Neural Network Application for Event Detection in Hybrid Dynamical Systems

Stefanie Winkler[1*], Andreas Körner[1], Felix Breitenecker[1]

[1] Department of Analysis and Scientific Computing, Vienna University of Technology, Wiedner Haupstraße 8–10, 1040 Vienna, Austria; [*] *stefanie.winkler@tuwien.ac.at*

Abstract. This contribution investigates a feed-forward neural network approach for event detection in hybrid dynamical models. Machine learning algorithms are commonly used in software development. In recent years these approaches have also been increasingly applied in modelling and simulation of physical systems. A significant amount of these models use artificial neural networks. However, hybrid dynamical systems describe a combination of different methods to describe a continuous process, which experiences behavioural changes at discrete events. Accordingly, the models of such systems are based on a combination of discrete and continuous methods and are often illustrated as automaton. Based on these two areas an approach, to predict the event time of the discrete processes, is presented. The different required elements are defined and a general approach is outlined. The feasibility of this concept is examined on the basis of one examples. If the given imbalanced data is resampled, training can be successful. Unfortunately, even then, the generalised classification of events often does not work sufficiently. The evaluation of the approximation results of the discrete events in hybrid systems suggests that neural networks are not suitable to classify the system states with regard to the occurrence of an event. In the outlook we suggest an alternative approach to predict the event with neural networks.

Introduction

The goal of this contribution is to investigate the applicability of artificial neural networks in the event detection for hybrid dynamical systems. On the one hand, the use of neural networks has increased, especially in fields such as pattern recognition and software development. In recent years neural networks are also applied in different engineering applications. On the other hand, hybrid approaches present a possibility to model complex systems. A hybrid model benefits from combining the advantages of different methods. In the case of hybrid dynamical systems, a dynamical process, characterised by finite changes of the model description, is described. These changes are called discrete events. This contribution focuses on autonomous events where no external factors are involved initiating the event.

Based on the hybrid dynamical automaton, as discussed in [7, 9], a model for applying neural networks is formulated. Additionally, the basic structure of neural networks will be presented. A multi-layered neural network will be applied to substitute the event in the hybrid dynamical model. Combining continuous, discrete and machine learning approaches, a model will be defined.

For the hybrid system in the case study the bouncing ball is chosen. The state space of this examples is one-dimensional and the automaton of the simplified assumption consists of one dynamical process. The goal of the trained network will be to determine if a state vector is initiating an event or not. One of the challenges of this approach is to create a useful learning dataset. In hybrid systems, states without an event are more frequent than states where an event occurs.

1 Artificial Neural Networks

Neural networks and machine learning are a central part of modern technology. Various software packages in computer science and engineering apply machine learning methods. In system identification problems for example, neural networks are used to approximate suitable model structures. In the early 1950s the first neural network was introduced and consisted of so-called perceptrons [13]. The task of a perceptron is to test incoming signals against a predefined threshold, whereas

ARGESIM Report 59 (ISBN 978-3-901608-93-3), p 121-127, DOI: 10.11128/arep.59.a59017

exceeding this value results in 1 and otherwise in 0. Despite the application purpose, the structure of a neural network always consist of three different types of layers: the input layer, the hidden layer and the output layer.

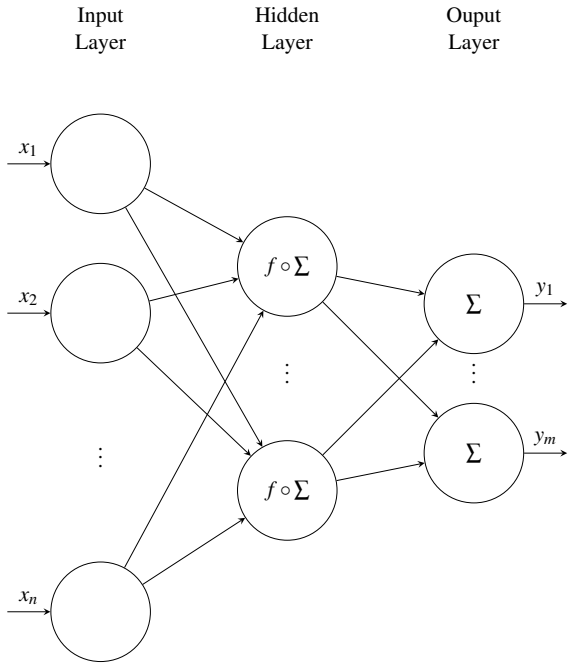


Figure 1: The structure of a basic artificial neural network, called multi-layer perceptron (MLP).

