Real Time Drunkness Analysis Through Games Using Artificial Neural Networks

Audrey ROBINEL
LAMIA, Université Antilles-Guyane
Pointe-à-Pitre, France
audrey.robinel@univ-ag.fr

Didier PUZENAT
LAMIA, Université Antilles-Guyane
Pointe-à-Pitre, France
didier.puzenat@univ-ag.fr

***Abstract*—In this paper, we describe a blood alcohol content estimation prototype based on a comportment analysis performed by artificial neural networks. We asked to subjects that had drunk alcohol to play a video-game after having measured their blood alcohol content with a breathalyser. A racing game was modified so that it could provide various data related to the use of the controls by the player. Using the collected data, we trained our neural network in order to be able to determine whether or not the subject had exceeded a blood alcohol content threshold. We also succeeded in estimating this blood alcohol content with a mean error of 0.1g/l. We could perform those estimations independently of the track played among the two ones used. It was also performed in "real time", e.g., using only the data collected within the last 10 seconds of playing.**

Keywords-User interfaces; Games; Neural network applications; Cognitive sciences; Psychology; Human factors.

I. INTRODUCTION

Driving under influence is not only forbidden, but also dangerous, being a major cause of accidents. Devices for measuring the blood alcohol concentration of a driver have thus been developed. The current and most common approach for a driver is to use a "breathalyser" [3]. Such a device measures the amount of alcohol contained in the exhausted air blown by the subject into the device by using a chemical reaction. The blood alcohol content is computed from this value. Law enforcement class devices are very reliable but expensive. Most of the low cost devices sold to regular citizen are unreliable and provide erroneous measures, and require regular calibration or parts replacement (we had a hard time finding a good device, which came at a higher price). Furthermore, if ever a good device is used, the driver must not forget to test himself before driving, which is very likely to happen for a drunk person.

But what if the car could estimate whether or not the driver is in condition for driving? Alcohol affects the driver and thus creates dangerous behavior on the road. This means that if we monitor the "low level" characteristics of the car controls used, we should be able to correlate the subject's blood alcohol content to his driving ability [9]. Our final goal is to create a natural interface embedded in the vehicle that would use the data collected by the car's computer to analyze the driver's behavior and actions to provide a diagnostic to the driver and warn him before he becomes really dangerous. For now, we present a prototype in order to demonstrate the feasibility of the concept. We will not use a real car but a video game, and gather data from the use of the game controls. In order to fulfill the objectives (detailed in the Section II), we had to instrument a game (Section III-A). We chose SuperTuxKart, a racing game for the reasons presented in the Section III-C. Following the measuring protocol described in Section III-E, we collected low level data (Section III-D) of 120 game runs. Using all those examples, and a K-fold cross-validation derived algorithm (Section IV-A), we measured the ability of the network to detect whether or not a subject's blood alcohol content exceed a threshold in Section V. We thereafter used the system to estimate the value of the blood alcohol content, in Section VI. We then showed how the system could perform this task independently of the track played, in Section VII. Subsequently, after defining what we call a "real time estimation" in Section VIII, we demonstrate the ability of our prototype to perform such an estimation in Section VIII. To conclude, we present in Section IX a global interpretation of the results and the perspectives opened.

II. OBJECTIVES

Our main goal is to demonstrate that using the data collected from an instrumented game played by a subject, a trained artificial neural network is able to determine whether or not a subject's blood alcohol content exceeds a fixed value, and even estimate the blood alcohol content value. Our secondary goal is to determine if the neural network is able to provide those estimations independently of the track or if it must be used on predefined tracks. Our last goal for the prototype is to perform in real time (we have a specific definition, presented in Section VIII), at any moment of the game run.

III. OUR SYSTEM

A. *"Instrumenting" a game*

We instrumented a video game that provided many low level measurements. By "Instrumenting" we mean taking measurements of player actions on input devices such as a joystick or a steering wheel. This devices response being progressive rather than binary, we can collect the state of the device and obtain a large set of "low level" data about the intensity of the subject's actions on the controls. We

 ISBN: 978-1-61208-117-5

gather these data continuously in time, which enables us to compute the evolution of some parameters over time.

