Learning from evolving data streams

Jacob Montiel

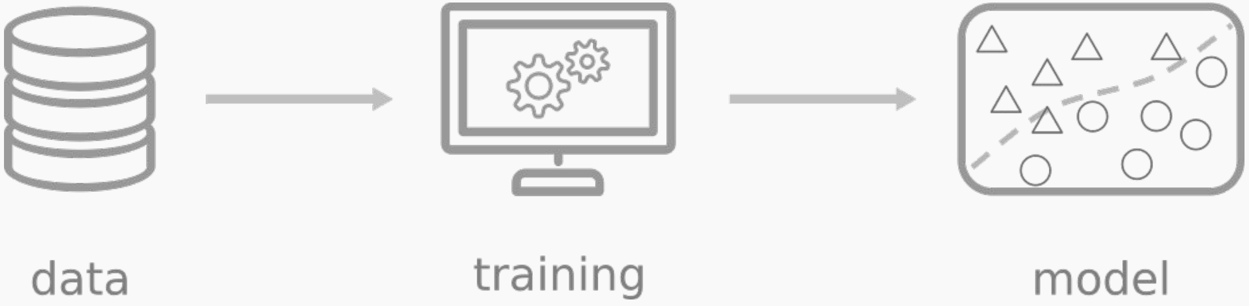
University of Waikato

KO TE TANGATA

SciPy2020

Standard machine learning

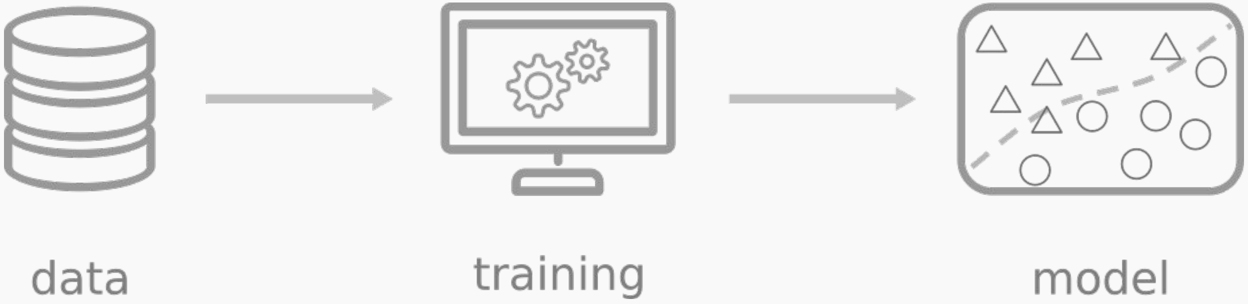
- Based on data batches (*batch learning*)
- State-of-the-art performance on multiple applications
- Batch learning pipeline:



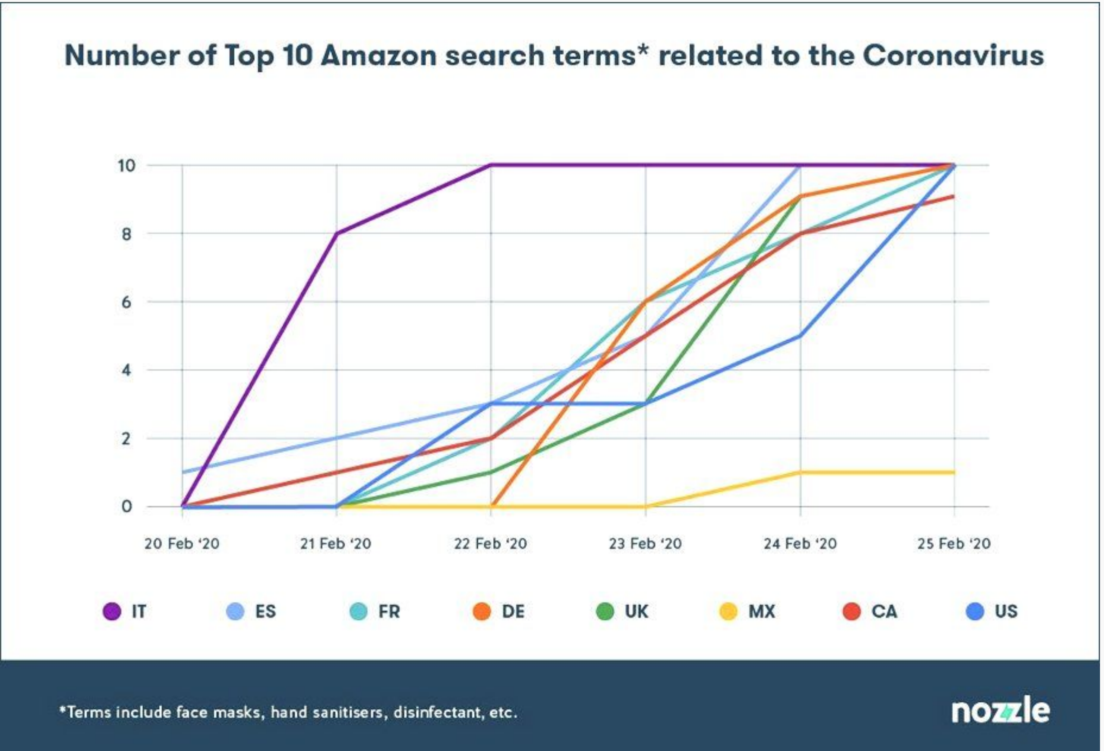
Standard machine learning

- Based on data batches (*batch learning*)
- What if data...
  a. is continuously generated (not available at once)
  b. changes over time

Keep pace with data!



Example: Supply chain



"It took less than a week at the end of February for the top 10 Amazon search terms in multiple countries to fill up with products related to covid-19."

"Our weird behavior during the pandemic is messing with AI models". Will Douglas Heaven. MIT Technology Review. May 11, 2020

Stream learning

- Data is assumed *infinite*
- Maintain models in an online fashion

- Unbounded training sets
- Incorporate data on the fly
- Resource-wise efficient
- Detect changes and adapt

