
Isolation Forest

Fei Tony Liu, Kai Ming Ting, e Zhi-Hua Zhou

Eduardo Braga

Tópicos

Motivação

Isolation Forest

Anomaly Score

Resultados

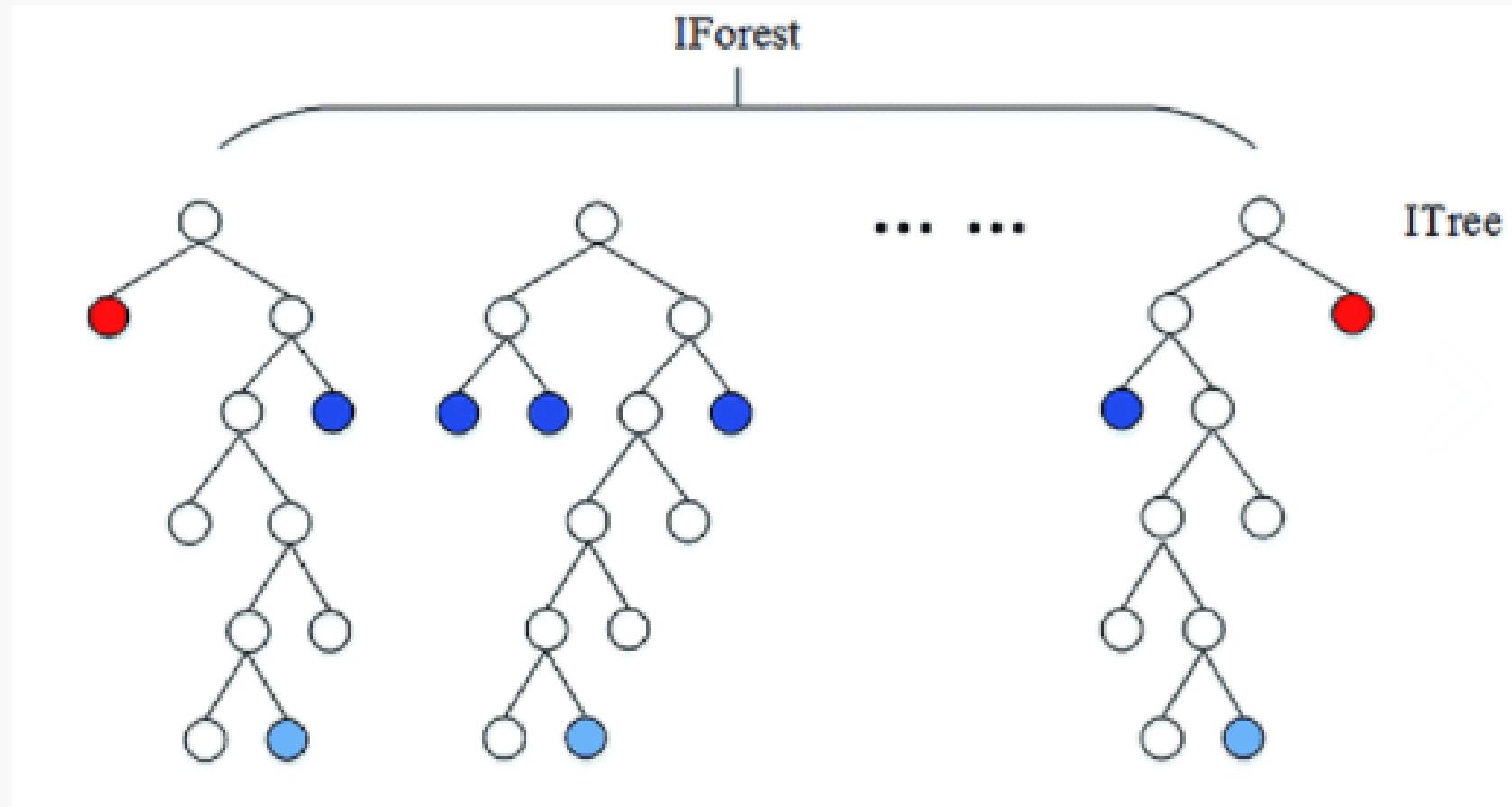
MTSA

Motivação

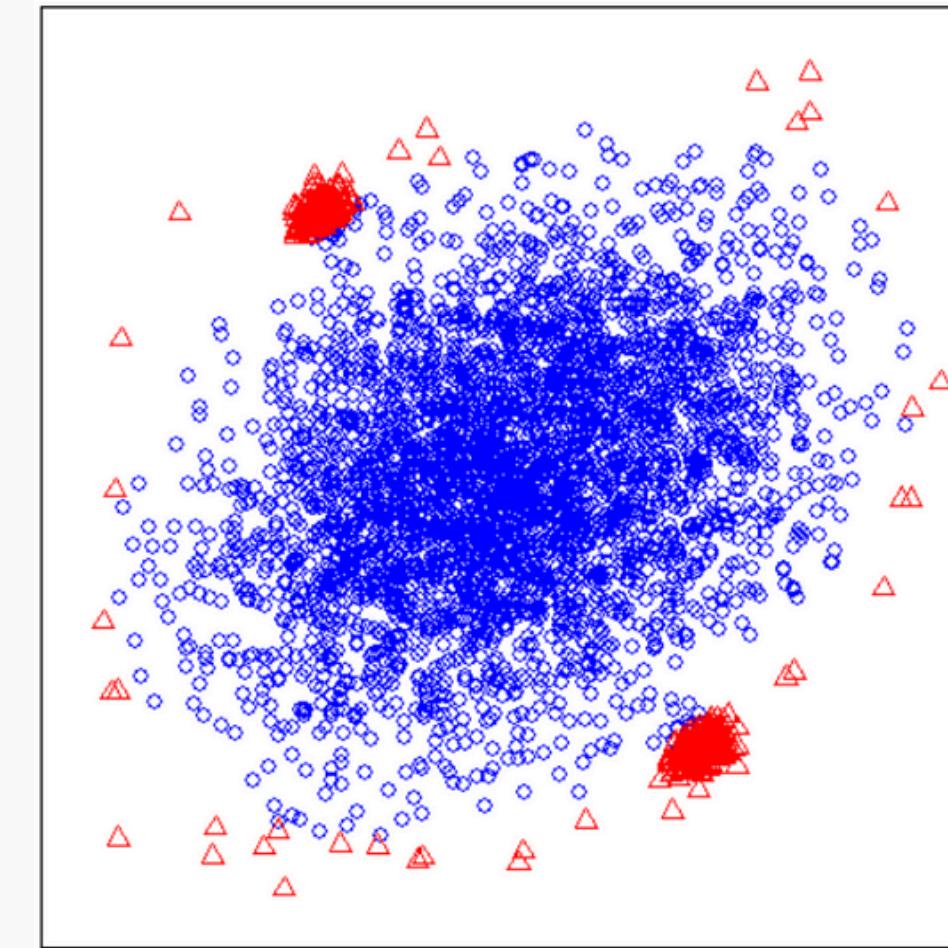
A maioria das abordagens existentes baseadas em modelos para detecção de anomalias constrói um perfil de instâncias normais e, em seguida, identifica as instâncias que não se conformam ao perfil normal como anomalias. Este artigo propõe um método baseado em modelos fundamentalmente diferente que isola explicitamente anomalias em vez de perfis de pontos normais. Até onde sabemos, o conceito de isolamento não foi explorado na literatura atual.

Motivação

Uma Mudança de Paradigma:
Modelar o normal x Detectar a anomalia



A Hipótese "Poucos e Diferentes":



1. elas são a minoria, consistindo em menos instâncias, e
2. elas têm valores de atributos que são muito diferentes daqueles das instâncias normais

