Tutorial on Learning Class Imbalanced Data Streams

Leandro L. Minku  
University of Leicester

Shuo Wang  
University of Birmingham

Giacomo Boracchi  
Politecnico di Milano
Outline

• Background and motivation
• Problem formulation
• Challenges and brief overview of core techniques
• Online approaches for learning class imbalanced data streams
• Chunk-based approaches for learning class imbalanced data streams
• Performance assessment
• Two real world problems
• Remarks and next challenges


Outline

• Background and motivation
• Problem formulation
• Challenges and brief overview of core techniques
• Online approaches for learning class imbalanced data streams
• Chunk-based approaches for learning class imbalanced data streams
• Performance assessment
• Two real world problems
• Remarks and next challenges
Data Streams

- Organisations have been gathering large amounts of data.
- The amount of data frequently grows over time.
Labelled Data Streams

Labelled data stream: ordered and potentially infinite sequence of examples \( S = \langle (x_1, y_1), (x_2, y_2), \ldots, (x_t, y_t), \ldots \rangle \) where \( (x_t, y_t) \in X \times Y \)

Classification problems:

- \( X \) is a (typically high dimensional) space.
  - Real values.
  - Ordinal values.
  - Categorical values.
- \( Y \) is a set of categories.

This tutorial will concentrate on supervised learning for classification problems.
Example of Data Stream: Tweet Topic Classification

Labelled data stream: ordered and potentially infinite sequence of examples \( S = \langle (x_1, y_1), (x_2, y_2), \ldots, (x_t, y_t), \ldots \rangle \) where \( (x_t, y_t) \in X \times Y \)

Each training example corresponds to a tweet.

\[ x = \{ \text{td-idf word 1}, \text{td-idf word 2}, \ldots \} \]

\[ y = \{ \text{topic} \} \]

td-idf (term frequency - inverse document frequency)
**Example of Data Stream: Fraud Detection**

**Labelled data stream:** ordered and potentially infinite sequence of examples $S = \langle (x_1, y_1), (x_2, y_2), \ldots, (x_t, y_t), \ldots \rangle$ where $(x_t, y_t) \in X \times Y$

Each training example corresponds to a credit or debit card transaction.

$x = \{ \text{merchant id, purchase amount, average expenditure, average number of transactions per day or on the same shop, average transactions amount, location of last purchase, etc} \}$

$y = \{ \text{genuine/ fraud} \}$
Example of Data Stream: Software Defect Prediction

Labelled data stream: ordered and potentially infinite sequence of examples $S = \langle(x_1, y_1), (x_2, y_2), \ldots, (x_t, y_t), \ldots \rangle$ where $(x_t, y_t) \in X \times Y$

Each training example corresponds to a software module.

$x = \{\text{lines of code, branch count, halstead complexity, etc}\}$

$y = \{\text{buggy/not buggy}\}$
Challenge 1: Incoming Data
Challenge 1: Incoming Data

• Data streams are ordered and potentially infinite.
• In the beginning there may be few examples.
• Rate of incoming examples is typically high.

Everyday, on average, around 600,000 credit / debit card transactions are processed by Atos Worldline.
Challenge 1: Incoming Data

• Data streams are ordered and potentially infinite.
• In the beginning there may be few examples.
• Rate of incoming examples is typically high.

Every second, on average, around 6,000 tweets are tweeted on Twitter (http://www.internetlivestats.com/twitter-statistics/)
Data Streams

Challenge 1: Incoming Data

[Strict] Online Learning

Chunk-Based Learning
Core Techniques: Online Supervised Learning

• Given a model $\hat{f}_{i-1}: \mathcal{X} \rightarrow Y$ and a new example $(x_t, y_t)$ drawn from a joint probability distribution $\mathcal{D}_t$.

• Learn a model $\hat{f}_i: \mathcal{X} \rightarrow Y$ able to generalise to unseen examples of $\mathcal{D}_t$.

• In strict online learning, $(x_t, y_t)$ must be discarded soon after being learnt.

• Potential advantages: fast training, low memory requirements.

