Progress Report, MSc. Dissertation: On-line Random Forest for Face Detection

German Alfonso Chaparro Alvarez

School of Computer Science
The University of Manchester

May 9, 2014

Contents

1 Introduction 2

2 Background 3

3 Research Methods 6
  3.1 What the project involves . . . . . . . . . . . . . . . . . 6
  3.2 The project plan and evaluation of the plan . . . . . . . . . . 8

4 Progress 10
  4.1 Quality attributes . . . . . . . . . . . . . . . . . . . . . . . 10
  4.2 Prototypes . . . . . . . . . . . . . . . . . . . . . . . . . . . . 11
    4.2.1 PGM Image . . . . . . . . . . . . . . . . . . . . . . . . . 11
    4.2.2 Working with Haar-like features and Integral Image . . . 12
    4.2.3 Accessing the Webcam Driver . . . . . . . . . . . . . . . . 13
    4.2.4 The On-line Random Forest . . . . . . . . . . . . . . . . 15
    4.2.5 The First version of the User Interface . . . . . . . . . . 15
  4.3 Open discussion about the On-line Random Forest . . . . . . . . 16

5 Next Steps and Conclusions 17

6 References 18
Abstract

This dissertation presents the complete process of design, implementation and test of machine learning model based on on-line decision trees which main goal consists in learning from data streams. This is a supervised learning model that changes the paradigm of having training and testing phases and proposes an on-line learning method in which the model learns while classifies. The selected model is called On-line Random Forest and a very specific problem such as face detection over video streaming will be the issue that we would try to solve. This technique increases the scope of traditional or off-line learning models. The on-line learning models have many real world applications: since the traditional email categorization problem, to the real time trading algorithms for financial sector, and surveillance systems in security companies are just a few examples.

1 Introduction

Machine learning models are being used broadly and successfully in many knowledge fields in order to enable systems to learn from data. Several techniques have been developed to reach this goal and is not difficult to realise that the brand new techniques are born as improvements over the first ones, based on the need to evolve according to new requirements. This is the case of study of this dissertation, which aim is to design, develop, test and analyse the results of a new way to work with Decision Trees.

Decision trees learning algorithms are the machine learning techniques that use tree data structures as core of their process. Off-line decision trees are those tree based models which learning process is done basically in two steps: training phase and testing phase. These algorithms have proved their high accuracy and very good performance the environments they have been used. However, when there is a huge bank of data the training and testing phases are generally expensive in time and resources, and if its required that the model learn from new data without “forget” previous information, is necessary to train and test the model with the whole set of old data and new data. All the characteristics of these type of models is shown in [4].

Based on the off-line decision trees, a new model has emerged which tries to preserve the accuracy of its predecessor but trying to solve the issue of having to retrain and retest the complete model once new samples must be added to the dataset. This technique has been called on-line decision trees, and its main goal is to learn while new information arrives to the model. This provides the
opportunity to treat with information that “arrives” to the model as an stream of data, so theoretically it can deal with infinity source of information. With this idea in mind, new potential applications can be done based on real-time data processing, for example trees that could learn on-line by processing signals such as videos or sounds. One very important characteristic of this type of trees is that they are memory friendly in the sense that they do not store the samples that are being processed but only they maintain the necessary statistics according to the features of the samples.

Finally by giving another step forward, and similarly to the off-line version, a ensemble of trees would increase even more the performance of the new on-line model. So the concept of On-line Random Forest emerges as the main subject of research of this dissertation. So the main objective of this dissertation is to design, develop and test an On-line Random Forest as a supervised learning method over a stream of data. The data input streams are focused on videos sources so the model will learn to detect faces in the videos.

2 Background

The research has been divided into two main parts. The first one is related to the investigation about the on-line learning model, and the second one related to the face detection problem. Hence, this section shows the papers related to these parts.

All the base concepts about decision tress used as a machine learning model such as the decision tree learning algorithm or ID3, the concept of depth of trees, the ideas of underfitting and overfitting, the split criterion, how to measure the information gain according to the cost of the splits, were taken from chapter 6 in [4]. In the same way the chapter 9 of [4] explains how by combining appropriately the output from multiple models and treat this combination as a new ensemble model, more accurate results are obtained in machine learning related problems. The concepts of bagging used for Random Forest models and boosting for Adaboost classifiers are also explained in this chapter.

