Supervised learning of drone actions

Seminar on AI, summer 2017

Monika Švaralová

1 Introduction

In this project I tried classifying actions using data from a drone. First, classifying single rows (sensor readings) and comparing results of various machine learning algorithms. Second, classifying sequences of sensor readings using LSTM model and also with addition of optical flow from the camera. Finally, I tried classifying more complex activities from whole flights.

2 Input data

As an input, I received sensor readings (navdata) from the drone, matched with the most recent command according to timestamps. There were 4 command columns in the data, each representing a different command type: $LeftRight, FrontBack, VerticalSpeed, AngularSpeed$. They could have either positive or negative values, and at max one of them was nonzero at each time. This results in 8 different possible commands + command "do nothing", represented by zeros in all of the 4 variables.

The main dataset used in the experiments was the longest flight: `flight_auto04`, which consisted of $\sim$ 100,000 sensor readings. I performed an additional preprocessing of the sensor data: scaling each input variable to zero mean and unit variance.

3 Classification of command types from single rows

In the first experiment, I compared the accuracy of standard machine learning algorithms (Logistic regression, K-nearest neighbors, Naive Bayes, Decision tree, Random forest) using 10-fold cross-validation on data from `flight_auto04`. I used either all features from a row of navdata as an input, or only rotation (pitch, roll, yaw). The target output of the classification algorithms was the command that the drone is currently performing. I filtered out zero commands, in order to classify into 8 classes which were more equally distributed. This resulted in a dataset of 60,000 samples, where the most common class had 23% of samples.

Results

Classifier	**Average accuracy**
Logistic regression	54.5%
K-nearest neighbors	96.1%
Naive Bayes	62.4%
Decision tree	98.4%
Random forest	98.5%

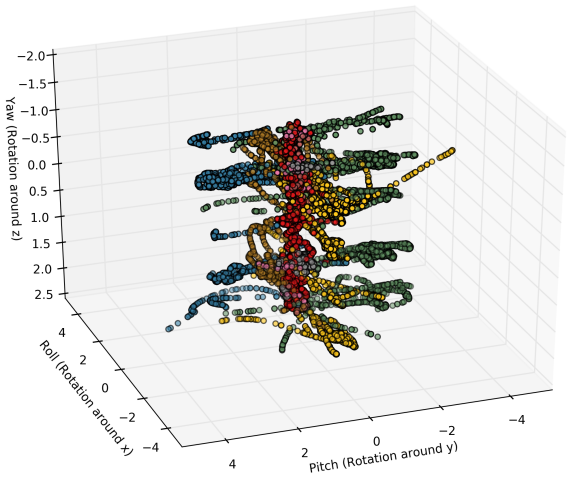


Figure 1: Visualization of rotation (pitch, roll, yaw). Each point represents a sensor reading from `flight_auto04`, color of the point corresponds to its current command.

The highest accuracy, 98.5%, was achieved by Random forest classifier, with only rotation (pitch, roll, yaw) as input. However, this was caused by the classifier over-fitting this particular flight despite using cross-validation. In Fig. 1, we can see that the flight creates continuous trajectories of rotation when a particular command is being executed. Therefore, it is possible for a classifier to learn that these correspond to a command, while it is only true in this particular flight.

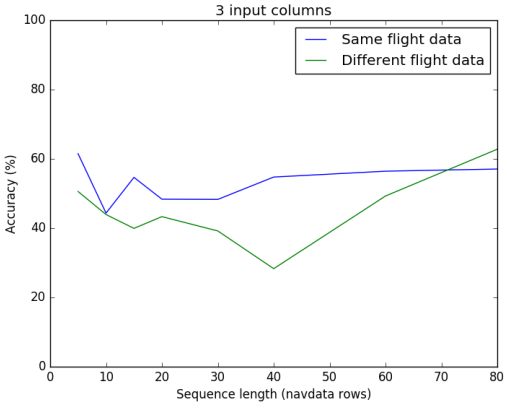
When testing the classifiers on a dataset from a different flight (`flight_auto01`), the accuracy is only 26.8% with Random Forest, when using rotation as the input, and 23.6% using all navdata as the input. Single sensor readings are not enough to identify the command, therefore we move on to classifying sequences of sensor readings.

4 Classification of sequences using LSTM

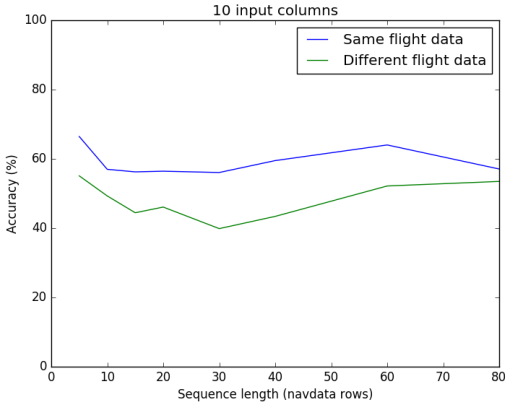
LSTM (Long short-term memory) is a type of recurrent neural network which has been successful in various applications of classifying or processing time series. As an input to the LSTM, I split the data into non-overlapping sequences of the same length. I tried various lengths of sequences, from 5 to 80 rows and compared results for each length. As the input variables, I tried 3 options: only rotation, (rotation, velocity, acceleration, altitude), or all 25 columns. The target output for each sequence of sensor readings was the most common command that occurs during that sequence. After removing zero commands, there were 8 remaining command types as target classes. LSTM has several parameters, which should be tuned, such as number of units, epochs and batch size. After some experimentation, I used 32 memory units and trained only for 3 epochs to prevent overfitting.