• Potential disadvantage: only one pass may result in lower predictive performance.
Core Techniques: Chunk-Based Supervised Learning

- Given a model $\hat{f}_{t-1}: \mathcal{X} \rightarrow \mathcal{Y}$ and a new chunk of examples $S_t = \{(x^{(1)}_t, y^{(1)}_t), (x^{(2)}_t, y^{(2)}_t), \ldots, (x^{(n)}_t, y^{(n)}_t)\} \subset S$, where $\forall i, (x^{(i)}_t, y^{(i)}_t) \sim \text{i.i.d. } D_t$.

- Learn a model $\hat{f}_t: \mathcal{X} \rightarrow \mathcal{Y}$ able to generalise to unseen examples of $D_t$.

- Typically, $S_t$ is discarded soon after being learnt.

- Potential advantages: easy to fit in any offline learning technique, processing examples from each chunk several times may help to increase predictive performance.

- Potential disadvantages: higher training time; requires to store chunk; in reality, examples from $S_t$ may not all come i.i.d. from $D_t$. 
Applying Online vs Chunk-Based Learning

Some applications lend themselves to a specific type of algorithm.

New training examples arrive separately.

New training examples arrive in chunks.
Applying Online vs Chunk-Based Learning

Online learning for applications where data arrive in chunks.

- Process each training example of the chunk separately.

Chunk-based learning for applications where data arrives separately.

- Wait to receive a whole chunk of data.
Learning Class Imbalanced Data Streams

Leandro Minku  http://www.cs.le.ac.uk/people/llm11/ Learning Class Imbalanced Data Streams
Challenge 2: Concept Drift

- The probability distributions $D_t$ and $D_{t+1}$ are not necessarily the same.

- **Concept drift**: change in the joint probability distribution of the problem. We say that there is concept drift in a data stream if, $\exists t, t+1 | D_t \neq D_{t+1}$.

- In **chunk-based learning**, $\forall i, (x_{t(i)}, y_{t(i)}) \in S_t$ are typically assumed to be drawn i.i.d. from the same $D_t$. However, in reality, it may be difficult to guarantee that.

- **Non-stationary environments**: environments that may suffer concept drift.
Types of Concept Drift

\[ D_t = p_t(x, y) = p_t(y|x) \cdot p_t(x) \]

\[ D_t = p_t(x, y) = p_t(x|y) \cdot p_t(y) \]

Concept drift may affect different components of the joint probability distribution.

Potentially, more than one component can be affected concurrently.
Change in $p(y|x)$
E.g.: a software module that was previously buggy may not be buggy in the new version of the software anymore, despite having similar input features.
Change in $p(x)$

![Diagram showing the change in decision boundary between true and learnt boundaries in a 2D space with features $x_1$ and $x_2$. The left side shows the update before, and the right side shows the update after. The blue triangles represent the positive class, while the red circles represent the negative class. The dotted line represents the old learnt decision boundary, and the solid line represents the new learnt decision boundary.]
Example of Potential Change in $p(x)$

E.g.: fraud strategies and customer habits may change.
Change in $p(y)$
Example of Change in $p(y)$

FIFA Confederations Cup  $\rightarrow$  FIFA World Cup

E.g., tweet topic becoming more or less popular.
Concept Drift and the Need for Adaptation

Concept drift is one of the main reasons why we need to continue learning and adapting over time.
Learning Class Imbalanced Data Streams

Leandro Minku  
http://www.cs.le.ac.uk/people/llm11/
Core Techniques:
The General Idea of Concept Drift Detection

Data Stream

<table>
<thead>
<tr>
<th>Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calculating Metrics</td>
</tr>
<tr>
<td>Change Detection Test</td>
</tr>
<tr>
<td>Concept Drift Detection Method</td>
</tr>
<tr>
<td>Concept Drift?</td>
</tr>
</tbody>
</table>

Potential advantage: tells you that concept drift is happening.

Potential disadvantage: may get false alarms or delays.

Normally used in conjunction with some adaptation mechanism.
Core Techniques: The General Idea of Adaptation Mechanisms

- Adaptation mechanisms may or may not be used together with concept drift detection methods, depending on how they are designed.

