DCC
DEPARTAMENTO DE
CIÊNCIA DA COMPUTAÇÃO

XGBOD (Extreme Gradient Boosting Outlier Detection)

Grupo 3 – 3CS
André Luís Ribeiro
Caio Alexandre Campos Maciel
Othávio Ruddá da Cunha Araújo

Segurança em sistemas de Computação (Tópicos em Ciência da Computação - TJ)
17 de outubro de 2019

Apresentação do Artigo

XGBOD: Improving Supervised Outlier Detection with Unsupervised Representation Learning

Yue Zhao and Maciej K. (Toronto, Canadá)

IJCNN: International Joint Conference on Neural Networks (2018)

Trabalho Correlato

- Extensão do trabalho de Micenková et al. (2014, 2015)
  - Proposta de framework para outlier detection
  - Scores de diversos algoritmos não supervisionados são utilizados para gerar novas features -> unsupervised feature engineering
  - As novas features são agregadas às antigas e enviadas a um classificador supervisionado final

Motivação

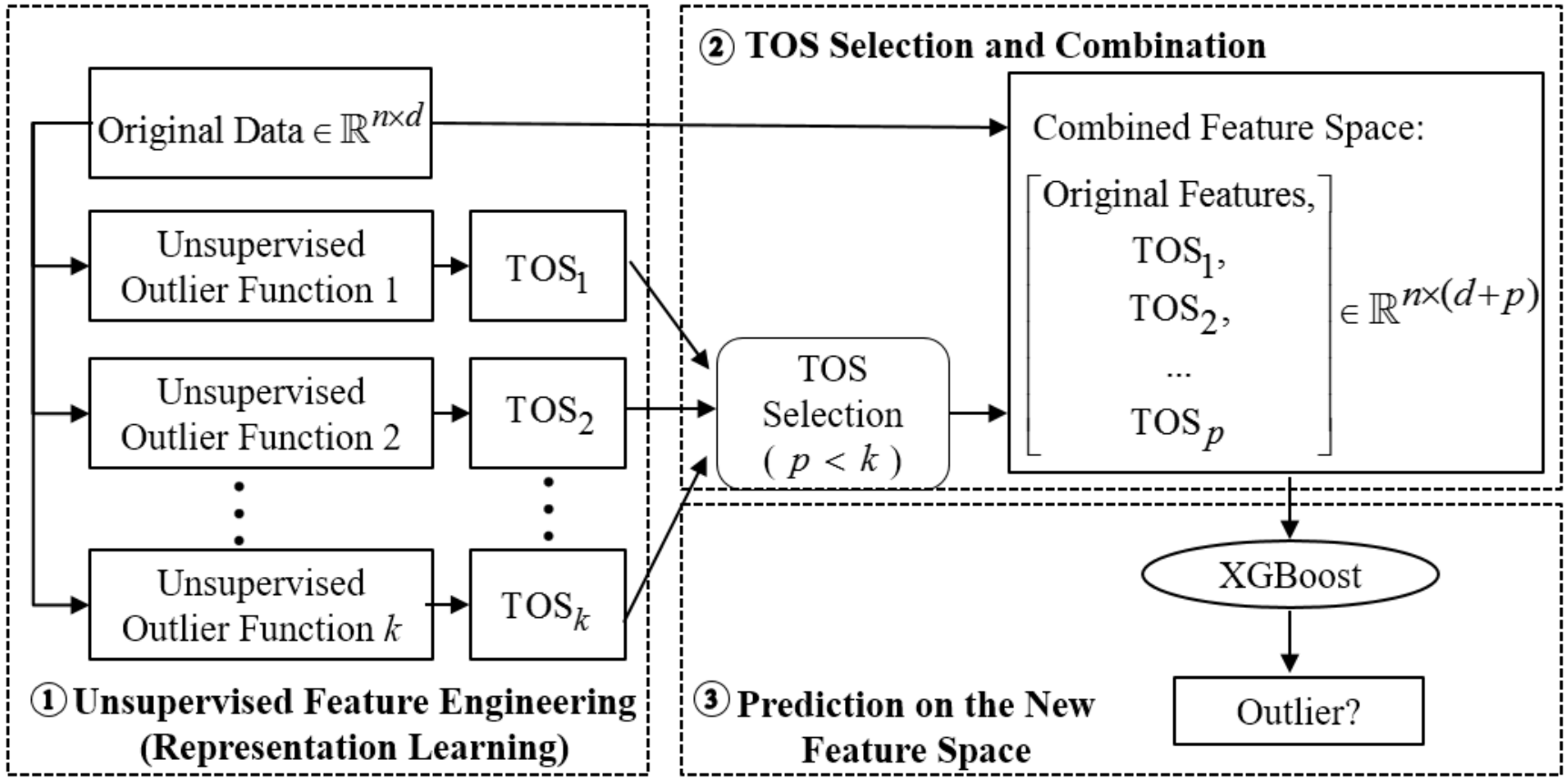
- Algoritmos não supervisionados são ótimos em identificar padrões complexos
- Outros algoritmos semelhantes utilizam Easy Ensemble para lidar com datasets desbalaceados -> custoso

