One-Class Learning & Concept Summarization for Vaguely Labeled Data Streams

Xavier & Tyler

What is the Problem we are trying to solve?

Well there are 2

1. Vague One-Class Learning Problem
   - One-class classification tries to determine whether a new test datum is a member of a specific class or not, instead of traditionally trying to determine which out of n-classes it best fits.
   - In Vague one-class learning, a labeled sample set has mixes of instances which may or may not be of interest to users. Effective mechanisms must be developed to differentiate positive samples in order to identify users' genuine interests.
2. One-Class Concept Summarization Problem
   - When testing data over a stream of the user's interests, the samples may contain multiple concepts due to the user's changing interestings such as rainfall, pressure, wind velocity etc. Therefore we need a method to ***best*** summarize the user's interests such that we can predict what else might interest the user.

How was the paper structured

25 Pages split into sections:

Section 2 defines the problem, and discusses simple solutions.

Section 3 discusses the overall framework of the proposed one-class learning and concept summarization system.

Section 4 discusses the vague one-class learning module.

Section 5 proposes a solution for concept summarization.

Section 6 analyzes the system time complexity.

Section 7 Experimental results reported.

Section 8 Review of related work

Section 9 Conclusion.

Their test data

Figure 1. Sensor Test Data

Data Stream data; 54 Sensors that contain **temperature, humidity, light & voltage**. Goal is to correctly predict which region a particular reading is coming from.

Power Test Data

Data stream data; **power supply from the main grid**, and the **power transforms from other grids** each hour. The goal is to predict which hour the current power supply entry belongs to.

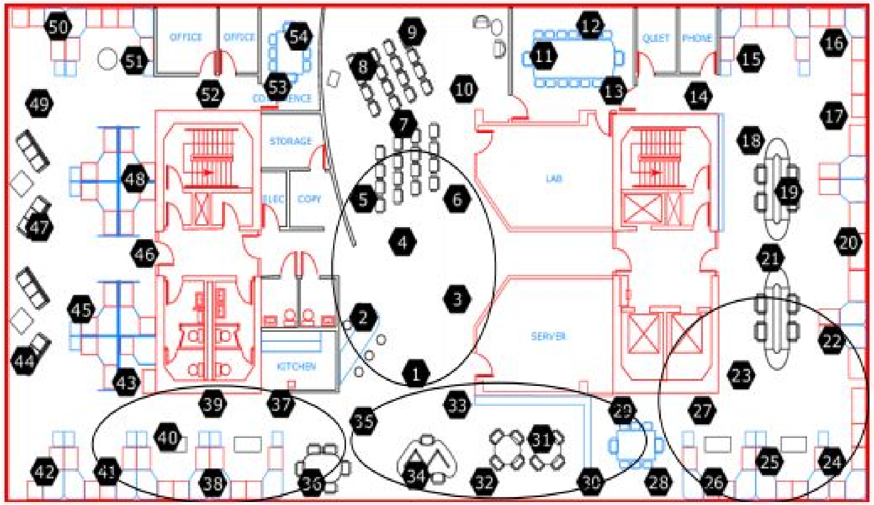
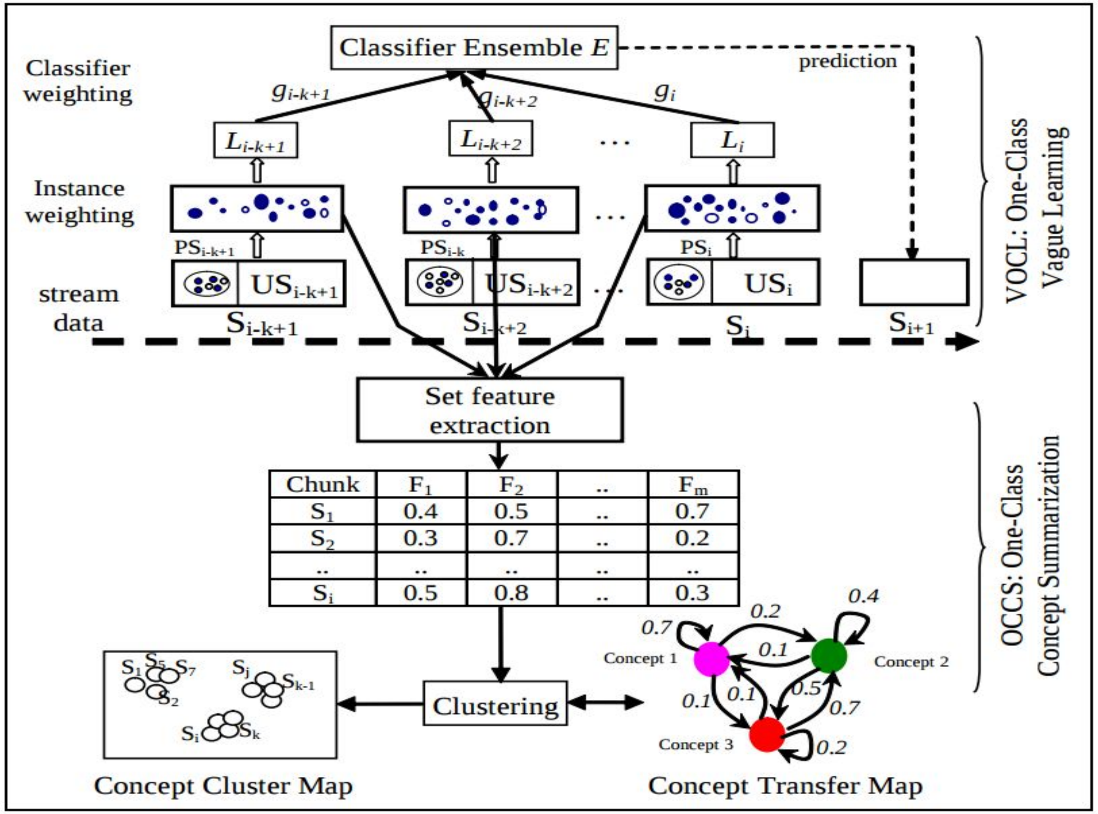


