An Ensemble Learning Approach to ImproveTracking Accuracy of Multi Sensor Fusion

by

Anoop Dasika, Praveen Paruchuri

Report No: IIIT/TR/2020/-1

Centre for Visual Information Technology
International Institute of Information Technology
Hyderabad - 500 032, INDIA
November 2020

An Ensemble Learning Approach to Improve Tracking Accuracy of Multi Sensor Fusion

Paper# 692

Abstract. Finding or tracking location of an object accurately is a key problem in defense applications and also object localization in the fields of robotics and computer vision. Radars fall into the spectrum of high-end defense sensors or systems on which the security and surveillance of entire world depends. There is a lot of focus on the topic of multi sensor fusion in recent years with radars as the sensors. Key issues of focus in this topic include the variable sampling rate of sensors due to which the time interval of the observational data can change irregularly, delays in communication between the multi sensor fusion system and sensors and protocol to use when targets enter and leave observation range of a sensor. In this paper, we focus on the problem of asynchronous observation of data which can reduce the tracking accuracy of a multi sensor fusion system comprised of radars of different types at different locations. Our solution utilizes a machine learning approach to train models on hundreds of hours of (anonymized real) multi sensor fusion data provided by radars performing tracking activity across Indian air space. Our approach comprises of 3 steps: In the first step, we train an ensemble model of logistic regression and Xgboost to predict one type of error namely splitting error, which can help to improve the accuracy of tracking. In the second step, we use Xgboost to predict the second type of error namely merging error, which can further improve the accuracy of tracking. The third step uses a nearest neighbour search to compensate for the predicted errors by retaining the data points removed in the first two steps while maintaining the tracking accuracy. Our experimental results show that the trained models provide good predictions of errors and increase accuracy of tracking by 15 percent while retaining around 100 percent of the data.

Keywords: Radars · Multi Sensor Fusion · Object Tracking.

1 Introduction

Determining or tracking the location of objects is an important problem in robotics [2], computer vision [3] and for defense applications in general [1]. Radars are generally used in defense applications when we need to locate objects over long distances and fall into the category of high-end security systems on which surveillance of entire world depends in current times. A radar is an electro-magnetic system [4] which is useful for detection and location of target objects such as aircraft, ships, spacecraft, vehicles, people and natural environment [5]. It can be viewed as an active sensor which can detect targets by emitting large power electromagnetic signals and also determines the angle, range or velocity of the targets. Radars work in all kinds of weather and this all-weather capability has contributed to their extensive use in various commercial applications such as autonomous robotics and mobile systems and defense applications such as tracking, surveillance and detection of targets in air, naval and ground.

The topic of **Multi Sensor Fusion (MSF)** has received a lot of attention in the recent years [4] due to the increasing automation across the world which resulted in a significant increase in the number of information collection devices. This is true in the context of radars too due to the following reason: To ensure smooth tracking of targets, radars have overlapping regions they monitor which can result in detection or collection of varying data on the same aspect of an issue or for the same target being tracked, which makes multi sensor fusion a necessity. Through the integration of data obtained from different sensors, the results can in general be optimized better and made continuous in terms of having a complete picture rather than having individual snapshots of a scenario obtained using multiple independent radars. Considering the fact that newer radars with different properties may replace some of the older ones over time, defense organizations would in general need to perform multi sensor fusion for a heterogeneous multi radar system.

Due to the differences in sampling rate of sensors, the communication delay between sensors and overlapping regions of observations for the various sensors, there can be asynchronity in the their observations which can significantly reduce the tracking accuracy of a multi sensor fusion system. There is therefore a strong need in the defence and commercial industry to develop solutions that improve the tracking accuracy of multi sensor fusion systems. We propose to use a machine learning based approach in this paper to tackle this issue. Rest of the paper is structured as follows: We provide brief introduction on radars and multi sensor fusion based tracking in Section 1, define the specific problem faced in defense industry in Section 2 and briefly describe the concepts used in the paper in Section 3. We propose our machine learning approach in Section 4, present the experimental results obtained using our approach in Section 5 and present the conclusions and future work in Section 6.

2 Problem Description

The Indian Air Force uses a multi radar system to detect flights pan India. These radars are located in different parts of the country and each radar has its own processing center. Measurement data is processed by each radar resulting in multi-target local tracks L_t. The processing involves usage of Kalman and advanced Kalman filters to compute the 3d location (in the form of distance from the sensor) and velocity of the above tracks. The processed information is then sent to fusion center of the system which performs multi sensor fusion (MSF) for data alignment, data interconnection, track filtering and association and track fusion to obtain the position in the form of latitude, longitude and altitude. The global track $\mathbf{G_t}$ is then computed using the local track information. Figure 1 captures the process flow involved.