Isolation Forest

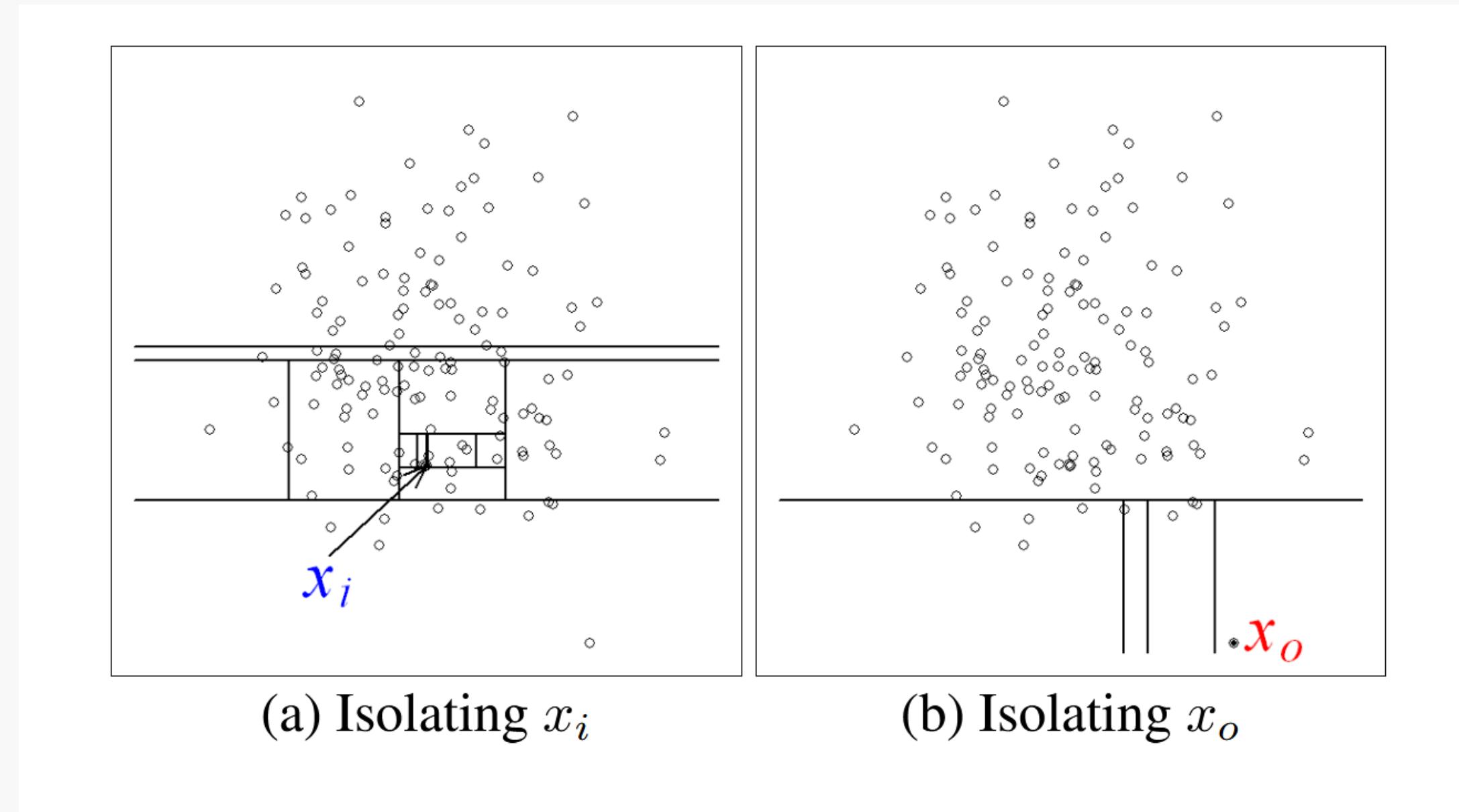
Algorithm 2 $iTree(X, e, l)$

Input: Input data - X ,
Current tree height t ,
Height limit l

Output: An iTree

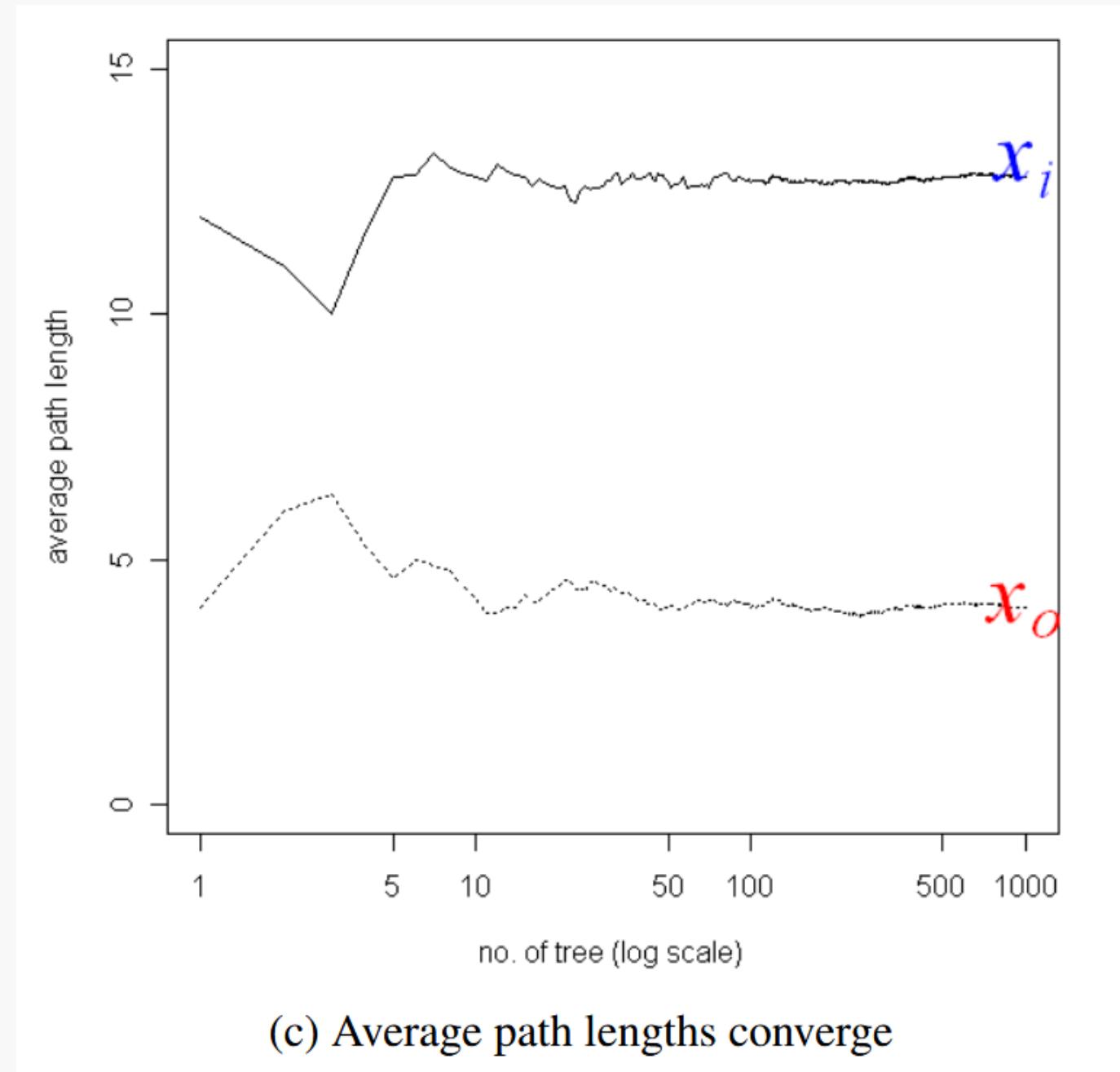
```
1: if  $e \geq l$  or  $|X| \leq 1$  then
2:     return exNode {size  $\leftarrow |X|$ }
3: else
4:     let  $Q$  be a list of attributes in  $X$ 
5:     randomly select an attribute  $q \in Q$ 
6:     randomly select a split point  $p$  from  $\max$  and  $\min$  values of attribute
     $q$  in  $X$ 
7:      $X_l \leftarrow filter(X, q < p)$ 
8:      $X_r \leftarrow filter(X, q \geq p)$ 
9:     return inNode{Left  $\leftarrow iTree(X_l, e + 1, l)$ ,
10:                  Right  $\leftarrow iTree(X_r, e + 1, l)$ ,
11:                  SplitAtt  $\leftarrow q$ ,
12:                  SplitValue  $\leftarrow p$ }
13: end if
```

Isolation Forest



Isolation Forest

Uma única iTree é inherentemente instável. Devido à natureza aleatória da seleção de atributos e valores de corte, o comprimento do caminho para um ponto específico pode variar drasticamente de uma árvore para outra.



Isolation Forest

Algorithm 1 *iForest*(X, t, ψ)