B. Artificial neural networks

Playing a video game implies mobilizing many skills. Depending on the game, it may be more or less necessary to think, to act quickly, to act with precision, etc. Thus, instrumenting a game should give valuable information on the player. However, the point is to know what to measure and how to go from the measurements to a conclusion. The prototype presented in this paper uses artificial neural networks to select and combine relevant measures to assess the player, and is based on previous works [9]. Once trained on a representative population, a dedicated neural network will be able to evaluate a new player. Our approach eliminates the need to explicit the significance of measures for a given goal, therefore it becomes possible to make "low level measurements", e.g., the frequency of corrections on the steering wheel during a racing car simulation.

Among all artificial neural network models, for all our present developments, we have chosen a classical Multi-Layer Perceptron (MLP) with a back-propagation learning algorithm [2]. The neural network has been implemented with the free open source *Fast Artificial Neural Network Library* (FANN) [8]. FANN is a C library facilitating the use of the developed neural network within the instrumented video game (or interface, embedded computer, etc.), e.g., to perform a real-time evaluation.

C. The chosen game

We chose a racing game because of the dynamic aspect of such games. Since races are played on tracks, it implies a predefined path. The subject has then a restricted freedom, which turns the runs (played games) into reproducible and comparable tasks, enabling us to create an uniform examples set. Furthermore, it can be controlled with a steering wheel [7], which is an intuitive device for controlling a car as presented in Figure 1.

We selected SuperTuxKart [5] because it is extremely easy to handle. Anyone who could drive a car, even if unfamiliar with computers can perform the tests. It has very simple tracks, on which it is impossible to loose one's way (every subject were able to finish the races). Tanks to the open source license (GNU GPL), we could edit the code to include our data collecting library.

D. The collected data

For each full lap of the circuit, a vector of 5 components is preserved and will feed the artificial neural network. Those components are (i) the average number of steps per rotation (the steering wheel is analogical), (ii) the number of accidents (collisions and falls), (iii) the total number of actions of the steering wheel, and (iv) the number of changes in direction (e.g., the driver was turning left and is now turning right). We define a "run" as a session played by a subject on a full track lap. When a subject plays a run, the system generates XML files containing all the race measurements. Each run then provides us an example (in the real time context, each run will provides us with multiple examples, more on that in Section VIII) that can be used for our problematics. Each subject does multiple runs, and the combination of all runs is our full examples set. For each run, we also measured the subject's blood alcohol content with a breathalyser.

Figure 1. The prototype during a run

E. The measuring and data collection protocol

In order to obtain a coherent example set, we used the same protocol for each subject. We had selected two tracks ("skyline" and "snowmountain", see Figure 2) for our data collection campaign, and every subject played on both tracks. Each subject played being sober on skyline, then on snowmountain, so that he could get used to the game controls. We then repeated it, still with a 0 g/l blood alcohol content, which provided us sober playing examples. Then the subject drank some alcohol, and we waited 20 minutes. After this delay, the subject's blood alcohol content were measured with a breathalyser (model AL7000 from Alcopass [1]). Two other games were then played (same tracks). The operation were repeated 15 to 20 minutes later, and for some subjects another time 15 to 20 minutes later again. This provided us with 8 to 10 examples per subject, two with a blood alcohol content of 0 g/l, and the other ones with various values greater than 0 g/l and lesser than 1 g/l.

 ISBN: 978-1-61208-117-5

At last, we obtained a 120 examples set containing game statistics distributed between 14 different subjects (in order to avoid subject-specific results) with various blood alcohol content, between 0 and 1 g/l. We ensured that, after the experiment, no subject drove a real car while having a blood alcohol content superior to 0 g/l.

Figure 2. The skyline track on the left, snowmountain on the right.

IV. Experimental protocol

Before presenting our results, we will explain the methodology used to obtain the numbers that will be showed later. Neural networks are trained on a learning set and then used in generalization, on examples that are not in the learning set. Generalization can be done either in production, or in order to test the network as presented in Section IV-A. We used a method derived from K-fold cross validation [6] to evaluate the generalization performance of the network. This provided the success rate and the average error of the network for each problematic.

