Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia

COPPE UFRJ

BUS LINE TRAJECTORIES CLASSIFICATION USING WEIGHTLESS NEURAL NETWORKS

Raul Bezerra Barbosa

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Felipe Maia Galvão França
Diego Moreira de Araujo Carvalho

Rio de Janeiro
Junho de 2018

BUS LINE TRAJECTORIES CLASSIFICATION USING WEIGHTLESS NEURAL NETWORKS

Raul Bezerra Barbosa

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

RIO DE JANEIRO, RJ – BRASIL
JUNHO DE 2018

Barbosa, Raul Bezerra

Bus line trajectories classification using weightless neural networks/Raul Bezerra Barbosa. – Rio de Janeiro: UFRJ/COPPE, 2018.

XII, 50 p.: il.; $29,7$cm.

Orientadores: Felipe Maia Galvão França
Diego Moreira de Araujo Carvalho

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2018.

Referências Bibliográficas: p. 39 – 46.

1. redes neurais sem peso. 2. classificação. 3. transporte público. I. França, Felipe Maia Galvão *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

...to our little star...

Acknowledgements

Primeiramente, gostaria de agradecer a Deus, se existir, por me manter vivo, apesar de todas as minhas tentativas de jogar minha vida fora. Eu ainda não entendo se existe uma aleatoriedade, ou se foi uma decisão de algum ser supremo, mas fato é que ainda estou aqui.

À minha família, pelo suporte dado ao longo desses anos sem o qual não seria possível continuar, em especial à minha sobrinha Ana, que me fez perceber que o dia se renova todo dia. À minha companheira, por preencher parte do vazio que existe em minha vida. Aos meus orientadores, Felipe França e Diego Carvalho, pelas ideias, boas discussões e ajuda ao longo de todo esse trabalho. Ao parceiro de trabalho, Douglas Cardoso, pelos conselhos e ajudas. Aos meus professores do mestrado e graduação, pelo conhecimento adquirido, de imensurável valor. Aos amigos de trabalho, UNIRIO e IBGE, que de alguma forma compartilharam os dramas ao longo desse período. Aos amigos do samba, sem os quais não posso ficar. Aos amigos da faculdade, do colégio e da vida, que certamente foram importantes na trajetória.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

CLASSIFICAÇÃO DE TRAJETÓRIAS DE LINHAS DE ÔNIBUS UTILIZANDO REDES NEURAIS SEM PESO

Raul Bezerra Barbosa

Junho/2018

Orientadores: Felipe Maia Galvão França
Diego Moreira de Araujo Carvalho

Programa: Engenharia de Sistemas e Computação

Dispositivos com localização espacial estão em toda parte hoje em dia. Dentre várias possíveis aplicações com a grande quantidade de dados gerada por esse tipo de equipamento, nosso trabalho foca em um problema crônico da cidade do Rio de Janeiro: seu sistema público de ônibus.

Apresenta-se neste texto uma arquitetura para classificação de trajetórias GPS, cujo foco é a identificação de rotas de ônibus do sistema público. Para isso, utilizamos o leve e versátil classificador baseado em redes neurais sem peso WiSARD. Para a geração da entrada da rede, experimentamos diferentes formas de binarização, fazendo uso de regras definidas pelo problema. Ainda, avaliamos uma forma de combinação das redes WiSARD com o uso de um grafo acíclico de decisões. Todas essas propostas resultam em diferentes sabores de um sistema de aprendizado neurossimbólico.

Tal arquitetura foi testada contra um vasto conjunto de dados construído a partir de dados fornecido em tempo real e de forma pública pelo sistema corrente da cidade do Rio de Janeiro. Os resultados obtidos indicam a aplicabilidade da solução proposta em um problema de classificação envolvendo mais de 500 classes. As comparações efetuadas indicam uma equiparação do modelo WiSARD com outros modelos em estado da arte. No mais, acreditamos que a metodologia aqui descrita possa ser utilizada com sucesso em outros domínios.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

BUS LINE TRAJECTORIES CLASSIFICATION USING WEIGHTLESS NEURAL NETWORKS

Raul Bezerra Barbosa

June/2018

Advisors: Felipe Maia Galvão França
Diego Moreira de Araujo Carvalho

Department: Systems Engineering and Computer Science

Geo-enabled devices are ubiquitous nowadays. Within a diversity of possible applications using the huge of amount data generated by this technology, our work focuses on a chronic problem of Rio de Janeiro city: its public bus system.

This text presents a framework for GPS trajectories classification, whose focus is the identification of bus routes of a public bus system. In order to do that, it was used the lightweight and versatile WiSARD, a weightless neural network classifier. Different binarization methods were used to adapt raw data to WiSARD's binary input, making use of a set of rules defined by the application domain. Yet, it is evaluated a way of combining WiSARD through decision directed acyclic graphs. All these approachs result in different flavors of a neuro-symbolic learning system.

The framework was tested against a vast data set created from open access and real-time data acquired from the current bus system of Rio de Janeiro city. Results obtained suggest the applicability of the proposed solution in a classification problem with more than 500 classes. Comparisons made also indicate an equivalent performance of WiSARD and other state-of-art and widely used machine learning methods. In addition, the framework described here is believed to be adaptable to other application domains.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

A diversity of geo-enabled devices is currently among us. These devices, that can accurately locate an object on Earth's surface, have been generating a deluge of data with spatial information. Such data mass is now being used in many applications, becoming an important research area that mixes professionals from many different fields such as geography, management, engineering and computer science. Some recent works varies from mathematical analysis, modeling and data mining to city management and urban planning [1, 2].

One of these applications is geo-located data mining, where there is the trajectory classification problem [1, 2]. The main goal of this kind of classification problem is to automatically categorize trajectories inside a well-known set of categories. Trajectory classification is just a special case of the traditional classification problem in machine learning [1], hence, all sort of machine learning techniques can be used, supported also by different feature extraction strategies.

In the literature, a diversity of machine learning techniques has been successfully tested to classify trajectories such as NaiveBayes [3], SVM, neural networks, forests and decision trees [4]. Some more generalist frameworks have also been proposed, like [5], where authors assess performance in various tasks, including classifying animal trajectories. Some recent works in the field include [6] where authors explore the problem of classifying hurricane trajectories and [3, 7, 8], in which the main goal is to identify the transportation mode of a trajectory.

