AI Seminar – Drone

Jan Pacovský

September 21, 2017

1 Introduction

Our interest is to explore the benefits of preprocessed input on learning algorithms. To do that we have to firstly preprocess data and then learn on the corresponding raw and enhanced data. First section of this article describes format, meaning and some basic properties of the raw input data. Then we focus on how was the data put together from different sources and finally we show how was the date enhanced.

2 Enhacing the input data

2.1 Description and analysis of input tables

Data collection for each flight is composed of three table types – navdata, commands and flow. For most of the flights include only the first two tables. The extra flow table was obtained from the camera attached to the drone.

2.1.1 Navdata table

Navdata table usually contains between 10k and 110k rows for longer flights and about 2k for short ones. For the flight with videos one file contains about 16000 rows and the other only 148 rows. That indicates error in recording, but we did not discard this data because this was the only flight with video input not ending with crash. Columns in navdata table in their order are:

Recorder Time, State, Battery level, Magnetometer x, Magnetometer y, Magnetometer z, Pressure, Temperature, Wind speed, Wind angle, Wind compensation: pitch, Wind compensation: roll, Pitch (Rotation around y), Roll (Rotation around x), Yaw (Rotation around z), Altitude, Velocity in x, Velocity in y, Velocity in z, Acceleration in x, Acceleration in y, Acceleration in z, Motor 1 power, Motor 2 power, Motor 3 power, Motor 4 power, Board Time.

2.1.2 Commands table

This table is the only one that contains commands transmitted to the drone. Usually this file contains around 1000 commands, which however due to the repetitiveness (each command is sent over and over again to let the drone know that it is still in the range) can be reduced to low tens of commands. As an example we can take file `camera_auto01` it contains 597 rows before reduction and only 7 afterward [Table 1].

Both tables - original as well as the reduced contains five columns: *Timestamp, Left-right tilt, Front-back tilt, Vertical speed, Angular speed.* The first column after the reduction does not represent a timestamp as before but the total time of the command. Getting the timestamp would be just summing the length of the events and adding time of begging of the flight.
In each row only one of the columns different from timestamp may contain value different from zero. Command that contains only zeros is a special one. It commands the drone cease in current movement and hover over the reached position.

Table 1: Reduced command list

0.0	0.0	0.0	0.0	0.0
7735.0	0.0	-0.25	0.0	0.0
5832.0	0.0	0.0	0.0	0.0
3087.0	0.0	0.0	0.25	0.0
5386.0	0.0	0.0	0.0	0.0
824.0	0.0	0.25	0.0	0.0
5362.0	0.0	0.0	0.0	0.0

2.1.3 Flow table

Flow table contains information extracted using visual flow or object tracking methods and the number of rows corresponds one to one to the number of frames in the video. It contains timestamp and 18 columns indicating the orientation of the main direction of movement in certain sector of the image.
We have two flights with including the camera input containing around 1500 rows for both of the flights. However not all the data was relevant due to the delays of recording start/end on the beginning and the end of the video. For the video that landed well the percentage of relevant data was 83%. For the video ending with crash it was only 49%. Only relevant, hand picked part of this flow table type was merged with another table containing rest of the navigation data (navdata).

2.2 Merging of the Tables

The merging of the tables is done using transformed timestamps as the join variable. There were essentially 2 or 3 problems first is the alignment of the data caused by different starting times, end of one of the files before the relevant part in the other and lag of the system which was discussed in the original article [1] in the section "Merging of Tables". We choose another approach in dealing with a delay of the system, instead of shift of the constant number of rows (as in original article) we use shift by certain amount of time. This approach produces different order of rows caused by jitter of the system. With learning we are searching for optimal value of this shift.
Side-note: *The navdata file contains two different timestamps one corresponds to the on-board time and the other to the computer. After examination of the data the on-board time seemed better. The other timestamp is probably assigned by the computer at the time of arrival of wireless transmission.*
Another side-note: *Time was normalized to milliseconds for each of the file and*