A. The K-fold cross-validation derived algorithm

Our algorithm was the following : at start, we take an example base (in this example, it has 100 examples). We initialize the algorithm by setting a loop index `i`, variable `s` (successes) and variable `f` (failures) to 0. We then split our example base in two: examples 0, 1 and 2, constitute the generalization set and examples from 3 to 99 form the learning set. We train the network on the learning set, and test it on the generalization set. We evaluate the network's answer and increment `s` or `f` in case of success or failure. We then increment `i`, and start again the same with generalization set containing examples 1, 2 and 3, and learning set containing examples 4 to 99 and example 0. We keep doing that until `i` reaches 99 (in that case, the generalization set is constitute of examples 99, 0 and 1, while learning set is the rest). Thus, on each pass, we trained the network on a subset of examples, and tested it on another disjoint subset. Therefore, the network was *never* tested in generalization on any example used in the learning set. We obtain in the end the total number of successes and failures. We can then compute the success rate of the neural network as accurately as possible.

B. Determination of right and wrong network answers

We will now explain how we decide when the neural network response is right or wrong. When the network returns a value for an example, we can compare it to the real value, e.g., the estimated blood alcohol content and the measured value. We define ϵ as the distance between the expected value v and the neural network output o: $\epsilon = \|v - o\|$. We define the total cumulated error as the sum of all ϵ obtained for each example. We then obtain the average error by dividing this sum by the total number of examples tested.

We will set a maximal tolerated value (ϵ_{max}) for ϵ: if ϵ exceed ϵ_{max}, the network answer will be considered wrong (a failure). It will be right (a success) otherwise. A value of ϵ_{max} is defined for each experiment.

From now on, when we write that the success rate in generalization is x% for a value of epsilon, it means that for x% of the tested examples, the error of the network were less than ϵ_{max}. We always give the success rate in generalization, and never the learning rate.

	(i) skyline	(ii) snow	(iii) both tracks
successes	129	121	233
failures	3	5	25
success rate (%)	97.73	96.03	90.31
mean error (g/l)	0.0587	0.0972	0.1873

Table I
THRESHOLD EXCEEDING DETERMINATION

V. Determination of threshold exceeding

We first tried to determine if the neural network is able to detect when a subject's blood alcohol content exceeds a fixed threshold of 0.4 g/l. When the subject's measured blood alcohol content was over 0.4 g/l, the network was expected to return a value of 1 on the output, and a value of 0 otherwise.

We present in Table I the success rate and mean error of the network when using an example set of runs played on (i) skyline, (ii) snowmountain and (iii) both tracks.

This first experiment is the easiest for the network, and offers the best success rates, over 95% of good determinations. The results are similar for skyline and snowmountain, while they slightly decrease when we use both tracks in the same base (it will be studied further in section VII).

VI. Estimation of blood alcohol content

This experiment was conducted in order to see if the neural network could give an estimation of the subject's blood alcohol content. The network is expected to return (as output value) the subject's measured blood alcohol content, between 0 g/l and 1 g/l.

We note in Tables II and III lower success rates than in the previous experiments (especially for lower values of ϵ). However, this was expected. As we try to return an accurate value, we define a lower tolerated error (ϵ_{max}). This indeed decreases the success rate.

 ISBN: 978-1-61208-117-5

ϵ	successes	failures	success rate
0.10	85	47	64.39%
0.12	102	30	77.27%
0.14	108	24	81.82%
0.15	112	20	84.85%
0.16	114	16	87.88%
0.18	119	13	90.15%
0.20	122	10	92.42%

Table II
BLOOD ALCOHOL CONTENT ESTIMATION ON SKYLINE, IN FUNCTION OF ϵ. THE AVERAGE ERROR IS 0.083860 G/L.

ϵ	successes	failures	success rate
0.10	70	53	56.91%
0.12	71	52	57.72%
0.14	78	45	63.41%
0.15	81	42	65.85%
0.16	84	39	68.29%
0.18	94	29	76.42%
0.20	98	25	79.67%

Table III
BLOOD ALCOHOL CONTENT ESTIMATION ON SNOWMOUNTAIN, IN FUNCTION OF ϵ. THE AVERAGE ERROR IS 0.1158 G/L.