1.1 Motivation

Since 2013[9], the bus fleet of the city of Rio de Janeiro is being monitored by a GPS enabled system. At this time, some legislation by city councilor Marcelo Queiroz was proposed [10] and GPS real-time monitoring was put as one of the main objectives of city administration[11].

Such public transit monitoring system or automatic vehicle location (AVL) system is not exclusive to Rio. There are several (and recent) works related to real-time public transit management and monitoring [12–14]. An interesting common characteristic of these systems is that their data available is mostly available to general public, enabling its use by third parties applications.

In Rio's case, the data is freely made available through the web, via an API and, yet recent, there are some works already using it. In [15], the authors detail a framework for data acquisition and storage. Such framework is essential for any posterior work (since most of the data is a real-time stream) and a similar approach is used in this work. Yet, [16] shows a real-time monitoring system, where one goal is to keep track of bus travel times of the public transit system.

Normally, any monitoring application could straightforwardly use any GPS data from the buses or other AVL system. In Rio de Janeiro it does not work that way, though. It is just another contradiction in this city: data that was supposed to use as a regulation data is not reliable enough to confirm that the service being offered is the same agreed in the public contracts.

A preliminary analysis over all records of 2014 (over 200 million records) shows that more than 25% could be discarded by an user or the city management, simply because they would lack a very important information: which route is being served. This is aligned with past reports in the literature [16], where the fact is also mentioned and numbers are similar. Also, it was previously verified that some vehicles are officially labelled with one route tag but actually serving another.

It becomes mandatory to verify if the city regulations are being respected. One step towards it is to make use of trajectory classification knowledge to enhance the available data and empower the government in such a case of urban chaos.

1.2 Main Goal and Contributions

In the context of trajectory classification and using the above cited motivation in the Rio de Janeiro urban planning and monitoring, the main goal of this work is to present a viable solution to identify the route of each bus running on the streets based on its trajectory.

In order to that, different alternatives that rely on machine learning techniques to identify public buses routes from their preexisting digitized GPS trajectories were proposed and evaluated. It was performed experimental tests using a lightweight classification architecture known as the WiSARD [17] to help with the high amount of data. The WiSARD model has been used in many different previous works and contexts [18–20], always with competitive running time, accuracy and low standard deviation. It was also compared WiSARD performance with other mainstream classifiers [4] such as Random Forest and Naive Bayes.

The approaches developed here are examples of neuro-symbolic learning [21] methods, which combine neural and symbolic reasoning for optimal performance. A hybrid strategy was adopted, using weightless neural networks combined with a set of pre-defined rules.

This strategy was first presented in [22] and expanded in [23], which are the kernel of this work. Another kind of neuro-symbolic learning and reasoning was developed by [24], where neural networks are used as building blocks for a complex reasoning system.

Concluding, this work:

- Presents a viable solution to the real problem of identifying the route of running buses of the city of Rio de Janeiro based of their GPS data;
- Puts weightless neural networks closer to the trajectory classification field with a reusable methodology;
- Compares weightless neural networks with other highly used classifiers in a domain where agility and accuracy are indispensable;
- Makes a step towards a better public transit management in Rio de Janeiro.

Its contribution also aims to, in a near future, use geolocation data from city's buses to pinpoint deviations from a regular operation. Besides, the approaches described here might be useful for other geospatial and non-geospatial classification problems.

1.3 Structure of the Work

The remainder of this document is organized in 5 chapters besides this first introductory chapter. In Chapter 2, Rio de Janeiro's public transit is described, with problem's base data model. Also some basic geography background is observed, together with explanations about the machine learning models and methods considered in the work, such as the WiSARD model. Chapter 3 describes the methodology and the approaches to the problem, which relies on different trajectory image generation. The following chapter details the experiments using different datasets, parameters and models, then shows and analyses its results, showing how one can benefit from using different pre-processing procedures. At last, Chapter 5 wraps up providing an overview of this work's findings, as well as some ideas for future related works.

Chapter 2

Background

2.1 Rio de Janeiro's Public Bus System

2.1.1 History & Current Status

Rio de Janeiro's public bus system has started in the early 1900s, when a first route was opened between two important places in its commercial center: Passeio Público and Praça Mauá [25]. Facing some competition from electric trams during the 1920s and the 1930s [25], it became the dominant transportation mode in Rio in the early 1960s [25]. Nowadays, according to the most recent FETRANSPOR's projections, the city bus system represents around 60% of Rio de Janeiro's metropolitan area public transportation system (see Figure 2.1).

FETRANSPOR is the main responsible for the bus system in Rio de Janeiro state, being a a congregation of syndicates of bus companies [2]. One of these syndicates is *Rio Ônibus*, main responsible for all 42 bus companies operating in Rio de Janeiro city.

As of FETRANSPOR' 2016 report[27], the city bus system has:

- 733 official bus routes;
- more than 8,000 vehicles serving these routes;
- more than 1,500,000 trips[3];
- more than 60,000,000 kilometers traveled[3];
- more than 100,000,000 passengers[3].

[1]Values for year 2016 were projections. Data is reported in [26]
[2]http://www.fetranspor.com.br/
[3]All monthly

