Improving Credit Scoring Performance Using Delay-Aware Streaming Analytics

Joshua Plasse, Niall Adams

Department of Mathematics, Imperial College London

August 2017
Population drift remains a challenge in credit-risk classification.

- Unpredictable changes over time create modeling difficulties.
- Ideally, we seek to use information as soon as it becomes available.
- In principle, streaming classification methods can be used which are able to handle drift and incorporate information as it arrives and are well studied.
- Delays in the arrival time of classification labels with respect to their corresponding feature vectors create issues for the deployment of statistical methods and are much less studied.
Purpose of the Talk:

This talk is about:

- highlighting the importance of delayed labels in streaming classification,
- showing that a simple classifier, that makes realistic assumptions on the delays, can increase performance,
- demonstrating improved performance incorporating delays for unsecured personal loan (UPL) score cards.
A data stream is an ordered sequence of data items arriving at high frequency and whose generating process is likely to drift (change) in time.

Streaming classification is the act of mapping each observation from a data stream to a fixed, predetermined set of classes.

Due to dynamics of the stream, streaming classifiers must be:

(⋆) Sequential,
(⋆) Single-pass,
(⋆) Adaptive.
**Setting the Stage:**

- **Delayed labels** are the true classes for *previously observed* feature vectors which become available after some time (or lag).

- Aspect of streaming data which is:
  - **under researched**,
  - and (most) literature assumes an **unrealistic** label-arrival process (e.g. “1-tick label arrival”).

- **Example:** a stream consisting of daily *UPL applications* where the status of the loan (default or not) arrives at a later time, i.e., are **delayed**. **Incorrect** to assume that the loan status becomes available on the next day.
Notation:

- Data streams assumed to take the form:
  \[
  \langle d_1, \ldots, d_t, \ldots \rangle, \quad d_t = (\mathcal{X}_t, \mathcal{L}_t).
  \]

- \(\mathcal{X}_t\) - feature vectors arriving at time \(t\).
  - Each \(x \in \mathcal{X}_t\) has a class label \(\ell \in \{c_i\}_{i=1}^{K}\).

- \(\mathcal{L}_t\) - delayed labels for vectors observed before time \(t\).

\[
\mathcal{X}_t = \left\{ x_t^{(k)} \right\}_{k=1}^{\lvert \mathcal{X}_t \rvert}, \quad \mathcal{L}_t = \left\{ \ell_k^{(m)} \right\}_{(k,m) \in \mathcal{I}_t}, \quad \mathcal{D}_t = \left\{ x_k^{(m)} \right\}_{(k,m) \in \mathcal{I}_t}
\]

- \((\mathcal{D}_t, \mathcal{L}_t)\) - a labeled collection available at time \(t\).
Arrival Process Example:

\[ X_1 = \{ x_1^{(1)}, x_1^{(2)}, x_1^{(3)} \} \]
\[ L_1 = \emptyset \]

\[ X_2 = \{ x_2^{(1)} \} \]
\[ L_2 = \{ \ell_1^{(2)}, \ell_1^{(3)} \} \]

\[ D_1 = \emptyset \]

\[ X_3 = \{ x_3^{(1)}, x_3^{(2)} \} \]
\[ L_3 = \{ \ell_1^{(1)}, \ell_2^{(1)} \} \]

\[ D_2 = \{ x_1^{(2)}, x_1^{(3)} \} \]

\[ D_3 = \{ x_1^{(1)}, x_2^{(1)} \} \]
Interpretation of Delay:

- Each feature vector is associated with the triple \((x_t, \ell_t, \tau_t)\)
  - \(\ell_t\) is the true label for \(x_t\)
  - \(\tau_t \in \mathbb{Z}^+ \setminus \{0\}\) is the delay
- If \(\tau_t = s\), then the label for \(x_t\) will arrive at time \((t + s)\):

\[
X_t = x_t, \quad L_{t+s} = \ell_t.
\]


- This paper proposes a much more general framework to streaming classification using delayed labels.
Dealing with Drift:

