Machine Learning Engineer Nanodegree

Capstone Project

Juan José Madrigal Martínez
February 5, 2017

Madrid, Spain
(+0034) 600 86 32 48
juanjomadrigal326@gmail.com
LinkedIn GitHub Kaggle Udacity

I. Definition

Project Overview

This project aims at building a video analysis system for surveillance purposes. It basically finds and indexes the movement events filmed by a fixed camera.

Video surveillance has become a major and widely used tool for multiple issues [1] and is supported by many companies [2] [3]. But the huge ammount of information which is usually dealt with has led to the need to use Machine Learning techniques to extract patterns, predictions and other refined information. This approach is being implemented [4] and there is much work for Machine Learning engineers to do in this field.

An efficient implementation of Machine Learning techniques to video surveillance would prevent users from dealing with a sequential (and manual) search through the (perhaps many hours long) video source, which is rather inefficient, boring and error prone, thus providing a major tool for a wide range of purposes.

[1] Wikipedia - Surveillance / Cameras

[2] VideoSurveillance.com

[3] Tyco

[4] Briefcam

Problem Statement

To apply unsupervised Machine Learning clustering algorithms to detect and quantify the movement events filmed by a fixed camera. The algorithms are to be applied to a tridimensional binary array obtained from the original video after some careful video-preprocessing (see below). Further refinements may also be tackled, such as finding time-parametric curves accurately fitting each movement event.

Metrics

The direct metric to tune the main parameters of DBScan (`eps` and `min_samples`) will be the Sum of Square Errors according to the expected value of clusters, which is known on beforehand for each of the training videos.

Once the algorithm is accurately tuned, the metric(s) used to estimate the quality of our clustering will comprise

- Silhouette Coefficient
- Calinski-Harabaz Index

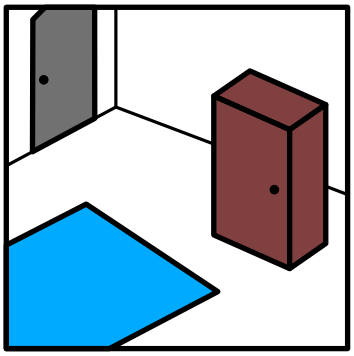
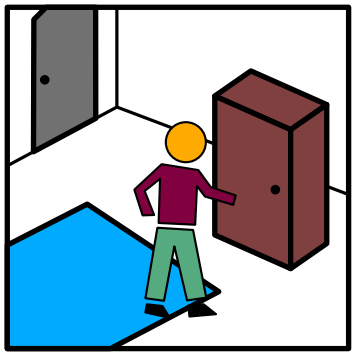
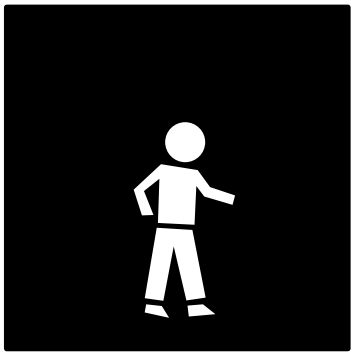
because these do not require knowledge of the ground truth classes.