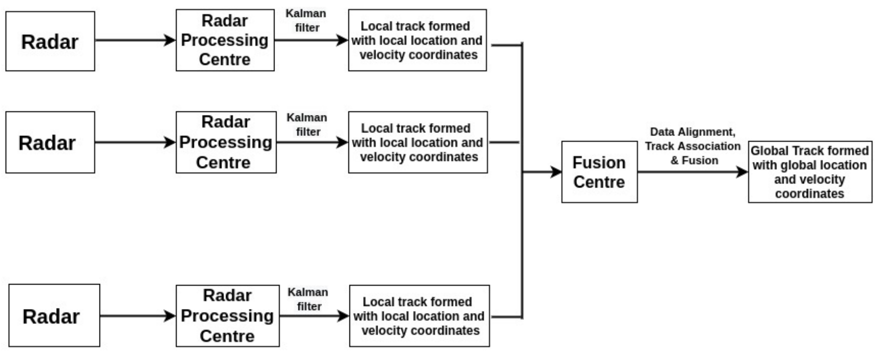
Radar
Radar Processing Centre
Kalman filter
Local track formed with local location and velocity coordinates
Radar
Radar Processing Centre
Kalman filter
Local track formed with local location and velocity coordinates
Radar
Radar Processing Centre
Kalman filter
Local track formed with local location and velocity coordinates
Fusion Centre
Data Alignment, Track Association & Fusion
Global Track formed with global location and velocity coordinates

Fig. 1. Working of Multi Sensor Fusion System

A real world multi sensor fusion system has a lot of engineering involved apart from track association and alignment algorithms, which typically include human generated rules tailored to specific use cases. As the use cases expand, as heterogeneous radars get added, as the nature of targets change and as the requirements for accuracy increase, the system gets more prone to errors due to limitations in the multi sensor fusion logic. Consider a scenario in which there are say 4 air targets. Sensors gather data from these air targets and generate Air Situation Picture (ASP) displaying 4 air targets with relevant attributes. However, there will be times where the actual number of air targets are shown as 4+/4- for certain period of time and then again shown as 4. If the MSF system observes 4+ air targets we call the error as a **splitting error** $\mathbf{E_s}$ i.e., sensing multiple targets when only a single one is present. When it observes 4- air targets we call the error as **merging error** $\mathbf{E_m}$ (sensing a single target when multiple targets are present). $\mathbf{E_s}$ can lead to the system assuming that there is an enemy flight in air even though there isn't any. $\mathbf{E_m}$ can lead to the system assuming that there is no enemy flight in air even though there may be one (or more). Both these errors in air surveillance can result in errors in threat evaluation by the system which would in turn result in wrong action getting taken leading to serious threats and security issues.

3 Our proposed solution

Figure 2 presents block diagram of our proposed solution to improve the tracking accuracy by using a machine learning approach in the data obtained after MSF. The MSF data is collected on a daily basis and is stored as a history dump (past

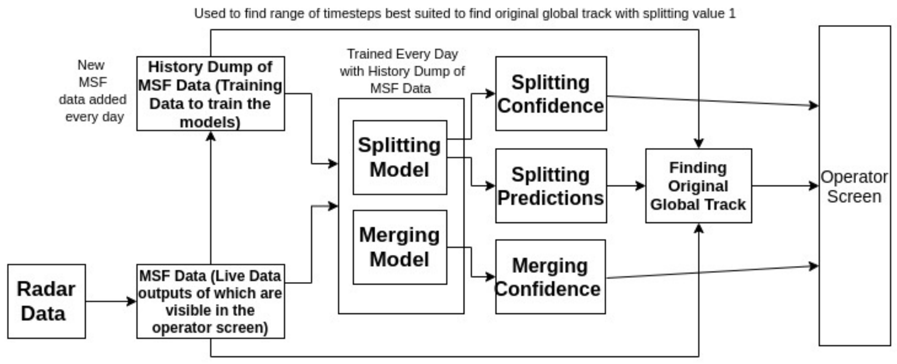
Used to find range of timesteps best suited to find original global track with splitting value 1
New MSF data added every day
History Dump of MSF Data (Training Data to train the models)
Trained Every Day with History Dump of MSF Data
Splitting Model
Merging Model
Splitting Confidence
Splitting Predictions
Merging Confidence
Finding Original Global Track
Operator Screen
Radar Data
MSF Data (Live Data outputs of which are visible in the operator screen)

