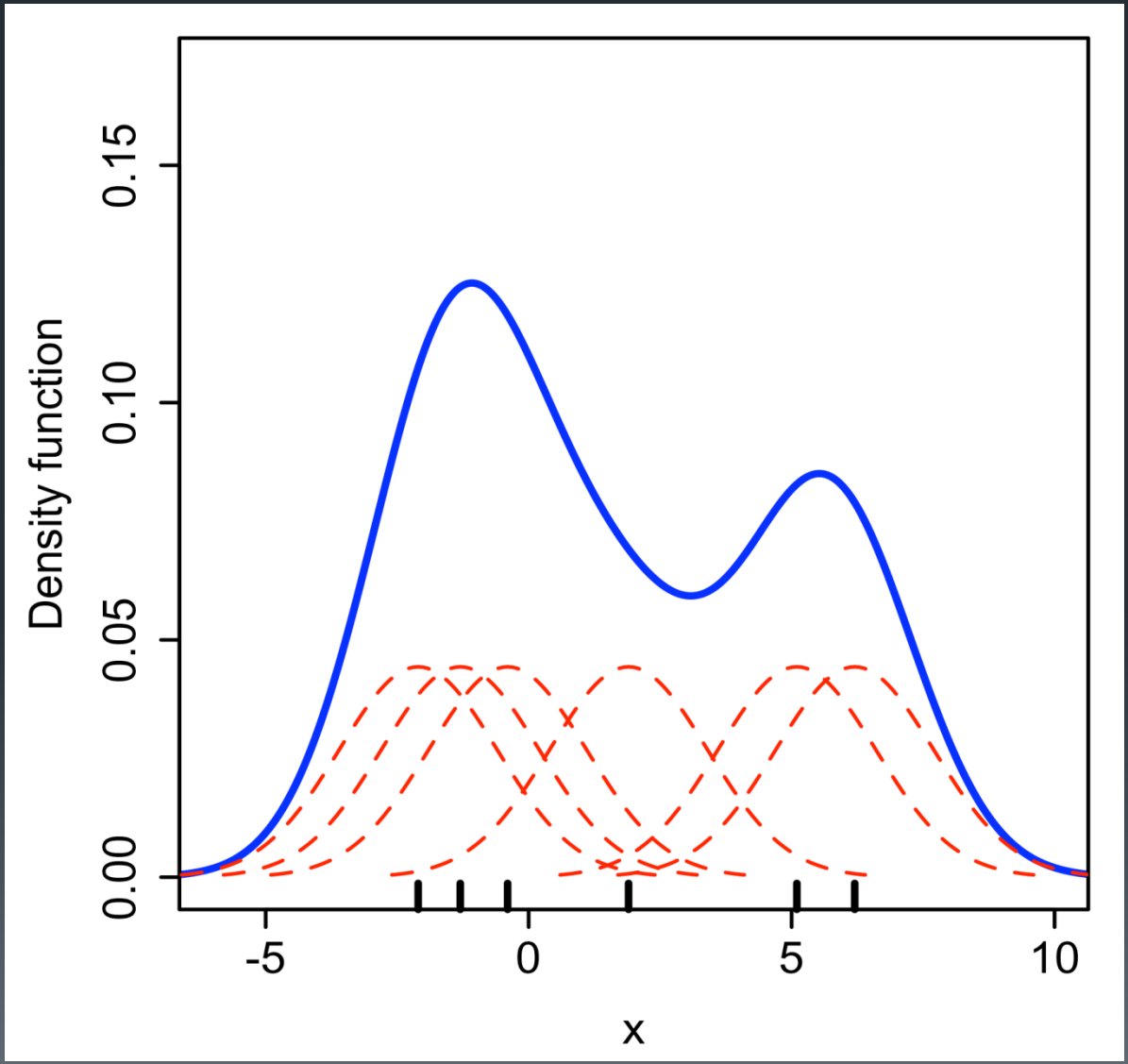
Mixture Model Summary

- Can fit very complex distributions
- EM is a local search algorithm
  - Different random initializations can find different fitted models
  - There are some new moment-based methods that find the global optimum
- Difficult to choose the number of mixture components
  - There are many heuristics for this

Kernel Density Estimation

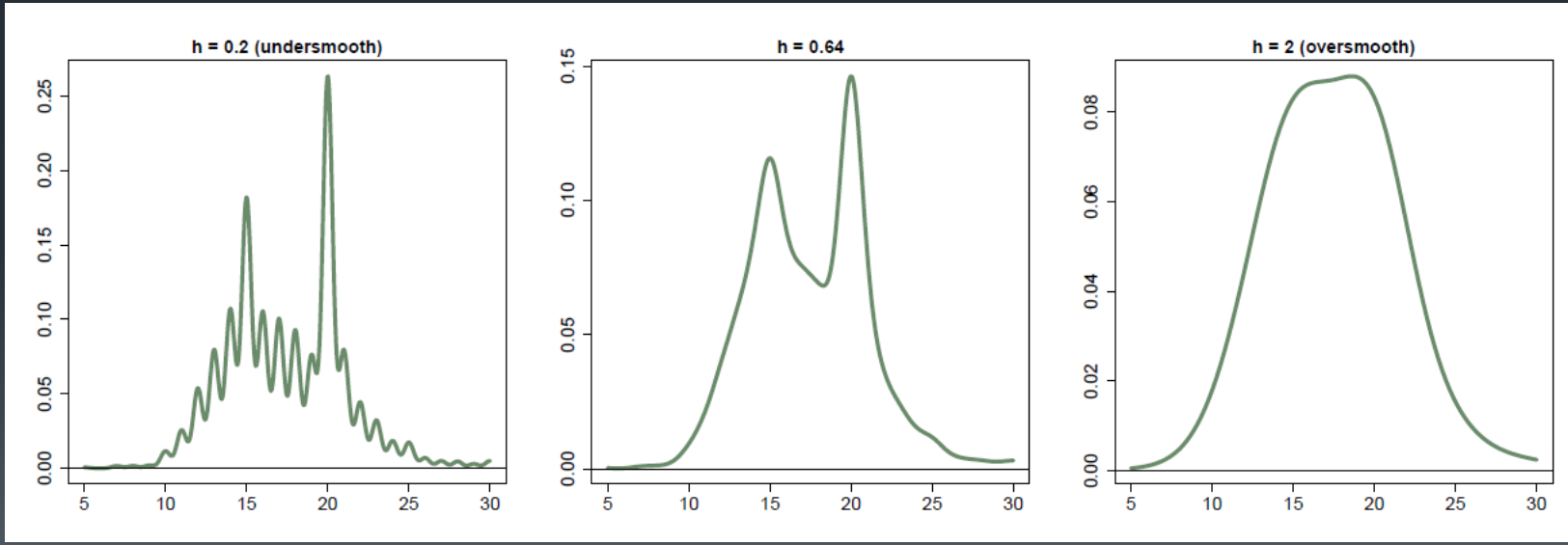
- Define a mixture model with one mixture component for each data point
  - $\hat{P}(x) = \frac{1}{N}\sum_{i=1}^{N} K(x - x_i, \sigma^2)$
  - Often use a Gaussian Kernel $K(x, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{x^2}{2\sigma^2}\right]$
  - Often use a fixed scale σ^2. The scale is also called the "bandwidth"

One-Dimensional Example



Design Decisions

- Choice of Kernel: generally not important as long as it is local
- Choice of bandwidth is very important



Challenges

- KDE in high dimensions suffers from the “Curse of Dimensionality”
- The amount of data required to achieve a desired level of accuracy scales exponentially with the dimensionality d of the problem: $\exp\frac{d+4}{2}$