II. ANALYSIS

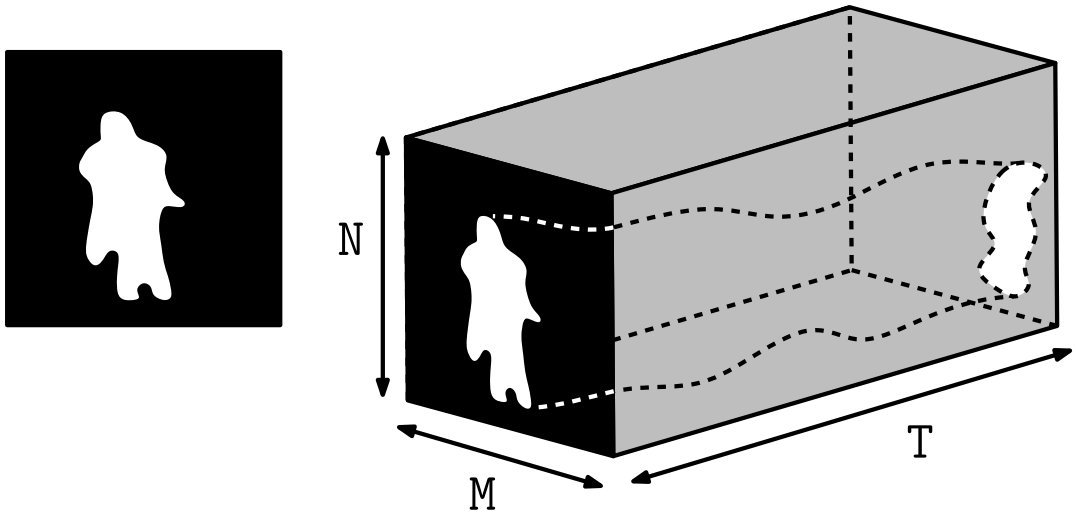
Data Exploration

The datasets and inputs for this project will be obtained after some video preprocessing made on sample videos. The selected videos are located in capstone_project/data/unprocessed_videos and they have been downloaded from VIRAT Video Dataset [1]. The video preprocessing is carried out using the free packages SciKitImage and SciKitVideo and has been generously provided by WardenAutomation.

The video preprocessing takes as input a `.avi` $\mathtt{M} \times \mathtt{N}$ pixels video and a `.png` $\mathtt{M} \times \mathtt{N}$ pixels picture (*background*). For each frame in the video, the frame and the background are compared pixel-wise, and the difference is recorded in a $\mathtt{M} \times \mathtt{N}$ binary array, where 1 denotes difference (above some threshold) and 0 denotes no difference.

If the video has $\mathtt{T}$ frames, then the video preprocessing gives as output a $\mathtt{M} \times \mathtt{N} \times \mathtt{T}$ binary array, which is supposed to capture the movement in our scene and that will be the primary input for the project. These arrays are saved not as arrays as such but as `.avi` files, and are located in capstone_project/data/processed_videos.



The total ammount of data in this latter folder sums up to 10 videos (and a exploratory `video1.avi`) whose duration ranges between half and three minutes. This should be enough to tune our algorithm (see below) for the purposes of similar (fixed-camera) videos.

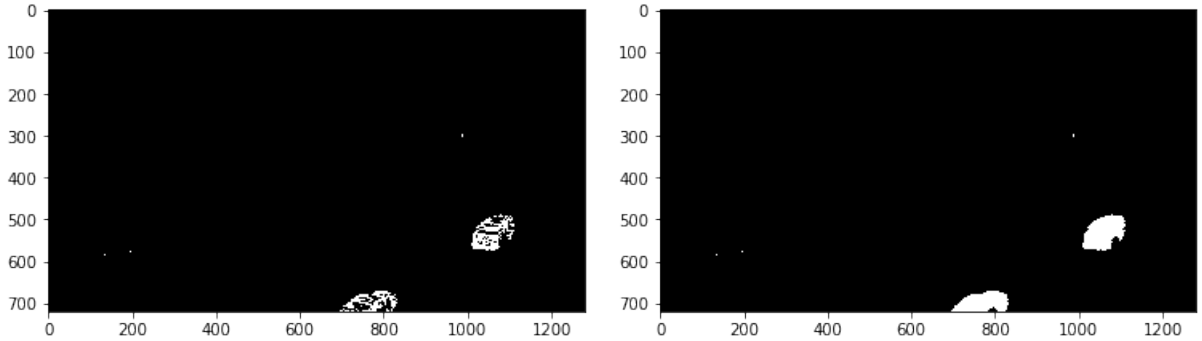
[1] **A Large-scale Benchmark Dataset for Event Recognition in Surveillance Video**, Sangmin Oh, Anthony Hoogs, Amitha Perera, Naresh Cuntoor, Chia-Chih Chen, Jong Taek Lee, Saurajit Mukherjee, J.K. Aggarwal, Hyungtae Lee, Larry Davis, Eran Swears, Xiaoyang Wang, Qiang Ji, Kishore Reddy, Mubarak Shah, Carl Vondrick, Hamed Pirsiavash, Deva Ramanan, Jenny Yuen, Antonio Torralba, Bi Song, Anesco Fong, Amit Roy-Chowdhury, and Mita Desai, *Proceedings of IEEE Comptuer Vision and Pattern Recognition (CVPR), 2011*.