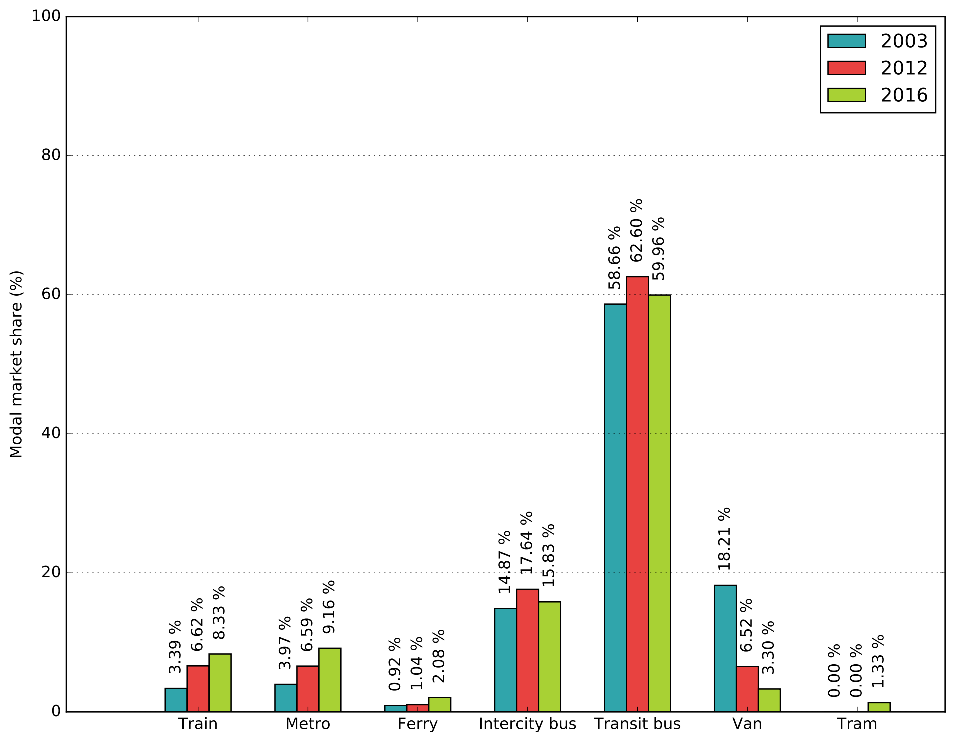


Figure 2.1: FETRANSPOR's modal market share survey [1].

The current system management is divided in regions managed by bus companies consortia in a concession agreement [28], as depicted by Figure 2.2. There are 5 regions of interest, named 'RTRs', from 1 to 5 (see Figure 2.3):

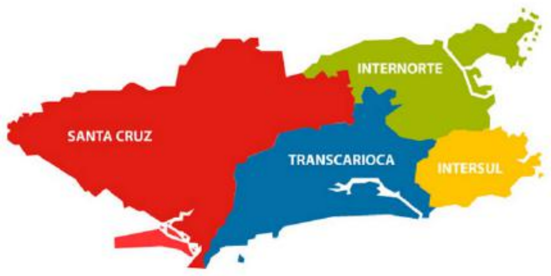


Figure 2.2: Rio de Janeiro city divided by bus consortia[4].

- RTR 1 (also called *Central*): consisting of the neighborhoods of *Centro* and *Tijuca*;
- RTR 2: comprising the South Zone (*Zona Sul*) neighborhoods;

[4]Adapted from [29]

- RTR 3: comprehending in a vast majority the neigborhoods of the North Zone (*Zona Norte*);
- RTR 4: consisting of some of the North Zone and some of the West Zone (*Zona Oeste*), like Barra and Jacarapagua neighborhoods;
- RTR 5: with the remaining neighborhoods of West Zone.

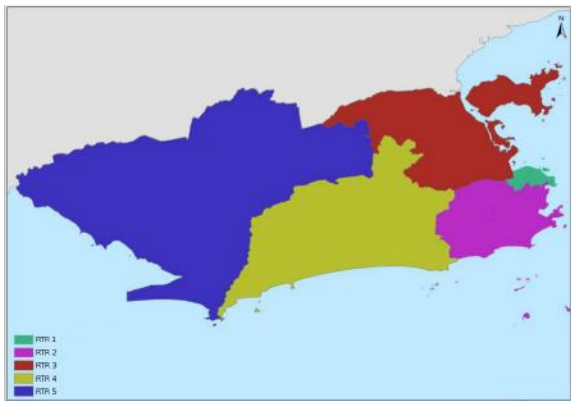


Figure 2.3: Rio de Janeiro city divided by RTRs[5].

The consortia are 4: Intersul, Internorte, Transcarioca e Santa Cruz, operating regions 2, 3, 4 and 5, respectively. Region 1 is a common operation region, but its internal routes are operated by Intersul consortium.

2.1.2 Data management

As leveraged by [25], every consortium is responsible for feeding FETRANSPOR data servers. FETRANSPOR's duty is to aggregate this data and then send to Rio's prefecture, which uses the data for city management and makes it public available and free of charge (like in Figure 2.4).

In Rio's system, as already cited, the data is provided as a public API (actually just one public URL)[6]. Every minute this URL response is updated, but it is always a JSON object consisting of:

- a timestamp - date and time of measure;
- vehicle id - an id that registers that car and is public visible in the car (see Figure 2.5);
- the route or line the bus is serving;
- latitude of the bus;

[5]Figure adapted from [30].
[6] http://dataṙio/dataset/gpsdeonibus

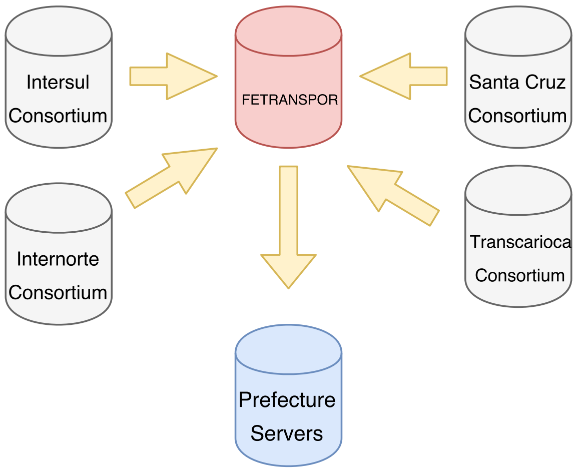


Figure 2.4: GPS measures data flow: from each consortium to prefecture's servers

- longitude of bus;
- current speed.