Factoring the Joint Density into Conditional Distributions

- Let $x = (x_1, \ldots, x_d)$ be a vector of random variables. We wish to model the joint distribution $P(x)$.
- By the chain rule of probability, we can write this as $P(x) = P(x_1)P(x_2|x_1)P(x_3|x_1, x_2) \cdots P(x_d|x_1, \ldots, x_{d-1})$
- We can model $P(x_1)$ using any 1-D method (e.g., KDE).
- We can model each conditional distribution using regression methods

Linear Regression = Conditional Density Estimation

- Let $\mathrm{x} = (x_1, \ldots, x_J)$ be the predictor variables and y be the response variable
- Standard least-squares linear regression models the conditional probability distribution was
- $P(y|x) \sim \mathrm{Normal}(y;\ \mu(x), \sigma^2)$
  - where $\mu(x) = \beta_0 + \beta_1 x_1 + \cdots + \beta_J x_J$
  - and σ^2 is a fitted constant
- Neural networks trained to minimize squared error model the mean as $\mu(x) = NNet(x; W)$, where W are the weights of the neural network

Deep Neural Networks for Density Estimation

Masked Auto-Regressive Flow (Papamarkarios, 2017)

- Apply the chain rule of probability
- $P\left(x_j \middle| x_{1:j-1}\right) = \text{Normal}\left(x_j; \mu_j, \left(\exp \alpha_j\right)^2\right)$ where
  - $\mu_j = f_j\left(x_{1:j-1}\right)$ and $\alpha_j = g_j\left(x_{1:j-1}\right)$ are implemented by neural networks
- Re-parameterization trick for sampling $x_j \sim P\left(x_j \middle| x_{1:j-1}\right)$
  - Let $u_j \sim \text{Normal}(0,1)$
  - $x_j = u_j \exp \alpha_j + \mu_j$
  - (rescale by the standard deviation and displace by the mean)

Transformation View

- Equivalent model: $\mathbf{u} \sim Normal(\mathbf{u}; 0, I)$ is a vector of J standard normal random variates
- $\mathbf{x} = F(\mathbf{u})$ transforms those random variates into the observed data
- To compute $P(\mathbf{x})$ we can invert this function and evaluate $\text{Normal}(F^{-1}(x); 0, I)$ "almost"
- $P(x) = \text{Normal}(F^{-1}(x); 0, I) \left| \det \left[\frac{\partial F^{-1}(x)}{\partial x} \right] \right|$
  - This ensures that P integrates to 1

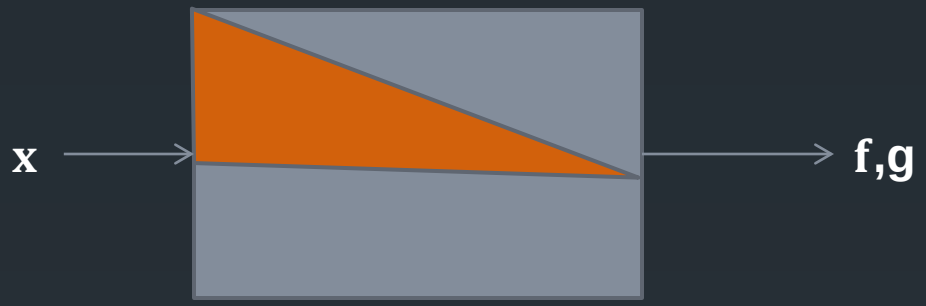
Derivative of F^{-1} is Simple

- Because of the “triangular” structure created by the chain rule,
- $\left|\det\left[\frac{\partial F^{-1}(x)}{\partial x}\right]\right| = \exp\left(-\sum_j \alpha_j\right)$

F is Easy to Invert

- $u_j = (x_j - \mu_j)\exp(-\alpha_j)$
  - where $\mu_j = f_j(x_{1:j-1})$ and $\alpha_j = g_j(x_{1:j-1})$