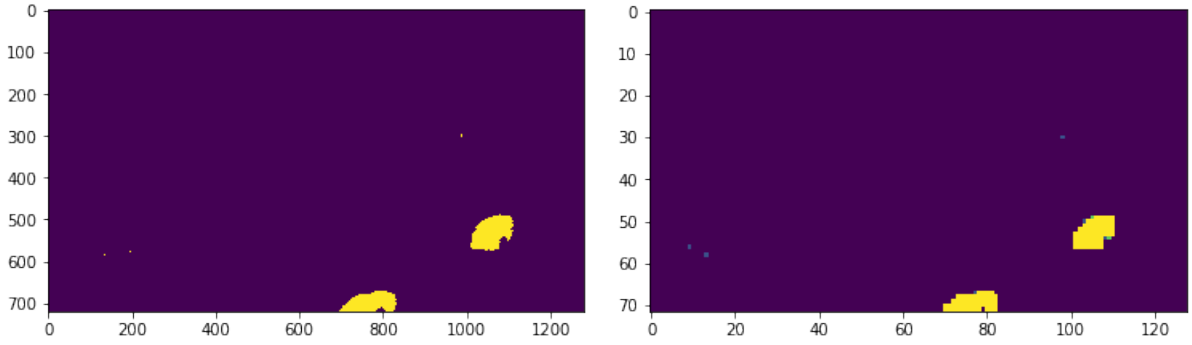
Exploratory Visualization

The two first `Jupyter` notebooks deal with some exploratory visualisation

capstone_project/1. data_extractor.ipynb

Some of the frames of an exploratory video:

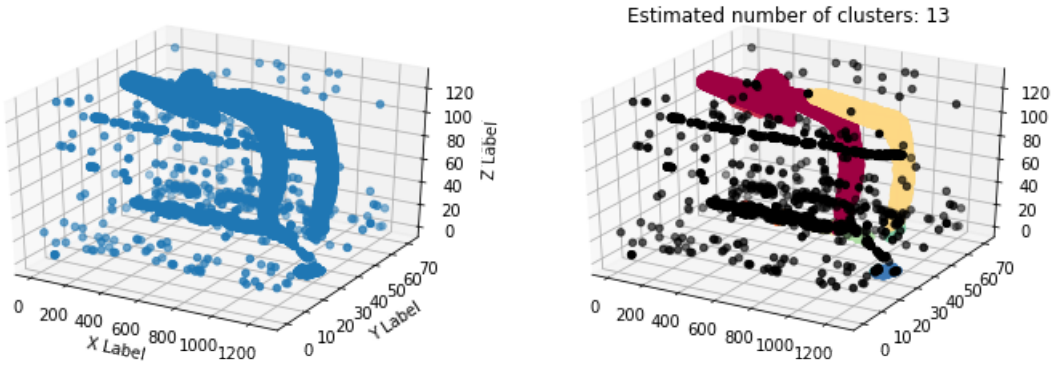




1. Single frame of video
2. Same frame after applying `dilation` and `erosion`
3. One layer of the previous frame
4. Dimensionality reduction (`rescale`)

capstone_project/2. exploration_clustering.ipynb

Visualisation of point clouds:



1. Cloud of points related to an exploratory video
2. Cloud of points after running `DSCan` algorithm

Algorithms and Techniques

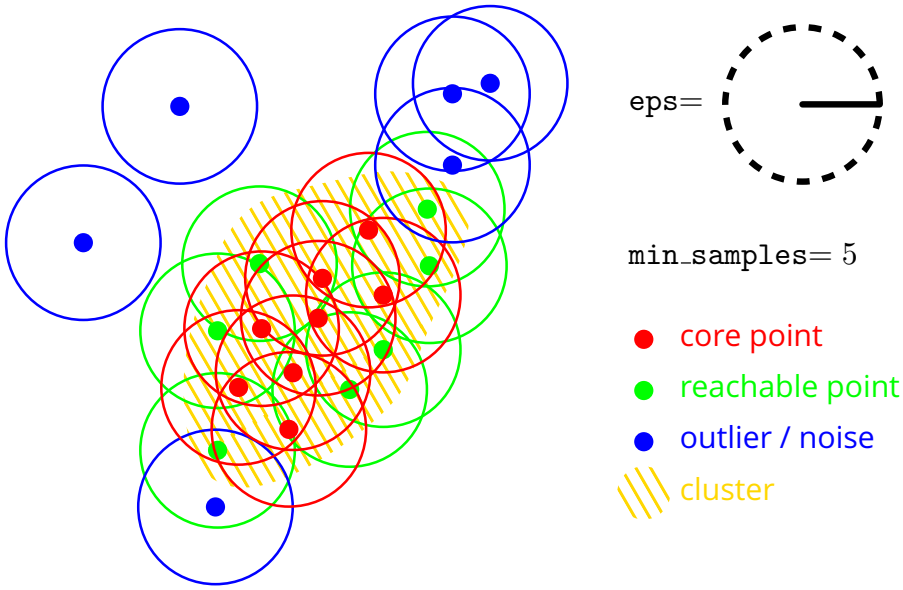
We propose the use of DBSCAN clustering algorithm, which is implemented in SciKitLearn. This clustering algorithm is well-suited for our generic clusters, which are uneven in size and non-convex.

