Our solution to the IDAO 2020 qualifiers

Max Halford
Raphaël Sourty
Robin Vaysse

Webinar on Data Analysis for Satellite Tracking
Sunday 12th April, 2020

Our team

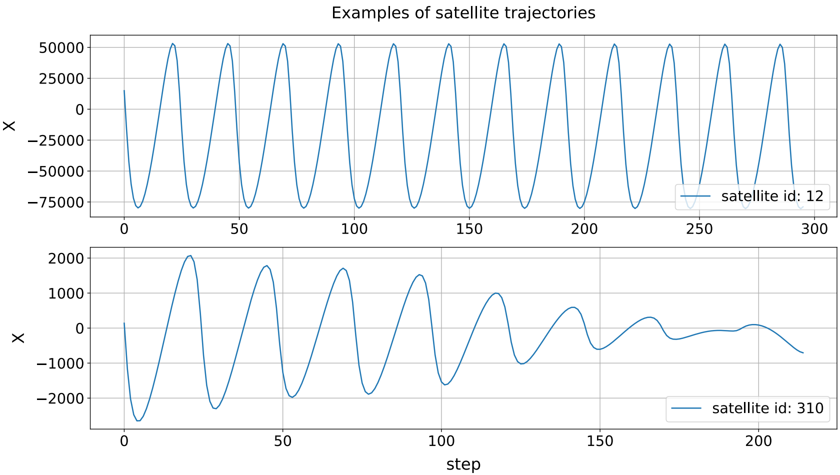
- Max Halford, 3rd year PhD student at IMT/IRIT
- Raphaël Sourty, 1st year PhD student at IRIT
- Robin Vaysse, 1st year PhD student at IRIT

We like competitive data science!

Context

- Satellite position forecasting
- Two tracks with separate leaderboards:
  1. Make the most accurate predictions possible
  2. Make accurate predictions with two constraints:
     2.1 Take less than 60 seconds
     2.2 Keep peak RAM usage under 500MB

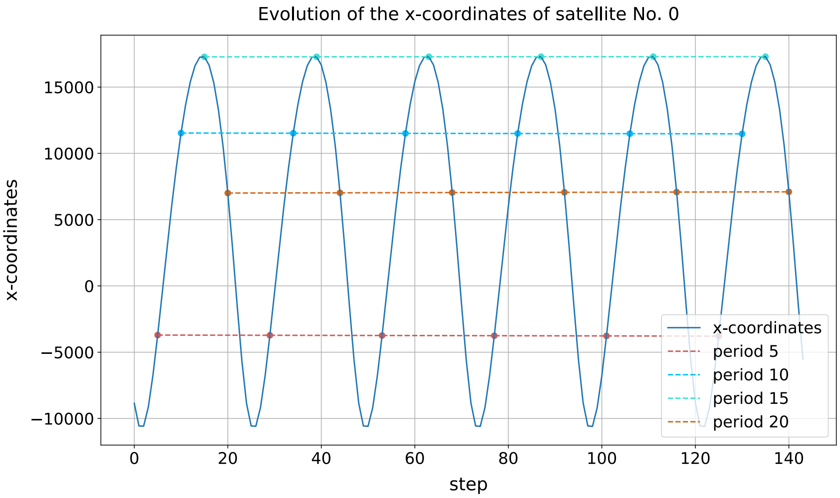
The data


Examples of satellite trajectories
50000
25000
0
−25000
−50000
−75000
x
0
50
100
150
200
250
300
satellite id: 12
2000
1000
0
−1000
−2000
x
0
50
100
150
200
step
satellite id: 310

Our solution in a nutshell

- We train one model per satellite and per coordinate ($300 \times 6 = 1800$ models)
- Each model is an autoregressive (AR) process of order $p = 48$
- In other words, we train a linear regression to predict y_{n+1} from $\{y_{n-48}, \dots, y_n\}$, that's all!
- To predict several steps ahead, we use the prediction at step $n + 1$ as a feature at step $n + 2$
- We validate locally on the last 40% of the data
- Our approach is simple enough to be used for both tracks without modifications

Starting simple



Auto-regression

- Using past target values makes sense because the data is very periodic
- For every satellite and coordinate, we build a vector of features
- Each vector contains the p past target values
- We obtain n feature vectors and n targets
- For forecasting into the future, we:
  1. Make a prediction for the next time step
  2. Append the prediction to the feature vector
  3. Remove the oldest value from the vector
  4. Repeat from step 1.
- Flexible framework:
  - Any regression model can be plugged in
  - Any feature can be added, provided it can be computed online

Dealing with speed

- AR models are slow at inference because of their sequential nature
- In `scikit-learn`, calling `.predict(X)` many times incurs a large overhead
- We “stripped” the `scikit-learn` classes we used to their bare minimum by overriding some of their methods

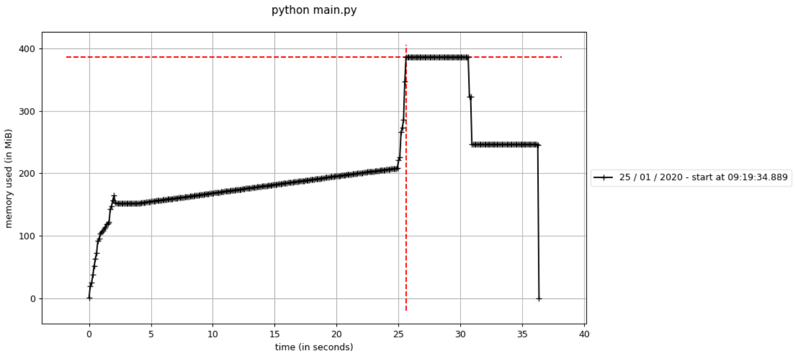
Overriding `scikit-learn`'s linear regression

```python
class StandardScaler(preprocessing.StandardScaler):
    """Barebones implementation with less overhead than sklearn."""

    def transform(self, X):
        return (X - self.mean_) / self.var_ ** .5

class LinearRegression(linear_model.LinearRegression):
    """Barebones implementation with less overhead than sklearn."""

    def predict(self, X):
        return np.dot(X, self.coef_) + self.intercept_
```

More information here. We've also learned about sklearn-onnx.

Dealing with memory usage

We used a Python package called memory_profiler to measure the memory usage of our script.



What didn't work

- Gaussian processes with sinusoidal kernels gave good training results, but fared poorly on the test set
- The N-BEATS[1] model fits perfectly to the training data but diverges in auto-regressive mode
- We got no improvement by training a multi-output linear regression to try capturing coordinate dependencies

[1] Boris N. Oreshkin et al. "N-BEATS: Neural basis expansion analysis for interpretable time series forecasting". In: *CoRR* abs/1905.10437 (2019). arXiv: 1905.10437. URL: http://arxiv.org/abs/1905.10437.

Production considerations

- Our model is essentially a linear regression
- Linear regression can be trained with stochastic gradient descent (SGD)
- SGD requires one sample at a time, and is thus enables online algorithm
- Online learning allows learning from a stream of data
- Predicting satellite positions is inherently a streaming problem, therefore models that can be trained online should be preferred

Shameless publicity: check out creme and chantilly for online learning

Our advice for competitive data science

- "*Keep it simple, stupid*" (KISS principle)
- Always start by setting up a local validation benchmark
- When your model improves, save your work (`git` is your friend)
- Doubt everything you do
- Don't be scared to try stuff, but don't tunnel vision

Code can be found on GitHub

Thank you for listening!