Proposta

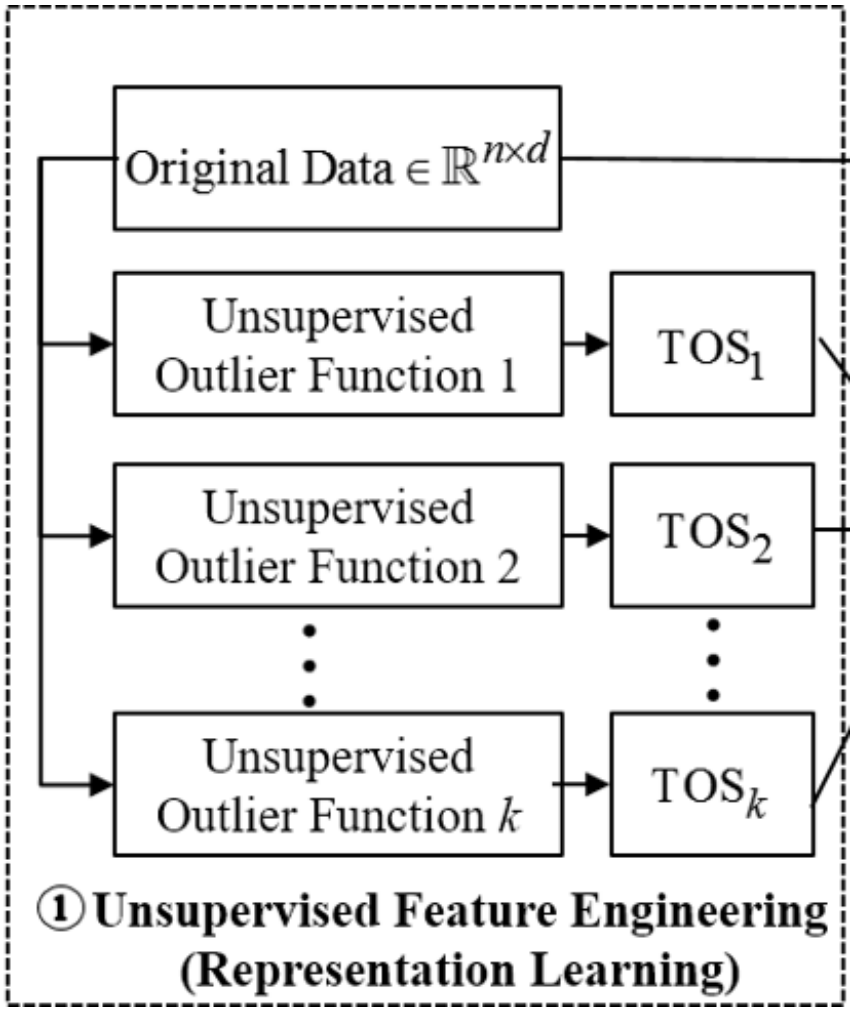
- Testar outro algoritmo como classificador final
- Avaliar diferentes seleções de algoritmos não supervisionados
- Desenvolver um método de seleção de algoritmos não supervisionados para o classificador final

Abordagem (Design do algoritmo)

- Framework de 3 fases



Fase 1: Representação do Aprendizado não-supervisionado



Fase 1: Representação do Aprendizado não-supervisionado

- Unsupervised outlier scores podem ser vistos como forma de representação dos dados originais;
- Dados Originais: $X \in \mathbb{R}^{n \times d}$
- Função que retorna vetor de scoring no dataset X, chamado de transformed outlier scores (TOS): $\Phi_i(X) \in \mathbb{R}^{n \times 1}$
- Matriz obtida combinando k funções de outlier scoring:

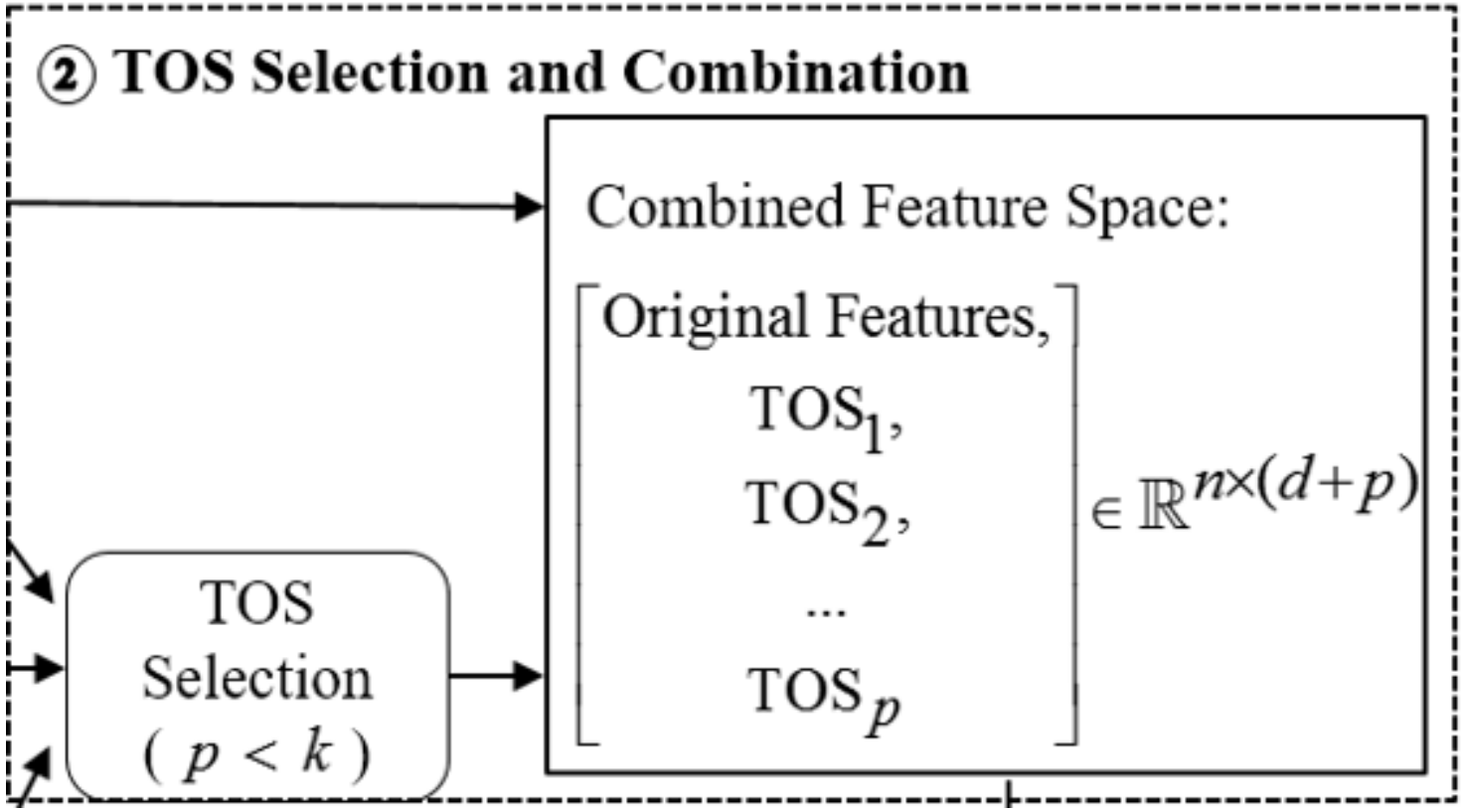
$$\Phi = [\Phi_1, \ldots, \Phi_k]$$

- Matriz de Outlier Scoring: $\Phi(X) = \left[\Phi_1(X)^T, \ldots, \Phi_k(X)^T\right] \in \mathbb{R}^{n \times k}$

Fase 1: Representação do Aprendizado não-supervisionado

- Trade-off entre Diversidade X Acurácia
- Utilizar detectores distintos melhoram a diversidade, mas com os risco de degradar a capacidade de predição
- Portanto um equilíbrio entre diversidade e acurácia deve ser mantido para se conseguir melhores resultados
- Neste estudo, diferentes tipos de métodos não-supervisionados foram utilizados e seus parâmetros trocados para gerar uma maior variação

Fase 2: Seleção TOS



Fase 2: Seleção TOS

- No trabalho de Micenková: $\text{Feature Space}_{new} = [X, \Phi(X)] \in \mathbb{R}^{n \times l}$

  onde $l = (d + k)$

- Neste trabalho, selecionou-se apenas $p\ (p \leq k)$ TOS de $\Phi(X)$ pois:
  - Há TOS que não contribuem para a predição;
  - Execução mais rápida;
  - O novo espaço será menor para o aprendizado.