Fig. 2. Proposed AI Model

1 week data which we refer to as dataset in the paper) to train (two separate) models that handle splitting and merging issues. We post process this history dump data which consists of flight code to classify **splitting** and **merging** (real time data doesn't contain flight code). The splitting model $\mathbf{S}$ would provide splitting confidence $\mathbf{C_s}$ i.e., probability of splitting. When a data point has high $\mathbf{C_s}$ then it is classified as splitting else it is real. We find the original Global Track number $\mathbf{G_t}$ of all data points classified as splitting. Similar to splitting model $\mathbf{S}$, the trained merging model $\mathbf{M}$ provides merging confidence $\mathbf{C_m}$ i.e., probability of merging.

Our approach comprises of 3 steps: In the first step, we train an ensemble model of logistic regression and Xgboost to classify all data points into splitting and real which can help to improve the accuracy of tracking. In the second step, we use Xgboost to classify all data points into merging and real which can further improve the accuracy of tracking. The third step uses a nearest neighbour search to modify data points classified as splitting into real ones. Figure 3 presents block diagram of these three steps.

We now present details of each of the three steps.

3.1 Step 1: Splitting Classification Model S $\mathbf{E_s}$

Based on understanding of the dataset, we identified the following features to use for development of splitting model $\mathbf{S}$ – latitude, longitude, altitude, speed and direction of object detected by MSF system at each time step precise to a millisecond, number of radars and a unique id depicting the specific combination of radars used in the MSF system for creation of individual data points.

MSF performed on the data provided by individual radars leads to assignment of global track number $\mathbf{G_t}$ to each of the newly detected objects. If for a particular flight, a new $\mathbf{G_t}$ is assigned even though there is an existing one, all the data points detected with a new $\mathbf{G_t}$ are a result of splitting from the original $\mathbf{G_t}$ of the flight and will be classified as **splitting** (data points created as a result

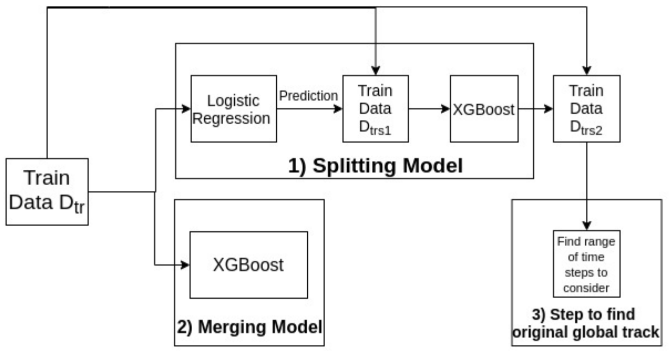

Fig. 3. Models Training Methodology

of Splitting Errors $\mathbf{E_s}$). Let our training dataset with features mentioned above be denoted as $\mathbf{D_{tr}}$ and test dataset with features mentioned above be denoted as $\mathbf{D_t}$

As part of the ensemble model, we first use a parametric machine learning algorithm Logistic Regression and train it with $\mathbf{D_{tr}}$. We add the predictions obtained using the logistic Regression to our train set $\mathbf{D_{tr}}$ and test set $\mathbf{D_t}$ and denote these new datasets as $\mathbf{D_{trs1}}$ and $\mathbf{D_{ts1}}$ respectively. Next we use a non parametric machine learning algorithm XGBoost and train it with the new train dataset $\mathbf{D_{trs1}}$. We obtain splitting confidence $\mathbf{C_s}$ from the above model. If the output has high $\mathbf{C_s}$ (based on threshold which gives the highest f1 score) then we classify it as splitting. We add the predictions obtained by using the XGBoost model to both our train set $\mathbf{D_{trs1}}$ and test set $\mathbf{D_{ts1}}$ and denote these new data sets as $\mathbf{D_{trs2}}$ and $\mathbf{D_{ts2}}$ respectively.