- **Potential advantages of not using concept drift detection:** no false alarms and delays, potentially more adequate for slow concept drifts.

- **Potential disadvantage of not using concept drift detection:** don’t inform users of whether concept drift is occurring.

- Several different adaptation mechanisms can be used together.
Core Techniques: The General Idea of Adaptation Mechanisms

Example of adaptation mechanism 1: forgetting factors

- Learner
  - Loss function with forgetting factor

- Calculating Metrics for Concept Drift Detection
  - Loss function with forgetting factor
Core Techniques: The General Idea of Adaptation Mechanisms

Example of adaptation mechanism 2: adding / removing learners in online learning
Core Techniques: The General Idea of Adaptation Mechanisms

Example of adaptation mechanism 3: adding / removing learners in chunk-based learning

Concept Drift Detection Method or Heuristic Rule

Add
Learner 1
Remove
[Optional]
Learner 2
Learner 3
Core Techniques: The General Idea of Adaptation Mechanisms

Example of adaptation mechanism 4: deciding how / which learners to use for predictions in online or chunk-based learning

Concept Drift Detection Method or Heuristic Rule

Add

Remove

[Optional]

Learner 1

Learner 2

Learner 3
Core Techniques:
The General Idea of Adaptation Mechanisms

Example of adaptation mechanism 5: deciding which learners can learn current data in online or chunk-based learning.
Core Techniques: The General Idea of Adaptation Mechanisms

Other strategies / components are also possible

Concept Drift Detection Method or Heuristic Rule

Add
Remove
Enable learning

[Optional]

[Optional]

Learner 1
Learner 2
Learner 3
Challenge 1: Incoming Data

[Strict] Online Learning

Chunk-Based Learning

Challenge 2: Concept Drift

Data Streams

Challenge 3: Class Imbalance

Concept Drift Detection

Adaptation Strategies
Challenge 3: Class Imbalance

Class imbalance occurs when $\exists c_i, c_j \in Y \mid p_t(c_i) \leq \delta p_t(c_j)$, for a pre-defined $\delta \in (0, 1)$.

- It is said that $c_i$ is a minority class, and $c_j$ is a majority class.
Challenge 3: Class Imbalance

Class imbalance occurs when \( \exists c_i, c_j \in Y \mid p_t(c_i) \leq \delta p_t(c_j) \), for a pre-defined \( \delta \in (0,1) \).

- It is said that \( c_i \) is a minority class, and \( c_j \) is a majority class.

Only \(~0.2\%\) of transactions in Atas Worldline's data stream are fraud.

Typically \(~20\%-30\%\) of the software modules are buggy.
Challenge 3: Class Imbalance

Why is that a challenge?

- Machine learning algorithms typically give the same importance to each training example when minimising the average error on the training set.
  - If we have much more examples of a given class than the others, this class may be emphasized in detriment of the other classes.
- Depending on $\mathcal{D}_t$, a predictive model may perform poorly on the minority class.
Data Streams

Challenge 1: Incoming Data

[Strict] Online Learning

Chunk-Based Learning

Challenge 2: Concept Drift

- Concept Drift Detection
- Adaptation Strategies

Challenge 3: Class Imbalance

- Algorithmic Strategies (e.g., Cost-Sensitive Algorithms)
- Data Strategies (e.g., Resampling)
Core Techniques: General Idea of Algorithmic Strategies

• Loss functions typically give the same importance to examples from different classes. E.g.: consider for illustration purposes:
  • \[ \text{Accuracy} = \frac{TP + TN}{P + N} \]

• Consider the fraud detection problem where our training examples contain:
  • 99.8% of examples from class -1.
  • 0.2% of examples from class +1.

• Consider that our predictive model always predicts -1.

• What is its training accuracy?
Core Techniques: General Idea of Algorithmic Strategies

• Consider again the following fraud detection problem:
  • 99.8% of examples from class -1.
  • 0.2% of examples from class +1.