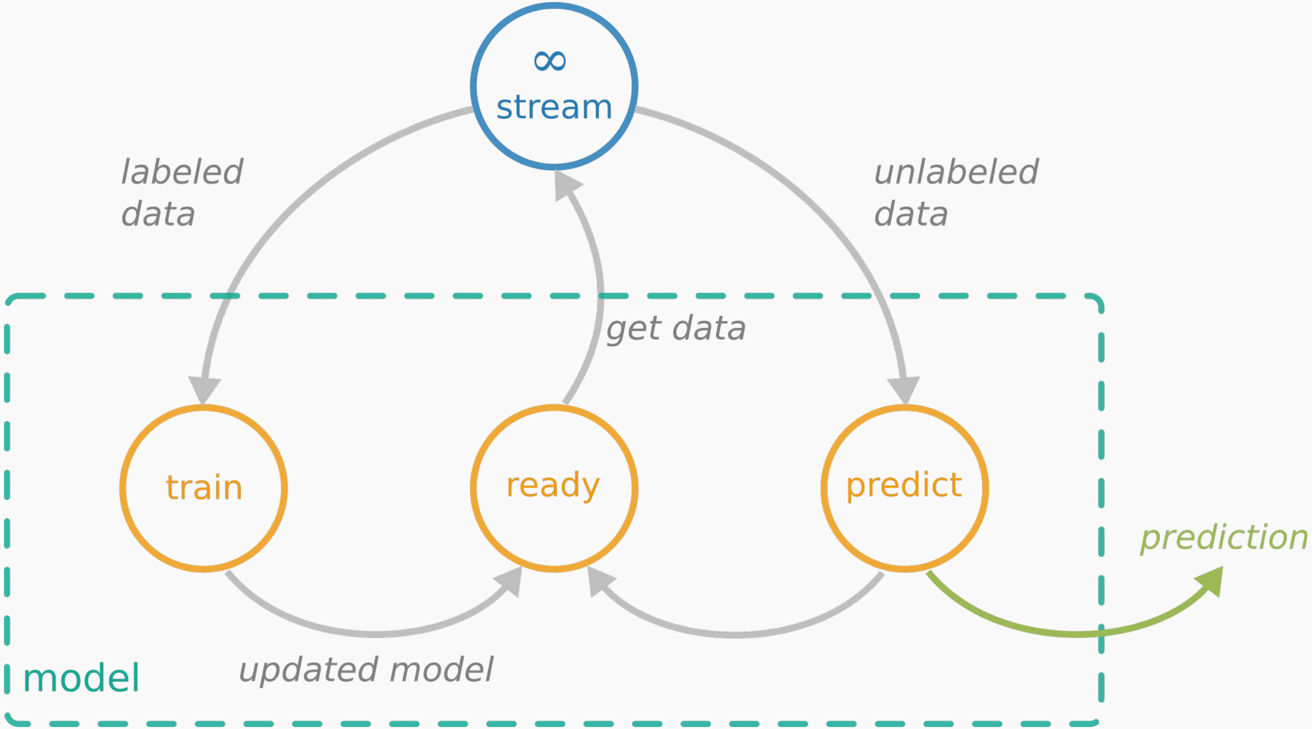
Image: OnAudience.com

Requirements

- Process one sample at a time, and inspect it only once
- Use a limited amount of memory
- Work in a limited amount of time
- Always ready to predict

Learning from data streams

Supervised learning



ⓘ One sample at a time

! Limited resources

! Predict at any time

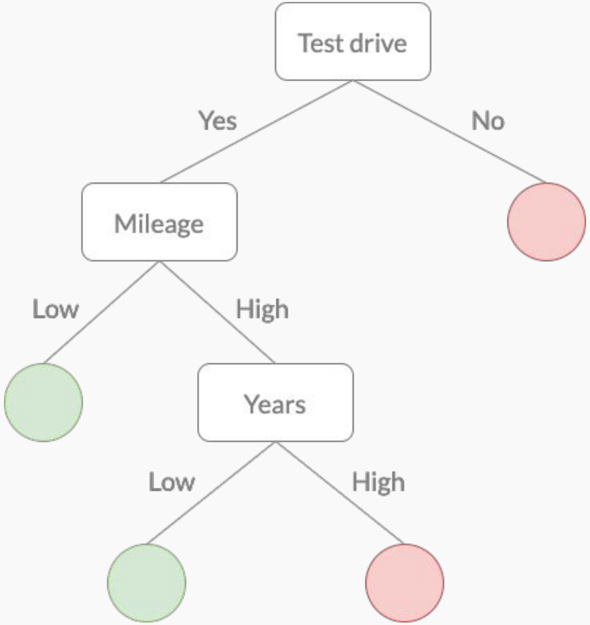
Decision Tree Classifier

- Popular **batch** method
    - Good performance + interpretability
- Greedy recursive induction
    - Sort all instances through tree
    - x_i = most discriminative attribute
    - New split node for x_i

      new branch for each value

      leaf node assigns class
    - Stop if no error or limit on #instances

Example: Buying an used car



Very Fast Decision Tree

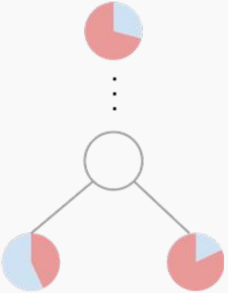
a.k.a. Hoeffding Tree

- Incrementally expand (split) nodes
  - A small sample can often be enough to choose a near optimal decision
  - Collect (sufficient) statistics
  - Estimate the merit of each attribute
  - Choose the sample size that allows to differentiate between the alternatives

Hoeffding bound

t_0

$t_1 = t_0 + \delta_1$



Pedro Domingos, Geoff Hulten, “Mining high-speed data streams”. In *KDD 2000*

Very Fast Decision Tree

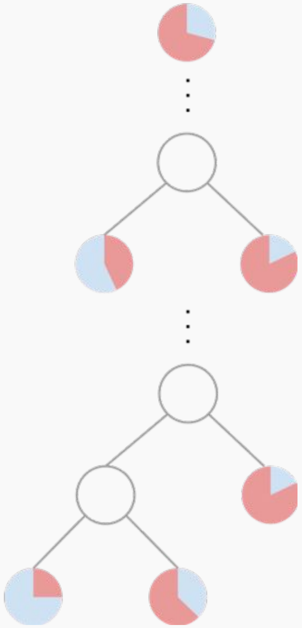
a.k.a. Hoeffding Tree

- The number of examples to expand a node depends only on the *Hoeffding bound*
  - error decreases as more data is observed
- Popular **stream** method
  - Low variance
  - Low overfitting
  - Asymptotically close to the batch model