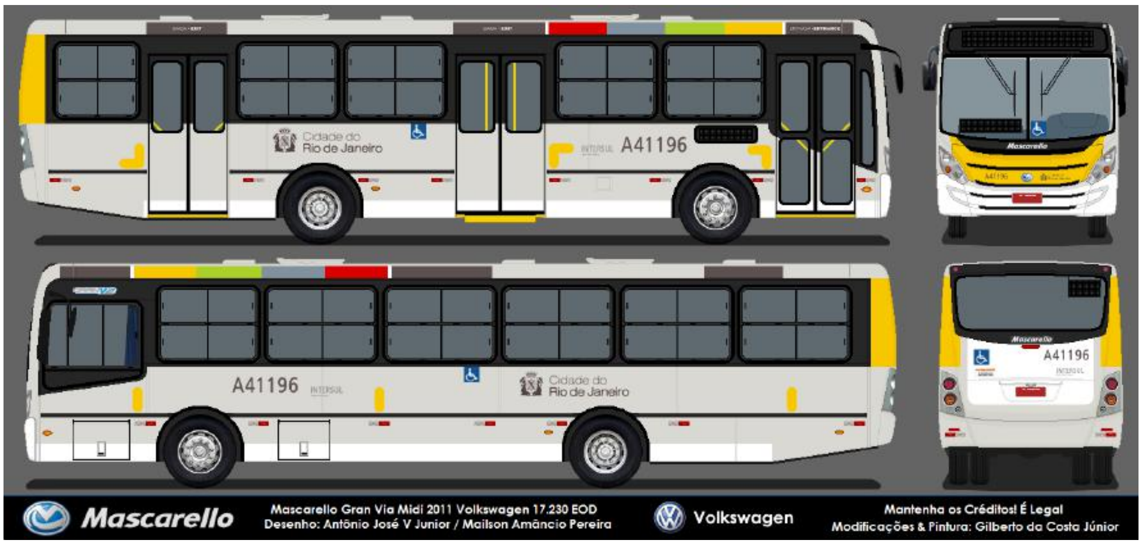

A41196
Cidade do Rio de Janeiro
Mascarello
Mascarello Gran Via Midi 2011 Volkswagen 17.230 EOD
Desenho: Antônio José V Junior / Mailson Amâncio Pereira
Volkswagen
Mantenha os Créditos! É Legal
Modificações & Pintura: Gilberto da Costa Júnior

Figure 2.5: A Rio de Janeiro bus car illustration with 'A41196' as its id[7].

In addition, Figure 2.6 shows an example of a JSON object obtained through the public API.

2.2 Geo-positioning and Cartography

2.2.1 GPS

The (NavStar) Global Positioning System, commonly referred as GPS [31], is an US-owned system to "accurately determine (...) position, velocity, and time in a

[6]Source: https://desenhos.onibusparaibanos.com. Authors: Antonio Mailson V Junior, Mailson Amâncio Pereira, Gilberto da Costa Júnior

```
{
    "COLUMNS": [
        "DATAHORA", //timestamp
        "ORDEM", // vehicle id
        "LINHA",  //route
        "LATITUDE",
        "LONGITUDE",
        "VELOCIDADE" //speed
    ],
    "DATA":[
        [
            "06-18-2015 08:28:47", //timestamp
            "C47705", //vehicle id
            861.0, // route
            -22.954453, // latitude
            -43.391499, //longitude
            19.0 //speed
        ],
        ...
    ]
}
```

Figure 2.6: Bus API response: a JSON example

common reference system, anywhere on or near the earth on a continuous basis”as detailed by [32]. Primarily designed as a military system [33], today it is available as a free public service, guaranteed by the US government, and can be used to locate mostly anything around the Earth. A recent study concludes that a common smartphone is typically accurate to within a 4.9 m radius under open sky [34].

According to [31], its structure follows a 3-segment approach: a space segment, a control segment and a user segment. The space segment consists of a constellation of at least 24 operational satellites around Earth’s orbit that transmit radio signal to users. The control segment is a network of ground facilities that operates and monitors the space constellation. Finally, the user segment is the user, most of times a GPS receiver or a GPS-enabled device, which receives the signals from the constellation and uses it providing three-dimensional location and timing.

The GPS system relies on precise atomic clocks in every satellite and the receiver (see figure available in [35]). The satellites send signals to the user and based on the time received and the time of reception, the receiver can calculate its distance to the satellite. With some geometry and at least 4 different satellite signals is possible to determine the position. The reader can refer to [31] for a more complete explanation behind GPS.

2.2.2 Coordinates

A position given by a GPS system is often expressed in terms of latitudes, longitudes and height (or depth), in some reference coordinate system. This is called a ellipsoidal or geographical coordinate system, since it is a more geographical way of viewing objects in the space. On the other hand, in mathematics, a more common way is to use the Cartesian coordinate system, expressed in terms of X, Y and Z.

The latitude (ϕ) of a point is defined as the angle between the equatorial plane, an imaginary plane that divides the Earth in two hemispheres, and a straight line that intersects the point and passes through the center of the Earth (or close to it).

The longitude (λ) of a point is defined as the angle between one reference meridian and a meridian that intersects the considered. A meridian is a half of a hypothetical great circle on Earth's surface that ends in North and South poles. It is common to use the Greenwich meridian (a meridian that crosses the British city of Greenwich) as the reference meridian, as it is established as the division between the Orient and the Occident.

The height (h) of a point is a more complicated problem, since it involves calculating the distance from the surface or center of the Earth to the given point. Since the Earth is nor a sphere neither a ellipsoid (actually a geoid) [33], researchers have come across the concept of different ellipsoids and reference systems to better fit their region of interest. Usually there are equations to transform point from one reference system to another. The reader can overview this scheme in Figure 2.7, as [33] covers this topic in depth.

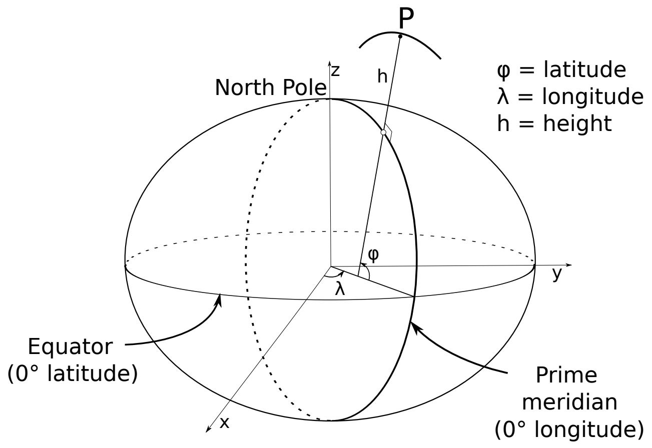


Figure 2.7: Geographical coordinates explained

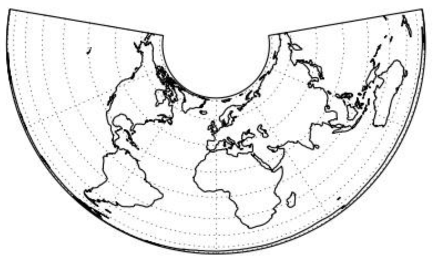
2.2.3 Projections

Knowing the latitude, the longitude and the reference system is not enough to precisely represent a point on a map. Since a map is usually a bi-dimensional canvas, and Earth's surface is a three-dimensional abstraction, researchers use projections to project points into a 2D-space, where x and y coordinates are given by function of the original latitude and longitude.

There is a huge number of projections [33], and the one used really depends on the application. Since it is impossible to have a perfect projection, there is always a trade-off. For instance, geodetic applications tend to use conformal projections, like Lambert's conical or cylindrical, which preserve the original angles from the ellipsoid on the map. Some maps or applications need an equivalent projections, which preserve the original distances, but not the angles, and can be used to calculate areas as well. Some examples of projections can be found in Figure 2.8.