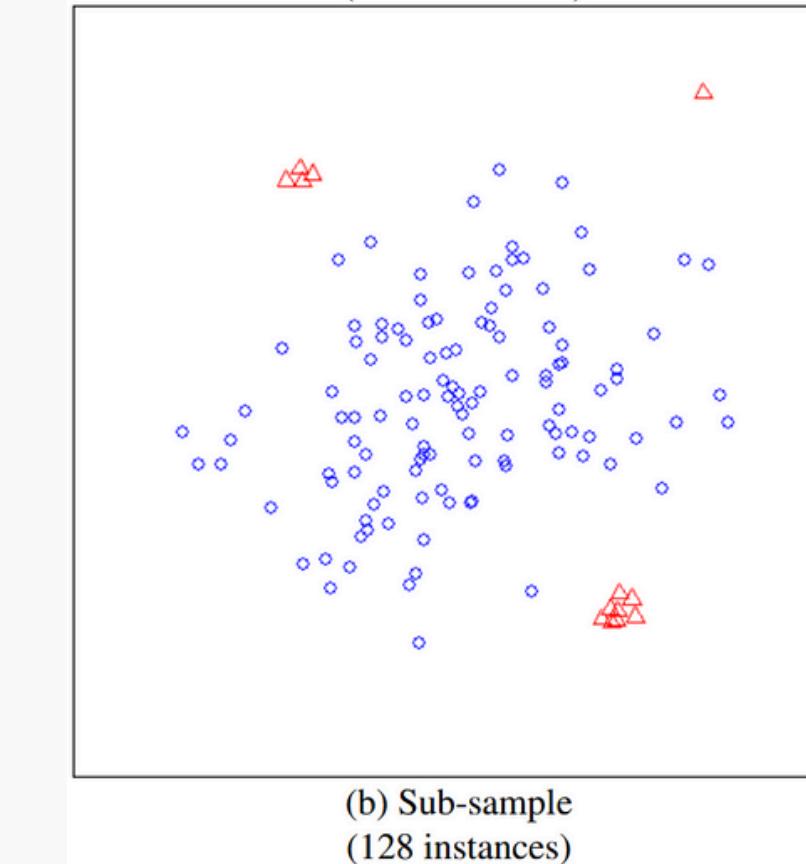
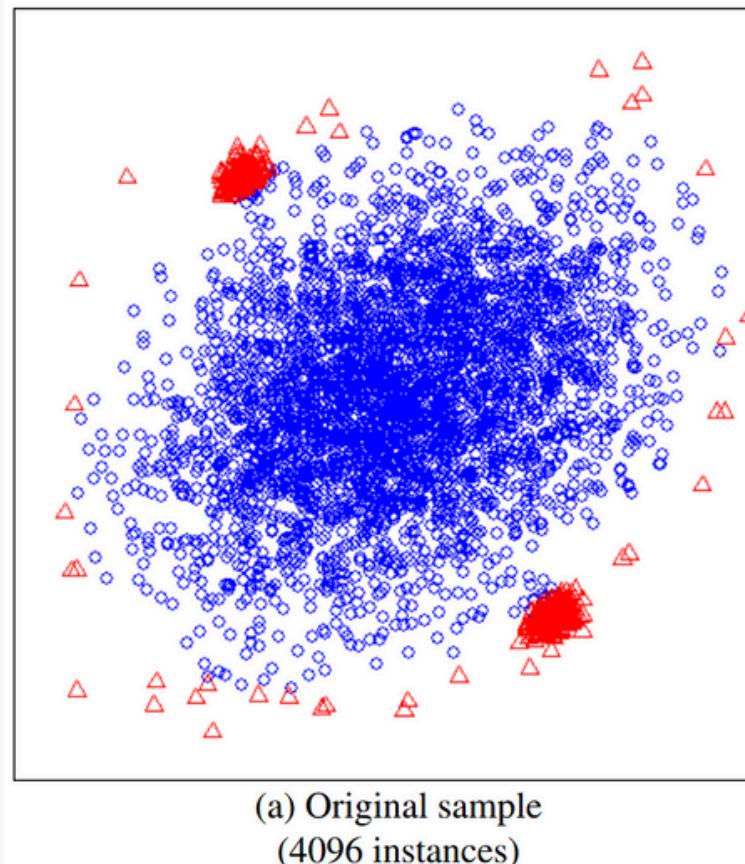
Input: Input data - X ,
Number of trees t ,
Sub-sampling size ψ

Output: A set of iTrees

- 1: Initialize Forest
 - 2: set height limit $l = \text{ceiling}(\log_2 \psi)$
 - 3: **for** $i = 1$ to t **do**
 - 4: $X' \leftarrow \text{sample}(X, \psi)$
 - 5: $\text{Forest} \leftarrow \text{Forest} \cup \text{iTree}(X', 0, l)$
 - 6: **end for**
 - 7: **return** Forest
-

Isolation Forest

Ao trabalhar com uma pequena amostra aleatória, o algoritmo cria uma versão simplificada do espaço do problema, onde as características definidoras das anomalias "poucas e diferentes" são amplificadas.



