The Past and the dubious: Concept Drift

Snehith allamraju
Typical ML workflow

DATA → TRAINING → ALGORITHM → EVALUATION → MODEL → PREDICTION

DATA → TESTING → ALGORITHM → EVALUATION → MODEL → PRODUCTION

DATA → TESTING → ALGORITHM → EVALUATION → MODEL → PREDICTION

DATA → TESTING → ALGORITHM → EVALUATION → MODEL → PRODUCTION

DATA → TESTING → ALGORITHM → EVALUATION → MODEL → PREDICTION

DATA → TESTING → ALGORITHM → EVALUATION → MODEL → PRODUCTION
Static model

Model does not change

Process changes

source: Evonik Industries
Types of drift in Machine Learning

- **Concept Drift**
  - Statistical properties of Target variable change
  - Hidden Context: A dependency not given explicitly in the form of input features.

- **Data Drift**
  - Statistical properties of the input data change
  - Seasonality, trend etc
  - No impact to previously labeled data

- **Upstream/pipeline changes**
  - Operational changes in data pipeline
  - Change in encoding, null value handling etc
The Past and the Dubious : Concept Drift

- Renders the model built on data, inconsistent with new data
- Differing decision boundary for the same input data over time

Examples
- Weather Prediction based on historical events
- Buying behavior based on customer preferences
- Fraud prediction based on payments
Types of concept drift

- **Abrupt Change in Class Distribution**: Class Distribution modified at a slower rate.
- **Drifts reoccur with time. Can be cyclic**.
What can we do / must be done

- **Monitor the distribution of the incoming data**: Small distribution changes may require an update to the model on more recent data, while large deviations may require a full re-training of the model.

- **Regular model refresh necessary**: Use SME to review model outputs periodically, and to determine new labels/data points that can be used to update the model.

- **Assume** that concept drift would occur, and periodically update.

- **Adapt to concept drift**, with limited resources (time and memory).

[https://www.dataversity.net/handling-concept-drift-in-interventional-machine-learning-systems/](https://www.dataversity.net/handling-concept-drift-in-interventional-machine-learning-systems/)
Handling Concept Drift

Three Basic approaches

- **Instance Selection**
- Instance Weighting
- Ensemble learning

Consists in generalizing from a window that moves over recently arrived instances and uses the learnt concepts for prediction only in the immediate future.

The window size can be fixed or heuristically determined (Adaptive)

*The Problem of Concept Drift: Definitions and Related Work - Alexev Tsymbalo paper. (April 29, 2004)*
Concept drift with Python - learning from evolving data streams

Drift Detection: `skmultiflow.drift_detection`

The `skmultiflow.drift_detection` module includes methods for Concept Drift Detection.

- `drift_detection.ADWIN`: Adaptive Windowing method for concept drift detection.
- `drift_detection.DDM`: Drift Detection Method.
- `drift_detection.EDDM`: Early Drift Detection Method.
ADWIN (ADaptive WINdowing)

- ADWIN efficiently keeps a variable-length window of recent items.
- Window has the maximal length statistically consistent with the hypothesis *there has been no change in the average value inside the window*.
- This window is further divided into two sub-windows (W0, W1) used to determine if a change has happened.
- ADWIN compares the average between W0 and W1 to confirm that they correspond to the same distribution.
- Concept drift is detected if the distribution equality no longer holds.
- Upon detecting a drift, W0 is replaced by W1 and a new W1 is initialized.
- ADWIN uses a confidence value $\delta = \in (0, 1)$ to determine if the two sub-windows correspond to the same distribution.
In [1]:
>>> # Imports
>>> import numpy as np
>>> from skmultiflow.drift_detection.adwin import ADWIN
>>> adwin = ADWIN()