Each layer consists of a particular number of nodes, referred to as perceptrons or neurons. If every neuron of each layer is interconnected to every neuron in the following layer, the network is called *fully connected* [6]. Artificial n eural n etworks (ANN) c an b e characterised by the way the signals are processed. If a neural network is executed and the signal flows from the input layer through the hidden layer and exits at the output layer, the network structure is categorised as *feed-forward*. Such networks are often called *multi-layer perceptrons* (MLP). For most of the applications, a multi-layer feed-forward network is sufficient [8]. In Figure 1 a one layered feed-forward network is depicted. It consists of one input, one hidden and one output layer. The shape of the input and output layer is defined by the given inputs $x \in \mathbb{R}^n$ and the corresponding outputs $y \in \mathbb{R}^m$. In contrast, there is no predefined structure for the hidden layers. The number of hidden layers as well as the amount of neurons in these layers are arbitrary. The connections between the neurons in the different layers are called edges. Each edge carries an individual weight w_{ij}^l, where $l \in \mathbb{N}$ defines the target layer of the connection and i, j specify the end and starting neuron, respectively. The weighted sum of the inputs together with a bias b_j^l form the input of the neuron in most applications. The neurons of the hidden layer as well as the neurons of the output layer apply an activation function to the incoming signal.

In order to obtain a good performing neural network, it is necessary to determine the biases and weights of the network structure. The iterative process determining these parameters is called training. The complexity of the model, the number of parameters as well as the accessibility, size and range of the given datasets affect the success of the training. In this study only supervised training methods are considered. The iterative optimisation process uses the given dataset to improve the results of a predefined cost function, also referred to as loss function. At the beginning of the training the parameters are initialised randomly using a certain distribution. The loss function of a neural network is similar to the cost function in optimisation problems. It evaluates the performance of the given neural network and by minimising the loss function the results are improving. For classification networks, a possible loss function for convex optimizers is the support vector machine (SVM) loss or Hinge loss and the cross entropy loss, respectively. There are different methods to determine the network parameters w and b. Apart form the Gradient Descent, common optimisation algorithms are the Newton's Method, Conjugate Gradient, Quasi Newton Method, Levenberg Marquardt and Adams Algorithm. Depending on the size of the input-output dataset, an adequate training algorithm has to be chosen. The loss function and the optimization algorithm are embedded in a learning method. The most common method is the back-propagation.

2 Hybrid Dynamical Automaton

In applied mathematics and computer science the creation of modelling standards is an important aspect. The advantage of a modelling frameworks is that they can be applied not only to one unique problem description but to an entire class of problems. Based in related work the structural and graphical framework for hybrid dynamical systems is defined. Characterisations of different event formulations and transitions are given.

The complex structure of hybrid systems is commonly illustrated using an automaton [9, 4]. Automata are often applied to depict abstract machines as well as theoretical concepts in computer science, such as combinational logic. It supports structuring mathematical tasks and illustrating finite state machines [2]. It shows current states, update rules as well as transition conditions. By expanding the description possibilities of the update rules, the automata concept can be applied to model hybrid dynamical systems, as shown in Figure 2.

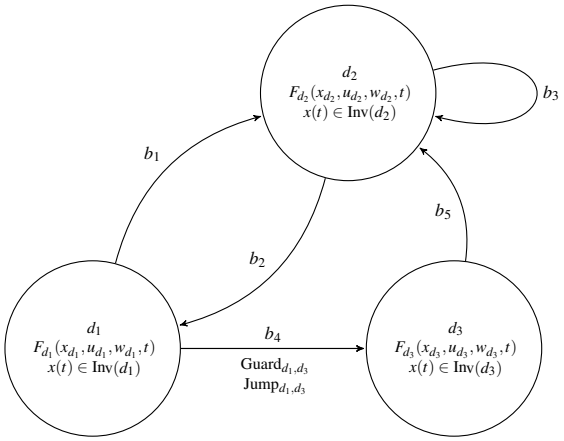


Figure 2: Conceptional structure of a hybrid automaton based on the work of [10].