In the original version of the random forest the bagging process is done by re-sampling randomly the examples from the original dataset to produce several training datasets or bootstraps, and each of those new datasets will be the input for each tree in the forest. So each tree will be trained by a bootstrap of the original dataset. In the on-line version of the random forest, the bagging process is not used because of the of the meaningless of knowing the size of the dataset (now the input is a stream of data so we don’t how many samples will arrive). [8] and [7] propose the on-line version of the bagging process. The authors of those articles probe that if the number of samples tends to infinite (that is the case
of a data streaming) the bagging process could be replaced by calculating how many times each single example must be sent to each single tree in the forest. This number of times is calculated based on a Poisson distribution. Their results show that the on-line model behaves similar (and identical when the dataset is large) to the original off-line version. The papers use the ITI algorithm to test and evaluate the results.

Another on-line model based on decision trees is explained in [1]. The technique is called Online Adaptive Decision Trees which is a neural network with the topology of a binary tree. The authors develop this algorithm taking advantage of the properties of the neural networks related to process information in on-line mode. Some characteristics about this model are: it uses the gradient descent learning algorithm due to is one of the traditional algorithms used for training neural networks, each of its nodes stores a decision function and an activation function, and the depth of the tree is the only parameter that affects the performance of the model. Their results show that underfitting could be present if the depth of the tree is low, but overfitting is not present while the depth of the tree is increase.

The article cited in [16] uses a Extremely Random Forest model to commit on-line learning and execute tracking tasks over videos. The authors of this article also explain the need of treat data streams as source samples for the learning process, and their technique is based on the idea expand trees with very few examples. To reach this goal they save the samples in the leaf nodes of the trees, and using the information of the samples in combination with the Gini Index they will decide to split the node into two new ones. The results shown in this paper are quite impressive by showing how tracking activities are done with very few samples, but the approach we will work in this dissertation is different in the sense of we are not going to store the samples in the trees.

Finally [9] shows the best approach to the research we want to do in this dissertation. As in [8], the authors proposed to replace the bagging process used in the off-line version of random forest by calculating the number of times each sample should be sent using a Poisson distribution. They use an extremely randomized forest with a growing procedure based on the statistics that are taken when the stream of data is arriving to the model and consequently to some nodes in the tree. The authors also make difference between the terms on-line and incremental arguing that in the on-line mode the samples are not stored in the tree while the incremental mode does, so this is one of the most important differences with [16]. The code was implemented in c++ and executed in a GPU. The model was executed to solve traditional machine learning classification problems, tracking tasks and interactive segmentation in well known datasets with very interesting results. This new technique converges to the performance of the off-line version of Random Forest while the more information arrives in the data stream. For the tracking tasks, the authors executed experiments to detect faces using haar-like features.
Using [9], new papers, articles and presentations have been done. This is the case of [11] in which the authors show some techniques based on Random Forests to Boost this technique and solve motion segmentation and face detection problems. Similarly to [9], the face detection problem is tackled by using haar-like features.

The second part of the research is related to the goal of detect faces. This is a well known theme in the computer vision area and many techniques have been developed to solve this problem. [12] shows the Eigenfaces algorithm, a well known technique based on PCA to detect and recognise faces. The algorithm consists basically in calculate the covariance matrix for a dataset of faces and calculate the eigenvalues and eigenvectors of that matrix. The eigenvectors are sorted in descending order according to their corresponding eigenvalues and the eigenvectors in the top of the list are called the eigenfaces of the solution that will represent all the faces of the original dataset.

Another interesting algorithm is the Fisherfaces model explain in [2]. It creates a method based on Fisher’s Linear Discriminant that produce well separated classes with strong changes in illumination or in the face expression, characteristics that traditional eigenfaces model fail to classify.

Later [13] propose a new model called TensorFaces that considers faces expressions, lighting and head poses to detect faces in a scene. It is based on represent the face image in a tensor that basically consists in a N-dimensional matrix, where each dimension corresponds to one of the previous three features in the image.

