Protecting Your Machine Learning Against Drift: An Introduction

Oliver Cobb - Applied Machine Learning Researcher
We will look at

- What drift is and why it pays to detect it.
- The different types of drift.
- How drift can be detected in a principled manner.
- The anatomy of a drift detector.
- Demystify concepts such as ‘online detectors’, ‘permutation tests’, ‘MMD test’ etc.
Preliminaries

- Wish to use $y$ for some prediction/output.
- Can't observe $y$, can observe $x=(x_1,\ldots,x_d)$ that are related.
- We fit a model $M$ to predict $y$ from $x$ and use $\hat{y} = M(x)$ as the prediction/output.

- Performance on held out data gives estimate for future performance...

  Assuming the process underlying $x$ and $y$ remains constant.
What is drift?

- When the process underlying x and y during deployment differs from the process that generated the training data.
- Can no longer expect the model's performance during deployment to match that observed on held out training data.

What is drift?

\[ p(x,y) = p(y|x)p(x) = p(x|y)p(y) \]
What is drift?

Labels available: **Supervised** drift detection

-> Can monitor model performance directly.
What is drift?

Labels unavailable: **Unsupervised** drift detection

-> Can't monitor model performance. Must operate in high dimensions.

No Drift

Concept Drift $\Delta p(y|x)$

Prior drift $\Delta p(y)$

Covariate Drift $\Delta p(x)$
Change or chance?

- We don't expect new data to look identical to training data.
- So how do we decide whether fluctuations are due to drift or natural fluctuations?
- Statistical hypothesis testing!
Statistical Hypothesis Testing

- Before observing data $X$, specify null and alternative hypothesis, $H_0$ and $H_1$.
- Specify test statistic $S(X)$ we expect to be small if $H_0$ and large if $H_1$.
- Observe data, compute $S(X)$, compute $\hat{p} = P(\text{such an extreme } S(X) \mid H_0)$.
- Low $p$-value discredits $H_0$.
  - Believe $H_0 = \text{believe we just observed something very unlikely (prob } \hat{p})$.
  - Typically specify a threshold $p$ (FPR) in advance and reject null if $\hat{p} < p$. 
Offline Drift detection

- Let $q_0$ be distribution underlying training data, $X_0$.
- Let $q_1$ be the distribution underlying a batch of new data, $X_1$.
- $H_0: q_0 = q_1$. $H_1: q_0 \neq q_1$.
- $S(X_0, X_1)$ small if $H_0$ true, large if $H_1$ true.
- Compute $\hat{p}$. Flag drift if $\hat{p} < p$ for desired FPR $p$. 

The hard part!
Online Drift detection

- Data points $z=(x,y)$ arrive in sequence, $z_1, z_2, \ldots$ and we’d like to detect drift ASAP.
- Assumption:
  
  \[
  z_i \sim \begin{cases} 
  q_0 & \text{for } i < T^* \\
  q_1 & \text{for } i \geq T^* 
  \end{cases}
  \]

  where $q_0$ underlied training data and $q_0 \neq q_1$.
- At each time $t$ we perform a hypothesis test of $\{H_0: T^* > t\}$ vs $\{H_1: T^* \leq t\}$.
  
  i.e. “has drift occurred yet”?

![Diagram](image)

- First plot: Data points $(x_1, x_2)$ before drift.
- Second plot: Data points $(x_1, x_2)$ after drift.
- Third plot: Hypothesis test result.
Online Drift detectors - desired properties

- When a change occurs (at time $T^*$) the detector is typically fast to respond.
  - i.e. Expected Detection Delay, $\text{EDD} = E[T' - T^*]$, is small.
- Ability to specify the frequency of false detections when no change occurs.
  - i.e. We can specify Expected Run Time, $\text{ERT} = E[T' | T^* = \infty]$.
- There's an ERT vs EDD tradeoff.
  - Low EDD requires sensitivity, which means more false positives and low ERT.

Often overlooked
Windowing Strategies

- Can be fixed sized and disjoint, fixed size and overlapping or adaptive.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>0.21</td>
<td>1.51</td>
<td>-1.03</td>
<td>-0.08</td>
<td>1.46</td>
<td>0.80</td>
<td>1.13</td>
<td>0.17</td>
<td>-0.35</td>
<td>-0.12</td>
</tr>
<tr>
<td>$x_2$</td>
<td>-0.41</td>
<td>0.05</td>
<td>0.59</td>
<td>-0.94</td>
<td>0.73</td>
<td>0.74</td>
<td>-1.52</td>
<td>0.66</td>
<td>-1.19</td>
<td>-0.72</td>
</tr>
</tbody>
</table>
Windowing Strategies

- Can be fixed sized and disjoint, fixed size and overlapping or adaptive.