- **Recall:** any classifier needs to be **adaptive** to handle drift.
- Temporal adaptivity introduced through **forgetting factors** (FFs):
  - a sequence of scalars that continuously **down-weights** historical data as new data arrives,
  - can be considered as a continuous analogue of a **sliding window**.
- FFs are incorporated into parameter estimation via a **weighted-maximum likelihood** approach.
- Consider a variation of **linear discriminant analysis (LDA):** need to consider class conditional mean vectors and covariance matrices (Gaussian) and class priors (multinomial).
Sequential Updates:

Adaptive estimates for the **mean vector** and **covariance matrix** are given, **recursively**, by:

\[
\begin{align*}
n_t &= \lambda_{t-1}n_{t-1} + \omega_t \\
\tilde{\mu}_t &= \left(1 - \frac{\omega_t}{n_t}\right)\tilde{\mu}_{t-1} + \frac{\omega_t}{n_t}x_t \\
\tilde{\Pi}_t &= \left(1 - \frac{\omega_t}{n_t}\right)\tilde{\Pi}_{t-1} + \frac{\omega_t}{n_t}x_t x_t^T \\
\tilde{\Sigma}_t &= \tilde{\Pi}_t - \tilde{\mu}_t\tilde{\mu}_t^T.
\end{align*}
\]

- \(\lambda_{t-1} \in [0, 1]\) \(\equiv\) a FF that affect how **quickly** (or slowly) the estimates react to change (**data driven**),
- \(\omega_t \equiv\) additional **weight** given to \(x_t\) (**delay driven**).
Sequential Updates:

Adaptive estimates can also be computed for the multinomial distribution. Recall:

- \( \{c_i\}_{i=1}^{K} \) are the classes
- \( \{\ell_i\}_{i=1}^{t} \) are the labels from the stream where for every \( i \):
  \[
  \ell_i \in \{c_i\}_{i=1}^{K} \quad \text{or} \quad \ell_i = \emptyset \quad \text{(hasn’t arrived)}.
  \]

An adaptive estimate for the \( j^{th} \) cell-probability is given by:

\[
\begin{align*}
  n_t &= \lambda_{t-1} n_{t-1} + \omega_t \\
  \tilde{p}^{(j)}_t &= \frac{\omega_t}{n_t} I(\ell_t = c_j) + \left( 1 - \frac{\omega_t}{n_t} \right) \tilde{p}^{(j)}_{t-1},
\end{align*}
\]

where we have assumed that \( \ell_t \neq \emptyset \).
Suppose the **fixed-forgetting** case where $\lambda_{t-1} \equiv \lambda \in [0, 1]$. Then

$$\tilde{\mu}_t = \frac{1}{n_t} \left[ \lambda^{t-1} x_1 + \cdots + \lambda x_{t-1} + x_t \right].$$

- $\lambda = 1 \implies \tilde{\mu}_t = \bar{x}$, the **static** sample mean
- $\lambda = 0 \implies \tilde{\mu}_t = x_t$, the **most recent** observation

Case where the FFs are functions of time is referred to as **adaptive forgetting**.
Figure 1: **Red Line:** FF mean; **Blue Line:** static mean
The recursive estimates derived lay the groundwork for a **streaming linear discriminant analysis** algorithm.

**Assumptions:**

- \( X_t = x_t \) (for now),
- \( \ell_t \in \{0, 1\} - 2\text{-class problem}, \)
- fixed forgetting case (for now),
- typical aggregated covariance and allocation rule used.
Constructing the Classifier:

- Suppose at time $t$ we have the labeled ordered set:
  \[
  \{(x_k, \ell_k, \tau_k)\}_{k \in \mathcal{I}_t}, \quad k < t.
  \]

- For example: $x_k$ could be a feature vector corresponding to a loan application received on day $k$, $\ell_k$ is whether the applicant defaults or not and $\tau_k$ represents the number of days it takes to get $\ell_k$.

- On the stream, the parameters associated with class $\ell_k$ are updated by incorporating $x_k$ into the parameter estimation with some specified weight.
Constructing the Classifier:

Let

- \( \lambda_j \in (0, 1) \) be the fixed FF for class \( j \in \{0, 1\} \),
- \( \lambda^{pr} \in (0, 1) \) be the fixed FF for the prior probabilities.