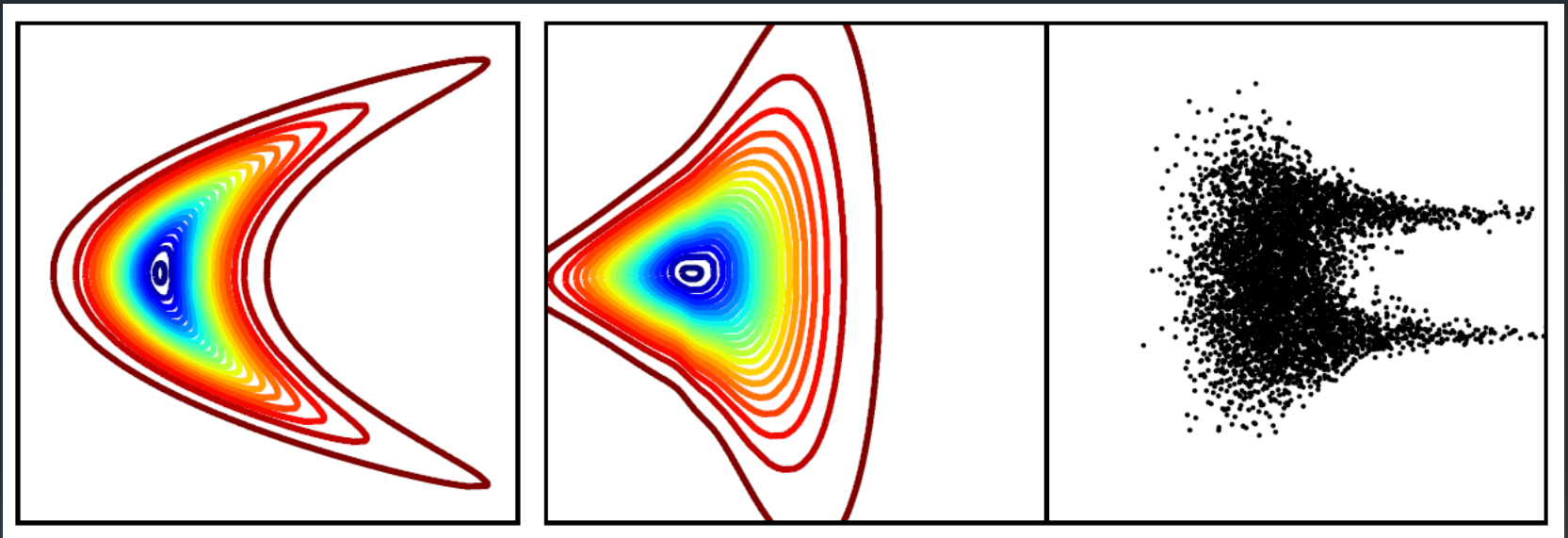
Masked Auto-Regressive Flow



- "mask" ensures that f_j and g_j only depend on $x_{1:j-1}$

Stacking MAFs

- One MAF network is often not sufficient

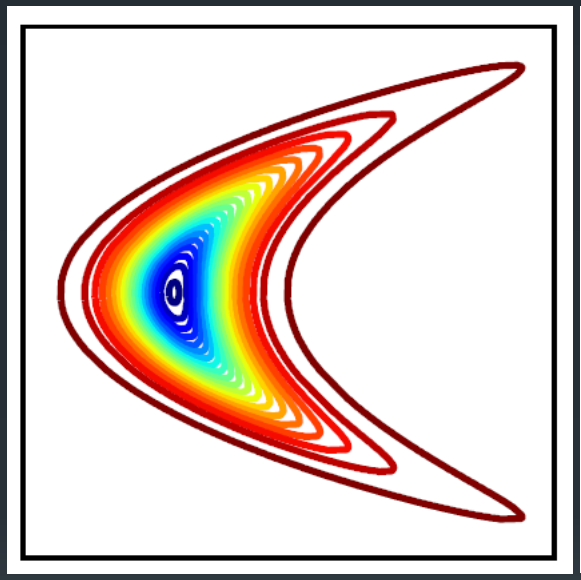


True Density

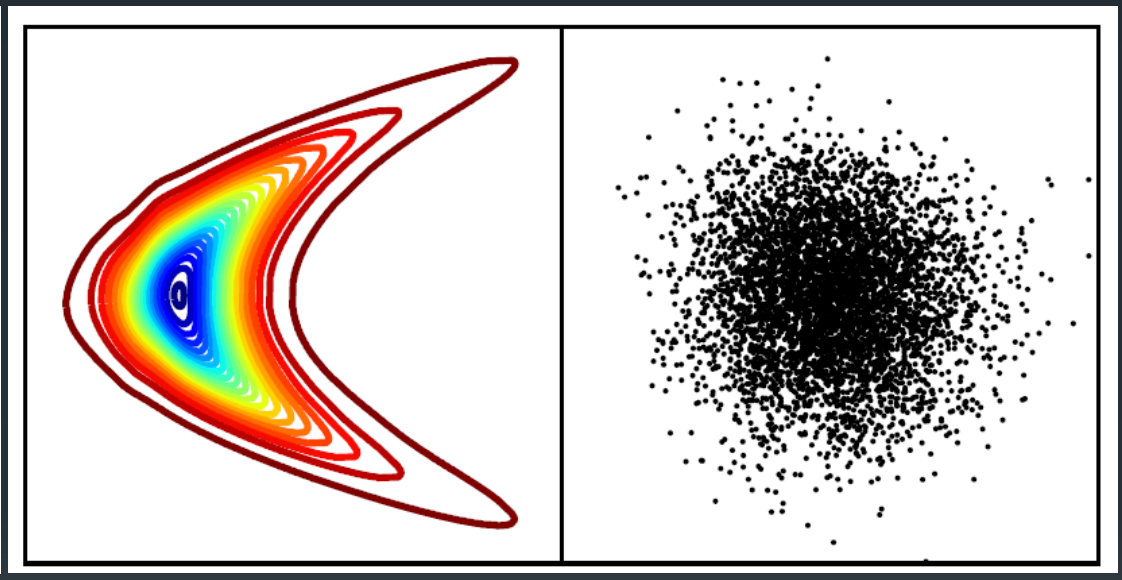
Fitted Density from single MAF network

Distribution of the $\mathbf{u}$ values

Stack MAFs until the $\mathbf{u}$ values are Normal(0,I)



True Density



Fitted Density from stack of 5 MAFs

Distribution of the $\mathbf{u}$ values

Test Set Log Likelihood

	POWER	GAS	HEPMASS	MINIBOONE	BSDS300
Gaussian	-7.74 ± 0.02	-3.58 ± 0.75	-27.93 ± 0.02	-37.24 ± 1.07	96.67 ± 0.25
MADE	-3.08 ± 0.03	3.56 ± 0.04	-20.98 ± 0.02	-15.59 ± 0.50	148.85 ± 0.28
MADE MoG	$\mathbf{0.40 \pm 0.01}$	8.47 ± 0.02	$\mathbf{-15.15 \pm 0.02}$	-12.27 ± 0.47	153.71 ± 0.28
Real NVP (5)	-0.02 ± 0.01	4.78 ± 1.80	-19.62 ± 0.02	-13.55 ± 0.49	152.97 ± 0.28
Real NVP (10)	0.17 ± 0.01	8.33 ± 0.14	-18.71 ± 0.02	-13.84 ± 0.52	153.28 ± 1.78
MAF (5)	0.14 ± 0.01	9.07 ± 0.02	-17.70 ± 0.02	$\mathbf{-11.75 \pm 0.44}$	155.69 ± 0.28
MAF (10)	0.24 ± 0.01	$\mathbf{10.08 \pm 0.02}$	-17.73 ± 0.02	-12.24 ± 0.45	154.93 ± 0.28
MAF MoG (5)	0.30 ± 0.01	9.59 ± 0.02	-17.39 ± 0.02	$\mathbf{-11.68 \pm 0.44}$	$\mathbf{156.36 \pm 0.28}$

Outline

- Definition and Motivations
- Density Estimation
  - Parametric Density Estimation
  - Mixture Models
  - Kernel Density Estimation
  - Neural Density Estimation
- Anomaly Detection
  - Distance-based methods
  - Isolation Forest and LODA
  - RPAD theory

Part 2: Other Anomaly Detection Approaches

- Do we need accurate density estimates to do anomaly detection?
- Maybe not
  - Rank points proportional to $-\log P(x)$
  - Detect low-probability outliers

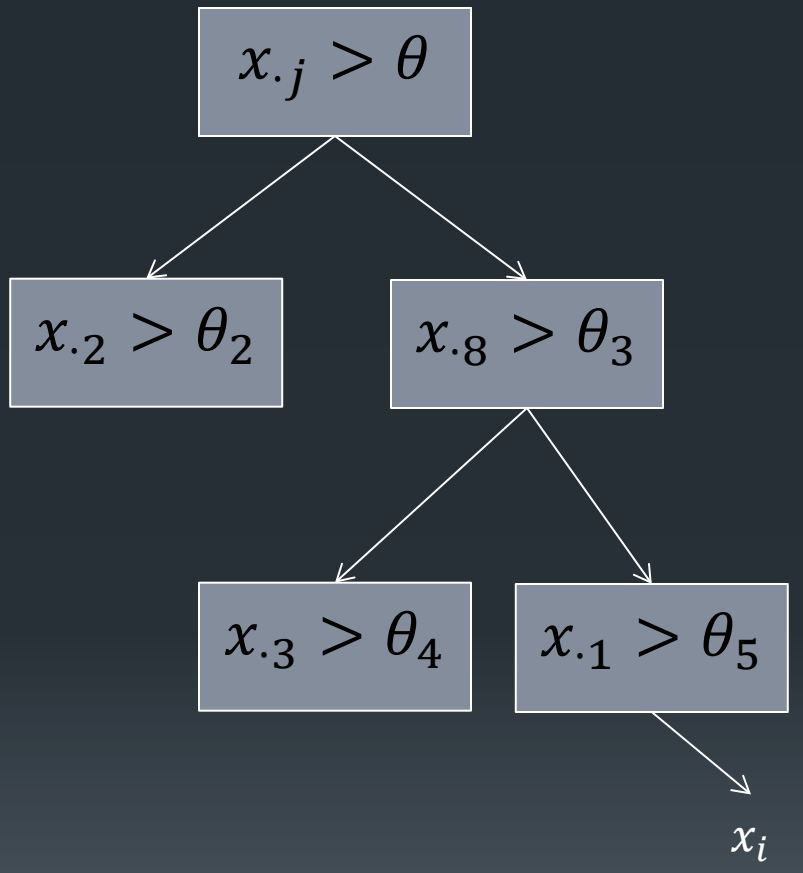
Distance-Based Anomaly Detection

- Assume a distance metric $||x - y||$ between any two data points x and y
- $A(x)$ = anomaly score = distance to k-th nearest data point
- Points in empty regions of the input space are likely to be anomalies

- There are many refinements of this basic idea
  - Learned distance metrics can be helpful