(a) Lambert conformal projection

(b) Azimuthal equidistant projection


(c) Albers projection (with equal area)

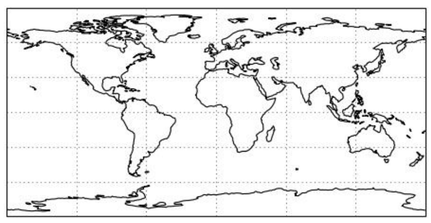

(d) Plate Carrée projection

Figure 2.8: Projection examples. [8]

2.3 The WiSARD weightless neural network architecture

The WiSARD[17] model is a weightless neural network and has its name derived from its authors (**Wi**lkie, **S**tonham e **A**leksander's **R**ecognition **D**evice). It was initially developed as a hardware architecture for image recognition [37] and it is very different from other models such as the *perceptron*, derived from McCulloch & Pitts' neuron [38].

The McCulloch & Pitts' neurons are human neurons abstractions, which activate or not, depending on a linear combination u of a weight vector p and an input vector x:

[8]Source:[36]

$$u = \sum_{n}^{i=1} w_i * x_i$$

where signal y of neuron is activated if it reaches some threshold θ:

$$y(u) = \begin{cases} 1, & u >= \theta \\ 0, & u < \theta \end{cases}$$

On the other hand, the WiSARD weightless neural network can be viewed as an implementation of the Bledsoe & Browning's n-tuple classifier [39]. It works based on read and write operations of binary values in RAM memories, making use of pseudo-random mappings on input vectors.

The network itself is mainly organized in *discriminators* which are sets of X one-bit word RAMs with n inputs (also called neurons), being one discriminator frequently used to determine just one class of many given as input to the classifier, hence its name. This model can be categorized as a supervised learning method[40], although it has been expanded to unsupervised learning as well [41]. Like every supervised learning method, it is characterized by a training phase (learning from examples) and a classification (or prediction and sometimes test) phase.

2.3.1 Training Phase

During the training phase, a number of binary vectors (examples) are shown to the classifier. Each input vector is mapped to a *retina*, which is a division of the input vector in X tuples of n bits, commonly in a pseudo-random manner. Following this strategy, each n-bit tuple is used as an address of a RAM memory of 2^n positions, with the n tuples mapping X memories.

When a new example of a new class C is presented to the network, a discriminator is then created to this new class. This discriminator will usually have X RAM memories, with all their 2^n positions with no data. Learning from that example is just as simple as writing a '1' in all memory addresses mapped by the tuples of that example.

2.3.2 Classification Phase

The classification phase is when an input vector without a class is presented to the network and we use the network to give it a class. It is handled by mapping an input in the same way training examples were mapped.

Afterwards, we present the input to all discriminators previously trained. At this moment, every tuple of the input can point to either a marked address (1) or

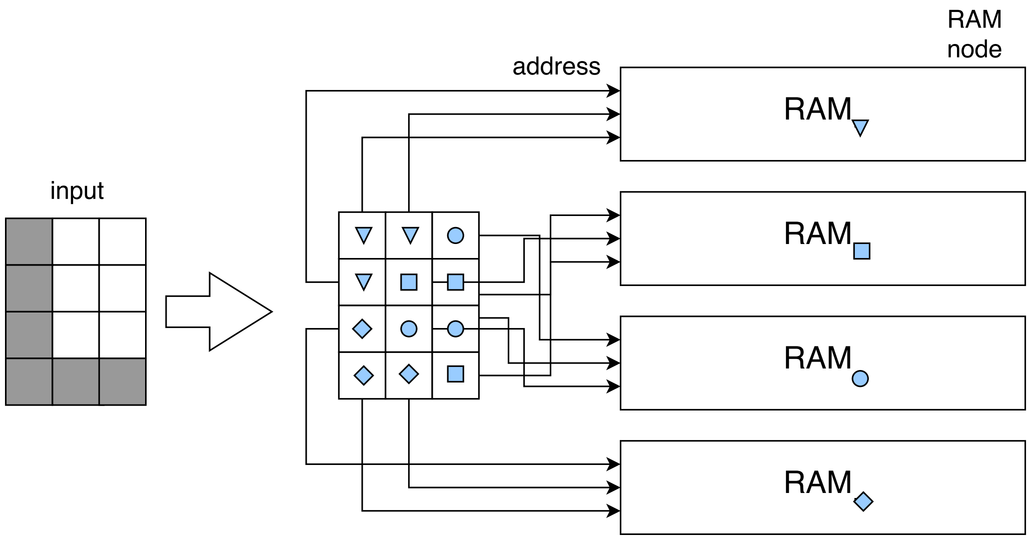


Figure 2.9: Training phase using WiSARD: the input binary vectors are pseudo-randomly mapped into n-bit tuples ($n = 3$) that become addresses of X RAMs ($X = 4$)

a non-written address(0) of RAMs. Here, we have the neuron activation: if a tuple addresses a written address, that RAM will be activated, as a neuron, indicating a pattern previously seen.

The score r_y for a discriminator y given an input x is calculated by counting the number of memories mapped by the classification example whose addresses were previously written by any training example. The discriminator that has the greatest score (r_{max}) is chosen as the output class for x, with ties being broken randomly. Literature[37] also establishes a confidence measure for the result r_{max} (denoted here by C), given by:

$$C = \frac{r_{max} - r_{max-1}}{r_{max}} \tag{2.1}$$

where, again, r_{max} is the score of the output class discriminator and r_{max-1} is the runner-up discriminator's score.

2.3.3 Binarization & Pre-processing Steps

Since the WiSARD architecture was designed to operate with bit vectors and not with integers or other types of values, pre-processing an input dataset to use WiSARD is a very common step. The so-called binarization, transforming or encoding a vector into a bit vector is needed when one does not have a binary input. Many binarization approaches have been proposed in the literature [42–44]. The generation of black and white images (which can be seen as binary inputs) is diversely explored in [45], although in a different context.