Finally, based on [9] and [11], the technique proposed in [14] and [15] is reviewed. These are well known papers in the computer vision field and they expose how by applying the concept of integral image over an image and with very simple rectangular features (ranked by their performance in face detection over faces and non-faces images using adaboost), they use a cascade of strong classifiers based on subsets of these rectangular features (obtained as well using adaboost learning model) to detect faces in a very fast way compared to previous papers of that time. [17] shows how by using the same concept in [14] they are able to detect facial features such as eyes, nose and mouth. Later [3] also base their work in [14] to implement the face detection algorithm in a GPU. [10] collects and summarizes the use of haar-like features and shows how they are being applied to detect different types of objects such as pedestrian in a street or cardiac structures in medical images.

The final decision of which faces detection strategy should be used was based on the quality attributes explained in section 4.1. The Haar-like features technique proposed in [14] was the chosen one due to its performance in execution time. Moreover, this technique can be modified to detect several types of objects by changing the features. So in theory, the online decision tree model would be
able to learn to detect whatever object in the image according to set of features it uses.

3 Research Methods

This section of the report explains how the “big problem” was divided into two main blocks, and how a methodology based on the construction of prototypes that solve small targets related to the main of the project is being done. By applying this methodology, in this early stage of the project, some parts of the final version of the software product are already working with very interesting results. The final integration of these prototypes and the future ones will lead into the final deliverable. The second part of this section shows the progress of the project over the original project plan. Moreover, this subsection shows how the this plan is related to the small targets and prototypes based methodology and finally explain how some delays according to the initial planned dates could be recovered due to the actual achievements that were reached because of the prototypes.

3.1 What the project involves

This project is divided into two main big parts: The first one, an more important, is related to the on-line learning model based on tree structures. For this dissertation, the chosen one after some meetings and literature review was On-line Random Forest due to it could be seen as an extension of the Machine Learning lecture I took during the master. Moreover, the intention I have to leave a learning tool to teachers and future students of the School of Computer Science interested in Machine Learning as branch of the Artificial Intelligence, motivates even more the importance of this part of the dissertation. The papers show that this technique could be applied to solve machine learning traditional problems and it will behave as good as its off-line predecessors when more and more data arrives to the model. According to the literature review many applications can be develop using this technique and until now have not been draw on its full operational potential. Furthermore its on-line nature allows actual models to evolve and work with “big-data”. Now, thinking in the applications, the second part of the project arise as a very specific way to see how the on-line learning model performs in real time. Here I am talking about process video to check how the model behaves while trying to learn to detect faces.

The methodology applied during the process of this dissertation is being based on building prototypes that solve small parts of the big problem and therefore all of them oriented to achieve the aim of the dissertation. The following list of
steps explains in a better way how this method has been applied.

- Build a list of small targets that points to the main goal of the project.
- Choose a target from the list.
- Look for papers, information, software APIs, etc that describe the selected target in previous point. This papers could already belong to the bank of papers exposed in the section 2.
- Filter the information and get the most relevant papers information (1 or 2 papers, choose the more adequate API) according to the research.
- Understand completely the selected papers to be able to apply the concept in the subject of research.
- Build a software design for a prototype, based on the goal of target and on the information extracted from the papers.
- Develop and test the prototype.
- Integrate the new prototype with previous prototypes.
- Test the integrated software and fix the bugs that appear in the integration process.
- Document the last prototype according to the results. This documentation should include some information related to the design and development documentation. Moreover this documentation should include the information related to the integration process in order to update the information that will become the final product documentation.
- Check the list of targets to add or remove targets according to the results of the integrated system.
- Repeat the process until the list of targets is empty.

This process is modelled in the diagram of figure 1

During the meetings with the supervisor, several targets have been set up and until now some of them are already developed and integrated. The figure 2 show the Master List of Targets. In the section 4 a detailed view of this progress is explained.

With this methodology is easier to follow the progress of the project and to take corrective actions on time in case of delay.
3.2 The project plan and evaluation of the plan

The figure 3 shows the current progress of the project according to the original plan. The gray colour of the bars in the Gantt Chart shows the actual progress of the project.