Fase 2: Seleção TOS

- Três métodos de seleção foram definidos para compor o conjunto S:

1) Seleção Aleatória: Seleciona p TOS aleatoriamente e adiciona à S sem reposição
2) Seleção Precisa: Seleciona as p TOS mais precisas, tomando como medida a ROC curve, por exemplo. $ACC_i = ROC(\Phi_i (X)^T, y)$

Fase 2: Seleção Balanceada TOS

3) Seleção Balanceada: Mantém o equilíbrio entre diversidade e acurácia selecionando as TOS que são ambas precisas e diversas.

Uma função que desconta a acurácia é aplicada:

$$\Psi(\Phi_i) = \frac{ACC_i}{\sum_{j=1}^{\#(S)} \left|\rho(\Phi_i, \Phi_j)\right|}$$

$$\cos(\alpha) = \rho = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} = \frac{\operatorname{cov}(X, Y)}{\sqrt{\operatorname{var}(X) \cdot \operatorname{var}(Y)}}$$

$$\text{subject to } \Phi_j \in \{S\}, ACC_i \geq 0$$

Fase 2: Algoritmo de Seleção Balanceada de TOS

Algorithm 1 Balance Selection

Input: $\Phi = \{\Phi_1, ..., \Phi_k\}$, ground truth y, # of TOS = p
Output: The Set of Selected TOS: S

Initialize: Selected TOS: $S = \{\}$

1. $\Phi(X)_{\max} = max\left(ACC\left(\Phi(X)\right)\right)$ /* most accurate*/
2. $S \leftarrow S \cup \Phi(X)_{\max}$ /*add selected TOS to set S */
3. $\Phi(X) \leftarrow \Phi(X) \setminus \Phi(X)_{\max}$ /*remove from the pool*/
4: **while** $\#(S) < p$ **do**
5: **for** $\Phi(X)_i \in \Phi(X)$ **do**
6. $\Psi(\Phi_i) \leftarrow$ Eq. (5) /*discounted accuracy*/
7. $\Phi(X)_{\max} = \max\left(\Psi(X)_i\right)$
8. $S \leftarrow S \cup \Phi(X)_{\max}$ /*add the current best to set S */
9. $\Phi(X) \leftarrow \Phi(X) / \Phi(X)_{\max}$ /*remove from the pool*/
10. **end for**
11. **end while**
12. **return** S

O(K*p)

Fase 2: Seleção Balanceada de TOS

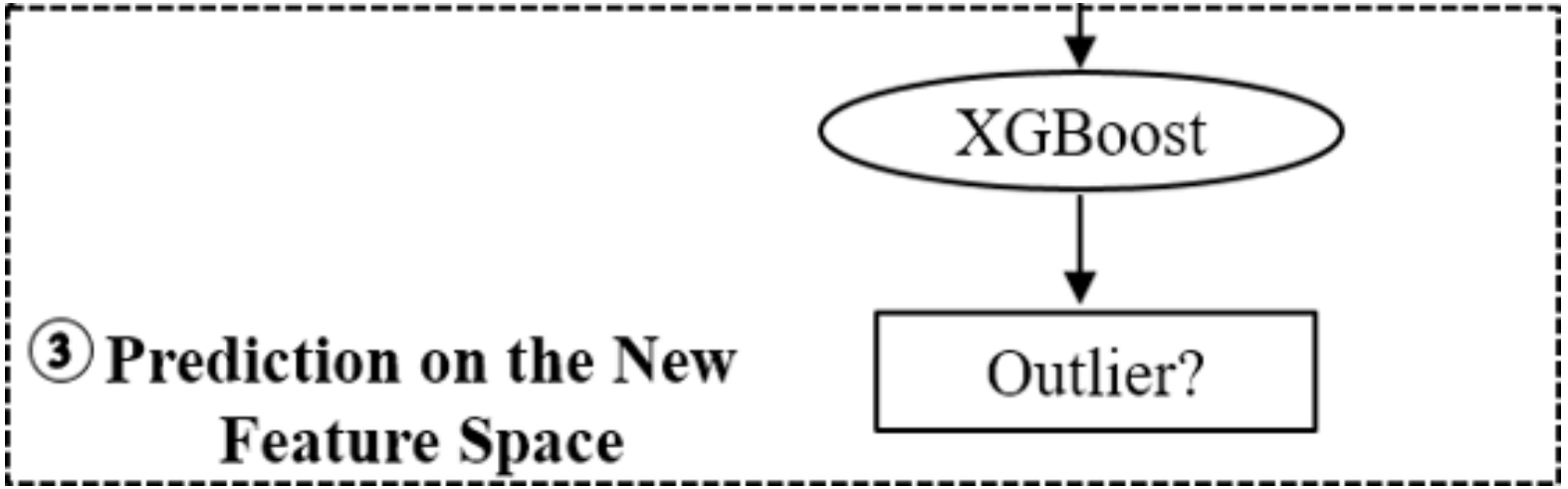
- Como resultado, tem-se p TOS selecionadas como $S \in \mathbb{R}^{n \times p}$