Swamping e Masking

Isolation Forest

Algorithm 3 *PathLength*(x, T, e)

Input: An instance - x ,
An iTree - T ,
Current path length - e ; to be initialized to zero when first called

Output: Path length of x

```
1: if  $T$  is an external node then
2:     return  $e + c(T.size)$  { $c(\cdot)$  is defined in Equation 1}
3: end if
4:  $a \leftarrow T.splitValue$  then
5:     if  $x_a < T.splitValue$  then
6:         return PathLength( $x, T.left, e + 1$ )
7:     else { $x_a \geq T.splitValue$ }
8:         return PathLength( $x, T.right, e + 1$ )
9: end if
```

Anomaly Score

O comprimento médio do caminho de uma busca mal sucedida em uma Árvore de Busca Binária (BST)

$$c(n) = 2H(n - 1) - \frac{2(n-1)}{n},$$

$H(n-1)$ pode ser estimado por:

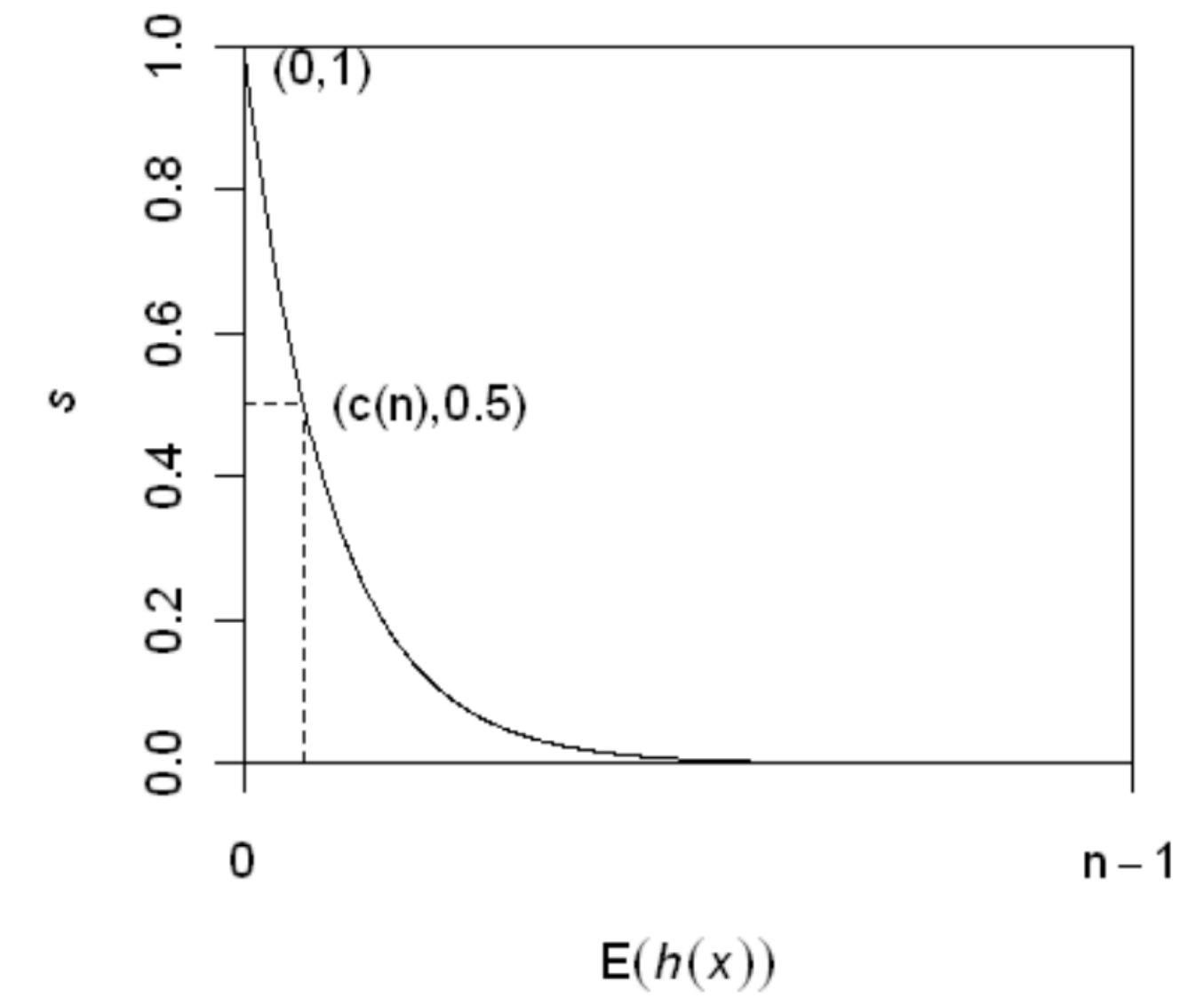
$$\ln(n-1) + 0.5772156649
(\text{a constante de Euler})$$