VII. TRACK INDEPENDENT ESTIMATIONS

In order to demonstrate that the system is able to estimate the blood alcohol content of the subject independently of the race track played, we conducted two other experiments. The first one, in Section VII-A, focuses on a network trained on a base containing examples from both tracks. The second one, presented in Section VII-B, shows how the network behaves when trained on a learning base containing examples from one track and is tested on a generalization base made of examples from another track.

A. Testing the network with examples from both circuits

We will start with the mixed tracks data sets. The example set includes data gathered from game runs played on the two tracks, skyline and snowmountain. Using K-fold cross validation, we measured the ability of the network do estimate the blood alcohol content of the subjects after being trained on such a base.

The results presented in the Table IV does not show a significant changes in neither the success rate nor the mean error of the network. This tends to indicate that the prototype is able to estimate the blood alcohol content without the need of a specific portion of road, implying that we could construct examples set from various road sections or tracks.

variable	value
successes	77
failures	22
success rate (%)	77.78
mean error (g/l)	0.0587

Table IV
RESULTS OF THE MIXED EXAMPLE SET FOR ϵ=0.20

values of epsilon	A : sky $\rightarrow$ snow	B : snow $\rightarrow$ sky
Success rate for ϵ=0.15	26.47%	40.62%
Success rate for ϵ=0.20	82.35%	62.62%
Success rate for ϵ=0.25	88.24%	71.88%
Average error (g/l)	0.185407	0.226358

Table V
TRACK INDEPENDENCE TEST RESULTS.

B. Training the network with examples from one track, and testing it with examples from another track

After this, we trained the network on the examples from the first track, and then checked it's generalization results on the examples of the second track. This configuration ensures that the second set of examples (games played on the second track) is unknown to the network.

This time, we created two examples sets instead of one: (i) the "sky set" contains all the game runs that were played on the "skyline" track; (ii) the "snow set" contains all the game runs that were played on the track "snowmountain". We will try two configurations: in the first one (A) we will use the sky set as training set and snow set as generalization set. In the second one (B), the the snow set as training and sky set as the generalization set. The results obtained for different values of ϵ are presented in Table V. On contrary of other experiments, we did not use the k-validation algorithm, since we had two distinct sets.

If we consider low values of ϵ, the success rate is quite low. We had to increase ϵ in order to maintain a success rate comparable to the previous ones. The obtained ϵ reaches really important max values, with in the worst case 0.25 g/l. But despite the fact that the mean error also increased, the network could still perform valuable estimations. This tends to indicate the ability of the prototype to be used to estimate the blood alcohol content independently of the track. Furthermore, this shows how our neural network is able to cope with data gathered on an unknown portion of track. This avoids the need to either create an exhaustive learning set containing all possible situations or to use the system in the same environment that were used for training. Of course it is far from being perfect as, in these conditions, the prototype is less accurate with much lower success rates in some cases (experiment B, for example) and significantly higher average errors (reaching 0.22 g/l for experiment B). But the prototype tends to demonstrate that it is able to generalize from one track to another.

Of course, we could have more meaningful results with more tracks, but the game did not provided enough tracks that met our requirements, detailed in section III-C.

 ISBN: 978-1-61208-117-5

VIII. REAL TIME ESTIMATIONS

We will here demonstrate the ability of the system to perform "real time" estimations of the measured blood alcohol content.

A. Definition of "real time estimation"

We define as "real time" an estimation done using the data gathered within the last elapsed seconds. We define n as the length in seconds of the interval of play used. An n seconds real time estimation means that the estimation is performed using the data of n seconds of race instead of using the full length of the race. The interval of play used for this determination will be called a "window" and n is defined as the size of the window.

In order to obtain comparable data, we normalize the variables by dividing their values by n. By this mean we also ensure that the trained network can estimate the blood alcohol content for various values of n, e.g., it can be used indifferently to provide a real time estimation based on 10 or 20 seconds.