- Por fim, o novo espaço é criado concatenando X com S:

$$\text{Feature Space}_{comb} = [X, S] \in \mathbb{R}^{n \times (d+p)}$$

- Interessante notar que $(k-p)$ TOS foram descartadas, para melhorar a eficiência e predição do algoritmo

Fase 3: Predição com XGBoost

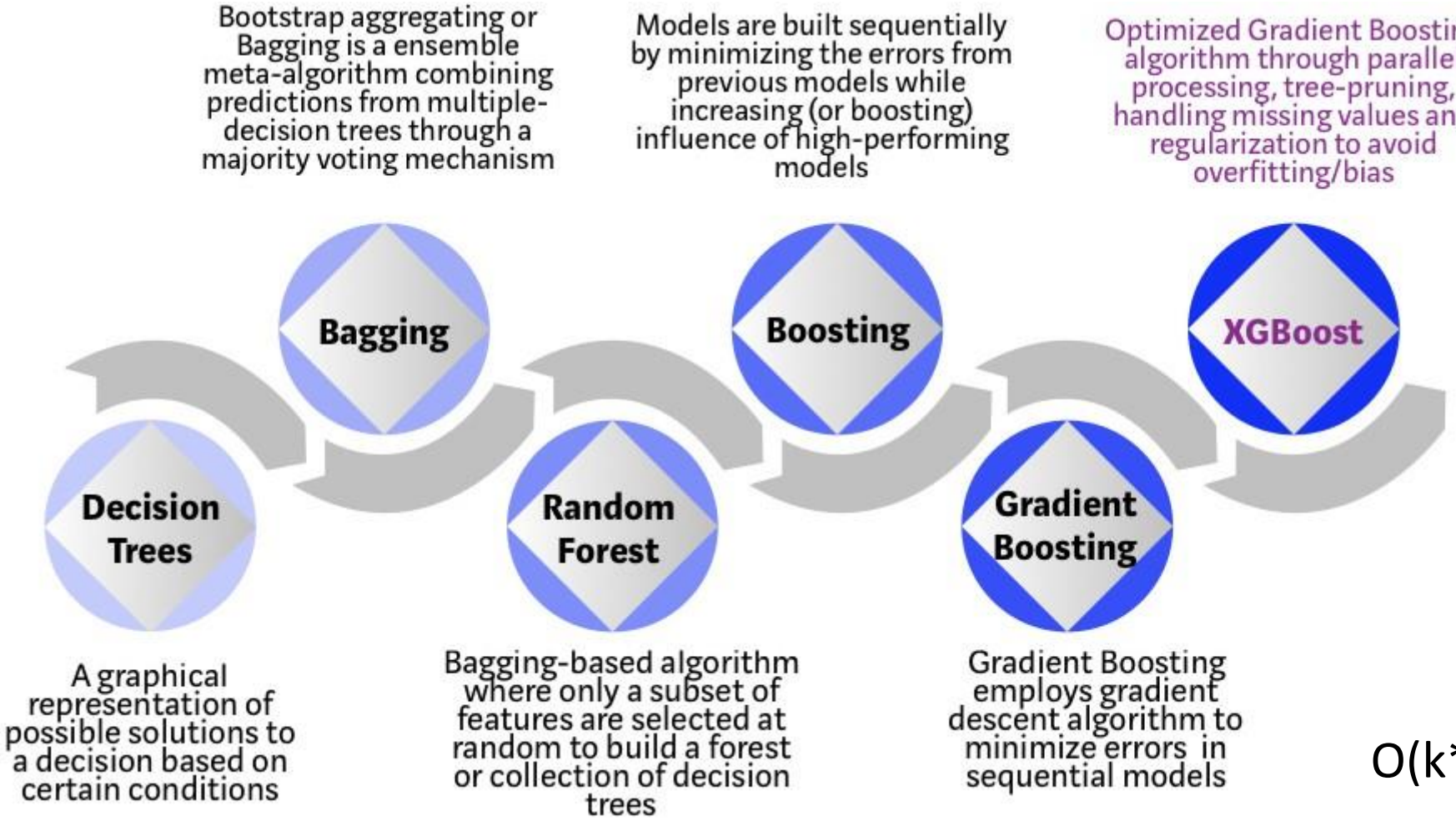


XGBoost

- Design voltado para performance
  - Processamento paralelo
- Conhecido como campeão de desafios no Kaggle
- Lida bem com datasets desbalanceados
- Utiliza um termo de regularização para penalizar funções de alta complexidade (evita overfitting)
- Capaz de gerar *feature importance* do modelo

*Notar que outro algoritmo poderia ser utilizado como classificador final

XGBoost: Visão Geral



$O(k*d||x||\log n)$

Experimentos

- Dois experimentos:
  - Experimento 1: Compara a utilização de todos ou nenhum TOS
  - Experimento 2: Compara o tipo de seleção de TOS aplicado

- 7 datasets
  - Arrhythmia
  - Letter
  - Cardio
  - Speench
  - Satellite
  - Mnist
  - Mammography

- 107 TOS com variação de hiperparâmetros
  - kNN
  - K-Median
  - Avg-kNN
  - One-Class SVM
  - LOF
  - LoOP
  - iForest

Experimento 1: Resultados

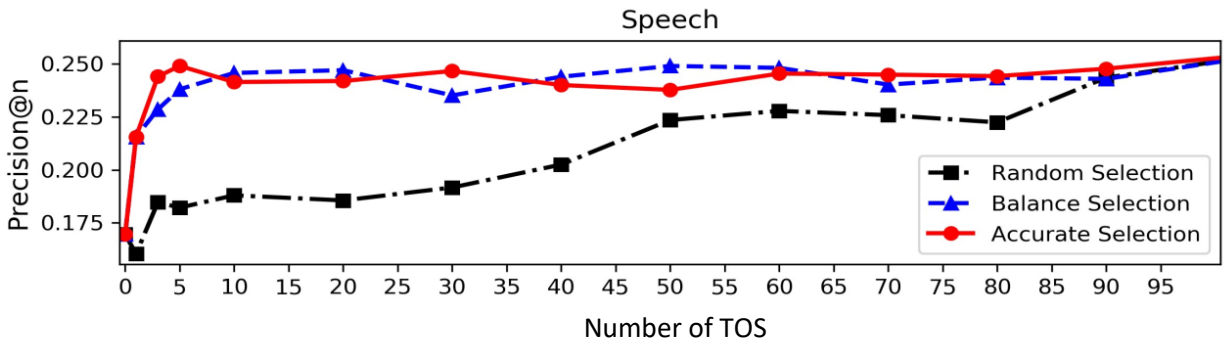
Area under ROC			
Datasets	XGBoost sem TOS	XGBoost com todas as TOS	XGBoost somente TOS
Arrhythmia	0.8698	**0.8816**	0.8810
Letter	0.9399	**0.9729**	0.9593
Cardio	0.9966	**0.9976**	0.9868
Speech	0.7593	**0.8591**	0.7819
Satellite	0.9656	**0.9666**	0.9254
Mnist	0.9963	**0.9999**	0.9980
Mammography	**0.9515**	0.9431	0.9105

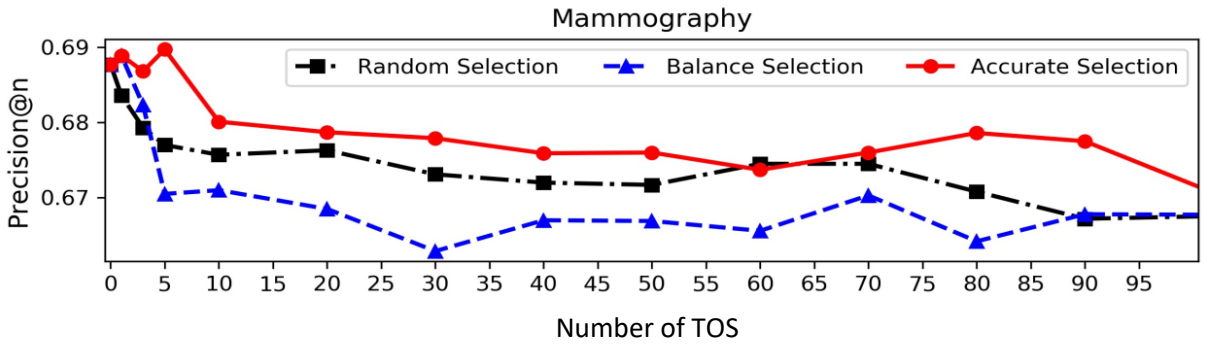
Performance do modelo em diferentes datasets