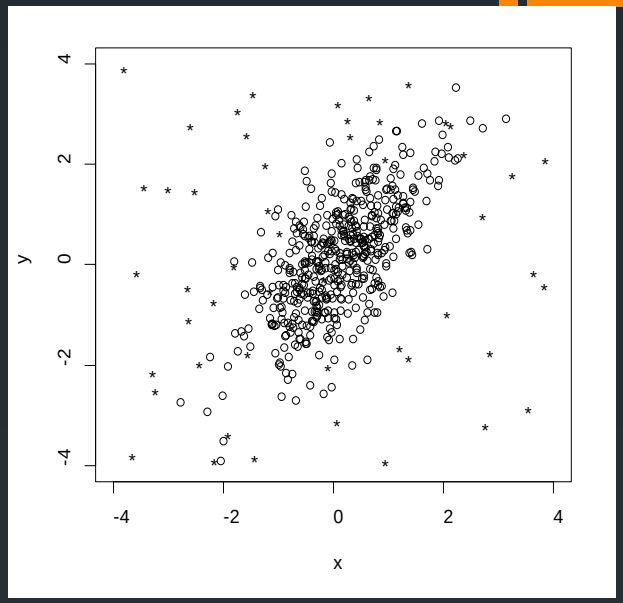
Isolation Forest [Liu, Ting, Zhou, 2011]

- Construct a fully random binary tree
  - choose attribute j at random
  - choose splitting threshold θ uniformly from $[\min(x_{.j}), \max(x_{.j})]$
  - until every data point is in its own leaf
  - let $d(x_i)$ be the depth of point x_i
- repeat 100 times
  - let $\bar{d}(x_i)$ be the average depth of x_i
  - $A(x_i) = 2^{-\left(\frac{\bar{d}(x_i)}{r(x_i)}\right)}$
    - $r(x_i)$ is the expected depth



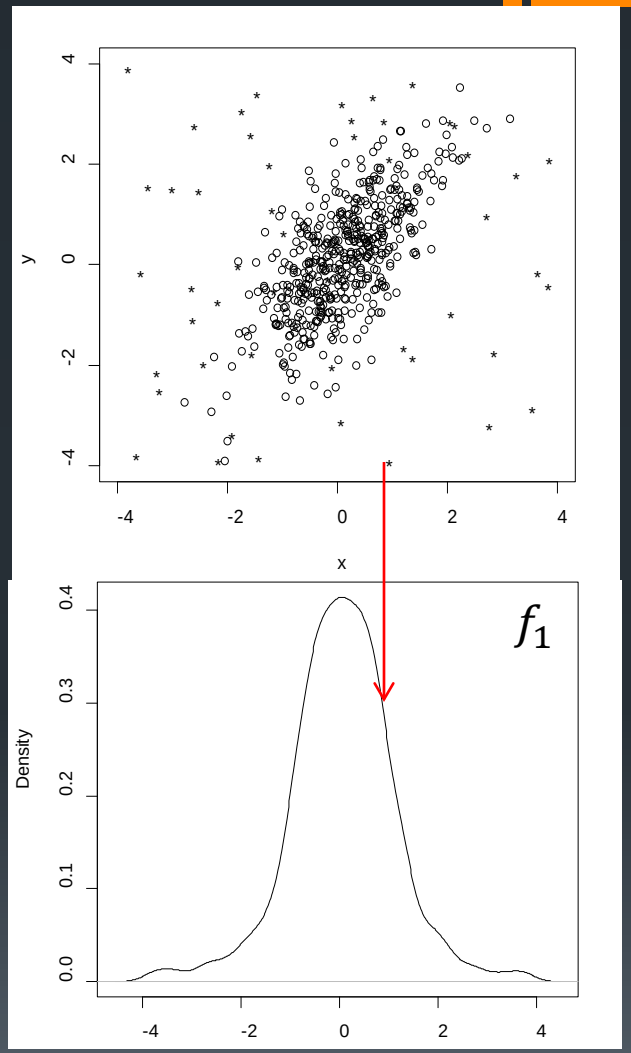
LODA: Lightweight Online Detector of Anomalies [Pevny, 2016]

- $\Pi_1, \ldots, \Pi_M$ set of M sparse random projections
- $f_1, \ldots, f_M$ corresponding 1-dimensional density estimators
- $S(x) = \frac{1}{M}\sum_m -\log f_m(x)$ average "surprise"



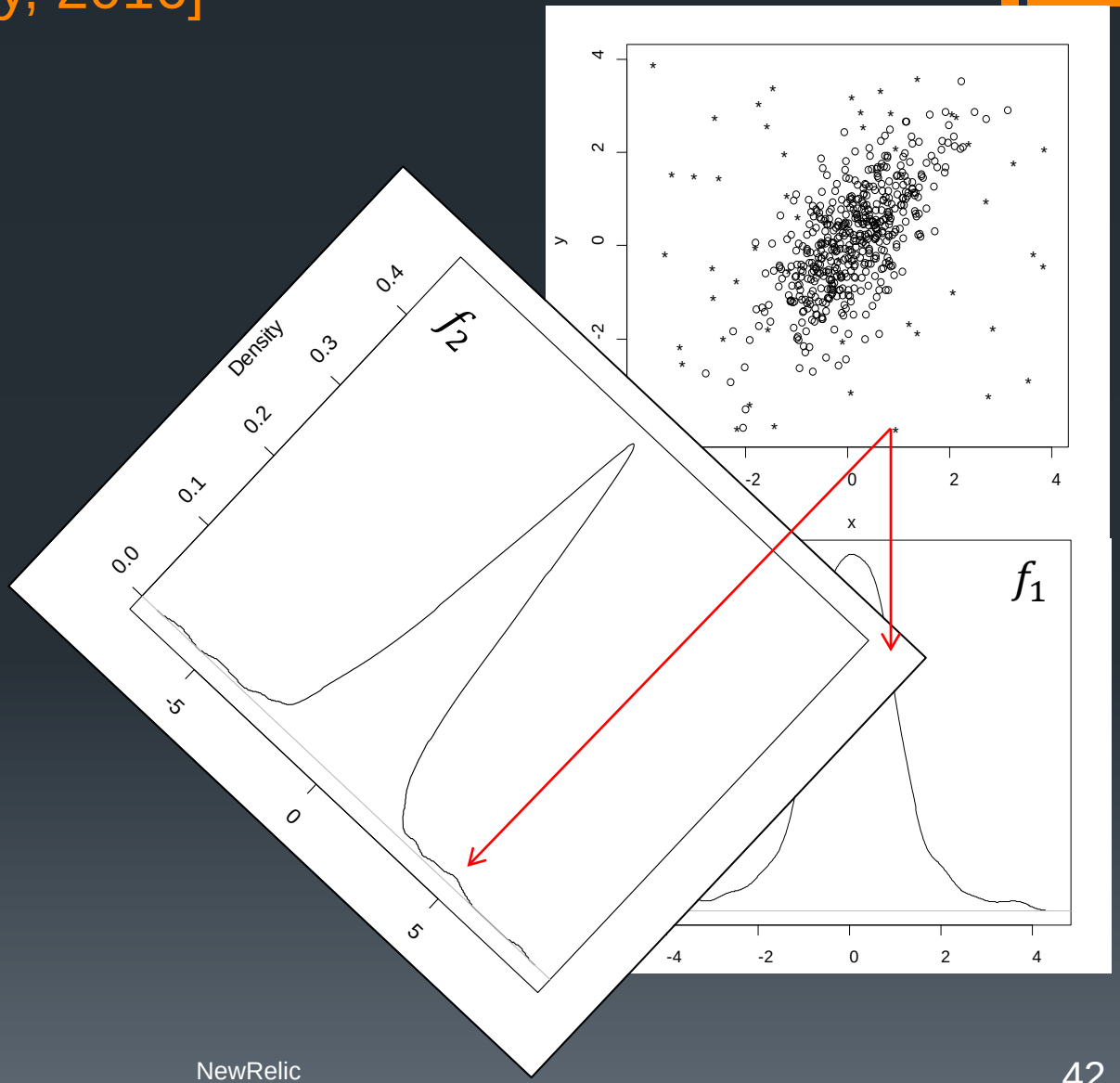
LODA: Lightweight Online Detector of Anomalies [Pevny, 2016]