In terms of layout, an automaton is an ambiguous description. It just characterises the basic structure of the model in a compact way. The nodes of the automaton describe different local dynamics whereas the connecting lines, called edges, define transition conditions. If fulfilled, the transition from one location (node) to the next is initiated.

Focusing on modelling of hybrid dynamical systems, a mathematical definition of the hybrid automaton is given [7, 4, 10].

Each node of the hybrid automata contains an individual model description, e.g. differential-algebraic equations. The state and time events, respectively, enable the transition from one node to the next. A characterisation of the different possible changes during the event, transitioning from one subsystem to the next, is formulated based on [10].

3 Artificial Hybrid Events

3.1 Idea & Concept

The illustration of the hybrid dynamical automaton in Figure 2 suggests a possible scenario to apply neural network for event detection in hybrid dynamical models. It proposes to apply neural networks to administrate the event handling.

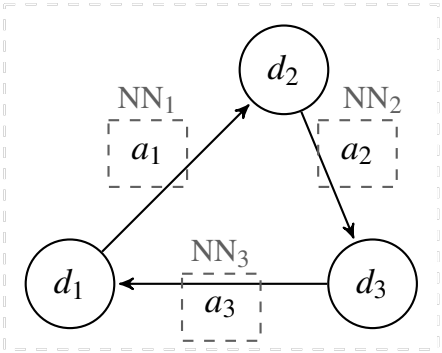


Figure 3: Three framework applications for including neural network concepts (red dashed rectangles) in modelling hybrid dynamical systems.

3.2 Network Approach

The definition is given independently from any specific feed-forward network structure and facilitates every event type. The characterisation of the event, as introduced in the hybrid dynamical automaton, requires at least three elements. The mapping Act is considered to be a-priori knowledge since the local processes and descriptions are accessible. The guard G defines the set of states initiating the events, whereas the jump map J executes the actual changes happening during the transition from one local description to the next.

Considering that the local dynamics are known, the jump relation is already implicitly defined. An investigation of the given dynamical descriptions determines a-priori if J describes a coordinate transform. A neural network approximation of that transform is unnecessary. Secondly, assuming a hybrid system with changing parameters and variables the dynamical behaviour of the system can be characterised mathematically and the jump relation is a consequence of the model description. Hence, the following framework focuses on the

approximation of the guard region.

The local descriptions and their state variables are known, whereas the guard region has to be determined. In contrast to both previously explained applications, the output data for the training set of the ANN has to be defined differently. There is no feasible output of the system, which can be directly applied for training. Instead the data has to be analysed and classified. Two different categories can be defined:

$$O_j = \begin{cases} 0, & \text{no event occurred at } x_j, \\ 1, & \text{an event occurred at } x_j. \end{cases}$$

With this classification a training set for the network A^b_{ae} can be given. The *input data* for training the neural network include

- the state vector of the system $x(t), t \in T \subset \mathbb{R}^+$,
- the given initial conditions x_0,
- input u and
- external variables w including any system parameters p.

Hence, the input can be given as $I(x_0, x, u, w, t)$. The *output data* for the training process is $O \in \{0,1\}^q, q \in \mathbb{N}$, where q depends on the number of classified state values included in the dataset. After embedding the trained network into the model structure, at each time step the state vector of the dynamical process is classified by the network and an event is initiated if the state vector is labelled 1.

Both previous definitions a ttempt t o approximate the relation between input and output values. In contrast, the approximation of the event guard represents a classification t ask. Hence, neural network structures such as EQL and HONN are not applicable instead MLP suited for classification tasks are required for this framework application. Therefore, datasets containing past state vectors classified w ith r egards t o t he occurance of an event can be very imbalanced [4]. As suggested in [5], resampling methods for training will be applied to level out the imbalance and its results will be compared.

4 Case Study

For the application of the approach a hybrid dynamical systems is chosen. The example is used to investigate the applicability and the experimental results are analysed and compared with the original data.