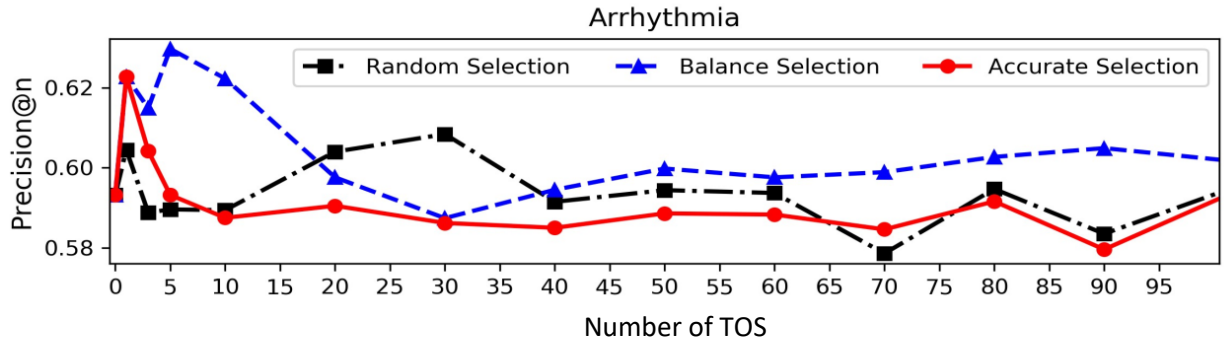
Conclusões empíricas – Experimento 1

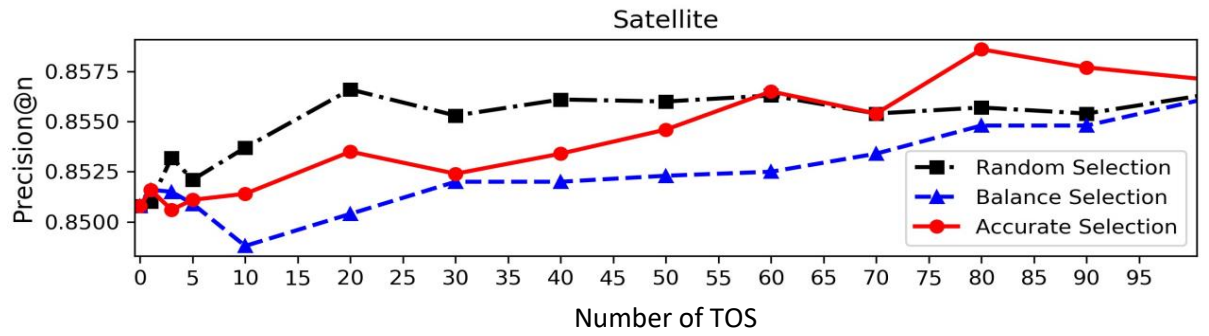
- Em geral, concatenar as features originais com os TOS traz melhorias
- As TOS sozinhas se mostraram significativas
  - Boas representações do dataset original

Experimento 2 - Resultados









Conclusões empíricas – Experimento 2

- Seleção balanceada parece funcionar melhor em datasets com muitas features (Speech, 600 e Arrhythmia, 274)
- Seleção por acurácia funciona melhor em datasets com poucas features (Mammography, 6 e Satellite, 36) -> espaço pequeno, então inserir variância pode não ser tão necessário
- Seleção aleatória normalmente é a pior das três
- Mesmo selecionar poucas TOS tende a melhorar o modelo

Limitações e trabalhos futuros

- Alto custo computacional para processar todos os TOS

- Incorporação e análise de mais TOS
- Efetuar feature selection antes de calcular os TOS
- TOS podem ser combinados ao invés de selecionados

Implementação PyOD: Parâmetros

Parameters

- **estimator_list** (*list, optional (default=None)*) – The list of pyod detectors passed in for unsupervised learning
- **standardization_flag_list** (*list, optional (default=None)*) – The list of boolean flags for indicating whether to take standardization for each detector.
- **max_depth** (*int*) – Maximum tree depth for base learners.