- $\Pi_1, \ldots, \Pi_M$ set of M sparse random projections
- $f_1, \ldots, f_M$ corresponding 1-dimensional density estimators
- $S(x) = \frac{1}{M}\sum_m -\log f_m(x)$ average "surprise"



LODA: Lightweight Online Detector of Anomalies [Pevny, 2016]

- $\Pi_1, \ldots, \Pi_M$ set of M sparse random projections
- $f_1, \ldots, f_M$ corresponding 1-dimensional density estimators
- $S(x) = \frac{1}{M}\sum_m -\log f_m(x)$ average "surprise"



Benchmarking Study
[Andrew Emmott]

- Most AD papers only evaluate on a few datasets
- Often proprietary or very easy (e.g., KDD 1999)
- Research community needs a large and growing collection of public anomaly benchmarks

[Emmott, Das, Dietterich, Fern, Wong, 2013; KDD ODD-2013]
[Emmott, Das, Dietterich, Fern, Wong. 2016; arXiv 1503.01158v2]

Benchmarking Methodology

- Select 19 data sets from UC Irvine repository
- Choose one or more classes to be “anomalies”; the rest are “nominals”
- Manipulate
  - Relative frequency
  - Point difficulty
  - Irrelevant features
  - Clusteredness
- 20 replicates of each configuration
- Result: 25,685 Benchmark Datasets

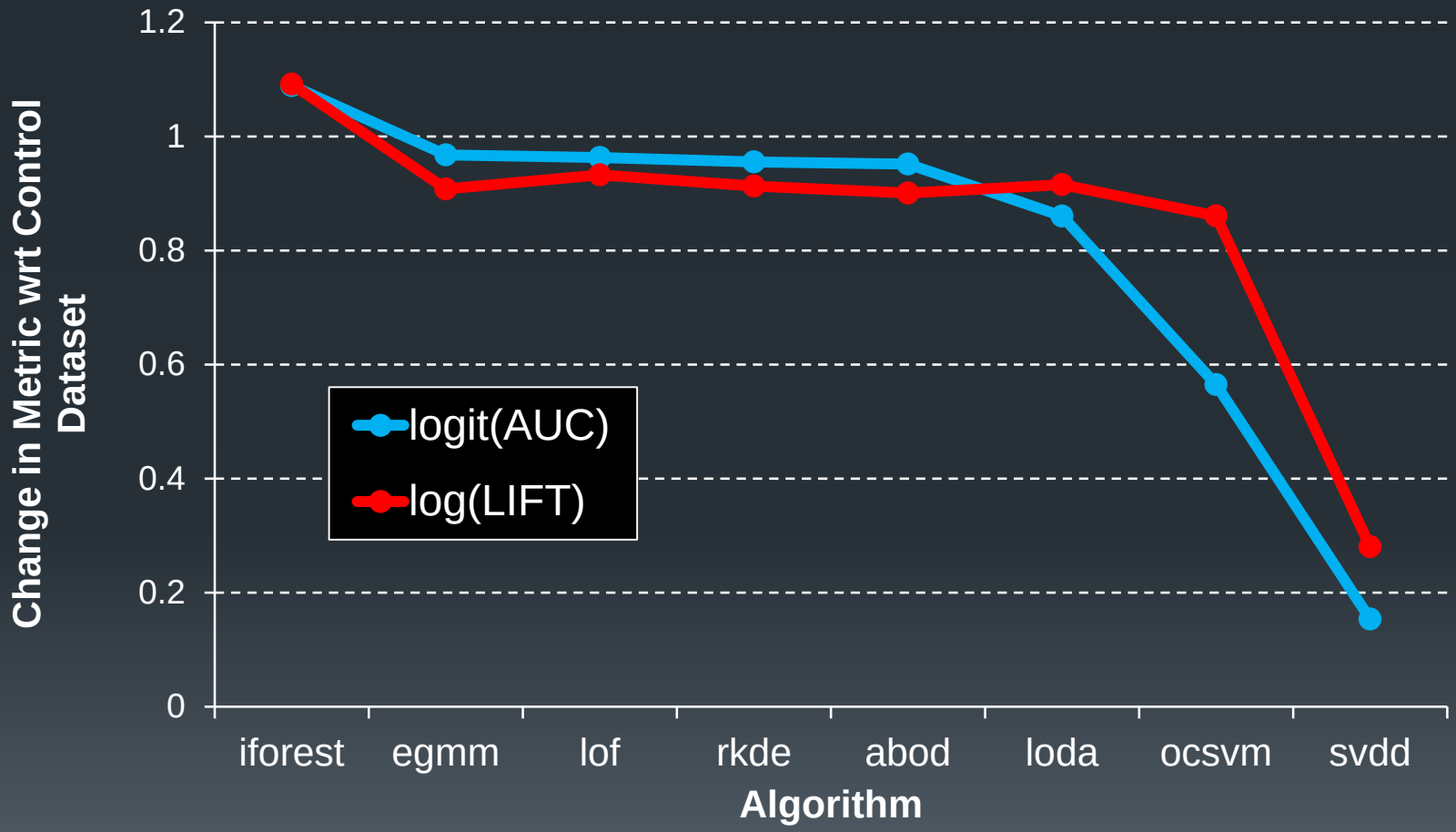
Algorithms

- Density-Based Approaches
  - RKDE: Robust Kernel Density Estimation (Kim & Scott, 2008)
  - EGMM: Ensemble Gaussian Mixture Model (our group)
- Quantile-Based Methods
  - OCSVM: One-class SVM (Schoelkopf, et al., 1999)
  - SVDD: Support Vector Data Description (Tax & Duin, 2004)
- Neighbor-Based Methods
  - LOF: Local Outlier Factor (Breunig, et al., 2000)
  - ABOD: kNN Angle-Based Outlier Detector (Kriegel, et al., 2008)
- Projection-Based Methods
  - IFOR: Isolation Forest (Liu, et al., 2008)
  - LODA: Lightweight Online Detector of Anomalies (Pevny, 2016)

Analysis

- Linear ANOVA
  - $metric \sim rf + pd + cl + ir + mset + algo$
    - rf: relative frequency
    - pd: point difficulty
    - cl: normalized clusteredness
    - ir: irrelevant features
    - mset: “Mother” set
    - algo: anomaly detection algorithm
- Validate the effect of each factor
- Assess the *algo* effect while controlling for all other factors

Algorithm Comparison



How Much Training Data is Needed?

- We looked at learning curves for anomaly detection
- On most benchmarks, we only need a few thousand points to get good detection rates
- That is certainly better than $\exp\frac{d+4}{2}$
- What is going on?

