Machine Learned Model Quality Monitoring in Fast Data and Streaming Applications

Emre Velipasaoglu

Lightbend

Strata DATA CONFERENCE
model quality
* same data generating distribution

(Some algorithms tolerate violation of this to a certain degree.)
core problem

stream
core problem

stream

population change
core problem

stream

sensor failure
core problem

concept drift

stream
core problem

stream

emerging concept
can active learning help?
can active learning help?
can active learning help?
can active learning help?
can active learning help?
Can active learning help?
a better solution

data

feature extraction

classifier

predictions
a better solution

data

feature extraction

classifier

monitoring

predictions
a better solution

- Data
- Feature extraction
- Classifier
- Monitoring
- Predictions
a better solution

data

feature extraction

classifier

monitoring

predictions
a better solution

data

feature extraction

classifier

predictions

monitoring
a better solution

- data
  - feature extraction
    - classifier
      - predictions
    - monitoring
  - labeling
a better solution

data -> feature extraction -> classifier -> change detection -> predictions

labeling -> adaptation
monitor how?

- supervised
  - statistical process control
  - sequential analysis
  - error distribution monitoring

- unsupervised
  - clustering / novelty detection
  - feature distribution monitoring
  - model-dependent monitoring
adapt how?

explicit mechanisms:
- windowing
- weighting
- sampling

implicit mechanisms:
- pure methods
- ensemble methods
which method?
monitor how?

- supervised
  - statistical process control
  - sequential analysis
  - error distribution monitoring
- unsupervised
  - clustering / novelty detection
  - feature distribution monitoring
  - model-dependent monitoring
ML theory: samples \rightarrow errors
Drift Detection Method [DDM]

- # of errors is Binomial:

\[ \mu = np_t \]

\[ \sigma = \sqrt{\frac{p_t(1-p_t)}{n}} \]

- alert:

\[ p_t + \sigma_t \geq p_{min} + 3\sigma_{min} \]
statistical process control

- Drift Detection Method [DDM]
  - # of errors is Binomial:
    \[ \mu = np_t \]
    \[ \sigma = \sqrt{\frac{p_t(1 - p_t)}{n}} \]
  - alert:
    \[ p_t + \sigma_t \geq p_{min} + 3\sigma_{min} \]

- Early Drift Detection Method [EDDM]
  - distance between errors better for gradual drift
  - warn & start caching:
    \[ \frac{p_t + 2\sigma_t}{p_{max} + 2\sigma_{max}} < 0.95 \]
  - alert and reset max:
    \[ \frac{p_t + 2\sigma_t}{p_{max} + 2\sigma_{max}} < 0.90 \]
monitor how?

supervised

- statistical process control
- sequential analysis
- error distribution monitoring

unsupervised

- clustering / novelty detection
- feature distribution monitoring
- model-dependent monitoring
## Sequential Analysis

- **Linear Four Rates (LFR)**
  - Stationary data $\Rightarrow$ Constant contingency table

<table>
<thead>
<tr>
<th>Predicted</th>
<th>True</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>TN</td>
<td></td>
<td>FN</td>
</tr>
<tr>
<td>1</td>
<td>FP</td>
<td></td>
<td>TP</td>
</tr>
</tbody>
</table>
sequential analysis

- Linear Four Rates [LFR]
  - stationary data => constant contingency table
  - calculate four rates

<table>
<thead>
<tr>
<th>Predicted</th>
<th>True</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>TN</td>
<td>FN</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>FP</td>
<td>TP</td>
<td></td>
</tr>
</tbody>
</table>

\[ P_{npv} = \frac{TN}{TN + FN} \]
\[ P_{ppv/precision} = \frac{TP}{TP + FP} \]
\[ P_{tnr/specificity} = \frac{TN}{TN + FP} \]
\[ P_{tpr/recall} = \frac{TP}{TP + FN} \]
### Sequential Analysis

- **Linear Four Rates [LFR]**
  - stationary data => constant contingency table
  - calculate four rates
  - incremental updates

<table>
<thead>
<tr>
<th>Predicted</th>
<th>True</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>TN</td>
<td>FN</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>FP</td>
<td>TP</td>
<td></td>
</tr>
</tbody>
</table>

