Online Feature Selection for Semantic Image Segmentation

Rishav Rajendra, Michael Christopher, Elias Ioup, Md Tamjidul Hoque, Mahdi Abdelguerfi

University of New Orleans
email: rrajendr@uno.edu

10/07/2020
Presenter Information

- Senior at the University of New Orleans.
- Graduating with a Bachelor's of Science in Computer Science with a minor in Mathematics.
Presentation Outline

• Introduction to Online Feature Selection
  • Difference between "data stream" and "feature stream."
  • Why "feature stream" is widely adopted?
  • What is online feature selection?

• Existing Online Feature Selection Algorithms Overview

• Results Using Existing Online Feature Selection Algorithms
What does "online" data mean?

• As data arrives sequentially over time from a source, it is referred to an "online" data.
• "Online" data can be readily seen in the real-life. For example: live news, twitter, blog posts etc.
• There are two ways to process online data:
  • Data Stream
  • Feature Stream
Streaming Data vs Streaming Features

• Streaming Data
  • In a data stream, the number of features remains the same, but the number of data points increases over time.
  • Also, candidate instances in streaming data are generated dynamically if the size of the instances is unknown.

• Streaming Features
  • In a feature stream, the number of data points is fixed but the candidate features are generated dynamically if the size of the features is unknown.
In a data stream, as pictures arrive, new data points extracted from the pictures are stacked below existing values. The number of features remains the same throughout the process.
In a feature stream, as pictures arrive, new features extracted from the pictures are stacked horizontally. The number of data points remain the same but the number of features increase.
Why streaming feature selection is better?

• We extract 32 features from every image. Each feature has 270,000 data points.

• In a data stream framework, after ten pictures, even if we select just five features, we have a total of \((270,000 \times 10) \times 5 = 13,500,000\) data points.

• In a feature stream structure, after ten pictures, if we select five features, we will have \(270,000 \times 5 = 1,350,000\) data points.

• The number of data points in a data stream framework will continue to rise even if we don’t select any additional features. This is not scalable and could easily overflow.

• Thus, the feature stream framework avoids implicitly handling feature redundancy and efficiently eliminates features that are not required by explicitly managing redundancy found in the features [1].

What is online feature selection?

- In traditional "batch" learning, feature selection is conducted in an off-line fashion where all the features of training instances are given priority [2].
- After all features arrive, the feature selection process starts before training start. Every time we want to include new data gathered time, the entire process has to be restarted.
- This is very computationally expensive with the processing time growing over time.
- Online feature selection process data as they arrive and updates our machine learning model in real-time. This allows us to have a fast and scalable framework which can adapt to changes in real-time.

Existing online feature selection algorithms

• Alpha-investing:
  • Alpha-investing can handle infinitely large feature set, but evaluates each feature exactly once, without considering the redundancy of the selected features.
  • Alpha-investing controls the false discovery rate by dynamically adjusting a threshold on the p-statistic for a new feature to enter the model.
  • The threshold, $\alpha_i$, corresponds to the probability of including a spurious feature at step $i$. It is adjusted using the wealth, $w_i$, which represents the current acceptable number of future false positives.
  • Wealth is increased when a feature is added to the model.
  • Wealth is decreased when a feature is not added to the model, in order to save enough wealth to add future features [3].

OSFS selects strongly relevant and non-redundant features from a sequentially streaming data source using **conditional independence**.

OSFS decides the conditional independence using the chi-squared test and then dynamically identifies and eliminates redundant features from the selected features.

If a subset exists within the selected features, if the features outside the subset, Y, is conditionally independent to the class label, Y is discarded.

Fast-OSFS is an improvement on the efficiency of OSFS and provides faster results [4].

• Koller proposes and theoretically justifies a classification of input features, X, with respect to their relevance to a target, T, in terms of conditional independence [5][6].

• Conditional Independence: In a variable set S, two random variables X and Y are conditionally independent given the set of features Z, if and only if:
  • \( P(X|Y,Z) = P(X|Z) \), denoted as \( \text{Ind}(X,Y|Z) \)

• OSFS determines conditional independence using the chi-square test. The chi-square test of independence is used to determine if there is a significant relationship between the features.

• The null hypothesis of the chi-square test is that there is no association between the variables. For the hypothesis test for the chi-square the test, the test statistic is computed and compared to a critical value.

• The critical value of the chi-square statistic is determined by the level of significance (typically 0.05) and the degrees of freedom.

• If the chi-square test statistic is greater than the critical value, the null-hypothesis is rejected.


Our algorithm

- OSFS achieves a very high prediction accuracy with a few number of pictures in datasets with highly redundant features [2].
- Our water bodies dataset is highly redundant with multiple channels for the same pictures.
- For real-time image segmentation for aerial water bodies pictures, we propose a framework centered around OSFS.
- To further increase processing, we also propose a distributed approach to OSFS using the Spark ecosystem.