Rare Pattern Anomaly Detection
[Siddiqui, et al.; UAI 2016]

- A pattern $h: \Re^d \rightarrow \{0,1\}$ is an indicator function for a measurable region in the input space
  - Examples:
    - Half planes
    - Axis-parallel hyper-rectangles in $[-1,1]^d$
- A pattern space $\mathcal{H}$ is a set of patterns (countable or uncountable)

Rare and Common Patterns

- Let μ be a fixed measure over $\Re^d$
  - Typical choices:
    - uniform over $[-1, +1]^d$
    - standard Gaussian over $\Re^d$
- $\mu(h)$ is the measure of the pattern defined by h
- Let p be the "nominal" probability density defined on $\Re^d$ (or on some subset)
- $p(h)$ is the probability of pattern h
- A pattern h is τ-rare if

$$f(h) = \frac{p(h)}{\mu(h)} \leq \tau$$

- Otherwise it is τ-common

Rare and Common Points

- A point x is τ-rare if there exists a τ-rare h such that $h(x) = 1$
- Otherwise a point is τ-common

- Goal: An anomaly detection algorithm should output all τ-rare points and not output any τ-common points

PAC-RPAD

- Algorithm $\mathcal{A}$ is PAC-RPAD for
  - pattern space $\mathcal{H}$,
  - measure μ,
  - parameters τ, ϵ, δ

  if for any probability density p and any τ, $\mathcal{A}$ draws a sample from p and with probability $1 - \delta$

  detects all τ-rare points and rejects all $(\tau + \epsilon)$-common points in the sample

- ϵ allows the algorithm some margin for error
- If a point is between τ-rare and $(\tau + \epsilon)$-common, the algorithm can treat it arbitrarily
- Running time polynomial in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, and $\frac{1}{\tau}$, and some measure of the complexity of $\mathcal{H}$

RarePatternDetect

- Draw a sample of size $N(\epsilon, \delta)$ from p
- Let $\hat{p}(h)$ be the fraction of sample points that satisfy h
- Let $\hat{f}(h) = \frac{\hat{p}(h)}{\mu(h)}$ be the estimated rareness of h
- A query point x_q is declared to be an anomaly if there exists a pattern $h \in \mathcal{H}$ such that $h(x_q) = 1$ and $\hat{f}(h) \leq \tau$.

Results

- Theorem 1: For any finite pattern space $\mathcal{H}$, RarePatternDetect is PAC-RPAD with sample complexity

$$N(\epsilon, \delta) = O\left(\frac{1}{\epsilon^2}\left(\log|\mathcal{H}| + \log\frac{1}{\delta}\right)\right)$$

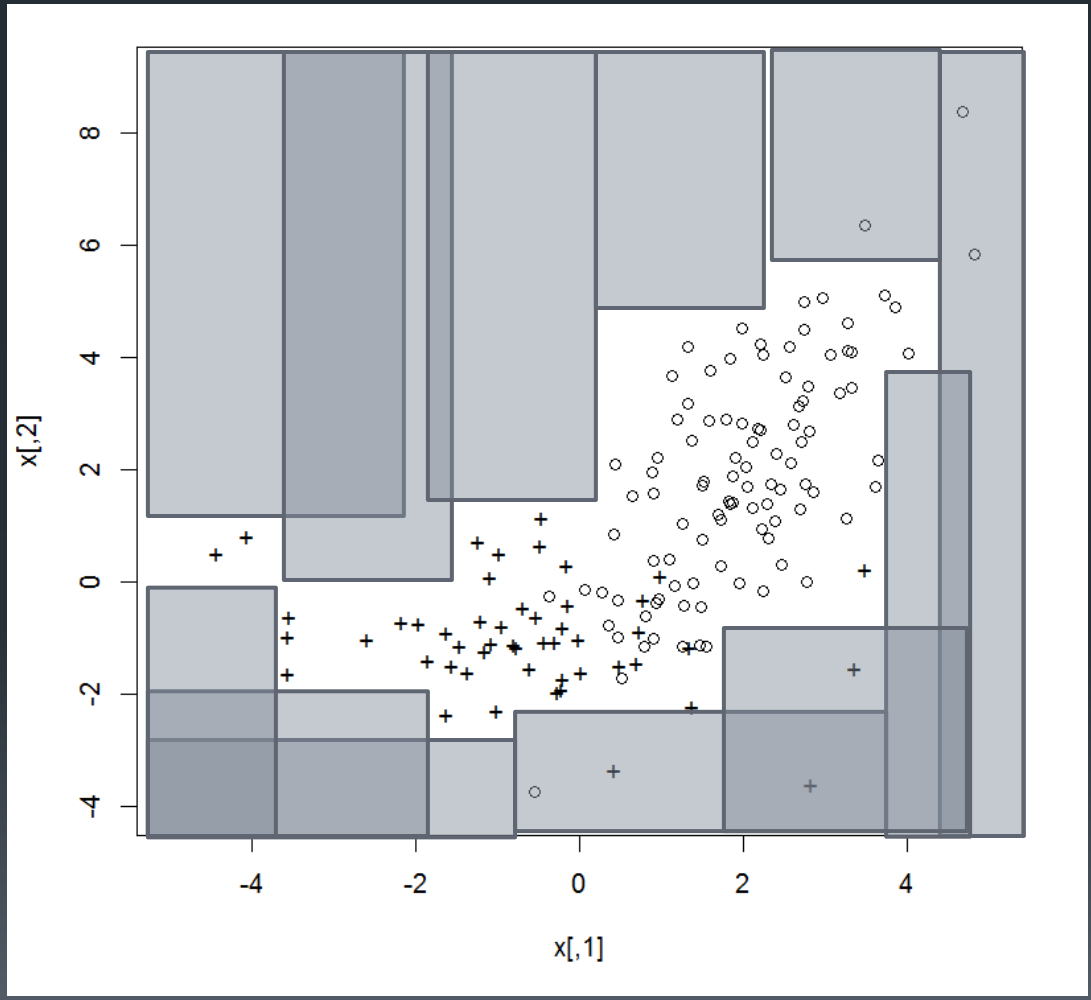
- Theorem 2: For any pattern space $\mathcal{H}$ with finite VC dimension $\mathcal{V}_{\mathcal{H}}$, RarePatternDetect is PAC-RPAD with sample complexity

$$N(\epsilon, \delta) = O\left(\frac{1}{\epsilon^2}\left(\mathcal{V}_{\mathcal{H}} \log\frac{1}{\epsilon^2} + \log\frac{1}{\delta}\right)\right)$$