$C(n)$ representa o comprimento de caminho esperado teórico para um ponto em uma árvore binária construída aleatoriamente com n elementos. Ela serve como uma linha de base universal para o que um comprimento de caminho "médio" ou "normal" deveria ser, independentemente da distribuição dos dados.

Anomaly Score

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

- Se $E(h(x)) \rightarrow 0$ (muito curto), então $s \rightarrow 1$. Isso indica uma anomalia clara.
- Se $E(h(x)) \rightarrow c(n)$ (comprimento médio), então $s \rightarrow 0.5$. Isso indica um ponto normal.
- Se $E(h(x)) \rightarrow n-1$ (muito longo), então $s \rightarrow 0$. Isso indica um ponto inequivocamente normal.

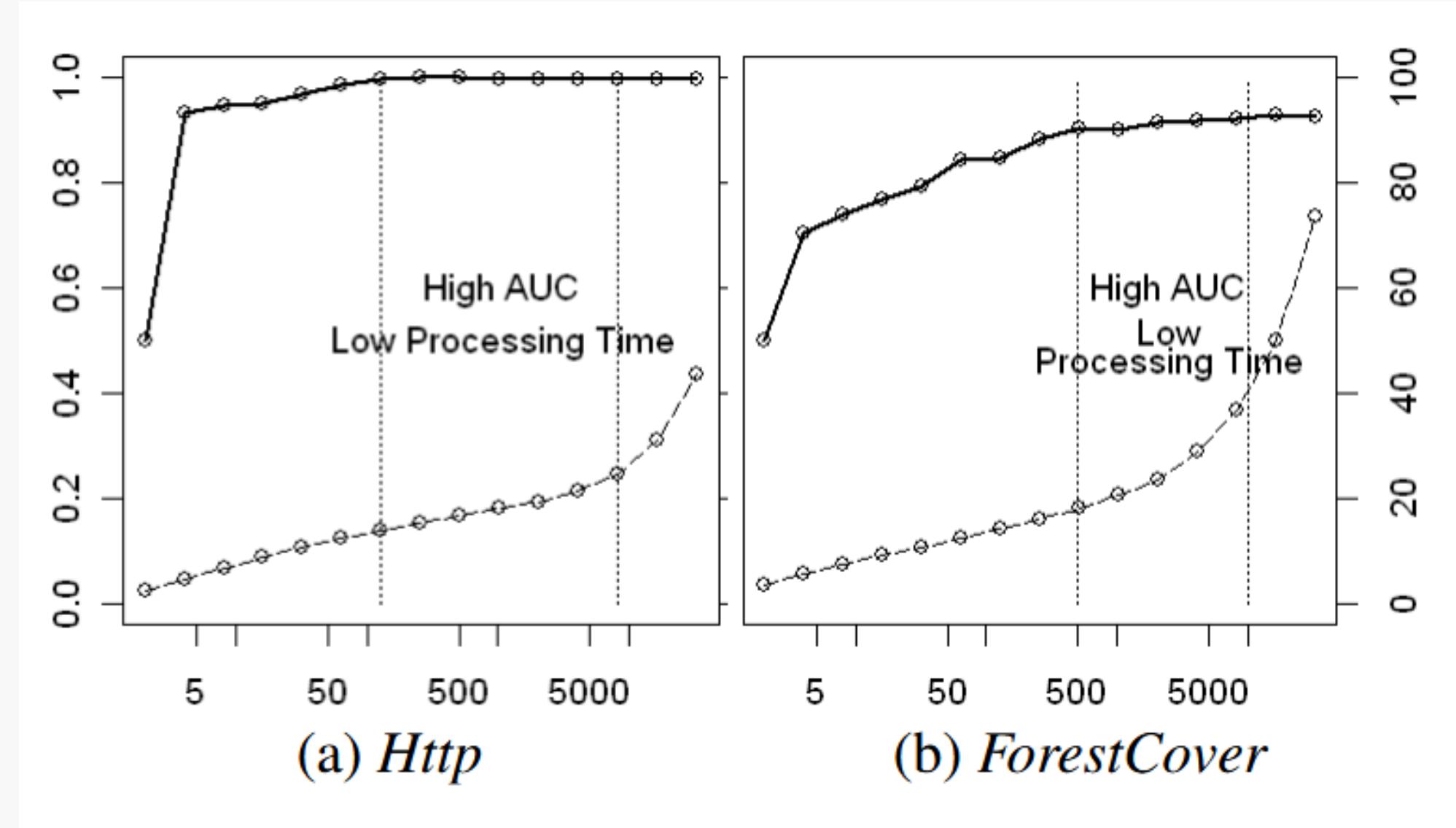


Resultados

	AUC				Time (seconds)							
	iForest	ORCA	LOF	RF	iForest			ORCA	LOF	RF		
					Train	Eval.	Total					
Http (KDDCUP99)	1.00	0.36	NA	NA	0.25	15.33	15.58	9487.47	NA	NA		
ForestCover	0.88	0.83	NA	NA	0.76	15.57	16.33	6995.17	NA	NA		
Mulcross	0.97	0.33	NA	NA	0.26	12.26	12.52	2512.20	NA	NA		
Ssmtp (KDDCUP99)	0.88	0.80	NA	NA	0.14	2.58	2.72	267.45	NA	NA		
Shuttle	1.00	0.60	0.55	NA	0.30	2.83	3.13	156.66	7489.74	NA		
Mammography	0.86	0.77	0.67	NA	0.16	0.50	0.66	4.49	14647.00	NA		
Annthyroid	0.82	0.68	0.72	NA	0.15	0.36	0.51	2.32	72.02	NA		
Satellite	0.71	0.65	0.52	NA	0.46	1.17	1.63	8.51	217.39	NA		
Pima	0.67	0.71	0.49	0.65	0.17	0.11	0.28	0.06	1.14	4.98		
Breastw	0.99	0.98	0.37	0.97	0.17	0.11	0.28	0.04	1.77	3.10		
Arrhythmia	0.80	0.78	0.73	0.60	2.12	0.86	2.98	0.49	6.35	2.32		
Ionosphere	0.85	0.92	0.89	0.85	0.33	0.15	0.48	0.04	0.64	0.83		

$O(t.n.\log n + n.t.\log n)$

Resultados