3.2 Step 2: Merging Classification Model M $\mathbf{E_m}$

We use the initial training set $\mathbf{D_{tr}}$ and the initial test set $\mathbf{D_t}$ for purposes of developing this model. If for a particular global track number $\mathbf{G_t}$, a new flight number gets associated by MSF system even though there is an existing one, all the data points detected with this $\mathbf{G_t}$ and the new flight number are a result of merging of 2 different flights and will be classified as merging (data points are combination of multiple data points (flights) merging together after MSF at some time step). We use a non parametric machine learning algorithm XGBoost and train it with the train dataset $\mathbf{D_{tr}}$. We obtain merging confidence $\mathbf{C_d}$ from the above model. If the output has high $\mathbf{C_m}$ (based on threshold which gives the highest f1 score) then we classify it as merging. We then insert a new data point with the same features as input data points but with a new global track number. So now every data point classified as merging is split into two different tracks which in turn lead to reduction of merging error E_m.

3.3 Step 3: Correcting data point classified as splitting

In the first two step we determined which data point is tracking accurately and which is not. But we cannot just delete the data points which have splitting issues as we may lose important tracking information of targets. The third step therefore helps in correcting/retaining the data points which were classified as splitting. First we create a new train $\mathbf{D_{trgt}}$ and test dataset $\mathbf{D_{tgt}}$ from our train $\mathbf{D_{trs2}}$ and test dataset $\mathbf{D_{ts2}}$ obtained by splitting model $\mathbf{S}$ and retaining entries with predicted splitting value $\mathbf{P_s}$ of 1. Every data point in our dataset forms at a particular time precise to a millisecond. In dataset $\mathbf{D_{trgt}}$, for every data point we construct a small new dataset with all the data points captured in the immediate past of this data point. For every data point, we find the $\mathbf{G_t}$ of the nearest neighbour by calculating the Euclidean distance between the features latitude, longitude and altitude of every data point in the constructed dataset and the data point whose original $\mathbf{G_t}$ we are trying to determine.

4 Experiments and Results

We performed a survey of different papers at start of this work and could not find ones that can be of significant help. Most state of the art multi sensor fusion process operate on small domain settings and use machine learning during the multi sensor fusion process whereas our work involves monitoring a very large area with lots of objects to track in real-time(scalability) and proposed algorithm uses the data obtained from the multi sensor fusion process and improves on it so would work for any organization where sharing the multi sensor fusion process is a security risk.

As part of our experimentation, we compare the performance of our splitting $\mathbf{S}$ and merging $\mathbf{M}$ models against the following algorithms: Logistic Regression, Gaussian Naive Bayes, Support Vector Machines, Gaussian Nearest Neighbour, Random Forest Classifier, XGBoost, Stacking Logistic Regression with XGBoost and Artificial Neural Networks.

Dataset: Our dataset contains 3.3 million data points obtained as part of 1 week of MSF history dump. It is post processed by us as per method given section 4.1 and 4.2 as it doesn't contain splitting and merging values which we get using flight code provided in the MSF data. After post processing of MSF data, we get around 653k (k = 1000) data points classified as **splitting** (i.e. they are formed as as a result of splitting error) and around 73k data points classified as **merging**(i.e. they are formed as a result of merging error).This translates to an error rate of 22 percent (i.e. tracking accuracy of 78 percent),20 percent due to splitting error and 2 percent due to merging error. Due to this imbalance in , instead of using accuracy or mean squared error as criterion, we use f1 score which is the harmonic mean of precision and recall.

3.2 million random data points are our test set $\mathbf{D_t}$ and the other 100k data points from the above are used as train set $\mathbf{D_{tr}}$. We use such a small train set $\mathbf{D_t}$ compared to the whole data because any increase in data with the algorithms used in our experiments, lead to over fitting or no further increase in accuracy.

4.1 Splitting Classification Model S

Table 1 presents prediction results for eight different techniques (including our technique Stacking Logistic Regression with XGBoost) obtained using comparing splitting prediction $\mathbf{P_S}$ and **splitting** (target variable) value (obtained using post processing of MSF data which isn't available real time) in the test dataset $\mathbf{D_t}$. Note that the dataset $\mathbf{D_{tr}}$ was used for training the splitting model **S** along with hyperparameter tuning. A number of structures were considered for designing the neural network that provides the best feasible performance.