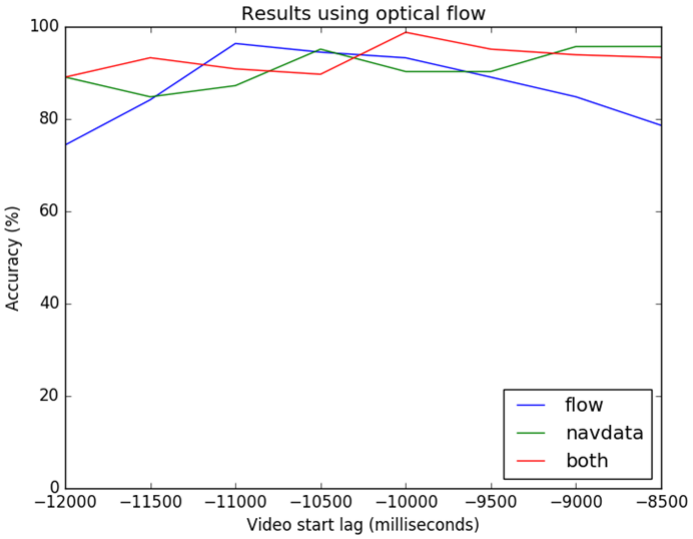

Figure 1: Searching for a right fit for Flow table data, using LSTM accuracy as a referee – video indicated 10s mark, which was optimum reached by LSTM on whole data.

resulting merged file has a begging at the zero time.
When merging the Flow table with rest of the data we had to face a different starting times of the video input and drone, from the audio and video footage we know roughly when the drone started its engines. This was than manually aligned and the learning algorithm itself decides the best time shift on a limited interval. The results of LSTM, 5-fold cross-validation and with given lag are shown in the Figure 1. The merged file is than created with the shift found by the learning, this file is than used for other methods as well.

2.3 Preprocessing

During the work on preprocessing we found out that the drone does some preprocessing on its own, so the data that we obtain is not a raw data. An example how we can see this is when drone is tilting we do not see any difference in between engine speeds. Another example is the Altitude column in Navdata - the drone combines two sensors and present us only with this value.

The main task of preprocessing is to get more data and get cleaner data. What we want is to help the learning methods distinguish better what is important and what is not.

2.3.1 Sensor fusion and changes to used columns

The goal of sensor fusion is to extract additional (hidden) information, another view may be that it is a form of compression that leaves the important characteristics only (if we remove original columns) hence making the search space smaller.

We were able to fuse data to two resulting columns. Readings from the magnetometer x and y were used to compute compass direction to the north. This was then normalized to fix initial position to 0 degrees. While all 4 motor power columns on our data had always the same values we have replaced them with their sum. We also identified useless columns for our task: Columns containing word *"Wind"* and columns *"Battery level, Temperature"* were removed to prevent overfitting by reducing the search space. Other columns were scaled to obtain zero mean and unit variance.