• Consider a modification in the accuracy equation, where:
  • class -1 has weight 0.2%
  • class +1 has weight 99.8%
  • Accuracy = \(\frac{0.998 \times TP + 0.002 \times TN}{0.998 \times P + 0.002 \times N}\)

• What is the training accuracy of a model that always predicts -1?
Core Techniques: General Idea of Algorithmic Strategies

- Use loss functions that lead to a more balanced importance for the different classes.

  - E.g.: cost sensitive algorithms use loss functions that assign different costs (weights) to different classes.
Core Techniques: General Idea of Data Strategies

• Manipulate the data to give a more balanced importance for different classes.
  • E.g.: oversample the minority / undersample the majority class in the training set, so as to balance the number of examples of different classes.

• Potential advantages: applicable to any learning algorithm; could potentially provide extra information about the likely decision boundary.

• Potential disadvantages: increased training time in the case of oversampling; wasting potentially useful information in the case of undersampling.
Data Streams

Challenge 1: Incoming Data

Challenge 2: Concept Drift

Challenge 3: Class Imbalance

[Strict] Online Learning

Chunk-Based Learning

Concept Drift Detection

Adaptation Strategies

Algorithmic Strategies (e.g., Cost-Sensitive Algorithms)

Data Strategies (e.g., Resampling)
Challenge 4:
Dealing with the three challenges altogether
Outline

• Background and motivation
• Problem formulation
• Challenges and core techniques
  • Online approaches for learning class imbalanced data streams
  • Chunk-based approaches for learning class imbalanced data streams
• Performance assessment
• Two real world problems
• Remarks and next challenges
**Learning Class Imbalanced Data Streams**

**[Strict] Online Learning**

**Chunk-Based Learning**

**Challenge 1:** Incoming Data

**Challenge 2:** Concept Drift

**Challenge 3:** Class Imbalance

**Data Streams**

**Algorithmic Strategies** (e.g., Cost-Sensitive Algorithms)

**Data Strategies** (e.g., Resampling)

**Concept Drift Detection**

**Adaptation Strategies**
DDM-OCI: Drift Detection Method for Online Class Imbalance Learning

Detecting concept drift $p(y|x)$ in an online manner with class imbalance.

- Metric monitored:
  - Recall of the minority class $+1$.
  - Whenever an example of class $+1$ is received, update recall on class $+1$ using the following time-decayed equation:

$$R_+^{(t)} = \begin{cases} 
1_{[\hat{y}=+1]}, & \text{if } (x,y) \text{ is the first example of class } +1 \\
\eta R_+^{(t-1)} + (1 - \eta)1_{[\hat{y}=+1]}, & \text{otherwise}
\end{cases}$$

where $\eta$ is a forgetting factor.

DDM-OCI: Drift Detection Method for Online Class Imbalance Learning

- Change detection test:

\[
R_+^{(t)} - \sigma_+^{(t)} \leq R_+^{min} - \alpha \cdot \sigma_+^{min}
\]

Adapting from concept drift \( p(y|x) \):
- Resetting mechanism.

Learning class imbalanced data:
- Not achieved.

Other Examples of Concept Drift Detection Methods

• **PAUC-PH**: monitor the drop of Prequential AUC
  

• **Linear Four Rates**: monitor 4 rates from the confusion matrix.
  
  H. Wang and Z. Abraham, “Concept drift detection for streaming data,” in *the International Joint Conference on Neural Networks (IJCNN)*, 2015, pp. 1–9.
[Strict] Online Learning

Chunk-Based Learning

Challenge 1: Incoming Data

Data Streams

Challenge 2: Concept Drift

Concept Drift Detection

Adaptation Strategies

Challenge 3: Class Imbalance

Algorithmic Strategies (e.g., Cost-Sensitive Algorithms)

Data Strategies (e.g., Resampling)
Dealing with concept drift affecting p(y):

• Time-decayed class size: automatically estimates imbalance status and decides the resampling rate.

\[ w_k^{(t)} = \eta w_k^{(t-1)} + (1 - \eta) 1_{(y(t)=c_k)} \]

where \( \eta \) is a forgetting factor.