### Overlapping

<table>
<thead>
<tr>
<th>$t$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>0.21</td>
<td>1.51</td>
<td>-1.03</td>
<td>-0.08</td>
<td>1.46</td>
<td>0.80</td>
<td>1.13</td>
<td>0.17</td>
<td>-0.35</td>
<td>-0.12</td>
</tr>
<tr>
<td>$x_2$</td>
<td>-0.41</td>
<td>0.05</td>
<td>0.59</td>
<td>-0.94</td>
<td>0.73</td>
<td>0.74</td>
<td>-1.52</td>
<td>0.66</td>
<td>-1.19</td>
<td>-0.72</td>
</tr>
</tbody>
</table>
Windowing Strategies

- Can be fixed sized and disjoint, fixed size and overlapping or adaptive.
Windowing Strategies

- Can be fixed sized and disjoint, fixed size and overlapping or adaptive.
- Different windowing strategies call for different test statistics and threshold determination processes.
- They also each have their own pros and cons.
Disjoint Window Detectors

- The strategy adopted by existing alibi-detect detectors.
- Can compute p-value corresponding to any test statistic $S(X_0, X_1)$!
- This is achieved via a permutation test:
  - shuffle($X_0, X_1$) : $(X_0, X_1) \mapsto (X^*_0, X^*_1)$
  - alt_stats = [$S(shuffle(X_0, X_1))$ for _ in range(B)]
  - $\hat{p} = (\text{alt_stats} > S(X_0, X_1)).\text{mean()}$
- Performed anew for each test, so can learn new test statistic $S$ for each test.
  - Using a portion of the data.
  - Solves problem of specifying suitable $S$.
- Sensitive to choice of window size.
- Slow to respond to severe drift.
Overlapping Window Detectors

- Use test statistics that can be updated incrementally.
- Controlling the ERT much more difficult due to correlation between test outcomes.
- I’ve been working on a method to get around this.
- It allows any test statistic $S$ to be used.
- However it must be pre-specified. Problem of specifying $S$ remains.
- If a suitable test statistic can be specified, relative to disjoint detectors:
  - Computationally light.
  - Fast to respond to severe drift.
Adaptive Window Detectors

- Typically accumulate some notion out ‘outlierness’.
  - (At least those which operate at fixed cost).
- Hard to set threshold corresponding to a given ERT.
- I worked on a method to target a given ERT.
- However accumulating ‘outlierness’ not good for EDD!
- Why not?

Image credit: Basseville and Nikiforov (1993) - Detection of Abrupt Changes: Theory and Application
Drift = persistent outliers?

$q_0 = N(0,I_2)$
Drift = persistent outliers?

$q_0 = N(0, I_2)$

$q_1 = N(\mu, \sigma I_2)$
Drift = persistent outliers?

$q_0 = N(0,I_2)$

$q_1 = N(\mu,\sigma I_2)$
Drift = persistent outliers?

$q_0 = N(0, I_2)$

$q_1 = N(\mu, \sigma I_2)$
Drift = persistent outliers?

$q_0 = \mathcal{N}(0, I_2)$

$q_1 = \mathcal{N}(\mu, \sigma I_2)$
So if outlierness-based test statistics are not sufficient - what is?
Test statistics which estimate distance between $q_0$ and $q_1$.
This could be achieved by first estimating $\hat{q}_0$ and $\hat{q}_1$ and evaluating $d(\hat{q}_0, \hat{q}_1)$.
But with limited samples it is more efficient to directly estimate $d(q_0, q_1)$.
$d(q_0, q_1) = \text{MMD}_k(q_0, q_1)$ features prominently in alibi-detect.
Maximum Mean Discrepancy (MMD\(_k\))

- Transforms problem from specifying a distance \(d(q_0,q_1)\) between distributions to specifying a similarity (kernel) \(k(z_0,z_1)\) between data points.
  - Which is much more intuitive and possible for all data modalities.
- Typically \(k(z_0,z_1)=\Phi(z_0)^T\Phi(z_1)\) for some projection \(\Phi\).
  - E.g. For \(\Phi\) can use pretrained ConvNet (images), transformer (text) components.
- Then:

\[
\text{MMD}_k(q_0,q_1) = \text{Avg. similarity between data points within training set} \\
+ \text{Avg. similarity between data points within current window} \\
- 2*\text{Avg. similarity between training data point and window data point}
\]

\[
= \frac{1}{n_x(n_x - 1)} \sum_{x_1 \neq x_2 \in \mathcal{X}} k(x_1, x_2) + \frac{1}{n_y(n_y - 1)} \sum_{y_1 \neq y_2 \in \mathcal{Y}} k(y_1, y_2) - \frac{2}{n_x n_y} \sum_{(x, y) \in \mathcal{X} \times \mathcal{Y}} k(x, y)
\]
Maximum Mean Discrepancy (MMD)

\[ \text{MMD}_k(q_0, q_1) = \text{Avg. similarity between data points within training set (small)} \]
\[ + \text{Avg. similarity between data points within current window (large)} \]
\[ - 2 \times \text{Avg. similarity between training data point and window data point (small)} \]
\[ = \text{large (e.g. } 0.1 + 0.8 - 2 \times 0.1 = 0.7) \]
Maximum Mean Discrepancy (MMD) - Permuted

\[ MMD_k(q_0, q_1) = \text{Avg. similarity between data points within training set (small)} \]
\[ + \text{Avg. similarity between data points within current window (small)} \]
\[ - 2 \times \text{Avg. similarity between training data point and window data point (small)} \]

= small (e.g. 0.1 + 0.09 - 2 \times 0.09 = 0.01)
Summary and demo

ALIBI DETECT

github.com/SeldonIO/alibi-detect
Thanks for watching!

github.com/SeldonIO/alibi-detect

@SeldonResearch

oc@seldon.io