Um tamanho pequeno de subamostragem fornece tanto um alto AUC (eixo y esquerdo, linhas sólidas) quanto um baixo tempo de processamento (eixo y direito, linhas tracejadas, em segundos). O tamanho da subamostragem (eixo x, escala logarítmica) varia em $\psi = 2, 4, 8, 16, \dots, 32768$."

```

131
132 Algoritmo 1: Build
133
134 Input:
135     X          // Conjunto de dados de áudio brutos para treinamento
136     t          // Número de árvores (n_estimators)
137     ψ          // Tamanho da subamostragem (max_samples)
138     c          // Taxa de contaminação (contamination)
139     ?          // Taxa de amostragem do áudio (sampling_rate)
140     ...        // Outros hiperparâmetros (max_features, bootstrap, etc.)
141
142 Output:
143     M_trained // O modelo treinado
144
145 -----
146

```

```

147 // INICIALIZAÇÃO
148
149 // Converter áudio para array NumPy
150 1: wav_to_array ← Wav2Array(sampling_rate=self.sampling_rate)
151 // Converter array de áudio para MFCCs com n_mfcc especificado
152 2: array2mfcc ← Array2Mfcc(sampling_rate=self.sampling_rate)
153 // Combinar as features existentes usando FeatureUnion
154 3: features ← FeatureUnion(self.features)
155
156 // Configuração do IsolationForest
157 4: iforest ← Initialize_IsolationForest(
158 5:   t,
159 6:   ψ,
160 7:   c,
161 8:   ...
162 9: )
163
164 // PIPELINE
165
166 // Define a sequência de operações
167 10: P_model ← Pipeline(
168 12:   steps ← [
169 12:     ("wav2array", wav2array),
170 13:     ("array2mfcc", array2mfcc),
171 15:     ("features", features),
172 16:     ("final_model", iforest)
173 17:   ]
174 18: )
175

```



Obrigado