# Simulating a data stream as a normal distribution of 1's and 0's
>>> data_stream = np.random.randint(2, size=2000)

# Changing the data concept from index 999 to 2000
>>> for i in range(999, 2000):
    data_stream[i] = np.random.randint(4, high=80)
    #print(data_stream[i])
    #if adwin.detected_change():
        #print('Change detected in data: ' + str(data_stream[i]) + ' - at index: ' + str(i))

# Adding stream elements to ADWIN and verifying if drift occurred
>>> for i in range(2000):
    adwin.add_element(data_stream[i])
    print(data_stream[i])
    if adwin.detected_change():
        print('Change detected in data: ' + str(data_stream[i]) + ' - at index: ' + str(i))
Change detected in data: 69 - at index: 1055
EDDM (Early Drift Detection Method)

- Method to detect changes in the distribution of the training examples which monitors the online error-rate
- Learning takes place in a sequence of trials
- When a new training example is available, it is classified using the current model.
- Warning level and Drift level are defined
- A new context is declared, if in a sequence of examples, the error increases reaching the warning level at example Kw, and the drift level at example Kd.
- This is taken as an indication of a change in the distribution of the examples.
- Works well with and gets good results with slow gradual changes

https://www.es.upc.edu/~abifet/EDD
EDDM - Early Drift Detection Method

In [2]:

```python
>>> # Imports
>>> import numpy as np
>>> from skmultiflow.drift_detection.eddm import EDDM
>>> eddm = EDDM()
>>> # Simulating a data stream as a normal distribution of 1's and 0's
>>> data_stream = np.random.randint(2, size=2000)
>>> # Changing the data concept from index 999 to 1500, simulating an increase in error rate
>>> for i in range(999, 1500):
...     data_stream[i] = 0
>>> # Adding stream elements to EDDM and verifying if drift occurred
>>> for i in range(2000):
...     eddm.add_element(data_stream[i])
...     if eddm.detected_warning_zone():
...         print('Warning zone has been detected in data: ' + str(data_stream[i]) + ' - of index: ' + str(i))
...     if eddm.detected_change():
...         print('Change has been detected in data: ' + str(data_stream[i]) + ' - of index: ' + str(i))
```

Warning zone has been detected in data: 1 - of index: 69
Warning zone has been detected in data: 0 - of index: 70
Warning zone has been detected in data: 0 - of index: 71
Warning zone has been detected in data: 1 - of index: 72
Change has been detected in data: 1 - of index: 73
Change has been detected in data: 1 - of index: 131
Some advanced methods

- Handling concept drift using Instance Weighting
  - Weighting by:
    - Age
    - Relevance to the current concept.

- Handling Concept Drift using Ensemble Learning
  - The Very Fast Decision Tree (VFDT) Algorithm
  - Ensembles of Classifiers
  - Support Vector Machines
To summarize

- Concept Drift is Real, and is a challenge!
- Most of the algorithms for handling concept drift consider incremental (online) learning environments as opposed to batch learning.
  - Because real life data often needs to be processed in an online manner.
    - Data Streams -> incremental learning
    - Databases -> batch learning
- Acknowledging that models must/will evolve continuously is the key to understand and fix concept drift
References/Bibliography

- [https://www.dataversity.net/handling-concept-drift-in-interventional-machine-learning-systems/](https://www.dataversity.net/handling-concept-drift-in-interventional-machine-learning-systems/)
- [https://github.com/topics/concept-drift](https://github.com/topics/concept-drift)
- [https://github.com/ogozuacik/concept-drift-datasets-scikit-mutiflow](https://github.com/ogozuacik/concept-drift-datasets-scikit-mutiflow)
- [https://scikit-multiflow.github.io/scikit-multiflow/documentation.html#module-skmultiflow.drift_detection](https://scikit-multiflow.github.io/scikit-multiflow/documentation.html#module-skmultiflow.drift_detection)
- [https://www.cs.auckland.ac.nz/courses/compsci760s2c/lectures/YunSing/conceptdrift.pdf](https://www.cs.auckland.ac.nz/courses/compsci760s2c/lectures/YunSing/conceptdrift.pdf)