We kept using the examples from both tracks for our real time experiments.

B. Protocol

We checked here if the network is able to perform a real time determination based on 10, 15, 20, 30, 40 and 50 s windows. In order to study the impact of n on the results, we kept all other parameters equal. In order to do that, when cutting the runs in order to obtain n seconds examples, we only kept the first example, so that the number of generated examples remained constant. We thus used data from the interval $[0, n]$ in order to perform the n seconds real time estimation.

C. Impact of n on the success rate and the mean error

We present in the Figure 3 the evolution of the success rate (as always in generalization) and in the Figure 4 the variation of the mean error in function of the value of n. The success rate increases with the length of the interval, while the mean error decreases. Increasing the window size improves the quality of the examples, which indeed enhances the results.

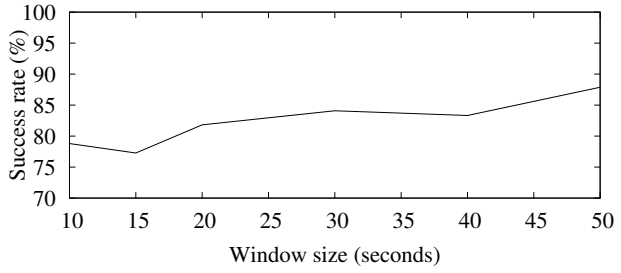

Figure 3. Evolution of the success rate in function of the window size

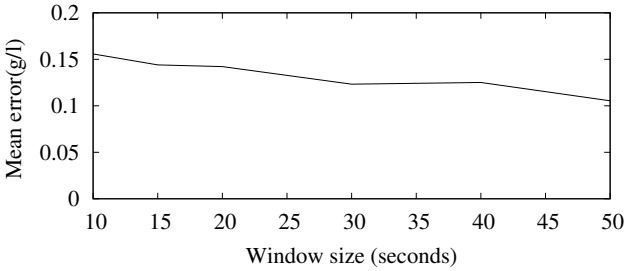

Figure 4. Network mean error as a function of the window size

	successes	failures	success rate	mean error
10s	104	28	78.79	0.155820
15s	102	30	77.27	0.143969
20s	108	24	81.82	0.142155
30s	111	21	84.09	0.123291
40s	110	22	83.33	0.125081
50s	116	16	87.88	0.105408

Table VI
REAL TIME BLOOD ALCOHOL CONTENT ESTIMATION ON A MIXED SET (CONTAINING BOTH TRACKS), FOR A VALUE OF ϵ OF 0.2 FOR DIFFERENT VALUES OF THE WINDOW SIZE (n).

D. Other real time results using all the available windows

The previously presented results focus on the impact of the window size on the success rate and mean error. Furthermore, we obtained similar results when using all the available windows. Indeed, when we split our 60 seconds long runs in windows of 20 seconds, we obtain 3 examples. In the previous experiments we kept only the first one. Keeping all the obtained examples gave similar results. This also confirms the track independent estimations capability of the network, since the windows [0-20], [20-40] and [40-60], as an example, are 3 different sections of the track. Indeed, only one lap is performed. Thus, the subject never drives two times on the same section during a run.
We also obtained comparable results for the threshold exceeding detection (with better success rates), which we do not present here as they are quite similar (slightly better) than the ones detailed earlier in Section VIII-C.

IX. CONCLUSION

We have demonstrated that it is possible not only to detect threshold exceeding but also to estimate the blood alcohol content of a subject. These estimations were achieved independently of the track. Considering that it is hardly feasible to create an exhaustive learning base containing all possible situations, it makes possible to only fill the learning base with the most common situation patterns. We then rely on the generalization capability of the neural network to provide estimations on new patterns. We also managed to obtain a real time estimation, and still independently of the track played. The prototype is able to provide an estimation

 ISBN: 978-1-61208-117-5

using a 10 seconds window (e.g., after only 10 seconds of playing), and to improve the reliability and the accuracy of the estimation by using a larger window. It should thus be possible to conceive an embedded interface that would be able to provide estimations on the driver after only a short while, and enhance these estimations after more time. The fact that the estimations are not bound to a fixed length of time allows many configurations.