Learning class imbalanced data in online manner with concept drift affecting \( p(y) \):

For each base learner \( f_m \) \((m = 1, 2, \ldots, M)\) do:

- If \( y_t = +1 \) and \( w^+_t < w^+_o \) for OOB
  - Set \( K \sim \text{Poisson} \left( \frac{w(t)}{w^+_o} \right) \)
- Else if \( y_t = -1 \) and \( w^-_t < w^-_o \) for OOB
  - Set \( K \sim \text{Poisson} \left( \frac{w(t)}{w^-_o} \right) \)
- Else
  - Set \( K \sim \text{Poisson} \left( 1 \right) \)

Update \( f_m \) \( K \) times

+1 is a "minority"
+1 is a "majority"
-1 is a "minority"
-1 is a "majority"

Undersample \((\lambda < 1)\)

Problem: can’t handle multi-class problems, and concept drifts other than \( p(y) \).
Other Examples of Algorithms

MOOB and MUOB: extensions of OOB and UOB for multi-class problems.

Data Streams

Challenge 1: Incoming Data

Challenge 2: Concept Drift

Challenge 3: Class Imbalance

Algorithmic Strategies (e.g., Cost-Sensitive Algorithms)

Data Strategies (e.g., Resampling)

[Strict] Online Learning

Chunk-Based Learning

Concept Drift Detection

Adaptation Strategies
DDM-OCI + Resampling

Detecting concept drift $p(y|x)$ in an online manner with class imbalance and adapting from it:

- DDM-OCI.

Learning class imbalanced data in an online manner with concept drift $p(y)$:

- OOB or UOB.

Other Examples of Algorithms

ESOS-ELM: Ensemble of Subset Online Sequential Extreme Learning Machine

- Also uses algorithmic class imbalance strategy for concept drift detection and online resampling strategy for learning, but
- it preserves a whole ensemble of models representing potentially different concepts, weighted based on G-mean.

Learning Class Imbalanced Data Streams

[Strict] Online Learning
Chunk-Based Learning

Data Streams

Challenge 1: Incoming Data

Challenge 2: Concept Drift

Concept Drift Detection
Adaptation Strategies

Algorithmic Strategies (e.g., Cost-Sensitive Algorithms)
Data Strategies (e.g., Resampling)

Challenge 3: Class Imbalance
RLSACP: Recursive Least Square Adaptive Cost Perceptron

Loss function:

\[ E_t(\beta) = \sum_{i=1}^{t} w_i(y_i) \cdot \lambda^{t-i} \cdot e_i(\beta) \]

\[ e_t(\beta) = \frac{1}{2} (y_t - \phi(\beta_t^T x_t))^2 \]

\((x_t, y_t)\) is the training example received at time step \(t\); \(\phi\) is the activation function of the neuron, \(\beta_t\) are the neuron parameters at time \(t\);

\(\lambda \in [0,1]\) is a forgetting factor to deal with concept drift \(p(y|x)\);

\(w_t(y_t)\) is the weight associated to class \(y_t\) at time \(t\), to deal with class imbalance.

RLSACP: Recursive Least Square Adaptive Cost Perceptron

Learning class imbalanced data in an online manner with concept drift affecting $p(y|x)$:

$$E_t(\beta) = w_i(y_i) \cdot e_i(\beta) + \lambda \cdot E_{t-1}(\beta)$$

$\beta$ are the neuron parameters;

$\lambda \in [0,1]$ is a forgetting factor to deal with concept drift;

$w_t(y_t)$ is the weight associated to class $y_t$ at time $t$, to deal with class imbalance.

RLSACP: Recursive Least Square Adaptive Cost Perceptron

Dealing with concept drift affecting $p(y)$:

• Update $w_t(y_t)$ based on:
  • Imbalance ratio based on a fixed number of recent examples.
  • Current recalls on the minority and majority class.

Problem: single perceptron.

Other Examples of Algorithms

ONN: Online Multi-Layer Perceptron NN model.