- \( P_{npv} = \frac{TN}{TN + FN} \)
- \( P_{ppv/precision} = \frac{TP}{TP + FP} \)
- \( P_{inv/specificity} = \frac{TN}{TN + FP} \)
- \( P_{prv/recall} = \frac{TP}{TP + FN} \)

\[
P_{*}^{t} \leftarrow \eta_{*} P_{*}^{t-1} + (1 - \eta_{*})I_{y_{t} = \hat{y}_{t}}
\]
sequential analysis

- Linear Four Rates [LFR]
  - stationary data => constant contingency table
  - calculate four rates
  - incremental updates
  - test for change
    - Monte Carlo sampling for significance level
    - Bonferoni correction for correlated tests
  - $O(1)$
  - Better than (E)DDM for class imbalance

<table>
<thead>
<tr>
<th>Predicted</th>
<th>True</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>TN</td>
<td>FN</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>FP</td>
<td>TP</td>
<td></td>
</tr>
</tbody>
</table>

\[
P_{npv} = \frac{TN}{TN + FN} \quad P_{ppv/precision} = \frac{TP}{TP + FP}
\]

\[
P_{inv/specificity} = \frac{TN}{TN + FP} \quad P_{prep/recall} = \frac{TP}{TP + FN}
\]

\[
P_*^t \leftarrow \eta_* P_*^{t-1} + (1 - \eta_*) I_{y_t = \hat{y}_t}
\]
monitor how?

supervised

- statistical process control
- sequential analysis
- error distribution monitoring

unsupervised

- clustering / novelty detection
- feature distribution monitoring
- model-dependent monitoring
error distribution monitoring

- ADaptive WINdowing [ADWIN]
  - Consider all partitions of a window
  - Drop the last element if any
  - Efficient version $O(\log W)$
    - Data structure for windows $\sim$ exponential histograms
    - Drop last window rather than last element

\[ |\mu_0 - \mu_1| > \theta_{\text{Hoeffding}} \]
resampling

- Prediction loss over random permutations vs. ordered training data
- Parallel permutation test version available
- Still expensive
- Only method directly applicable to regression setting
- Side note: Even with finite training set, drift could be problematic if model is developed naively.
monitor how?

supervised

statistical process control
sequential analysis
error distribution monitoring

unsupervised

clustering / novelty detection
feature distribution monitoring
model-dependent monitoring
clustering / novelty detection

- OLINDDA: K-means, periodically merge unknown to known or flag
- MINAS: micro-clusters, incremental stream clustering
- DETECTNOD: Discrete Cosine Transform to estimate distances efficiently
- Woo-ensemble: Treat outliers as potential emerging class centroids
- ECSMiner: Store and use cluster summary efficiently
- GC3: Grid based clustering
clustering / novelty detection

- OLINDDA: K-means, periodically merge unknown to known or flag
- MINAS: micro-clusters, incremental stream clustering
- DETECTNOD: Discrete Cosine Transform to estimate distances efficiently
- Woo-ensemble: Treat outliers as potential emerging class centroids
- ECSMiner: Store and use cluster summary efficiently
- GC3: Grid based clustering

Curse of Dimensionality
monitor how?

- supervised
  - statistical process control
  - sequential analysis
  - error distribution monitoring

- unsupervised
  - clustering / novelty detection
  - feature distribution monitoring
  - model-dependent monitoring
Monitor individual features

Many ways to compare:
- Pearson correlation [Change of Concept - CoC]
- Hellinger distance [HDDDM] \( \sim O(DB) \)

Use PCA to reduce the number of features to track (top [PCA-1] or bottom [PCA-2] n%)
monitor how?

supervised

unsupervised

statistical process control
sequential analysis
error distribution monitoring

clustering / novelty detection
feature distribution monitoring
model-dependent monitoring
model-dependent monitoring

- Not all changes matter
- Posterior probability estimate
  - Use [A-distance] ~ generalized KS distance
    - designed to be less sensitive to irrelevant changes
model-dependent monitoring

- [Margin] distribution
  - rank statistic on density estimates for a binary representation of the data,
  - compare average margins of a linear classifier induced by the 1-norm SVM
  - based on the average zero-one or sigmoid error rate of an SVM classifier

- Generalized margin [MD3]:
  - Embed base classifier in a Random Feature Bagged Ensemble
  - Margin == high disagreement region of the ensemble
adapt how?