4.1 Example Description

he bouncing ball, as defined in [11] and [1], describes a ball, bouncing off the ground. The state variable of interest is the height over time t. The acting force in the observed system is the gravity, accelerating the ball to the ground. Thus, the ODE of the dynamical behaviour of the bouncing ball can be described as

$$\ddot{h}(t) = -g, \tag{1}$$

where g is the gravitational constant. Considering an initial height h_0 and velocity v_0, two state variables can be defined, namely the height $h(t) =: x_1(t)$ and the velocity $x_2(t) := \dot{x}_1(t)$. Hence, using the state vector $x = (x_1, x_2)^T \in \mathbb{R}^2$ equation (1) can be transformed into a state space description resulting in

$$\dot{x}(t) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} x(t) + \begin{pmatrix} 0 \\ -g \end{pmatrix}, \; x(0) = \begin{pmatrix} h_0 \\ v_0 \end{pmatrix}. \tag{2}$$

An advantage of the chosen academic example resides in the existence of an analytical solution given as

$$x(t) = \begin{pmatrix} -\frac{g}{2}t^2 + v_0 t + h_0 \\ -gt + v_0 \end{pmatrix}. \tag{3}$$

In Figure 4 (a), the height of the ball is depicted over time. Two different processes can be distinguished, the flying and falling phase of the ball, respectively and the bounce. The latter affects the behaviour of the ball and therefore defines the state event of the system.

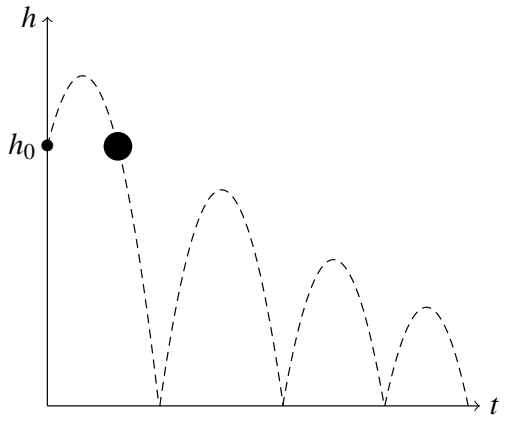
Due to the fact that the free fall phase is interrupted when the ball reaches the ground, the *event guard* $G_{d,d}$ can be defined. An event occurs if the state vector fulfils

$$x \in G_{d,d} := \left\{ x(t) \in \mathbb{R}^2 \colon x_1(t) = 0 \wedge x_2(t) \leq 0 \right\}. \tag{4}$$

This state event can be given in all three forms. The event function can be defined as $e_b(x) := x_1$ with $e_b(x) = 0$ initiates the event, whereas the threshold for the event can be given as $\Delta x_1 := 0$.

Model descriptions are often implying a certain degree of abstraction. In this application, the bounce is described by a simplified assumption without modelling the physical process in detail. When the ball hits the ground the behaviour of the ball is affected. The friction is realised as simple damping factor λ with $0 < \lambda < 1$.

(a)



(b)

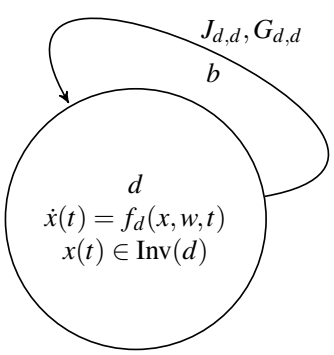


Figure 4: (a) Height of the ball over time. (b) Hybrid dynamical automaton of the bouncing ball.

The reflection of the ball results in inverting the velocity component. Hereby, the time delay due to any deformation work is neglected. Thus, the *jump map* $J_{d,d}$ at the event can be characterised by the linear transform

$$x(t^+) = J_{d,d}(x(t^-)) := \begin{pmatrix} 1 & 0 \\ 0 & -\lambda \end{pmatrix} x(t^-). \qquad (5)$$

4.2 Event Classification Results

Focusing on the problem of event detection with neural networks, possible datasets consist of different state values of the system and its label if an event occurred. A MLP is chosen as network structure. The framework application to replace the guard region in the hybrid model of the bouncing ball is given as

$$\begin{aligned} A_{\text{hae}} &= (\{d\}; \mathbb{R}^2; \{b\}; W_{\text{ae}}; E_{\text{ae}} = (d, b, G_{\text{ae}}, J_{d,d}, d); \text{Inv}; \text{Act}), \\ &\text{Act}(d) := F_d, \\ &G_{\text{ae}}(b) := A^b_{\text{ae}}. \end{aligned} \qquad (6)$$

where W, Inv and $J_{d,d}$ remain as defined in the hybrid automaton. The network definition does not depend on the application example. Therefore one network definition can be used for events of both systems, occurring during the bounce, the free fall and the pendulum phase.