Results

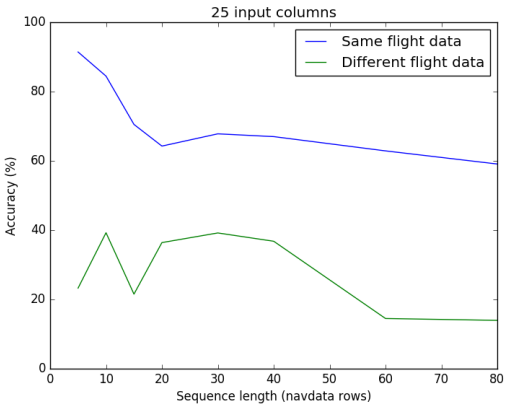
As test data, I used either a different subset of data from the same flight, or data from a different flight, to identify whether the model generalizes well. The best accuracy was around 90% on test data from the same flight (when using all input variables), and around 60% on test data from another flight when using only rotation as an input (Fig. 2). These results were obtained only from a single run, without cross-validation, which caused larger variance in the accuracy.



(a) Input variables: rotation



(b) Input variables: rotation, altitude, velocity, acceleration



(c) All variables – accuracy on test data from the same flight is significantly higher than on data from a different flight, meaning that it's overfitting the flight

Figure 2: Test accuracy of LSTM on data from the same and different flight, according to length of the sequences into which the data was split

5 Enriching the data with optical flow

In this experiment, I used the dataset `camera_auto01`, where I received sensor readings aligned with optical flow from the camera. Since the video and navdata had different start times, we had to identify the lag between them. I tried different lag values and evaluated average classification accuracy, expecting correctly aligned data to achieve the highest accuracy. I evaluated the accuracy using LSTM with 5-fold cross-validation.

As input features, I tried using either optical flow, navdata, or both. I split the data into sequences of 20 rows, assigning the most frequent target command as the output variable. There were only 3 distinct nonzero commands in this dataset (front tilt, vertical speed, back tilt), each of them in a single 5 second interval, therefore the classes were almost evenly distributed. In this way, I obtained 160 sequences of length 20.

Results

Maximum accuracy (98.8%) was achieved when using both navdata and flow, and video is aligned correctly around -10000ms (Fig. 3).

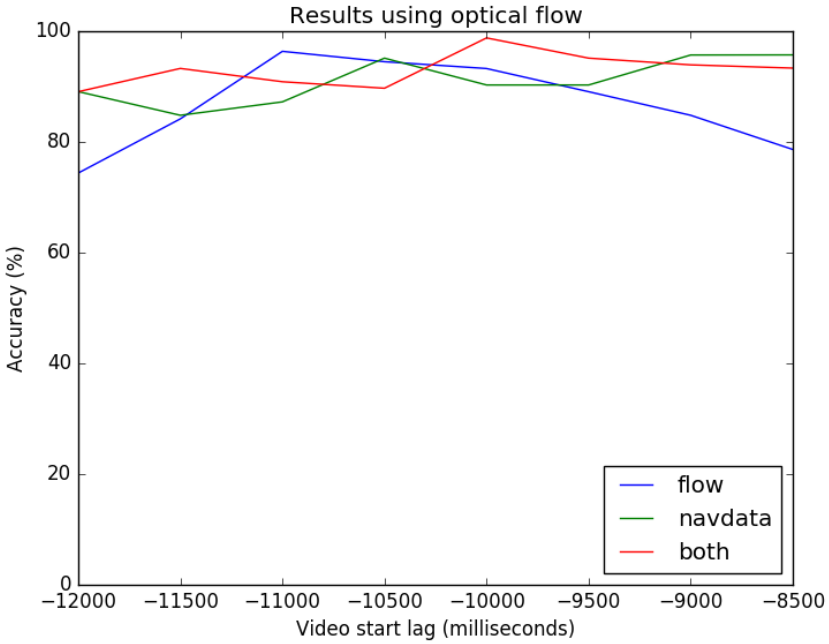


Figure 3: Average cross-validation accuracy, according to the lag between optical flow data from the video and sensor data from the drone. Different input variables used (flow, navdata or both).

6 Classification of activities

The final experiment consisted of classifying activities in the `short_flights` dataset. The dataset consisted of 35 short flights. In each of them a complex activity was performed, such as flying a circle, square etc. There were 7 types of these activities. The classes were almost evenly distributed, the most common one had 28% of samples.

Each flight was a single input sample, which means that the input sequences were long (the longest flight had 6500 rows), and the number of samples was small. We tried to preprocess the sequences to obtain shorter sequences with less noise that would still capture the activity. LSTM with 5-fold cross-validation was used.

Results

First, I used a smoothed sequence of navdata and commands from each whole flight as a sample. Simple smoothing was performed by averaging every 50 rows into 1 value. The resulting cross-validation accuracy was 40%. Another preprocessing method that we tried was to use a sequence of distinct commands along with duration of each command. This resulted in average accuracy of 37%. The results were slightly better than baseline.