$$t_0$$

$$t_1 = t_0 + \delta_1$$

$$t_2 = t_1 + \delta_2$$

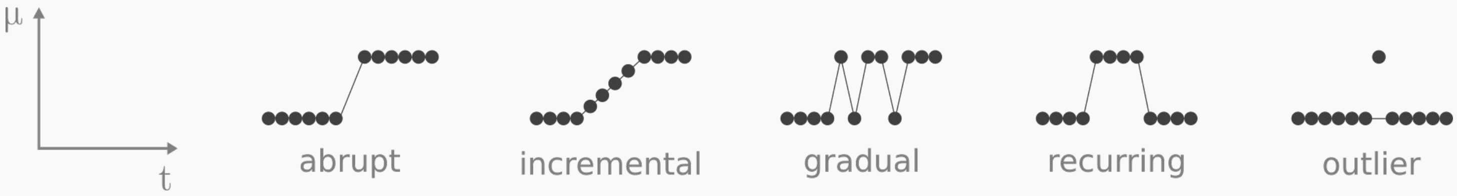


Pedro Domingos, Geoff Hulten, “Mining high-speed data streams”. In *KDD 2000*

Concept drift

In dynamic and non-stationary environments, the data distribution can change over time

- **Change detection:** Given an input sequence $\langle x_1, x_2, \dots, x_t, \dots \rangle$ raise an alarm signal at instant t if there is a distribution change
- **Application:** Detect changes in model performance

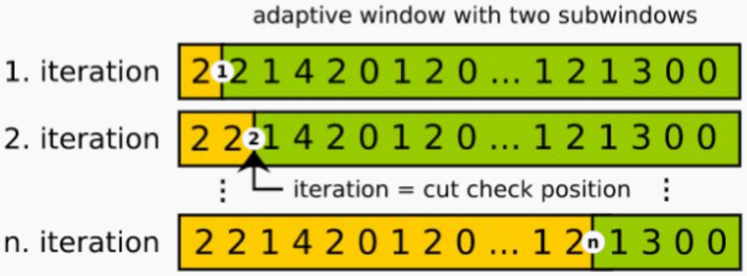


J. Gama et al., “A survey on concept drift adaptation.” In: ACM Computing Surveys 46.4 (2014)

ADWIN change detector

ADaptive WINdowing

- Adaptive window with two subwindows
  - Rise an alarm if subwindows exhibit “distinct enough” *averages*
  - Subwindows are *recomputed online* according to the rate of change
- Theoretical guarantees
  - Logarithmic memory and update time