Table 1. Results of Splitting model

Sr No	Algorithm	F1 Score	Recall
1)	Logistic Regression	0.55	0.68
2)	Gaussian Naive Bayes	0.49	0.69
3)	Support Vector Machines	0.49	0.37
4)	K Nearest Neighbours	0.38	0.38
5)	Random Forest Classifier	0.86	0.81
6)	XGBoost	0.89	0.84
7)	Stacking Logistic Regression with XGBoost	0.94	0.94
8)	Artificial Neural Networks	0.64	0.81

Table 1 shows that a stacking model which ensembles Logistic Regression with a XGBoost classifier has the best performance predicting the Splitting errors $\mathbf{E_s}$. We obtain the following scores from this model when applied to our test dataset $\mathbf{D_t}$:

Accuracy = 0.98
Precision = 0.95
Recall = 0.94
F1 Score = 0.94

So the tracking accuracy of the test dataset $\mathbf{D_t}$ can be increased from previous 78 percent in the first model to around 95 percent, if we delete the data points with $\mathbf{P_s} = 1$. Fig **??** shows the relative importance(average number of training samples classified correctly because of a particular feature) on x-axis and the different features used in the splitting model on the y-axis. We observe the following order of importance for the features namely logistic regression predictions followed by Flight type, global track number, no of radars used (the no of radars used to get this data point) and location parameters.

4.2 Merging Classification Model M

Table 2 presents prediction results for seven different techniques (including our technique XGBoost) obtained using comparing merging prediction $\mathbf{P_M}$ and

merging (target variable) value (obtained using post processing of MSF data which isn't available real time) in the test dataset $\mathbf{D_t}$. Note that the dataset $\mathbf{D_{tr}}$ was used for training and hyperparameter tuning the merging model $\mathbf{M}$. As with splitting model $\mathbf{S}$, a number of structures were considered for design of neural network to obtain the best feasible performance.

Table 2. Results of Merging model

Sr No	**Algorithm**	**F1 Score**	**Recall**
1)	Logistic Regression	0.1	0.19
2)	Gaussian Naive Bayes	0.04	1
3)	Support Vector Machines	0.12	0.4
4)	K Nearest Neighbours	0.11	0.11
5)	Random Forest Classifier	0.45	0.35
6)	XGBoost	0.91	0.89
7)	Stacking Logistic Regression with XGBoost	0.88	0.85
8)	Artificial Neural Networks	0.51	0.62

Table 2 shows that XGBoost classifier has the best performance predicting the Merging Errors $\mathbf{E_m}$. We obtain the following scores from this model when applied to our test dataset $\mathbf{D_t}$:

Accuracy = 1
Precision = 0.92
Recall = 0.89
F1 Score = 0.91

So the tracking accuracy of the test dataset $\mathbf{D_t}$ can be increased from previous 95 percent in the second model to around 97 percent, if we add an extra data point with new G_t for every $\mathbf{P_m} = 1$. Fig **??** shows the relative importance (y-axis) of the different features(x-axis) used in the merging model. We observe the following order of importance for the features namely Flight type, global track number, no of radars used (the no of radars used to get this data point) and location parameters.

4.3 Correcting data point classified as splitting

For finding the original track number, we first looked into clustering algorithms as we wanted to cluster all data points classified as splitting with a real data point. As cluster locations(i.e. feature values of data points classified as splitting) need to be fixed, there was no need for learning so clustering algorithms were unnecessary and euclidean distance was used. Via experimentation with our training data , we find that a time range of 20 seconds provides the most accuracy in predicting G_t for all data points in our train dataset. We therefore use this time range to obtain predicted global track number $\mathbf{G_{pt}}$ in our test dataset $\mathbf{D_{tgt}}$. By comparing $\mathbf{G_t}$ and $\mathbf{G_{pt}}$ we observe an accuracy of 61 percent in our test dataset $\mathbf{D_{tgt}}$. This means that due to the compensation of errors performed using Step 3, we have a 61 percent chance of being able to identify the correct global track $\mathbf{G_{pt}}$ of a flight which was classified as splitting. So our final accuracy without the third step is 97 percent but with only around 78 percent retention of data, after the third step our accuracy reduces to 91 percent but with 100 percent retention of the data.

4.4 End Results

Out of our 3.2 million data points in test dataset $\mathbf{D_t}$, with around 20 percent having **splitting** value 1 (formed due to splitting error $\mathbf{E_s}$) and 2 percent having **merging** value 1 (formed due to merging error) $\mathbf{E_m}$, we are able to predict 98 percent of the data points of the data points that have splitting error and 99 percent of the data points that have merging error. So if we delete the data points with splitting error we get an tracking accuracy increase from 78 percent to almost 95 percent but a reduction of data points from 3.2 million data points to 2.6 million in test dataset $\mathbf{D_t}$. /Using step 3, we can add back all the data points to the test dataset $\mathbf{D_t}$ but tracking accuracy reduces to 91 percent which is still a 13 percent increase from MSF. As merging is predicted with near 100 percent accuracy, we correct the 2 percent merging error and increase total tracking accuracy to 93 percent which boils down to a total 15 percent increase in tracking accuracy after MSF with 100 percent data retention.