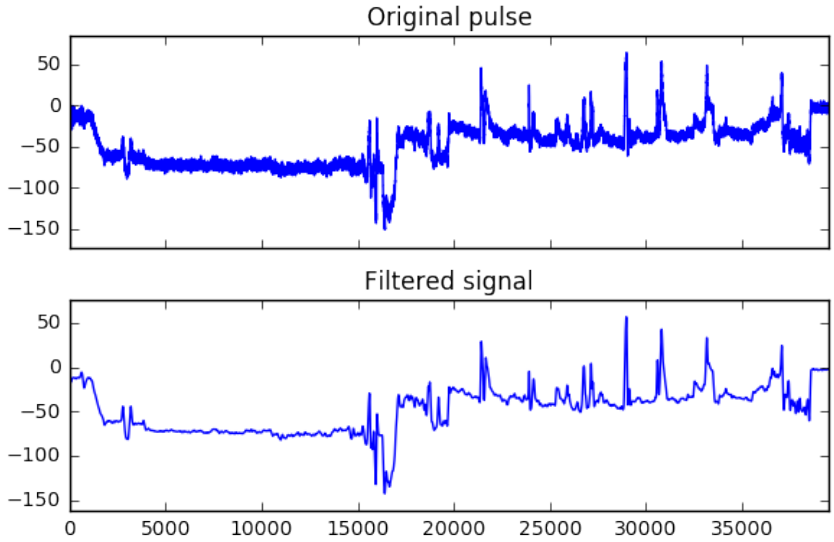

Figure 2: Denosing example

2.3.2 Denoising

When the drone operates in the air it vibrates a lot. Those vibrations are then seen as never ending spikes in the graph of accelerometers, hiding the overall behavior of the drone. Our drone operated at rate up to 200 readings per second.

The flight speeds in comparison to 200Hz are not high, specially the pitch rotation which takes couple of seconds. This allowed us to average multiple reading into only one. Different approach is not to change size of the input by using convolutions. Starting with median, we moved to flowing averages and to more complex ones as well.

By a small margin the most satisfactory results were obtained using convolution on Hann window. Convolution with Hann filter gives most weight to the data in the middle and gradually less and less as it approaches to the border of the window (Gaussian like curve). Since we had many reading per second we could make the window size big. We experimented with different lengths of the window but interestingly the results were almost same for lengths between 10 and 100 (window duration from 0.05s to 0.5s).

In fact the difference was negligible between any two methods that used windows of size at least 10. We think this is due to the small variance in spikes.

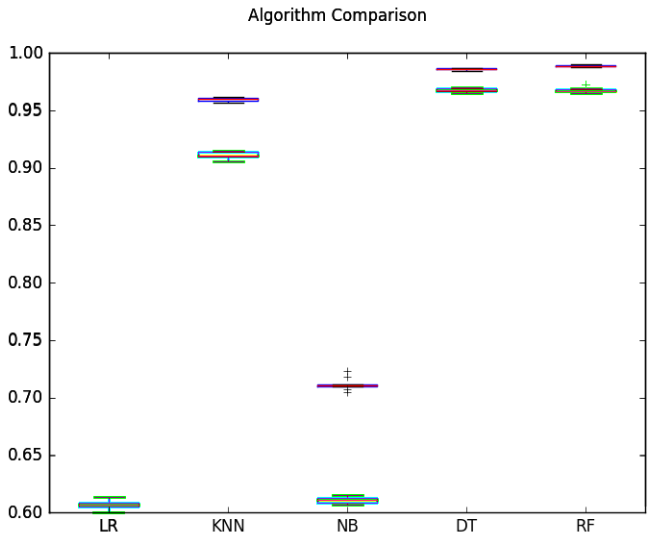

Figure 3: Effect of convolution on classification. Green boxes represent result before and violet after applying Hann filtering. Abbreviations stands for: Logistic regression, k nearest neighbors, Naive Bayes, Decision Trees and Random Forest. Only method that got worse is the Logistic regression with accuracy around 0.55.

We expected that the signal would be more noisy. But as seen in the Figure 2 the signal noise is small and does not blur much of the signal. We can clearly see that the denoising makes the line more stable and this is enough for the learning. Test were done using 10-fold cross validation on different machine learning algorithms. We used Logistic regression, K-nearest neighbors, Naive Bayes, Decision tree, Random forest on input consisting on pitch, roll and yaw and on all of the features as discussed earlier.

3 Results

Getting the results for the question if and how much has the preprocessing helped was straight forward. We used methods that were used in task for supervised learning of actions. Doing the classification on original data versus the enhanced data yielded result showing us that the preprocessing in form of convolution helps for most of the methods that were used. As seen in the Figure 3 the preprocessing helped slightly all but Linear regression methods to reach better performance.

References

[1] Roman Barták and Marta Vomlelová. Using machine learning to identify activities of a flying drone from sensor readings. In Vasile Rus and Zdravko Markov, editors, *Proceedings of the Thirtieth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2017, Marco Island, Florida, USA, May 22-24, 2017.*, pages 436–441. AAAI Press, 2017.