Image Segmentation Framework

1. Initialization
   • Best candidate feature set $BCF = []$, the target feature $T$

2. Image Augmentation
   • Randomly augment image with one of eight image augmentation methods

3. Feature Extraction
   • Extract feature $X$ from augmented image

4. Online relevance analysis
   • Determine if $X$ is irrelevant to $T$ or not
     • If not: add to $BCF$

5. Online redundancy analysis
   • If new feature added to $BCF$ perform redundancy analysis and discard redundant features

6. Update machine learning model with $BCF$
Pseudocode

```
Input: class_labels C, image_stream

BCF = []

while image_stream:
    get(image), get(image_mask)

    # Image augmentation
    augment_choices = [rotate, flip, shift, shear, channel, 
                       gray_scale, brightness, contrast]
    augment = random.choice(augment_choices)
    image, image_mask = augment(image, image_choice)

    # Feature extraction
    features = extract_features(image)

    for feature in features:
        # Online relevancy analysis
        if Conditional_Dependent(feature, C|∅):
            # Add relevant feature to BCF
            BCF = BCF.add(feature)

        # Online redundant analysis
        if ∃ subset ⊆ BCF/feature, s.t. Independent(feature, subset):
            # Remove redundant feature
            BCF.remove(feature)

    update_model(BCF)
```
OSFS Distributed Framework

Kafka Producer
- Load pictures from database
- Convert pictures into Byte Array
- Send Byte Array to Spark Streaming Client

Spark Streaming
- Convert Byte Array back to image
- Follow OSFS image segmentation framework
- Update MLib model

MLlib

Repeat until condition satisfied or no more images
Dataset preparation

• Dataset
  • The images used are aerial imagery of water bodies acquired during the agricultural growing seasons in the continental US by the NAIP.
  • Low number of images. Each image has four bands of data: red, green, blue and near-infrared. As a result, data is highly redundant.

• Image Augmentations methods
  • In order to increase randomness and size of the available dataset, I used the following image segmentation methods:
    • Rotate image a random amount of degrees.
    • Randomly flip images horizontally or vertically.
    • Shift Augmentation
    • Shear Augmentation
    • Random Channel Shift
    • Gray Scale
    • Random Brightness adjustment
    • Random Contrast adjustment
  • As images arrived sequentially, an augmentation method was randomly chosen and applied before feature extraction.
Feature Extraction

- The following features were extracted from each image:
  - Original image values
  - Gabor kernal features
  - Canny edge features
  - Gaussian blur
- We extract 32 features from each image band. We have 32 image bands. A total of 1024 features are extracted every run.
Performance Evaluation

- Accuracy score
- Balanced accuracy score
- Average precision score
- Precision macro
- Recall micro
- F1 score
Features selected as image channels increase
## Evaluation metrics of OSSF vs D-OSSF

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Balanced Accuracy</th>
<th>Average Precision</th>
<th>Recall</th>
<th>F1 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT</td>
<td>91.68</td>
<td>91.39</td>
<td>90.72</td>
<td>92.34</td>
<td>95.63</td>
<td>93.27</td>
</tr>
<tr>
<td>LR</td>
<td>91.71</td>
<td>91.39</td>
<td>90.73</td>
<td>92.16</td>
<td>95.72</td>
<td>93.30</td>
</tr>
<tr>
<td>RF</td>
<td>91.68</td>
<td>91.39</td>
<td>90.73</td>
<td>92.33</td>
<td>95.63</td>
<td>93.27</td>
</tr>
<tr>
<td>XGBC</td>
<td>91.71</td>
<td>91.39</td>
<td>90.72</td>
<td>92.34</td>
<td>95.63</td>
<td>93.27</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>Precision</th>
<th>Balanced Accuracy</th>
<th>Average Precision</th>
<th>Recall</th>
<th>F1 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>DT</td>
<td>90.45</td>
<td>90.72</td>
<td>90.45</td>
<td>90.25</td>
<td>95.25</td>
<td>94.82</td>
</tr>
<tr>
<td>LR</td>
<td>91.39</td>
<td>90.93</td>
<td>91.39</td>
<td>90.27</td>
<td>93.97</td>
<td>93.22</td>
</tr>
<tr>
<td>RF</td>
<td>89.95</td>
<td>90.72</td>
<td>89.95</td>
<td>91.01</td>
<td>94.87</td>
<td>93.22</td>
</tr>
<tr>
<td>XGBC</td>
<td>91.38</td>
<td>90.72</td>
<td>91.38</td>
<td>92.35</td>
<td>93.51</td>
<td>94.82</td>
</tr>
</tbody>
</table>
ROC AUC of OSSF and D-OSSF

(a) ROC AUC of OSSF

(b) ROC AUC of D-OSSF

Classifier
- DecisionTree
- LogisticRegression
- RandomForest
- XGBoost
Run-time of OSSF and Alpha-investing

(a) Processing time of OSSF

(b) Processing time of Alpha-investing

Classifiers:
- DecisionTree
- LogisticRegression
- RandomForest
- XGBoost

Time (in sec)

Image Channels
Run-time of OSSF and D-OSSF

(a) Processing time of OSSF

(b) Processing time of D-OSSF
Insights from the reports

- The results above are from our implementation of OSFS and Alpha-Investing in Python.
- Using Alpha-Investing, the number of features selected climbed as the number of pictures increased.
- Across multiple runs, even after images are randomly augmented, the number of features selected remain very low for OSFS.
- Decision Trees along with OSFS provided the best results with the final accuracy of 95.13%.
- The accuracy with Alpha-Investing was fluctuated frequently.
- Time taken to process an image with OSFS remains constant even though number of features increase.
- Overall time for Alpha-Investing increased as number of pictures increased.