Figure 1: the sensor distribution map deployed in Intel Berkeley Research Lab

Algorithm Overview



Chunk	F_1	F_2	..	F_m
S_1	0.4	0.5	..	0.7
S_2	0.3	0.7	..	0.2
..	..	..	..	..
S_i	0.5	0.8	..	0.3

- KMeans clustering on all dimensions (all attributes)
- Instance weighting
  - Filter by attribute
  - *F* Classifiers via cross-validation
- Global Weighting
  - % Positive Classified from PSi
- Unified weighting formula -
  - favor Large local and global values
- Classifier weighting
  - samples from weighted sample distribution to accommodate for instance weight values for learning
  - Ensures instances Independently sampled from given distribution

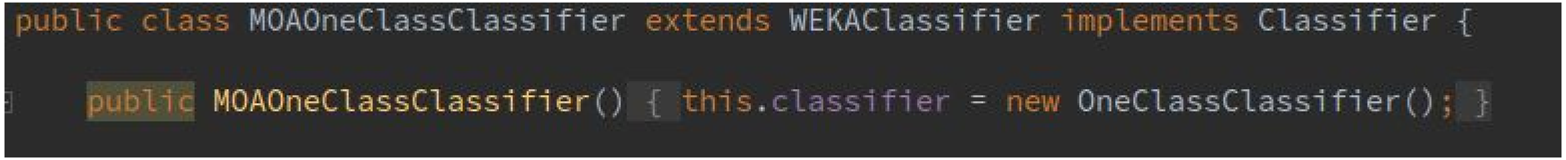
Our Implementation - Structure

Classes from the VOCL Package

- ClusterVagueLabeling.java
  - Parititon into PSi USi subsets using KMeans clustering
- LocalWeighting.java
  - Cross-validation, Train Classifier per fold, PSi which classifier as positive 1.0 weight, else 0.5, USi classified as positive 0.5 - 1.0 weighting formula using f classifiers
- GlobalWeighting.java
  - Percentage of ensemble classifiers predicting instance as positive
- OneClassClassifierEnsemble.java
  - WIP - Contains the weighting classifiers to perform a weighted vote prediction
- VOCL.java
  - Core Loop (Figure 5. VOCL Main procedure), along with necessary weighting, training & helper functions.
- VOCLRunner.java
  - Reads in the Arff stream and passes it to our VOCL module

Our Implementation - Weka <-> Moa Conversion

Writing a wrapper for the Weka OneClassClassifier that extends the Moa WEKAClassifier parent and implementing the MOA Classifier interface so that it can be used with both Weka and Moa components.