Within many proposed approaches, one common approach to encode real values

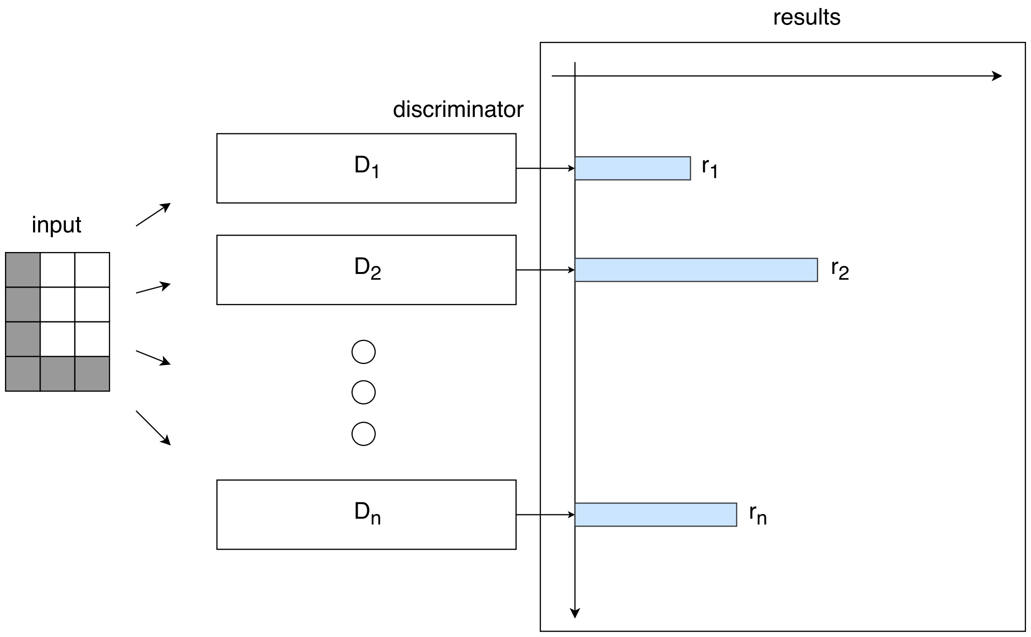


Figure 2.10: Classification in the WiSARD: the input (mapped the same way training examples were) is presented to the network and for all trained classes (discriminators), the one with the best result r is chosen.

in binary values is the *termometer* transformation[44], when a fixed-length bit array is used to discretize the values. As an example, suppose that 10 bits were chosen to represent an real-valued attribute a. Hence, maximum and minimum values that a can assume are:

$$a_{\text{max}} = (1111111111)$$

$$a_{\text{min}} = (0000000000)$$

with the median being a value like:

$$y_{\text{med}} = (1111100000)$$

When dealing with categorical values, another common transformation is the *one-hot encoding*. In order to use it, a fixed-length array is again chosen, as also a marker size (some bits). In example, supposed that some attribute *color* can assume one between three values: red, yellow, blue, and a 3 bit marker will be used. The final representation of the attribute will have 9 bits, with the possible values:

$$\text{color}_{\text{red}} = (111000000)$$

$$\text{color}_{\text{yellow}} = (000111000)$$

$$\text{color}_{\text{blue}} = (000000111)$$

It is important to note that pre-processing steps are also common when using other machine learning approaches, being an almost mandatory step in classification tasks [1]. One of most common steps are data standardization and normalization[46], used to reduce the effect that features of different scales and orders of magnitude might produce over the final result. Even though very common, pre-processing steps may play a key-role to achieve a final good result, especially when weightless neural networks are being used [41]. In WiSARD, a good binarization is somewhat related to finding a good distance measure, which is a very important factor in search and query problems [1].

2.3.4 Bleaching & DRASiW

The performance of the standard WiSARD model can be affected when dealing with very noise data. For instance, when many discriminators receive similar examples as if they belong to their class, the network performance can degenerate to a complete random answer. This can also happen if small tuples are being used, leading to very common patterns, easily exausted. This phenomenon is called *saturation* of the RAMs.

Saturation happens when many classes have high responses from the discriminators. If all RAMs have most of its addresses written, they will see the same patterns in almost every class. It can be viewed as a form of overfitting the WiSARD model. Such concept means that we are fitting more data than a network can absorb while being trained.

An immediate aid to overcome this problem is the adoption of the *bleaching* [47] technique with the so-called DRASiW. DRASiW [48] is a modification of WiSARD where instead of using one-bit word RAMs, q-bit word RAMs are used. Hence, different threshold values can be used to activate a RAM, instead of simply marking whether an address was written or not. Although it becomes possibly more accurate, it comes with a trade-off in speed[49].

Yet, this modification also enables the user to reverse the training process and see what the network has absorbed in the form of a mental image (see Figure 2.11).

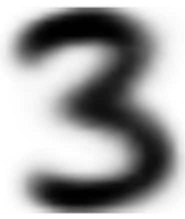


Figure 2.11: A DRASIW's mental image example when WiSARD is trained on a dataset of handwritten digits (is this case: digit three).

Although very fast and competitive, the WiSARD model is just the simplest model of a whole family of weightless neural networks as presented in [37].

2.4 Decision Trees

Decision trees classfiers [50] are widely used in machine learning [51]. Most common use is done by building or inducting trees (acyclic graphs) capable of answering questions or taking a decision in a methodological form. A decision tree classifier is a decision tree capable of labeling an example in a path from its root to any of its leaves (see Figure 2.12). It is a tree that consists of essentially three types of nodes:

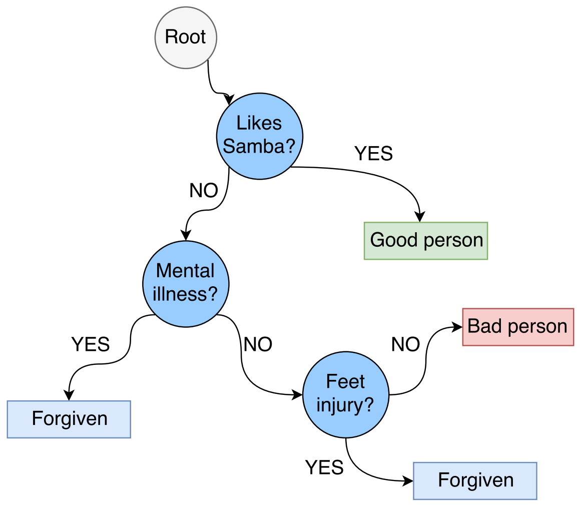


Figure 2.12: A decision tree example. Here the tree distinguishes between three classes: forgiven, good or bad.

- **Root node**: Where the process starts, with 0 incoming edges and possibly more than zero outgoing edges;
- **Internal nodes**: Where questions are asked and paths can be chosen, with exactly one incoming edge and at least two outgoing edges;
- **Leaf nodes**: Where questions are answered, with exactly one incoming edge and no outgoing edges.