The figure 3 shows that most of the activities have been done according to the proposed schedule. In this point of the project, the Initial Report was done, the Progress Report was done also, and most of the literature related to the project has been chosen and it has been extensively reviewed. So the first part of the project was based in the search of information related to On-line decision trees and Face Detection techniques. This set of documents become now the base for the research and the main idea of each of them is explained in section 2. Now, this bank of papers is the core of the big set of information that was used to build the first prototypes explained in section 4.2. The documents are also part of the literature that must be reviewed according the small targets in the prototype based methodology explained in section 3.1.

Although according to the plan there is a delay of 4 days, it means that for the current date the Definition of requirements should be almost done, this activity can be finished completely based on the prototypes that were already built. More over, this prototypes increase the progress of the activities of Design of the solution and Implementation of the solution in 10% each.
<table>
<thead>
<tr>
<th>Target</th>
<th>Developed</th>
<th>Integrated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read a PGM Image and calculate its integral image</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Build a set of Haar-like features</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Apply a Haar-like feature to a PGM image</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Access to the webcam driver and be able to take screenshots and videos</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Transform the images taken from the webcam to PGM Images</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Read the features from a already tested dataset of Haar-like features</td>
<td>In Progress</td>
<td>No</td>
</tr>
<tr>
<td>Build a prototype of user interface according to the requirements for building the forest and control the webcam</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Integrate the user interface with the webcam test</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Apply Haar-like features over the images taken from the webcam</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Build a ID3 algorithm</td>
<td>In Progress</td>
<td>No</td>
</tr>
<tr>
<td>Build a Random Forest</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Define, design and develop the transport interface between the data-sources of the stream of data and the forest</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Build a module able to replace the bagging process of the Random forest by calculating the number of times a message sample should be sent to the forest</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Build an adapter to read information from files, transform it to the format the forest requires and send it to the forest interface.</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Build an adapter to read information video source such as the webcam, transform it to the format the forest requires and send it to the forest interface.</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Build a module able to calculate Entropy measure and Gini Index that will be used as split criterion for the nodes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Build a module to calculate the gain respect to a test, based on Entropy measure or Gini index</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Modify the nodes of the forest to work with the new the split criterion according to the parameters that were set up</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Adapt the forest to work with Haar-like features to recognize faces.</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Figure 2: Master list of Targets
With this in mind we are ready to follow the schedule of the project and start with the activity related to the definition of the requirements of the system.

4 Progress

This section will show the progress of the dissertation by explaining first some design decisions based on quality attributes that have been taken during the process, then a detailed explanation of the prototypes that have been done, and finally a discussion about future important decisions should be done to continue based on the learning model to implement.

4.1 Quality attributes

Performance - Execution time: The nature of the model related to the on-line learning process makes this quality attribute the most relevant to the project. The model should be able to work with streams of data that continuously will arrive waiting to be processed. So the model should be fast enough to deal with this kind of characteristic. In the videos case, 15 frames per second for will be generated from the video source with resolutions up to 640x480 pixels.

Efficiency: The provided software must use hardware resources in reasonable way without overloading the use of memory, disk space or CPU resources. Again, because of the nature of the on-line learning model, the software must be able to deal with infinity size data sources without generate overloads in the use of hardware resources while it is being executed.
Portability: Based on the main goal of the project related to the software could be used as a teaching tool, the final product will be installed in many types of machines with different operative systems. So is necessary to build a program easy to install and execute in several types of machines.

Traceability: Is important to leave a trace of failures in case some unexpected data or situation occurs during the execution of the software in order to trace the error an recover the execution of the program.

Security: No relevant to this project in terms of confidentiality of data, data integrity and data availability.

Fault tolerance: This is not a relevant issue for the final product in this project. It is not necessary that the system could recover automatically in case of a fail, and a manual procedure should be done in order to detect the error that caused the failure and to start again the process.

Based on previous quality attributes, the programming language selected to build the software was Java. Due to its Oriented Object Programming paradigm, many architectural, design patterns and best practices rules can be applied to reach the performance and efficiency attributes. Moreover actual hardware architectures based on multi-core CPUs allow Java to take advantage of its multi-threading and concurrency tools. The portability attribute is in some way “natural” for Java codes due to that responsibility is already solved by the Java Virtual Machine or in general by the Java Runtime Environment installed in each machine. So the same binary class executable files will run in a machine with Java installed. The traceability will be solved by doing a right exception handling in design and development time in order to leave a trace of errors with enough information to find what is going wrong in any given moment. With Java is also feasible to find several open source APIs available on the web, so already working code can be use as modules of the main application.