5 Conclusion and Future Work

The topic of Multi Sensor Fusion has received great attention in the recent years due to its application both in military and commercial applications. In this paper, we focus on the issue of improving accuracy of the output of an existing Multi Sensor (Radar) Fusion System used in the context of tracking flights across India. Due to the asynchronous observation of targets by the radars as well as overlapping regions of observation, a number of errors happen during the fusion process primarily categorized into merging and splitting errors. In this paper, we introduce a three step machine learning approach to improve the confidence in tracking and reduce the errors observed. In particular, we train models to predict splitting (step 1) and merging (step 2) of tracks in data and and use nearest neighbour search (step 3) to compensate for the identified errors that would help to improve the tracking accuracy while retaining most of the data. Using this three step process, we are able to improve the accuracy of tracking of the MSF system from 78 percent to about 93 percent while retaining 100 percent of the data. Along with high performance we also score well on explainability (important criterion for deployment).

Another aspect we plan to look into is the following: While our algorithm can potentially help with identifying and compensating for the errors in tracking, given the sensitive nature of defense operations there can still be uncertainty in the mind of an operator overseeing a system of this nature. There will therefore be a need for consideration of other actions depending on the confidence in tracking. We therefore plan to look into usage of a formal planning model like Partially Observable Markov Decision Process (POMDP)[17], which can enable tracking of belief and specification of different actions based on the computed belief in the tracking process.

References

1. D Lyon,"Theorizing surveillance" - 2006
2. Alexander Andreopoulos, Stephan Hasler, Heiko Wersing, Herbert Janssen, John K. Tsotsos and Edgar Korner,"Active 3D Object Localization Using a Humanoid Robot" - IEEE Transactions on Robotics (Volume: 27 , Issue: 1 , Feb. 2011)
3. Mustafa Ozuysal, Vincent Lepetit and Pascal Fua,"Pose estimation for category specific multiview object localization" - 2009 IEEE Conference on Computer Vision and Pattern Recognition
4. Ruotsalainen Marja and Juha Jylhä, "Learning of a tracker model from multi-radar data for performance prediction of air surveillance system" - Evolutionary Computation (CEC) 2017 IEEE Congress on. IEEE, 2017.
5. Ke Ma, Hanguang Zhang, Rentao Wang and Zhimin Zhang,"Target tracking system for multi-sensor data fusion" - 2017 IEEE 2nd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)
6. Jason Brownlee,"Master Machine Learning Algorithms" - 2016
7. DG Kleinbaum, K Dietz, M Gail, M Klein, M Klein,"Logistic Regression" - 2002,Springer
8. I Rish,"An empirical study of the naive Bayes classifier" - IJCAI workshop, 2001
9. B Schlkopf, AJ Smola, F Bach,"Learning with kernels: support vector machines, regularization, optimization, and beyond" - 2018
10. MH Hassoun,"Fundamentals of artificial neural networks" - 1995
11. JR Quinlan ,"Induction of decision trees" - Machine learning, 1986 - Springer
12. A Liaw, M Wiener,"Classification and regression by randomForest" - R news, 2002
13. RE Schapire,"Explaining adaboost" - Empirical inference, 2013 - Springer
14. T. Chen and C. Guestrin,"XGBoost: A scalable tree boosting system" - In Proceedings of the 22nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining, pages 785–794, San Francisco, CA, 2016.
15. R Polikar, "Ensemble learning" - 2012, Springer
16. Faezeh Hajiaghajani, Marjan Naderan,Hossein Pedram, Mehdi Dehghan,"HCMTT: Hybrid Clustering for Multi-Target Tracking in Wireless Sensor Networks" - 4th International Workshop on Sensor Networks and Ambient Intelligence 2012, Lugano (23 March 2012)
17. SCW Ong, SW Png, D Hsu and WS Lee,"POMDPs for robotic tasks with mixed observability" - Robotics: Science Systems, 2009
18. Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, Andrey Gulin,"CatBoost: unbiased boosting with categorical features" - Advances in Neural Information Processing Systems 31 (NIPS 2018)