One of its biggest advantages is the fact that it is a 'white-box' method: it can give the user an explanation why a given class was the output. Other advantages include the fact that it is a simple method that can fit a innumerable amount of datasets. This can lead to of its main disadvantage: to not generalize well if trained with few data and be very specific [52].

2.4.1 Traditional Decision Tree Classifier

Here we present a general approach when building a decision tree classifier (training phase):

- The tree has a root. There is a training dataset C and a attribute set A that every example is made of;
- A measure of interest is calculated for every attribute $a \in A$;
- The initial dataset is divided in subsets, using the best attribute and its best value according to the value of the measure of interest after the division;
- The attribute and its value become a node N, added to the tree (attribute is sometimes discarded from set A and no longer used);
- The procedure is then repeated to all datasets created by the division (recursively), having N as root;
- Stop the algorithm when some stop criteria is reached (given by the measure of interest, for example), or if the attribute set is exhausted (if attributes were discarded along the way).

There are two main decision tree classifiers in the literature: CART[53] and C4.5[54]. Both are quite available in science tools like scikit-learn[55] or Weka[56] and have similar tree induction algorithms.

Both algorithms are greedy algorithms, which may not find a perfect solution to the problem. Also, finding the best tree to divide a dataset is known to the NP-complete[57]. The main difference between the algorithms is in the measure of interest and number of possible outcomes. CART takes into account the Gini impurity of outcome nodes and only splits in two nodes. C4.5 uses information gain as its split measure and can split nodes in more than two. The reader can see that are other differences[51] between the algorithms, but they are not subject to discussion here.

A classification phase can be executed when one has an inducted tree. Classifying an example y without label is done by simply following the path from the tree's root to one of its leaves according to the internal nodes attributes and values (and also the example attribute values).

As mentioned before, decision trees can suffer from overfitting as well. Its size can grow out of control and make the tree very specifically fitted to the data presented to it. In that sense, the tree may not be able to label examples that are somewhat different from what was presented to it. A very common technique to tackle this problem is to prune the inducted tree [52]. This can be done in two general ways:

- **Pre-pruning:** When the tree inducting algorithm is stopped before terminating, with some eartly stopping condition. The algorithm will produce a tree that does not perfectly fit the training data, but might be better when seeing test data or classifying in practice;

- **Post-pruning:** When after the tree induction some subtrees are discarded, following some condition. This is done in a bottom-up fashion (leaf to root). Some research [52] shows that post-pruning may lead to better results, but might be more compute-intensive.

2.4.2 RandomForests

In an attempt to improve decision tree classifiers, Ho[58, 59] proposed random decision forests and more lately Breiman [60] improved the idea, creating the so-called RandomForests algorithm. Its idea is based on few key aspects: forests (a set of trees) instead of a single tree, combined with the use of *bagging* and *random feature selection* when splitting the sets.

Bagging - *bootstrap aggregating* [61] - is a metaheuristic that consists of combining solutions to build a solution that generalize better. From a training set C, N other smaller datasets are created by sampling it with replacement. These N datasets can be trained independently to induct N tree classifiers. The final answer of the model is given by a voting where each classifier has one vote. The higher scoring class (the one with more votes) will be chosen as output.

The other concept involved, *random feature(attribute) subspace selection*, means that for every node decision (every partition of a dataset), only a random subset of features is considered, with replacement. The rest of the algorithm remains the same.

RandomForests has been used in a plethora of works [62, 63] with good results and provides a good baseline for studies. Its random attribute subspace selection combined with bagging reportedly increases performance over traditional decision trees.

2.4.3 ExtraTrees

ExtraTrees or Extremely Randomized Tree [64] is another improvement over classical decision tree on top of the RandomForests algorithm. It is very similar to it, but it adds another layer of randomness, making the value of attribute splitting a random value between the maximum and the minimum possible values. The measure of interest is still evaluated though. Results are combined the same way as in RandomForest, with voting. Yet in [64], authors show that ExtraTrees can improve performance over RandomForests. If not by making it more accurate, at least it is shown that it may be faster without compromising the results.

2.5 Naive Bayes

The Naive Bayes classifier is a well-known and very simple classifier based on conditional probabilities [52]. It makes strong use of Bayes theorem, hence, its name. The word naive comes from the other core concept of it: a *naive* supposition of conditional independence of all attributes in attribute set A (something that most likely will not be true). Even being very simple, it still achieves good results in some scenarios[65].

Given an attribute set $A = a_1, a_2, ..., a_n$, a class set $C = c_1, c_2, ..., c_q$, a dataset $D = X_1, X_2, ..., X_m$ and every $X_j = (x_1, x_2, ..., x_n)$ and x_i the value of attribute a_i in an example X_j, the output class $y(X)$ of the Naive Bayes classifier will be given by:

$$y(X) = \underset{c \in C}{\operatorname{argmax}} P(c|X) \tag{2.2}$$

where $P(c|X)$ is the probability of class c given the input vector X.

In order to computer this probability, the Bayes theorem is used, such that for every class $c \in C$ we have:

$$P(c|X) = \frac{P(X|c) \times P(c)}{P(X)} \tag{2.3}$$

Since $P(X|c) = P(x_1, x_2, ..., x_n|c)$ and conditional independence of every attribute is assumed, it is possible to work on equation 2.3 using the chain rule for conditional probabilities, which gives us:

$$P(c|X) = \frac{\prod_{i=1}^{n} P(x_i|c) \times P(c)}{P(X)} \tag{2.4}$$

One can easily verify that $P(X)$ is constant for every class $c \in C$, leaving us with only the numerator part of equation 2.4. $P(c)$ can be estimated from frequency of class c in D. If all attributes in A are discrete or categorical, $P(x_i = k|c = c_q)$ can be also estimated from the relative frequency of value $x_i = k$ within all examples of class c in D.

So, in this case, there is no unknown value in equation 2.4 to estimate $P(c|X)$ and thus Naive Bayes can be executed. For other scenarios such as continuous attribute values and a complete reference, the reader can see [66].

Chapter 3

Methodology

3.1 Input data acquisition

According to [1], one can define a spatial trajectory as a sequence of points $T = p_1, p_2, .., p_n$ chronologically ordered, such that each point $p_i = (x, y, t)$, where x and y are geographical coordinates (latitude and longitude, mostly) and t is a time stamp associated with that position.