Furthermore, such a system could re-evaluate the driver continuously, and adapt the estimations to a changing situation. Indeed, someone who just drunk some alcohol may feel perfectly able to drive, but as alcohol is absorbed in blood, the side effects will alter his driving skills up to the point where driving becomes dangerous. An embedded device based on our concept could detect the evolution of the driver and warn him before he becomes really dangerous.

X. Limits and drawbacks

While we could solve the problematics, we had to degrade the accuracy of the system (ϵ) in some case in order to maintain acceptable success rates. As for track independence, we note in some cases a significant drop on success rate and an increase of the average error. But this experiment was performed to see how the network would cope with unknown tracks. Considering that we only had two tracks, only one remained for training. As shown, if more accuracy is required, we can use several tracks. Furthermore, if this system were to be used in a production environment, we would try to create a base containing a representative set of the situations that can be encountered.

We also noted the much higher average error for real time estimations when using small windows such as 10 s, but it was expected. It indeed was intended to show the neural network behavior in an extreme case, setting the lower bound of the window size.

In some cases we reached an accuracy of 0.11 g/l, which is close to the 0.1 g/l of the breathalyser. Owing to the fact that we calibrated our system with this breathalyser, improving the accuracy can only be achieved by using a better breathalyser. On some cases though, the accuracy were quite far from the breathalyser's. As the present prototype was created as a proof of concept, it was not intended to be on par the breathalyser in every experiment.

XI. Perspectives

We demonstrated the ability of the prototype to perform meaningful estimations even in a simple environment. Those results will be the base of our next works: we plan to improve and test the system on a more realistic environment in so that it might be embedded in a vehicle later on. The prototype demonstrated the viability of the concept in a simplified environment, we will now take it to a more complex virtual world before trying to use it in the real world. For now, much work is left in order to achieve this goal: we will have to find what data to gather from a more realistic system, or among the numerous data monitored by modern cars, and also a way to accurately measure those data. Considering the amount of completely different situations in a real driving environment (the "game play" of SuperTuxKart is very simplified), we will have to use either much more inputs, or other sorts of neural networks such as mixture of neural experts [4] to cut down the problematics in smaller and more simple ones, considering how different it is to drive on the highway and in congested urban areas.

No matter what neural network model we will use, it will anyway imply much more work to select meaningful inputs for the network, and probably require the use of parallel computing in order to explore configurations until we obtain the optimal ones for a given problem. Hopefully, generalization will always be possible on a low cost, low power consumption, and low computing power system, making the software easily usable in embedded devices.

Furthermore, we foresee that increasing the complexity of the environment and the amount of measured characteristics might enable us to perform not only more accurate estimations, but also many other estimations, such as detection of tiredness, attention drop, or the use of driving impairing drugs or medications. Those very similar problematics may use the same system with just another examples set.

References

[1] Alcopass. Alcopass AL7000 breathalyzer documentation., December 2010. http://www.alcopass.com/.

[2] C. M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.

[3] E. Bogen. Drunkenness, a quantitative study of acute alcoholic intoxication. *California and Western Medecine*, XXVI(6):778–783, 1927.

[4] P. Estvez, H. Paugam-Moisy, D. Puzenat, and M. Ugarte. A scalable parallel algorithm for training a hierarchical mixture of neural experts. *Parallel Computing*, 28(6):861–891, 2002.

[5] J. Henrichs, M. Gagnon, and C. Pelikan. SuperTuxKart, Free 3D kart racing game., December 2010. http://supertuxkart.sourceforge.net.

[6] Ron Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection, 1995.

[7] Logitech. Formula Vibration Feedback Wheel., December 2010. http://www.logitech.com/en-in/441/298.

[8] S. Nissen. Implementation of a fast artificial neural network library (FANN). Technical report, Department of Computer Science University of Copenhagen, October 2003.

[9] D. Puzenat and I. Verlut. Behavior analysis through games using artificial neural networks. In *Third International Conferences on Advances in Computer-Human Interactions*, pages 134–138, February 2010.

 ISBN: 978-1-61208-117-5