A brief introduction to online machine learning

Hong Kong Machine Learning - June 10th 2020

By Max Halford

Outline

What is online machine learning?

Why is it desirable?

Which models are available?

Live demo

Questions

Online machine learning in a nutshell

- An algorithm is “online” when it can **learn with one element at a time**
- For ML, this refers to models that can be updated with a single sample
- **Most ML is batch**; all the data is assumed to be available
- Many synonyms: incremental, streaming, sequential, lifelong
- Online ML has many benefits but isn’t very popular (**paradox**)
- Online ML requires rethinking the whole data science workflow in a new way

Thinking online

- Many algorithms have an online counterpart:
  - Mean, variance, skew, kurtosis → Welford's algorithm
  - Gradient descent → stochastic gradient descent
  - Fourier transform → SDFT algorithm
  - Histograms → see this paper
- Most of the time, these algorithms are actually easier to understand…
- … and even easier to implement

You might already be doing it ...

- Deep learning essentially boils down to stochastic gradient descent
- Bayesian inference is about updating a current model with new evidence
- (Dynamic) Bayesian networks can be updated online
- Most recommendation systems work online under the hood
- Tech giants are doing it all the time

… you just don’t know it!

- Online machine learning is not as popular as batch machine learning
- Before the Internet era, there was no need to process streaming data
- Universities don’t teach it very much, vicious circle
- Kaggle mentality: see every problem as a train and test scenario

Online machine learning is good for you

- You can update your model with new data
- You don’t have to retrain your model from scratch
- Your model is always up-to-date
- Your model can (usually) adapt to concept drift
- You can process huge datasets with a laptop
- Local model development is more realistic

Creme

- Library for online machine learning
- Written in Python (`pip install creme`)
- General-purpose (like scikit-learn)
- Started in January 2019
- Interest is growing at a steady pace
- github.com/creme-ml/creme

Usage example

```
from creme import compose
from creme import datasets
from creme import linear_model
from creme import metrics
from creme import preprocessing

X_y = datasets.Phishing()  # this doesn't put anything in memory

model = compose.Pipeline(
    preprocessing.StandardScaler(),
    linear_model.LogisticRegression()
)

metric = metrics.ROCAUC()

for x, y in X_y:
    y_pred = model.predict_proba_one(x)  # make a prediction
    metric = metric.update(y, y_pred)    # update the metric
    model = model.fit_one(x, y)          # make the model learn
```

Data representation

- Features are stored in dictionaries (called “dicts” in Python)
- Dicts are to lists what pandas DataFrames are to numpy arrays
- Dicts can naturally represent **sparse data**
- Dicts allow **naming features**
- Dicts are native to Python → **no overhead** from specialised data structures
- Dicts are **JSON-friendly**

Flask web routes example

```python
@app.route('/predict', methods=['GET'])
def predict():

    payload = flask.request.json
    x = payload['features']

    model = db.load_model()
    y_pred = model.predict_proba_one(x)

    return y_pred, 200
```

```python
@app.route('/learn', methods=['POST'])
def learn():

    payload = flask.request.json
    x = payload['features']
    y = payload['target']

    model = db.load_model()
    model.fit_one(x, y)

    return {}, 201
```

A refreshingly simple philosophy

- We favour **simplicity**: what works on your laptop should work everywhere
- Data preprocessing and cleaning should be doable online
- Pipelines are first-class citizens and we encourage using them
- Model deployment and maintenance shouldn’t be a headache

creme has got you covered

- Anomaly detection
- Linear models
- Imbalanced learning
- Recommendation systems
- Decision trees
- Streaming utilities
- Feature extraction
- Feature selection
- Ensembles
- Model selection
- Latent Dirichlet Allocation
- Time series
- Clustering
- Performance metrics
- Naïve Bayes
- Nearest neighbours

Decision trees

- By design, decision trees require scanning a dataset multiple times
- It's possible to start with a single leaf and create branches on the fly
- Two main variants:
  - Hoeffding trees → store sufficient statistics, decide to split every so often
  - Mondrian trees → build deep trees completely at random
- I'm working on a Cython implementation of Mondrian trees, with a twist

More information in these slides

Bayesian inference

- We begin with a **prior** model
- When new **evidence** arrives, we update out model with it
- Mixing a prior with evidence produces a **posterior distribution**
- **The posterior becomes the new prior**
- Bayesian inference provides a sound foundation to do this

Bayesian inference

likelihood

prior distribution

predictive distribution

posterior distribution

$$p(y_0|x_0) \propto p(y_0|x_0,\theta_0)p(\theta_0)$$

$$p(\theta_1|\theta_0,x_0,y_0) \propto p(x_0,y_0|\theta_0)p(\theta_0)$$

$$p(y_1|x_1) \propto p(y_1|x_1,\theta_1)\underbrace{p(\theta_1|\theta_0,x_0,y_0)}_{p(\theta_1)}$$

$$p(\theta_2|\theta_1,x_1,y_1) \propto p(y_1|x_1,\theta_1)\underbrace{p(y_0|x_0,\theta_0)p(\theta_0)}_{p(\theta_1)}$$

More details in this blog post

Disclaimer

- Online machine learning isn’t a one-size-fits-all solution
- Batch learning is perfectly adequate for many problems
- **It makes sense** when you are dealing with streaming data
- Both approaches are **complementary**; neither dominates the other
- **Use the right tool for the job!**

Speed considerations

- Many libraries implement gradient descent, which allows comparing them
- creme shines when samples arrive one by one:
  - 20x faster than scikit-learn
  - 50x faster than PyTorch
  - 180x faster than Tensorflow
- The figures are similar for learning and predicting

Click here to access the complete benchmarks

What about big data?

- Learning with one sample at a time incurs overhead and is **not efficient**
- To go big, we have to make use of **vectorisation**
- Therefore, we need to support **mini-batches**
- The next version of creme will support mini-batches **for some models**
- We will also integrate more tightly with dask and vaex
- However, pure online learning will remain our core philosophy

Exciting prospects, stay tuned!

Time for a demo!

https://git.io/JfM9W

maxhalford.github.io

github.com/MaxHalford

maxhalford25@gmail.com