Outline

• Background and motivation
• Problem formulation
• Challenges and core techniques
• Online approaches for learning class imbalanced data streams
  • Chunk-based approaches for learning class imbalanced data streams
• Performance assessment
• Two real world problems
• Remarks and next challenges
Learning Class Imbalanced Data Streams

[Strict] Online Learning

Chunk-Based Learning

Challenge 1: Incoming Data

Data Streams

Challenge 2: Concept Drift

Concept Drift Detection

Adaptation Strategies

Algorithmic Strategies (e.g., Cost-Sensitive Algorithms)

Data Strategies (e.g., Resampling)

Challenge 3: Class Imbalance
Uncorrelated “Bagging”

Heuristic Rule:
- add new ensemble for each new chunk
- remove old ensemble

Problem: minority class may suffer concept drift.

Other Examples of Algorithms

- **SERA** — uses the N old examples of the minority class with the smallest distance to the new examples of the minority class.

  S. Chen and H. He. "SERA: Selectively Recursive Approach towards Nonstationary Imbalanced Stream Data Mining", in the *International Joint Conference on Neural Networks*, 2009.

- **REA** — uses the N old examples of the minority class that have the largest number of nearest neighbours of the minority class in the new chunk.

Learning Class Imbalanced Data Streams

- [Strict] Online Learning
- Chunk-Based Learning

Data Streams

- Challenge 1: Incoming Data
- Challenge 2: Concept Drift
- Challenge 3: Class Imbalance

Algorithmic Strategies (e.g., Cost-Sensitive Algorithms)
Data Strategies (e.g., Resampling)

Concept Drift Detection
Adaptation Strategies
Learn++.NIE: Learn++ for Nonstationary and Imbalanced Environments

Heuristic Rule:
- add new ensemble for each new chunk

Undersample (bootstrap) for each base learner

Weights calculated over time based on the error (e.g., cost-sensitive error) on all chunks seen by a given ensemble, with less importance to the older chunks.

Other Examples of Algorithms

Learn++.CDS: Learn++ for Concept Drift with SMOTE

- Also creates new classifiers for new chunks and combine them into an ensemble.
- Uses SMOTE-like resampling and boosting-like weights for ensemble classifiers.

Learning Class Imbalanced Data Streams

Data Streams

- Challenge 1: Incoming Data
- Challenge 2: Concept Drift
- Challenge 3: Class Imbalance

**Algorithmic Strategies** (e.g., Cost-Sensitive Algorithms)

**Data Strategies** (e.g., Resampling)

**Adaptation Strategies**

- Concept Drift Detection
- **[Strict] Online Learning**
- Chunk-Based Learning
Other Examples of Algorithms

- HUWRS.IP: Heuristic Updatable Weighted Random Subspaces-Instance Propagation
  - Trains new learners on new chunks, based on resampling.
  - Uses cost-sensitive distribution distance function to decide weights of ensemble members.
  - Cost-sensitive distance function could be argued to be a concept drift detector.

Outline

• Background and motivation
• Problem formulation
• Challenges and core techniques
• Online approaches for learning class imbalanced data streams
• Chunk-based approaches for learning class imbalanced data streams
• Performance assessment
• Two real world problems
• Remarks and next challenges
Performance on a Separate Test Set

Problem: typically infeasible for real world problems.
Prequential Performance

\[ \text{perf}^{(t)} = \begin{cases} \text{perf}_{ex}^{(t)}, & \text{if } t=1 \\ (t-1)\text{perf}^{(t-1)} + \frac{\text{perf}_{ex}^{(t)}}{t}, & \text{otherwise} \end{cases} \]

Problem: does not reflect the current performance.
Exponentially Decayed Prequential Performance

\[
perf^{(t)} = \begin{cases} 
perf^{(t)}_{ex}, & \text{if } t=1 \\
\eta \cdot perf^{(t-1)} + (1 - \eta) \cdot perf^{(t)}_{ex}, & \text{otherwise}
\end{cases}
\]

• Alternative for artificial datasets: reset prequential performance upon known concept drifts.

Chunk-Based Performance
Variations of Cross-Validation

Performance Metrics for Class Imbalanced Data

- Accuracy is inadequate.
  - \((TP + TN) / (P + N)\)

- Precision is inadequate.
  - \(TP / (TP + FP)\)

- Recall on each class separately is more adequate.
  - \(TP / P\) and \(TN / N\).