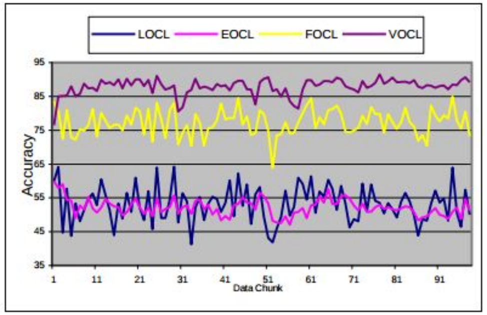
```java
public class MOAOneClassClassifier extends WEKAClassifier implements Classifier {

    public MOAOneClassClassifier() { this.classifier = new OneClassClassifier(); }
```

Issues

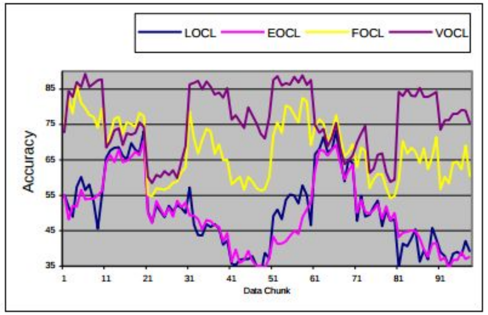
How to Interop MOA and WEKA

- How to use MOA Ensemble Classifiers with WEKA Single Classifiers
  - Specifically how to use the WeightedMajorityAlgorithm with my wrapped WEKA OneClassClassifier which extends the MOA WEKAClassifier.
- Conflicts between buildClassifier() and trainOnInstance() where some parts will work and others won’t
  - E.g buildClassifier() creates internal variables needed by classifyInstance()
  - E.g trainOnInstance() allows getVotesForInstance() to work.

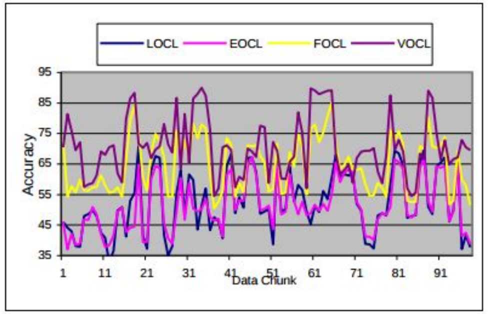
Expected Results



(a) constant interests



(b) regular shifting (10 chunks/shifting)



(c) probability shifting

FOCL

- cross validation to the labeled portion in Si (i.e., PSi) with incorrectly classified positive samples directly excluded from PSi
- crosss-validation classifiers trained from PSi are also used to classify unlabeled samples in Si (i.e., USi) and all instances in Si-1,
- A classifier trained from PSi is used to predict instances in Si+1

Results

All Pieces of VOCL Module implemented

- Clustered Vague Labeling -> PSi USi subsets
- Local Weighting, Global Weighting, Unified Weighting implemented
- Training new weighted classifiers for each Si chunk
- Generating classifier weightings from {i-k+1..i} classifiers
- Weighted Voting OneClassClassifier Ensemble Hacked together

However Accuracy is incorrect - Interop problems?

Summary

- Lot of layers to algorithm -> bigger bite than we could chew…
  - Scope narrowed down to just the VOCL module
- Vagueness on how pieces fit together
  - Paper implemented with Weka
  - Assignment implemented with Moa -> Conflict (Trying to cross-match ideas and data structures)
- Further development and improvements to algorithm for future:
  - Would like to experiment with more accurate clustering such as DBSCAN or HDBSCAN
  - Instead of KMeans