Intuition: Surround the data with rare patterns

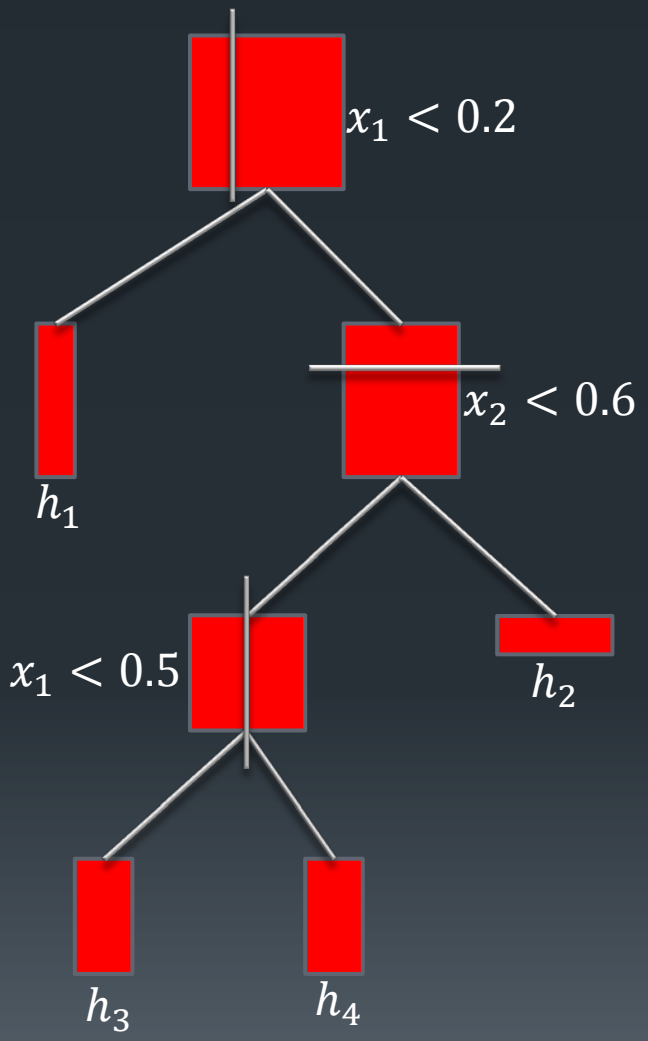
If a query point hits one of the rare patterns, declare it to be an anomaly

We only estimate the probability of each pattern, not the full density

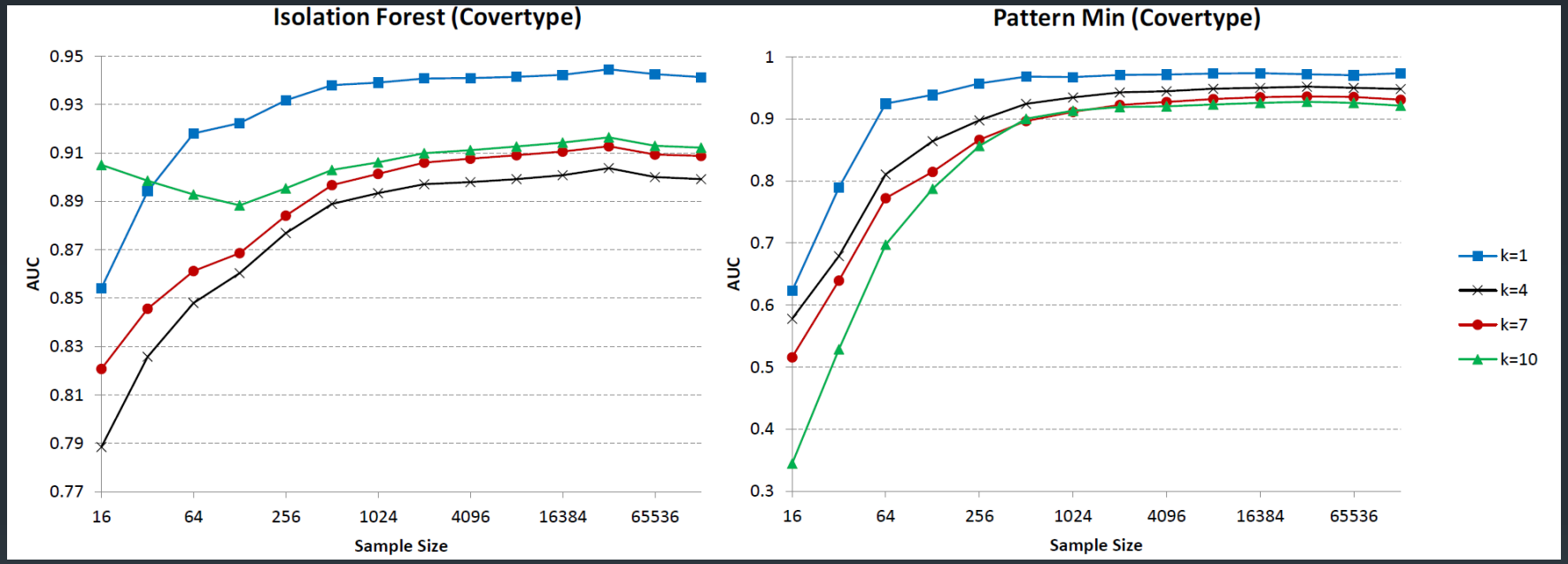


Isolation RPAD

- Grow an isolation forest
  - Each tree is only grown to depth k
  - Each leaf defines a pattern h
  - μ is the volume (Lebesgue measure)
  - Compute $\hat{f}(h)$ for each leaf
- Details
  - Grow the tree using one sample
  - Estimate $\hat{f}$ using a second sample
  - Score query point(s)