- F-measure: not very adequate.
  - Harmonic mean of precision and recall.

- G-mean is more adequate.
  - \(\sqrt{TP/P \times TN/N}\)

- ROC Curve is more adequate.
  - Recall on positive class \((TP / P)\) vs False Alarms \((FP / N)\)
Prequential AUC

• We need to sort the scores given by the classifiers to compute AUC.

• A sorted sliding window of scores can be maintained in a red-black tree.

• Scores can be added and removed from the sorted tree in $O(2\log d)$, where $d$ is the size of the window.

• Sorted scores can be retrieved in $O(d)$.

• For each new example, AUC can be computed in $O(d+2\log d)$.

• If size of the window is considered a constant, AUC can be computed in $O(1)$.

Outline

• Background and motivation
• Problem formulation
• Challenges and core techniques
• Online approaches for learning class imbalanced data streams
• Chunk-based approaches for learning class imbalanced data streams
• Performance assessment
• Two real world problems
• Remarks and next challenges
Tweet Topic Classification

\[
x \rightarrow \text{Learner 1} \rightarrow \hat{y}
\]

\[(x, y)\]
Characteristics of Tweet Topic Classification

- **Online problem**: feedback that generates supervised samples is potentially instantaneous.
- Class imbalance.
- **Concept drifts** may affect $p(y|x)$, though not so common.

---

Characteristics of Tweet Topic Classification

- Gradual concept drifts affecting $p(y)$ are very common.
- Gradual class evolution.
  - Recurrence is different from recurrent concepts, as it does not mean that a whole concept reoccurs.

Class-Based Ensemble for Class Evolution (CBCE)

- Each base model is a binary classifier which implements the one-versus-all strategy.
  - Class represented by the model is the positive +1 class.
  - All other classes compose the negative -1 class.
- The class $c_i$ predicted by the ensemble is the class with maximum likelihood $p(x|c_i)$. 
Dealing with Class Evolution

• The use of one base model for each class is a natural way of dealing with class emergence, disappearance and reoccurrence.
Dealing with Class Evolution

- The use of one base model for each class is a natural way of dealing with class emergence, disappearance and reoccurrence.

\[
\begin{align*}
\text{Model } f_{t}^{c_1} \\
\text{Model } f_{t}^{c_2} \\
\text{Model } f_{t}^{c_3} \\
\text{Model } f_{t}^{c_4}
\end{align*}
\]
Dealing with Class Evolution

- The use of one base model for each class is a natural way of dealing with class emergence, disappearance and reoccurrence.
Dealing with Concept Drifts on $p(y)$ and Class Imbalance

- Tracks proportion of examples of each class over time as OOB and UOB to deal with gradual concept drifts on $p(y)$.
  - If a given class becomes too small, it is considered to have disappeared.

- Given the one-versus-all strategy, the positive classes are likely to be the minorities for each model.
  - Undersampling of negative examples for training when they are majority.
Dealing with Concept Drifts on $p(y|x)$

- DDM monitoring error of ensemble.
- Reset whole ensemble upon drift detection.

All these strategies are online, if the base learner is online.
Sample Results Using Online Kernelized Logistic Regression as Base Learner

<table>
<thead>
<tr>
<th>Friedman Test (Nemenyi test at $\alpha = 0.05$) Result Considering the G-mean Values in Synthetic &amp; Tweet Streams</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthetic Streams</td>
</tr>
<tr>
<td>Tweet Streams</td>
</tr>
</tbody>
</table>

CBCE outperformed the other approaches across data streams in terms of overall G-mean.

For some twitter data streams, DDM helped and for some it did not help.
The Fraud Detection Pipeline

Terminal

Purchase

TX request

Transaction Blocking Rules

Real time

TX auth. $x$

Scoring Rules

 Alerts score

Alerts

Near real time

Classifier

Feedbacks $(x,y)$

Investigators

Offline

Disputes / Delays $(x,y)$

Real-time TX auth.