DBSCAN clustering algorithm (*Density-based spatial clustering of applications with noise*) takes primarily two parameters

- `min_samples`, integer
- `eps`, float

and classifies the points in a metric space into three groups

- **Core points**: points whose `eps`-neighbourhood contains at least `min_samples` points
- **Reachable points**: points that are not core points themselves, but that cointain some core point in their `eps`-neighbourhood
- **Outliers or noise**: points that are not core points nor reachable points



Each group of mutually `eps`-path-connected core points with the associated reachable points forms a cluster. The way in which the algorithm is implemented may lead to some indeterminisms, such as reachable points reached from two different clusters, but the number of clusters and their intrinsic shape are well-determined.

This algorithm has some major advantages that make it well-suited for our clustering problem:

- The number of expected clusters is not to be specified. This is crucial, since our goal is to determine the number of clusters / movement-events
- The clusters may be of any shape
- The algorithm deals well with noise (our arrays, generated from video processing, always have noise)

[1] DBScan - Wikipedia

Benchmark

A major Benchmark Model may be found at Actions as Space-Time Shapes.

This work deals with the recognition of activities in a single movement-event. For this movement-event to be analysed, spectral clustering methods are used [1]. The performance is measured in terms of misclassifications (2.17%, 7.91% and 36.40% for different methods).

[1] Spectral Clustering - Wikipedia

III. Methodology

Data Preprocessing

The first step in the data preprocessing has been generously provided by WardenAutomation. It consists in the extraction of *movement masks*, with the method described above, processing the videos in unprocessed_videos into those of processed_videos.

The second step in the data preprocessing is carried out by the `Jupyter` notebook data_extractor.ipynb. For each frame in the movement masks, it applies

a) Several `dilation` and `erosion` processes

b) Extraction of one color layer

c) Dimensionality reduction (`rescale`)

and finally stores the coordinates of the movement points in `NumPy` arrays in the folder numpy_arrays.

Implementation

The implementation of the `DBScan` algorithm is remarkably easy:

```
db = DBSCAN(eps=3, min_samples=20).fit(data)
labels = db.labels_

# Number of clusters in labels, ignoring noise if present.
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)

print('Estimated number of clusters: %d' % n_clusters_)
```

where `data` is the arrays of points to be clustered.

Refinement

The notebook 3. parameter_tuning.ipynb deals with the identification of the optimal parameters `eps` and `min_samples` inside the `DBScan` algorithm. For each of the exploratory pairs

```
eps in [2,4,6,8,10,12,14,16,18,20]
min_samples in [40,80,120,160,200,240,280,320,360,400,
    440,480,520,560,600,640,680,720,760,800]
```

a `DBScan` method has been applied to the training data and a Sum of Square Errors (when comparing the output with the expected output registered by direct observation of the original videos) has been stored. The pair in which the best local minimal has been achieved is `eps = 6 , min_samples = 400`, and this values have been used for the final implementation.

IV. Results

Model Evaluation and Validation

The notebook 4. implementation_validation.ipynb deals with the evaluation and validation.

Our tuned `DBScan` algorithm is applied to our training videos. We compare the expected clusters and the output given. We also measure the Silhouette Coefficient [1] and the Calinski-Harabaz Index [2]. Some of the calculations give rise to a `Memory Error`, so we do not have the scores for every training video.

[1] Silhouette Coefficient - SKLearn

[2] Calinski-Harabaz Index - SKLearn

Justification

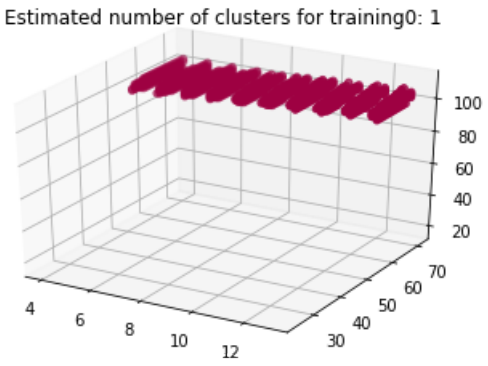
The results obtained (4. implementation_validation.ipynb) are quite good, but not excellent; further improvement should be done.