- explicit mechanisms
- implicit mechanisms

- windowing
- weighting
- sampling
- pure methods
- ensemble methods
explicit mechanisms for adaptation

Drop the last sub-window if threshold is exceeded. = Adaptively shrink window during drift.
explicit mechanisms for adaptation

*Adaptation goes through a similar refinement process.
adapt how?

explicit mechanisms

implicit mechanisms

windowing

weighting

sampling

pure methods

ensemble methods
explicit mechanisms for adaptation

Biased Reservoir Sampling

bias: \( f(r, t) = e^{-\lambda(t-r)} \)

capacity: \( N = \frac{1}{\lambda} \)

overwrite / exchange randomly w/ Prob{ %full } or append
adapt how?

- explicit mechanisms
  - windowing
  - weighting
  - sampling

- implicit mechanisms

- pure methods
  - ensemble methods
implicit mechanisms for adaptation

Ensemble Based Adaptation

ensemble 1

ensemble (N-1)

ensemble N

train new member
implicit mechanisms for adaptation

Ensemble Based Adaptation

- ensemble 1
  - retire / decay

- ensemble (N-1)

- ensemble N
  - train new member
implicit mechanisms for adaptation

Ensemble Based Adaptation

- **ensemble 1**
  - retire / decay

- **ensemble (N-1)**
  - recurring

- **ensemble N**
  - train new member
Ensemble Based Adaptation

- Online NonStationary boosting [ONSboost]
- NonStationary Random Forests [NSRF]
- Dynamic Weighted Majority [DWM]
- Learn++ for NonStationary Environments [Learn++.NSE]
<table>
<thead>
<tr>
<th>Method</th>
<th>Efficiency</th>
<th>Pros</th>
<th>Cons</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDM/EDDM</td>
<td>$O(1)$</td>
<td>no data stored</td>
<td>label cost</td>
<td>sampling necessary in case of fast data,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>false alarms</td>
<td>microservices architecture ideal</td>
</tr>
<tr>
<td>LFR</td>
<td>$O(1)$</td>
<td>class imbalance OK</td>
<td>label cost</td>
<td></td>
</tr>
<tr>
<td>ADWIN</td>
<td>$O(\log W)$</td>
<td>better change</td>
<td>label cost</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>localization</td>
<td></td>
<td></td>
</tr>
<tr>
<td>JIT</td>
<td>$O(\log W)$</td>
<td>no labels required</td>
<td>only for abrupt changes</td>
<td>best localization</td>
</tr>
<tr>
<td>Method</td>
<td>Efficiency</td>
<td>Pros</td>
<td>Cons</td>
<td>Notes</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------------------</td>
<td>-----------------------</td>
<td>---------------------------</td>
<td>----------------------------------------------------------------------</td>
</tr>
<tr>
<td>ECSMiner / GC3</td>
<td>$O(W^2 / k)$</td>
<td>emerging concepts</td>
<td>clusterable drift only</td>
<td>use if emerging concepts expected</td>
</tr>
<tr>
<td></td>
<td>$O(G \log C)$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HDDDM</td>
<td>$O(DB)$</td>
<td>no labels</td>
<td>not for population drift or class imbalance</td>
<td>better when combined with PCA</td>
</tr>
<tr>
<td>A-distance</td>
<td>$O(\log W)$</td>
<td>no labels</td>
<td>less false positives compared to HDDDM</td>
<td>good choice for unsupervised</td>
</tr>
<tr>
<td>Margin / MD3</td>
<td>Learning, detection, adaptation bundled</td>
<td>reduced false alarms</td>
<td>must use feature bagged ensembles</td>
<td>best choice but must commit to using the specific machine learning algorithms</td>
</tr>
<tr>
<td>Ensemble methods</td>
<td></td>
<td>recurring concepts</td>
<td>large batches</td>
<td></td>
</tr>
</tbody>
</table>
References

https://gist.github.com/emrev12/0d75dc2d6c3e80012d10a82712b8ced0
thank you
	emre.velipasaoglu@Lightbend.com