The framework application for the network is given as

$$\begin{aligned} A^i_{\text{ae}} &= (4, I, O, N_i, \{u^{(l)}_j\}^{l=2,3,4}_{j=1,\ldots,n_l}, T),\ i \in \{d, r, p\}, \\ &u^{(l)}_j(z^{(l)}_j) = \sigma(z^{(l)}_j),\ l = 2,3,\ j = \{1,\ldots,n_l\}, \\ &u^{(4)}(z^{(4)}) = z^{(4)}, \\ &T = (M, A, C, S, V), \\ &C = \text{L}_{\text{cross}}, \end{aligned} \qquad (7)$$

where A is the Scaled Conjugate Gradient Algorithm, M is the back-propagation method and N_i depends on the application example. The input data consists of the state vector for various initial conditions and various points in time. For each data point the occurrence of an event is evaluated

$$O_i = \begin{cases} 0, & \text{no event at } I_i, \\ 1, & \text{event occurs at } I_i. \end{cases}$$

The resulting dataset $O_i \in \{0,1\}, i \in \mathbb{N}$ forms the output data for training. The size and structure of the state vector depends on the application example. Hence, the input for the training data is given as

$$I_i = \big(t_i, x_1(t_i), x_2(t_i), g\big)$$

The event classification is especially challenging. In most hybrid systems the occurrence of an event is rather rare. A dataset for the pendulum with various initial conditions results in $\approx 0.6\%$ events. This problem is often referred to as binary classification of imbalanced datasets [16, 5]. In [3] different resampling methods are discussed, to balance the dataset and enable successful classification. One method suggests to create synthetic observations drawn from a uniform distribution within the data of the small category. Due to the fact that the

Ex	$S = (\mathrm{E}/\mathrm{NoE})$	k	V
BB_1	98485/6010853	–	$[0.75, 0.15, 0.15]$
BB_2	49243/49243	$\frac{1}{2}$	$[0.75, 0.15, 0.15]$
BB_3	49243/49243	$\frac{1}{2}$	$[0.75, 0.15, 0.15]$

Ex	Corr. pos.	Corr. neg.	False. pos.	False. neg.
BB_1	1.2%	98.4%	0%	0.4%
BB_2	49.9%	50%	0%	0.1%
BB_3	50%	50%	0%	0.001%

Ex	$S = (\mathrm{E}/\mathrm{NoE})$	k	N
BB_1	98485/12021707	–	$[4, 30, 20, 1]$
BB_2	98485/98485	1	$[4, 30, 20, 1]$
BB_3	98485/147728	$\frac{3}{2}$	$[4, 30, 20, 1]$

Ex	Corr. pos.	Corr. neg.	False. pos.	False. neg.
BB_1	0.6%	49.6%	49.6%	0.2%
BB_2	49.9%	50%	0%	0.001%
BB_3	40%	0.6%	59.4%	0.001%

Table 1: Comparision of the classification MLP for both examples.

events are defined by physics this approach is not applicable. Another possibility is to remove a certain amount of random data from the majority class to balance the dataset. A parameter k is introduced to coordinate the balance in the training set. For each data point characterising an event, k data points classified as 'no event occurred' are added to the training set.

In Table 1 the results for all three events of the two examples are given. The classification of imbalanced data with regards to hybrid events is a challenge. Even though some of the values for wrong categorisation seem sufficiently small, the reason is not a good approximation but the ratio between the two categories.

For all three event types this approach is inapplicable. Even resampling the dataset only improves the network performance in training but the testing is still not feasible. In case of the bouncing ball example, not even the events in the training set can be classify correctly by the network. In case BB_3 at least the new data is classified correctly but the 50 events which have been classified false in training, stay incorrect.