Scoring Rules

Alerts

Near real-time TX auth. $x$

Classifier
Characteristics of Fraud Detection Learning Systems

- **Class imbalance** (~0.2% of transactions are frauds).
- **Concept drift** may happen (customer habits may change, fraud strategies may change).
- Supervised information has a **selection bias** (feedback samples are transactions more likely to be fraud than the delayed transactions).
- Most supervised information arrives with a considerable delay (verification latency).

Characteristics of Fraud Detection Learning Systems

Delayed Information

day $t-\delta -1$  day $t-\delta$  ....

Feedbacks

day $t-3$  day $t-2$  day $t-1$

This is recent (valuable)

This is old (less valuable)
Learning-Based Solutions for Fraud Detection

Rationale: “Feedback and delayed samples are different in nature and should be exploited differently”

Two types of learners:

- Learn examples created from investigators’ feedback:
  \[ F_t = \text{TRAIN} \left( \{F_t, \ldots, F_{t-(Q-1)}\} \right) \]

- Learn examples with delayed labels.
  \[ D_t = \text{TRAIN} \left( \{D_{t-\delta}, \ldots, D_{t-(\delta+M-1)}\} \right) \]

Combination rule:

\[ P_{A_t}(+|x) = \alpha P_{F_t}(+|x) + (1 - \alpha) P_{D_t}(+|x) \]
Adaptation Strategies for Delayed Data

- Sliding windows:

- Ensemble
Sample Results Using Random Forest as Base Learner

<table>
<thead>
<tr>
<th>Classifier</th>
<th>Dataset</th>
<th>Average $P_k$ mean (std)</th>
<th>Average $P_k$ sum of ranks</th>
<th>Average AUC mean (std)</th>
<th>Average AUC sum of ranks</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A^W$</td>
<td>2014-2015</td>
<td>0.77 (0.21)</td>
<td>1796.50</td>
<td>0.94 (0.02)</td>
<td>1396.00</td>
</tr>
<tr>
<td>$F$</td>
<td>2014-2015</td>
<td>0.73 (0.23)</td>
<td>1632.00</td>
<td>0.87 (0.05)</td>
<td>409.00</td>
</tr>
<tr>
<td>$W$</td>
<td>2014-2015</td>
<td>0.61 (0.25)</td>
<td>1055.50</td>
<td>0.91 (0.04)</td>
<td>865.00</td>
</tr>
<tr>
<td>$W^D$</td>
<td>2014-2015</td>
<td>0.57 (0.26)</td>
<td>889.00</td>
<td>0.94 (0.03)</td>
<td>1315.00</td>
</tr>
<tr>
<td>$A^W$</td>
<td>2013</td>
<td>0.75 (0.20)</td>
<td>732.00</td>
<td>0.94 (0.03)</td>
<td>631.00</td>
</tr>
<tr>
<td>$F$</td>
<td>2013</td>
<td>0.73 (0.21)</td>
<td>693.00</td>
<td>0.89 (0.05)</td>
<td>229.00</td>
</tr>
<tr>
<td>$W$</td>
<td>2013</td>
<td>0.54 (0.25)</td>
<td>434.00</td>
<td>0.91 (0.05)</td>
<td>355.00</td>
</tr>
<tr>
<td>$W^D$</td>
<td>2013</td>
<td>0.50 (0.23)</td>
<td>345.00</td>
<td>0.93 (0.03)</td>
<td>539.00</td>
</tr>
</tbody>
</table>

Leandro Minku  http://www.cs.le.ac.uk/people/llm11/
Outline

• Background and motivation
• Problem formulation
• Challenges and core techniques
• Online approaches for learning class imbalanced data streams
• Chunk-based approaches for learning class imbalanced data streams
• Performance assessment
• Two real world problems
• Remarks and next challenges
Remarks and Next Challenges

• Overview of core techniques to deal with challenges posed by data streams.

• Learning class imbalanced data streams require a combination of several different core techniques to be used.

• Each technique has potential advantages and disadvantages based on the application to be tackled.

• Still, there are several challenges requiring more attention, when adopting more realistic scenarios, e.g.:
  • Class evolution.
  • Scarce supervised information.
  • Large delays in supervised information (verification latency).
  • Biased samples.

• Not many datasets are available in realistic conditions.