The additional weight given to \( x_k \) is chosen according to:

\[
\omega_k = \lambda^\tau_k \text{ or } (\lambda^{pr})^\tau_k .
\]

Observations get less weight the larger the delay.
Blocking and Averaging:

- Suppose now that multiple feature vectors arrive at each time $t$.
  - Several delayed labels can arrive for feature vectors that were observed at the same time.
- These vectors should be given the same weight in the parameter updating.
- Repeated application of the recursive update equations does not allow for this!
Blocking and Averaging:

- To alleviate this problem consider the blocks

\[ B_k^{(j)} = \left\{ x_k^{(m)} \in D_t \mid \ell_k^{(m)} = j \right\}, \quad \forall (k, m) \in \mathcal{I}_t, \quad j \in \{0, 1\}. \]

- Just ordering feature vectors by arrival time and label.

**Idea:** average over the blocks to get one feature vector to use in the updating of each class, for every \( k \).

- In credit scoring for an application \( x \in B_k^{(j)} \) it must be the case that:
  - the application \( x \) was received on day \( k \),
  - the status of the loan was revealed on day \( t \)
  - the label for \( x \) is class \( j \in \{0, 1\} \).

- Weights previously discussed can be scaled by the cardinality of the block.
Demonstration:

The Data:

- Over 36,000 unsecured personal loans from a major UK bank recorded over 1994-1995.

- Binary response gives the loan status of the applicant.

- Features: age of applicant, loan amount etc.

- Not high frequency, so why relevant?!
  - Exhibits population drift
  - Delay mechanism is inherent in the application.
The Delays:

- The data considered has no timing information associated with the labels.

**Fix:** synthetically introduce the delays which mimic how they may occur in reality:

- Any loan granted in the last three months never has their label arrive.

- Any loan granted in the first nine months has two options:
  - Bad (defaults) labels arrive randomly a minimum of three months from the start date.
  - Good (non-defaults) labels only arrive randomly in the last three months of the year.

- Streaming LDA was trained on 94 and tested on 95 and is compared to the baseline logistic regression classifier using area under the ROC curve as a performance measure.
Consider a grid \( G = \{0.9, 0.91, ..., 1\} \).

LDA was run for every combination of \((\lambda_0, \lambda_1, \lambda_{pr}) \in G^3\).

For each triple \((\lambda_0, \lambda_1, \lambda_{pr})\) the AUC from streaming LDA was compared to the AUC from logistic regression.

The upcoming plots report:

\[
\frac{\text{(LDA AUC)}}{\text{(Logistic AUC)}}
\]

Any value greater than one indicates that our method has a higher AUC when compared to logistic regression.
\( \lambda^{pr} \) Results:

![Graph showing \( \lambda^{pr} \) Results]

- LDA/Logit
- Joshua Plasse (ICL)
- Improving CS Performance
- August 2017
In the streaming paradigm a grid search may not always be possible.

Consider **adaptively tuning** the FFs according to a **stochastic gradient descent (SGD)** step:

\[ \lambda_t = \lambda_{t-1} - \eta \nabla \lambda \left[ J \left( \lambda_{t-1} | x_t, \tilde{\theta}_{t-1} \right) \right] \]

- \( \eta \equiv \text{a step-size} \)
- \( J \equiv \text{a continuous and differentiable cost function} \)
- \( \lambda_t \) could be for the class conditionals or the priors.

Applying SGD **and** incorporating delays is **tricky**.
Figure 2: Overall AUC: 0.785, 0.768
Conclusions:

- Handling delays in the proposed way has some promise for credit scoring and warrants further research.

- Delayed labels can be incorporated into a simple online classifier which:
  - better matches the structure of the real problem,
  - can perform better than a common classifier used in industry,
  - can remove the burden of having to subjectively choose the forgetting factors.

Acknowledgment: Research was supported by an Imperial College President’s PhD Scholarship.