5 Conclusion & Outlook

In the case of the event detection approach, the available descriptions of the local dynamics can be used to classify the available data. If the discrepancy between model output and system data increases an event can be identified. The gathered state vectors where the discrepancies are within a certain range, can be labelled 'no event'. This procedure results in an imbalanced dataset with two classes, the majority class 'no event' and the minority 'event'. The results listed in Table 1 show the results of a trained MLP for the classification of imbalanced binary datasets. If the data is resampled, as suggested in [15] and [3], the training can be successful. Unfortunately, even then, the generalised classification of events often does not work sufficiently. The case study confirms the findings in [14], that the traditional feed-forward neural network has difficulties to learn from imbalanced datasets [16]. To cope with imbalanced binary datasets in classification scenarios possible alternatives to neural networks are presented in [12]. Therefore, facilitating methods such as regression and projection into the third framework application is a future objective.

The focus of the framework was, to investigate different application scenarios for neural networks. In the case of classifying state vectors in 'event' and 'no event', the neural network performance was not sufficient. The classification experiments for both application examples have shown, that neural networks are not applicable for imbalanced datasets like this. Other methods might be more effective. Therefore, adding elements to the framework to enable an inclusion of other methods in the hybrid dynamical model is a future objective. In this context, also the preprocessing of the input data in neural network applications can be included in the framework definition. A future review of networks other than feed-forward, might require adding new elements. For example a recurrent network might require more descriptive elements than defined in the framework at the moment. If the dynamical description of each node involves several implicit formulations, it might be laborious or even impossible to formulate an explicit jump relation. In such cases, the measured output and input data of the different nodes could be used to train another network J^b_{ae} to replace the jump relation J in the model description. In addition, a combined framework application, to facilitate the replacement of events and local dynamics at the same time, can be one of the next steps.

References

[1] Berk Altın, Pegah Ojaghi, and Ricardo G. Sanfelice. A model predictive control framework for hybrid dynamical systems. *IFAC-PapersOnLine*, 51(20):128–133, 2018.

[2] Alberto Bemporad and Manfred Morari. Control of systems integrating logic, dynamics, and constraints. *Automatica*, 35(3):407–427, 1999.

[3] Evgeny Burnaev, Pavel Erofeev, and Artem Papanov. Influence of resampling on accuracy of imbalanced classification. *arXiv:1707.03905 [cs, stat]*, page 987521, 2015.

[4] Luca P. Carloni, Roberto Passerone, Alessandro Pinto, and Alberto L. Angiovanni-Vincentelli. Languages and tools for hybrid systems design. *Foundations and Trends® in Electronic Design Automation*, 1(1):1–193, 2006.

[5] Nitesh V. Chawla. Data mining for imbalanced datasets: An overview. In Oded Maimon and Lior Rokach, editors, *Data Mining and Knowledge Discovery Handbook*, pages 875–886. Springer US, 2010.

[6] Simon Haykin. *Neural networks and learning machines*. Prentice Hall, 3rd ed edition, 2009. OCLC: ocn237325326.

[7] Thomas A Henzinger. The theory of hybrid automata. In *Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science*, pages 278–292, 1996.

[8] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, January 1989.

[9] Andreas Körner, Stefanie Winkler, and Felix Breitenecker. Possibilities in state event modelling of hybrid systems. *SNE Simulation Notes Europe*, 28(3):109–111, 2018.

[10] Andreas Körner. *Mathematical Characterisation of State events in Hybrid Modelling*. phdthesis, Technische Universität Wien, 2015.

[11] Andreas Körner and Felix Breitenecker. State events and structural-dynamic systems: Definition of ARGESIM benchmark c21. *SNE Simulation Notes Europe*, 26(2):117–128, 2016.

[12] Sung-Chiang Lin, Yuan-chin I. Chang, and Wei-Ning Yang. Meta-learning for imbalanced data and classification ensemble in binary classification. *Neurocomputing*, 73(1):484–494, 2009.

[13] Warren S. McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[14] Yi L. Murphey, Hong Guo, and Lee A. Feldkamp. Neural learning from unbalanced data. *Applied Intelligence*, 21(2):117–128, 2004.

[15] Giang H. Nguyen, Abdesselam Bouzerdoum, and Son L. Phung. A supervised learning approach for imbalanced data sets. In *2008 19th International Conference on Pattern Recognition*, pages 1–4. IEEE, 2008.

[16] Jinghua Wang, Jane You, Qin Li, and Yong Xu. Extract minimum positive and maximum negative features for imbalanced binary classification. *Pattern Recognition*, 45(3):1136–1145, 2012.