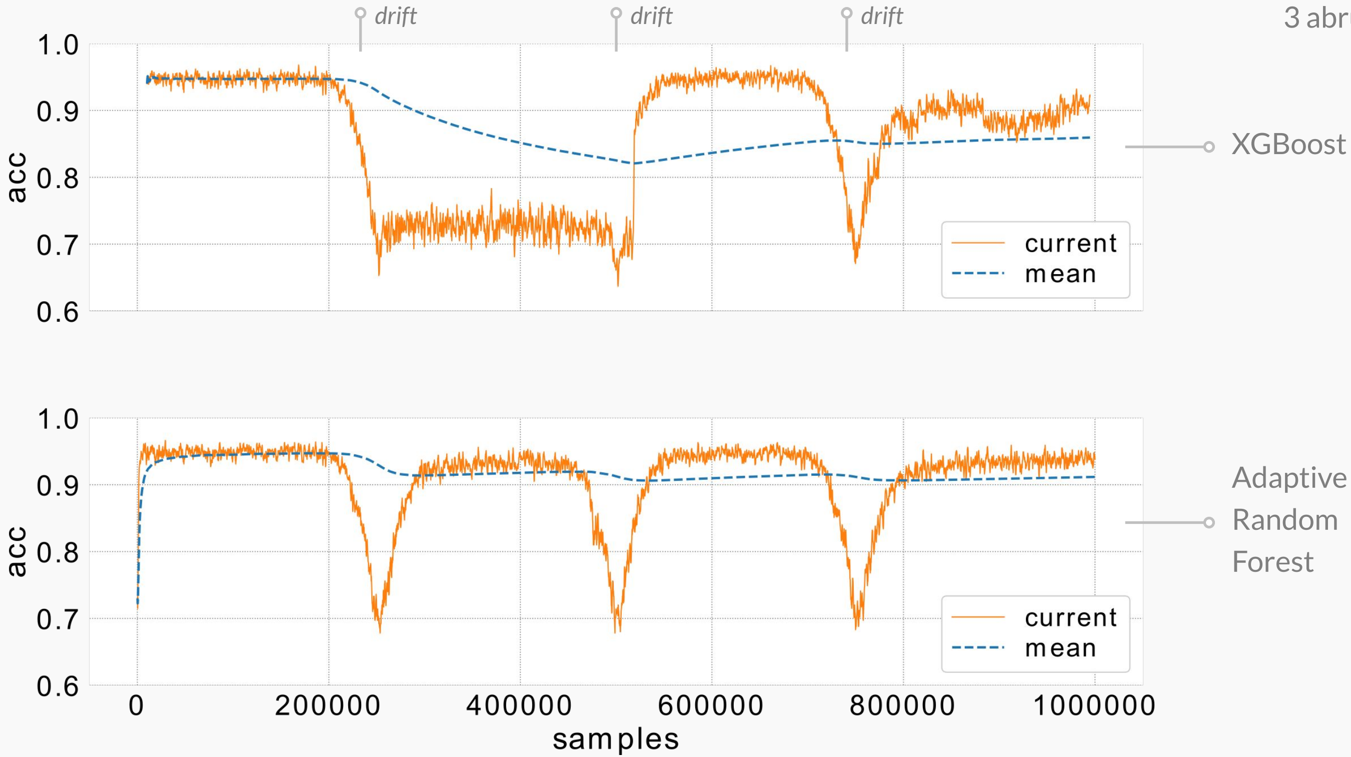
Iterations of the cut check procedure

A. Bifet, A., & R. Gavalda. "Learning from Time-Changing Data with Adaptive Windowing." SIAM ICDM, 2007

Learning from evolving data

batch **vs** stream

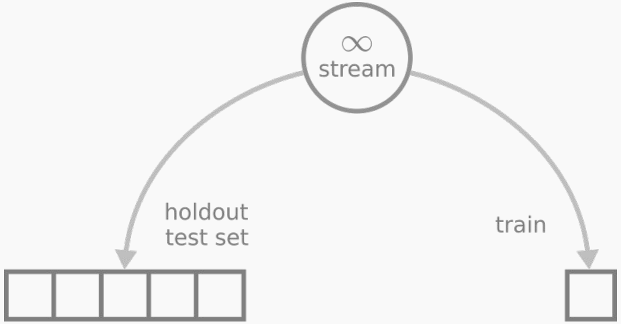
Agrawal generator data
3 abrupt drifts



Evaluation

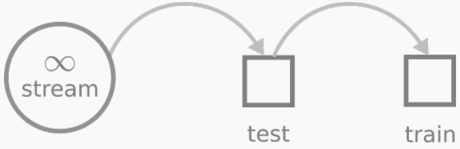
Holdout an independent test set

- Apply the current model to the test set, at regular time intervals
- *Unbiased* performance estimation
- Popular in *batch* and *stream* learning



Prequential

- Test *then* train each new instance
  - ○ Order matters!
  - ○ All data is used for training
- Performance is estimated on the sequence
- Popular in the *stream* setting



scikit-multiflow

A machine learning package for data streams in Python

- Easy to design and run experiments
- Easy to extend existing methods
- For users with any experience level
  - Low learning curve
  - Works in Jupyter Notebooks

- Contains
  - data generators
  - stream learning methods
  - change detectors
  - evaluators
  - and more

J. Montiel et al. “Scikit-Multiflow: A Multi-output Streaming Framework.” *JMLR*, 2018

Demo

Get scikit-multiflow

Multiple sources available

- `scikit-multiflow` works on Linux, macOS and Windows
- Recommended:
  - conda-forge

    ```
    $ conda install -c conda-forge scikit-multiflow
    ```
  - PYPI

    ```
    $ pip install scikit-multiflow
    ```
  - GitHub (latest development version)

    https://github.com/scikit-multiflow/scikit-multiflow

How can I contribute?

- We welcome contributions from the community
  - scikit-multiflow
  - gitter.im/scikit-multiflow/community
- We have a pool of projects in the following areas:
  - Classification
  - Regression
  - Clustering
  - Anomaly Detection
  - ...
- Or bring your own project/idea

Image: pikisuperstar / Freepik

Takeaways from this talk

✓ Stream learning is an alternative to standard (batch) learning

- data is continuously generated
- data is non-stationary, it evolves! (concept drift)

✓ `scikit-multiflow`

- machine learning for data streams in Python
- easy to design and run experiments
- easy to extend

Thank you

jacob.montiel [at] *waikato.ac.nz*