4.2 Prototypes

This section explains and shows each one of the prototypes that were built according to the research and literature review.

4.2.1 PGM Image

The images type PGM are an interesting format of images. PGM is the acronym of “Portable Gray Map”, and is a format that was designed to make easier the
A PGM image represents the image in grayscale. Moreover, many training datasets of faces are in PGM format. Figure 4a displays a face in PGM format taken from [5]. Figure 4b shows an example of an enlarged by eight times image face of 19x19 pixels, stored with PGM format. So in order to use them and due to the previous ideas, PGM format was chosen as the input format of the images that would be processed by the model. So the first prototype was to build a code or find an API able to read and write PGM images. The result of the prototype is shown in figure 4c.

4.2.2 Working with Haar-like features and Integral Image

One of the main concepts for face detection using the technique proposed by [14], is the concept of Haar-like features. The main idea of this definition is to try to detect objects by using simple rectangles that laid in the right position in an image and according to the amount of light each rectangle surrounds, the model would be able to detect common characteristics in faces. For example, in frontal faces images, generally the region in the image of the eyes is darker than the region in the image for the bridge of the nose. By combining three aligned rectangles over that region of the face and choosing a right threshold of how much light the region should display the model will be able to say if potentially that area corresponds to the eyes and nose in a face.

Due to the sum of the values of the pixels in the rectangles has to be done many times.
times in a single image, and the rectangles that describes the features should be moved around the whole image and with different sizes [14] propose the use of the Integral Image to calculate this sum in a very fast way. Those previous movements and changes in size of a feature, tend to increase the number of features to numbers that are bigger than the number of pixels in an image. So the calculation of the sum of the values of pixels in a single region becomes a critical part talking about the performance of the algorithm in terms on how fast this sum could be calculated. The general concept of the Integral Image is that the new value of a pixel of the image is the sum of values of the pixels that are to the left and top of the pixel. After doing this transformation to the image, calculate the sum of the values of the pixels in a region becomes in a single arithmetic operation between the values of the corners of the rectangle that surrounds the area. With this strategy the execution time for calculating the sum of the pixels of an area will be independent of the size of the area. A more detailed explanation of this concept is shown as well in [14].

With all the related information and research about the previous concepts, a new prototype was build in order to apply Haar-like features to an image. Figures 5c, 5d and 5e show the result of applying the Haar-like feature in figure 5a over the image in figure 5b. For all the examples a horizontal 3 pixels wide light line detector was used and the length of the line was changed with 3, 7 and 15 pixels. This example illustrates how a simple feature is able to detect some characteristics of the image. In the same way vertical light lines can be detected, horizontal and vertical dark and light lines with different length and thickness, and horizontal and vertical borders of different lengths from dark to light areas and vice versa.

4.2.3 Accessing the Webcam Driver

Based on the goal of the project related to process video to detect faces, a new prototype was set up in order to access the driver of the traditional webcams in common laptops or desktops. After some hours of investigation and several probes of concept, the research leads me to choose sarxos webcam-capture API. All the documentation related to the API can be found in [6]. The objective of this prototype consisted in access basic functions of the webcam, such as take a snapshot using the webcam, open and close the video driver, display the video in a panel, take snapshot from the video and save the image in different formats, modify the resolution of the video, change the image to gray-scale, take several snapshots of images while the video is being displayed, calculate the frames per second in a video source, etc. A screen-shot of this prototype is shown in figure 6.
Figure 5: Face image after applying Haar features to detect horizontal light lines of 3 pixels wide.

Figure 6: Screenshot of Webcam test.
4.2.4 The On-line Random Forest

According to the information of the papers, and the conclusions collected in the meetings some requirements over the configuration of the model have arose.

- From the 2nd meeting with the supervisor, we agree that the software should be able to save the state of the random forest and to be able to retrieve it. This only implies to store the information of the nodes, the statistics, the initial configuration or parameters of the forest, but no any of the samples it has processed. This to maintain the concept of on-line learning instead of incremental learning as was explain in section 2.