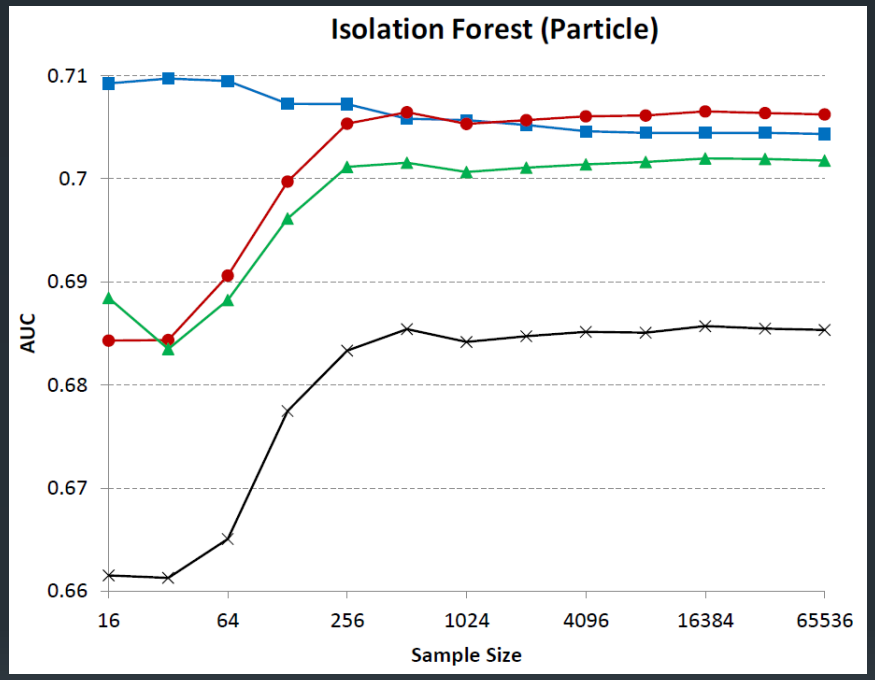


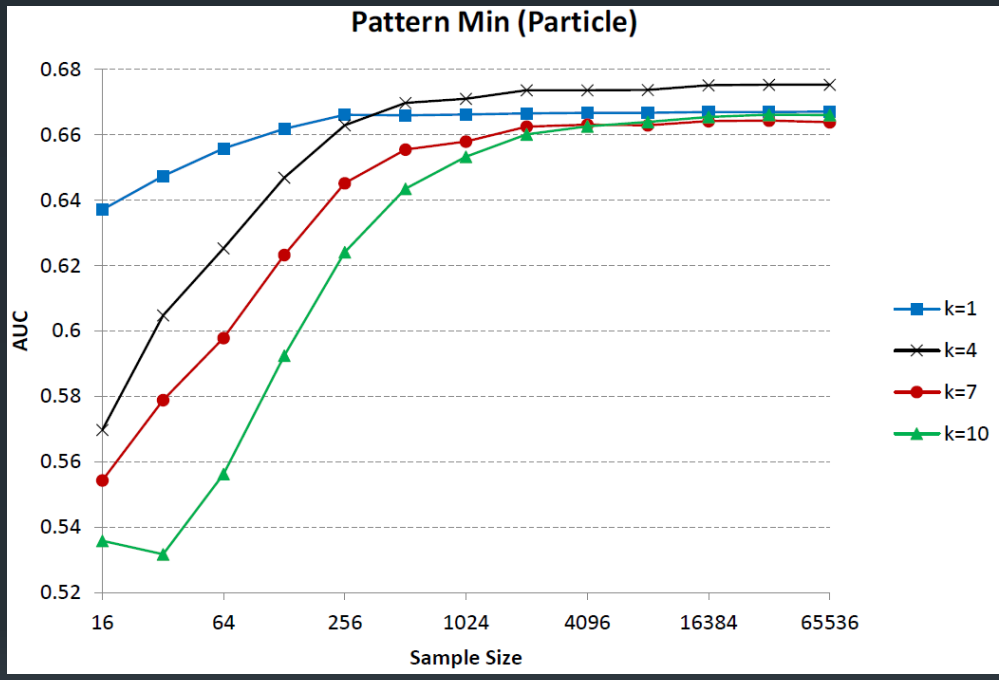
Results: Covertype



- PatternMin is slower, but eventually beats IFOREST

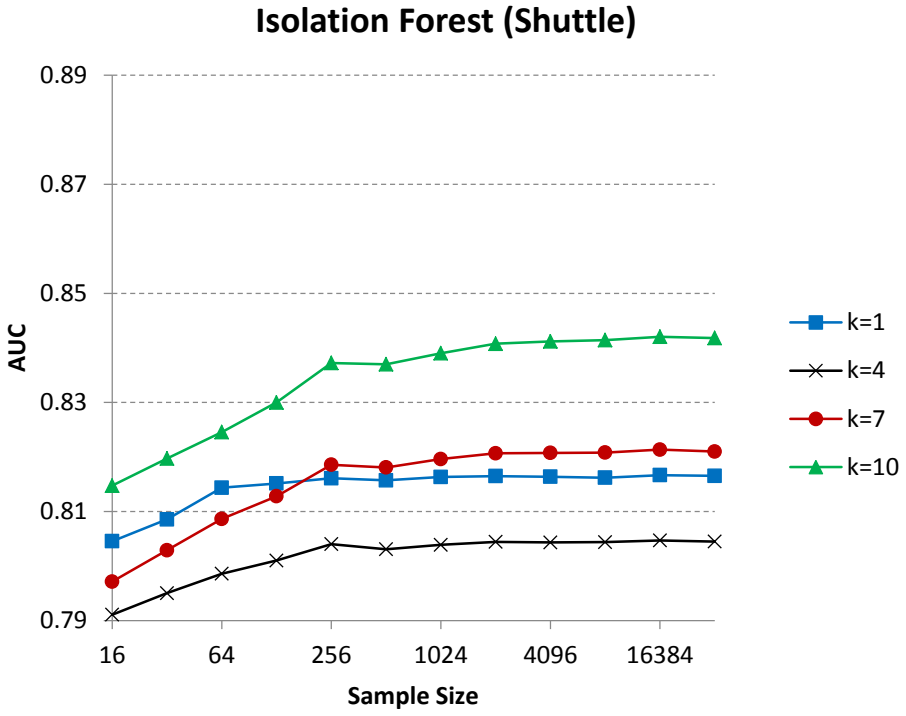
Results: Particle

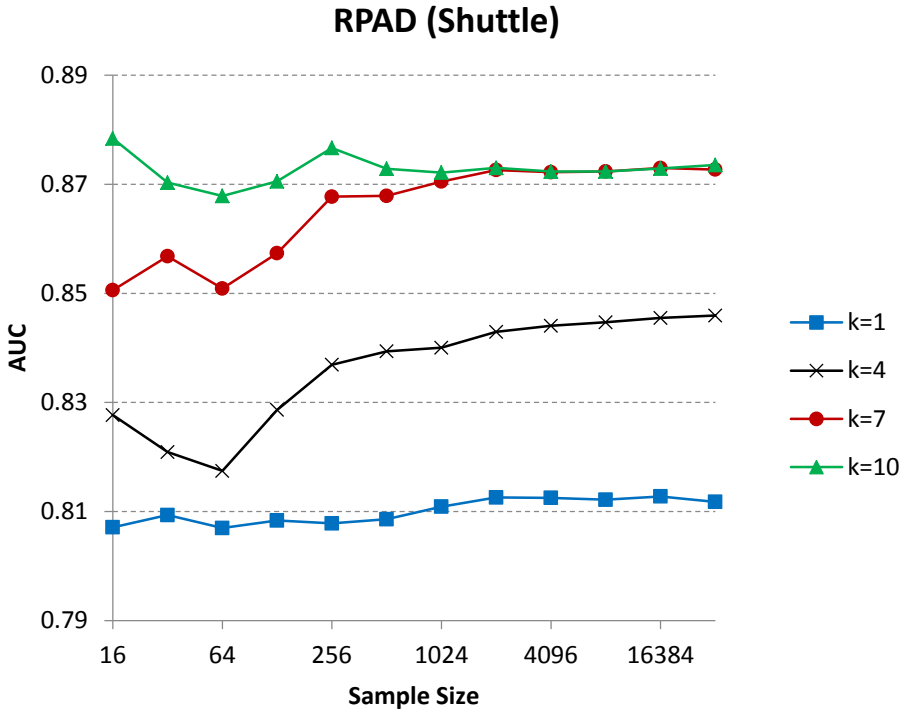




- IFOREST is much better better

Results: Shuttle





- PatternMin is consistently beats iForest for $k > 1$

RPAD Conclusions

- The PAC-RPAD theory seems to capture the behavior of algorithms such as IFOREST
- It is easy to design practical RPAD algorithms
- Theory requires extension to handle sample-dependent pattern spaces $\mathcal{H}$

Summary

- Density Estimation
  - Parametric Density Estimation
  - Mixture Models
  - Kernel Density Estimation
  - Neural Density Estimation
- Anomaly Detection
  - Distance-based methods
  - Isolation Forest and LODA
  - RPAD theory

Bibliography

- Silverman, B. W. (2018). *Density estimation for statistics and data analysis*. Routledge. [updated version of classic book]
- Sheather, S. J. (2004). Density Estimation. *Statistical Science*, *19*(4), 588–597. Recent survey on kernel density estimation
- Hastie, T., Tibshirani, R., & Friedman, J. (2011). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (2nd Edition). New York: Springer Verlag. Good textbook descriptions
- Papamakarios, G., Pavlakou, T., & Murray, I. (2017). Masked Autoregressive Flow for Density Estimation. In *NIPS 2017*. Retrieved from http://arxiv.org/abs/1705.07057
- Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation Forest. In *2008 Eighth IEEE International Conference on Data Mining* (pp. 413–422). Ieee. http://doi.org/10.1109/ICDM.2008.17
- Pevný, T. (2015). Loda: Lightweight on-line detector of anomalies. *Machine Learning*, (November 2014). http://doi.org/10.1007/s10994-015-5521-0
- Emmott, A., Das, S., Dietterich, T., Fern, A., & Wong, W.-K. (2015). *Systematic construction of anomaly detection benchmarks from real data*. http://arxiv.org/1503.01158v2
- Siddiqui, A., Fern, A., Dietterich, T. G., & Das, S. (2016). Finite Sample Complexity of Rare Pattern Anomaly Detection. In *Proceedings of UAI-2016*.