... (demais parâmetros do XGBoost)

Obviamente, X (matriz de observações x features) e Y (ground truth label)

Implementação PyOD: TOS default

```
# predefined range of n_neighbors for KNN, AvgKNN, and LOF
k_range = [1, 3, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]

# validate the value of k
k_range = [k for k in k_range if k < X.shape[0]]

for k in k_range:
    estimator_list.append(KNN(n_neighbors=k, method='largest'))
    estimator_list.append(KNN(n_neighbors=k, method='mean'))
    estimator_list.append(LOF(n_neighbors=k))
    standardization_flag_list.append(True)
    standardization_flag_list.append(True)
    standardization_flag_list.append(True)

n_bins_range = [3, 5, 7, 9, 12, 15, 20, 25, 30, 50]
for n_bins in n_bins_range:
    estimator_list.append(HBOS(n_bins=n_bins))
    standardization_flag_list.append(False)
```

```
n_bins_range = [3, 5, 7, 9, 12, 15, 20, 25, 30, 50]
for n_bins in n_bins_range:
    estimator_list.append(HBOS(n_bins=n_bins))
    standardization_flag_list.append(False)

# predefined range of nu for one-class svm
nu_range = [0.01, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99]
for nu in nu_range:
    estimator_list.append(OCSVM(nu=nu))
    standardization_flag_list.append(True)

# predefined range for number of estimators in isolation forests
n_range = [10, 20, 50, 70, 100, 150, 200, 250]
for n in n_range:
    estimator_list.append(
        IForest(n_estimators=n, random_state=self.random_state))
    standardization_flag_list.append(False)
```

Inicializa 68 seletores não supervisionados por default

Implementação PyOD: fitting function

```python
# keep the standardization scalar for test conversion
X_norm, self._scalar = standardizer(X, keep_scalar=True)

for ind, estimator in enumerate(self.estimator_list):
    if self.standardization_flag_list[ind]:
        estimator.fit(X_norm)
        self.X_train_add_[:, ind] = estimator.decision_scores_

    else:
        estimator.fit(X)
        self.X_train_add_[:, ind] = estimator.decision_scores_

# construct the new feature space
self.X_train_new_ = np.concatenate((X, self.X_train_add_), axis=1)
```

```python
self.clf_ = clf = XGBClassifier(max_depth=self.max_depth,
                                learning_rate=self.learning_rate,
                                n_estimators=self.n_estimators,
                                silent=self.silent,
                                objective=self.objective,
                                booster=self.booster,
                                n_jobs=self.n_jobs,
                                nthread=self.nthread,
                                gamma=self.gamma,
                                min_child_weight=self.min_child_weight,
                                max_delta_step=self.max_delta_step,
                                subsample=self.subsample,
                                colsample_bytree=self.colsample_bytree,
                                colsample_bylevel=self.colsample_bylevel,
                                reg_alpha=self.reg_alpha,
                                reg_lambda=self.reg_lambda,
                                scale_pos_weight=self.scale_pos_weight,
                                base_score=self.base_score,
                                random_state=self.random_state,
                                missing=self.missing,
                                **self.kwargs)
self.clf_.fit(self.X_train_new_, y)
```

Implementação PyOD: Predicting function

```
def _generate_new_features(self, X):
    X_add = np.zeros([X.shape[0], self.n_detector_])

    # keep the standardization scalar for test conversion
    X_norm = self._scalar.transform(X)

    for ind, estimator in enumerate(self.estimator_list):
        if self.standardization_flag_list[ind]:
            X_add[:, ind] = estimator.decision_function(X_norm)

        else:
            X_add[:, ind] = estimator.decision_function(X)
    return X_add
```

```
pred_scores = self.clf_.predict_proba(X_new)[:, 1]
```

Observações e Sugestões do grupo

- Algoritmo incompleto (biblioteca no beta)
  - Não implementa nenhum TOS selection (utiliza todos K TOS, não P)
- Modo verboso -> debug e tempo de execução
  - Alto tempo de execução (importante acompanhar o processo)

Paralelo com TP

- Performa uma análise de diversos algoritmos não supervisionados para detecção de ataques DDoS
- Interessante guideline para seleção prévia de TOS

Demonstração

Código

bit.ly/xgbod-seminar2019

*Posteriormente será disponibilizado no GitHub

Referência

- ZHAO, Yue; HRYNIEWICKI, Maciej. XGBOD: Improving Supervised Outlier Detection with Unsupervised Representation Learning. **International Joint Conference on Neural Networks.** 2018.

Obrigado!

Informações para contato:

andre.ribeiro@dcc.ufmg.br
caio.campos@dcc.ufmg.br
othaviorudda@dcc.ufmg.br