- The parameters of the forest correspond to the initial configuration of the model and must be set up at the beginning of the process. According to [9] these parameters are the number of trees in the forest, the depth of the trees, the minimum number of samples that should cross through a node of the tree before this node can be split or divided into two new node, and a measure of minimum value of gain in each node that means in general terms if the information that has passed through a node has been good enough to split the node.

- From the 4th meeting with the supervisor, we agree to have two operation modes for the model. The first one is when we are telling the forest that the arriving samples are labelled. In this case the forest is able to learn (that means collect statistics and modifies its internal values) and to classify the data (in this case to detect faces in the images). The second operation mode is when the arriving samples are not labelled, in this case the model will only classify the information without modifying its internal state (in this case it will only try to detect faces).

Based on previous bullet points, a prototype with a model of the panels in the user interface that will control the configuration of the on-line random forest was done. The figure 7 shows this prototype.

4.2.5 The First version of the User Interface

With the previous prototypes already working, a full integrated version that shows all the functionalities in a single program was done. The main goal of building this prototype was to ensure that all previous prototypes are compatibles among themselves and to fix all possible bugs that appeared in the integration test. So finally here we have a first integrated version with several features: First, the prototype is able to control the webcam through a panel with turn-on and turn-off, take a snapshot, and Start/Stop stream snapshots
buttons. Second, the panel of the images that show the video from the web-cam in gray-scale with the frames per second configuration of the video, the generated PGM image when a snapshot is taken, and the resulting image when a Haar-like feature for horizontal borders detector is applied over the integral image of the previous snapshot. Third, the panel that will configure the random forest is attached as the main control panel of the program. The figure 8 shows the look and feel of the graphical user interface.

4.3 Open discussion about the On-line Random Forest

Talking about the On-line Decision Trees, [9] propose the use of Extremely Randomize Forest by choosing randomly the value of the thresholds for each random test function in each node according to the set of features that were chosen randomly as well. It is interesting to measure how this decision impacts the performance of the model and compare it with a traditional Random Forest in which threshold value is calculated according to a quality function. In the case of Haar-like features for faces detection, this threshold is already calculated as result of the Adaboost process explained in [14] to rank this features.

Another point of discussion is the one related to how to choose a right quality measure to be used as splitting node criterion. [9] use the Entropy measure or
Gini index but there is not enough information of how to choose between the two of them for a given case. In this point of the project we can think in having a new parameter related to the selection of the quality measurement, so the user could select from a list the desired quality measure and then he will be able to compare results between many experiments.

5 Next Steps and Conclusions

As it was said in previous sections the research has been divided into two parts: the on-line learning model and the image treatment problem. The literature review and the investigation has been done over the two main blocks in parallel and good amount of information has been collected and analysed. The development of the first integrated prototype was done over the image treatment block based on [14] and [15] mainly. It showed interesting results about how an image taken from the web-cam can be treated up until the point before it reach the model. Now the project must focus in the development of the prototypes for the machine learning model based on the Targets master list, and by taking into account to build a very strong interface between the image processing block and the random forest input, due to its compatibility becomes a very important module for the complete software.

Talking about the design of the solution, it should contemplate the definition of the format for the features in the examples. With this characteristic the model
could be used not only to solve face detection problems but for tracking another types of objects.

Moreover, the documentation process should be done at the same time in order to complete the design, development and testing processes, and avoid to have delays for the project. This still fits with the research methods proposed in section 3.

In general, machine learning models and classifiers are based on mathematics procedures and functions. The On-line random forest is not excluded of these types of models, so a depth understanding of the maths related to the process must be done in order to complete the design of the software product and develop a good solution able to reach its general goal.

Because of the nature of the prototypes based methodology, many small programs could be left in a repository of information, in order to be accessed and taken as based for future new projects of the School of Computer Science. For example, the first version of the forest could lead to develop new machine learning models over it, or the module of apply Haar-like features over an image could be used for computer vision classes and projects.

Is expected as well that the number of hours assigned to the project must be increased in order to have greater daily progress than actual one. Is expected as well that new doubts and enquiries will come, so it is highly probable that more meetings should be schedule with the mentor to discuss these new issues and finally reach new ideas that could lead to the solution of the problems.

6 References