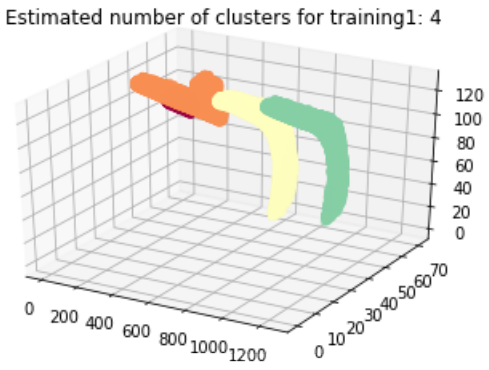
Comparing our results with those in our Benchmark Model Actions as Space-Time Shapes, we may appreciate that the last ones also run into a great number of misclassifications (2.17%, 7.91% and 36.40% for different methods). This shows that the movement-events recognition is in general quite a difficult problems where major Machine Learning techniques have to be studied and implemented.

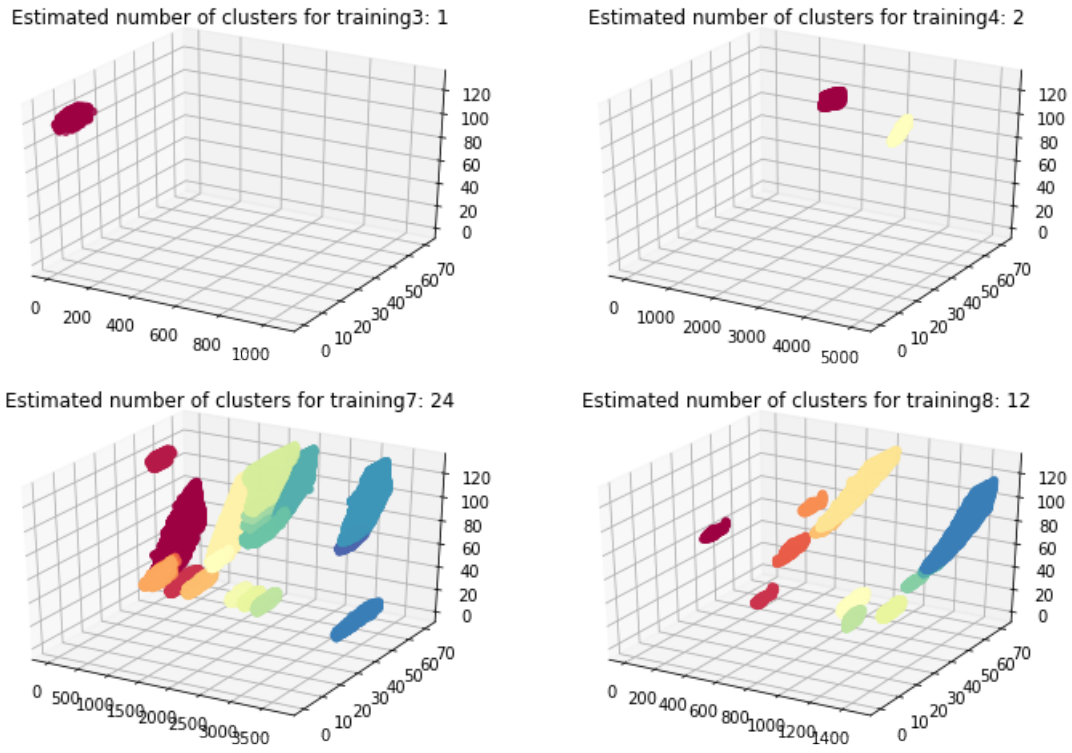
V. Conclusion

Free-Form Visualization

The notebook 5. final_visualisation.ipynb deals with the final visualisation.







Reflection

This Udacity ML Nanodegree Capstone Project has been a very enriching experience. It has been difficult but it has provided most encouraging results.

Perhaps the toughest part in the project has been the video preprocessing. It needed a lot of memory and time, so several methods had to be implemented (e.g. the use of Python generators instead of list comprehension) until finding one that could properly deal with such huge datasets.

The posterior tuning and validation has also been quite a subtle point. Almost all the clustering metrics assume knowledge of the ground truth classes, which in our setting is totally non-viable (even when the movement-events may be enumerated by direct visualisation).

Improvement

There are several aspects in which this method may be improved in further work:

- The movement masks extraction algorithm may be tuned to capture better the movement-events.
- The `eps` and `min_samples` parameters could be dynamically tuned in function of the input, if different fixed-camera contexts are to be considered. For instance, different parameters would be needed for driving cars or walking people.