The bus system API mentioned in the previous section does not provide any trajectory, only points, and the system itself does not keep track of past measurements. A trajectory can only be obtained if one keeps track of all API responses somehow.

In our work, it is used a data acquisition system to deal with this problem. A service running 24/7 makes requests every minute and stores every JSON response on a daily basis, like shown in Figure 3.1. Even doing this, there are only stored points. It is still necessary to group the stored points by vehicle id to have a final trajectory.

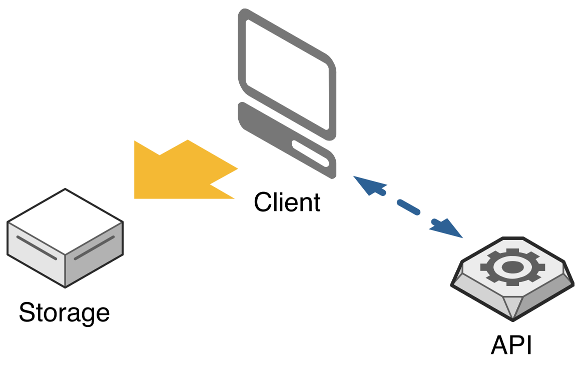


Figure 3.1: Storage scheme: API is called every minute by client code and results are stored in a file server

3.2 Generating the network input

As discussed in Section 2.3, the WiSARD neural network operates only with binary values (vectors). Since it the initial idea was to use the trajectories to classify buses routes using WiSARD, an encoding method to the stored GPS trajectory data is proposed here in order to use it as input data to our classifier.

3.2.1 The *footprint*

An initial approach to converting GPS trajectory data to a zero-or-one input is to generate a black-and-white image out of it. The GPS coordinates can be plotted as points in a 2-D canvas, within some reasonable bounding box such as the city limits. A similar approach was used and explored with good results in [67] to solve time series problems. This plotting can also be seen as a grid partitioning of the city in disjoint cells similar to [68, 69], but here we are ignoring the sequential and temporal characteristic of the trajectories.

In this initiative, the trajectories become some kind of photograph of the vehicle throughout a window of observation. This approach is called the *footprint*. The generated image can be easily converted to a binary vector if every painted pixel becomes a '1' and '0', otherwise.

One can note that there is a clear trade-off here: a lossy data compression of the trajectory [70]. However, it does not compromises the final goal, since it is likely to represent what the bus was doing in a period of time. Conversely, because of this compression, common problems of working with GPS data can be tackled, such as a brief or temporary loss of GPS signal[1], as it uses a window of observation and all points in this window are used together as an example.

This can be interesting to the bus case, where routes tend to be cyclic and a temporary loss might not be even noted if we look at a day's picture, for example. Also, this grid strategy collapses similar GPS positions, aiming towards a more complete and maybe redundant trajectory image, avoiding occasional noises and natural GPS imprecision.

It is essential to note that due to a WiSARD's characteristic, even if two or more examples have non-overlapping gaps but represent the same class, a complete example can be classified correctly, as the final memory will have a "neural" representation that is equal to the union of all training examples.

Figure 3.2 shows a single trajectory – the above-cited footprint – performed by a vehicle that served only one route during a period of time of day. In order to understand the complexity of the classification task, Figure 3.3 presents the trajectories of all vehicles, serving all lines during the same time period of time. Hence, the footprint of a vehicle is used as the unit of classification, representing an example of

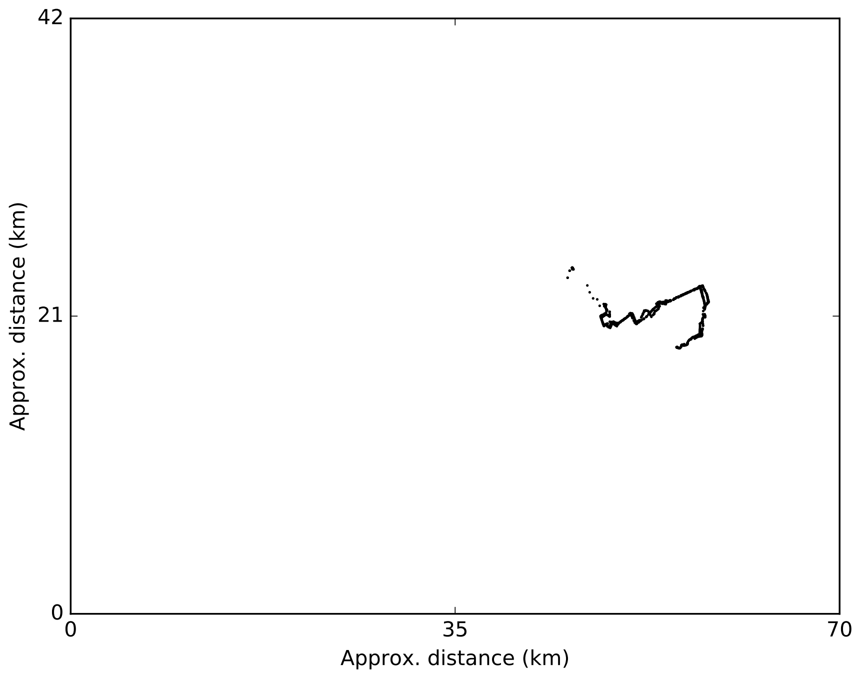


Figure 3.2: The footprint of a vehicle.

a bus route for the WiSARD classifier.

3.2.2 A smart gridded image

One way to refine the footprint image is to add some prior knowledge about the nature of the problem being solved. This is often employed in classical applied machine learning, where feature extraction/engineering plays a key role [46, 71]. Domain experts work along with statistical methods and metrics to try to come up with the most descriptive feature set to use as input to solve a problem using neural network.

For instance, in the information retrieval field there are metrics like *Document frequency threshold* and *TF-IDF* [72], commonly used to discard attributes based on words relevance. Many times algorithms like PCA/SVD [73, 74], SIFT[75] are used to extract relevant features out of other serveral features, possibly reducing a problem's dimension [46, 71].

In this application, as depicted by Figure 3.3, it is worth to perceive the distribution of the buses' GPS measures (over a latitude/longitude space). There are some regions where markers are abundant (like regions of interest) and others with not a single marker. This image is a good visual evidence of how Rio's bus system works and gives a hint about Rio de Janeiro's topography. In fact, there are some regions where traffic is very dense, the city center. Besides that, a landscape of mountains

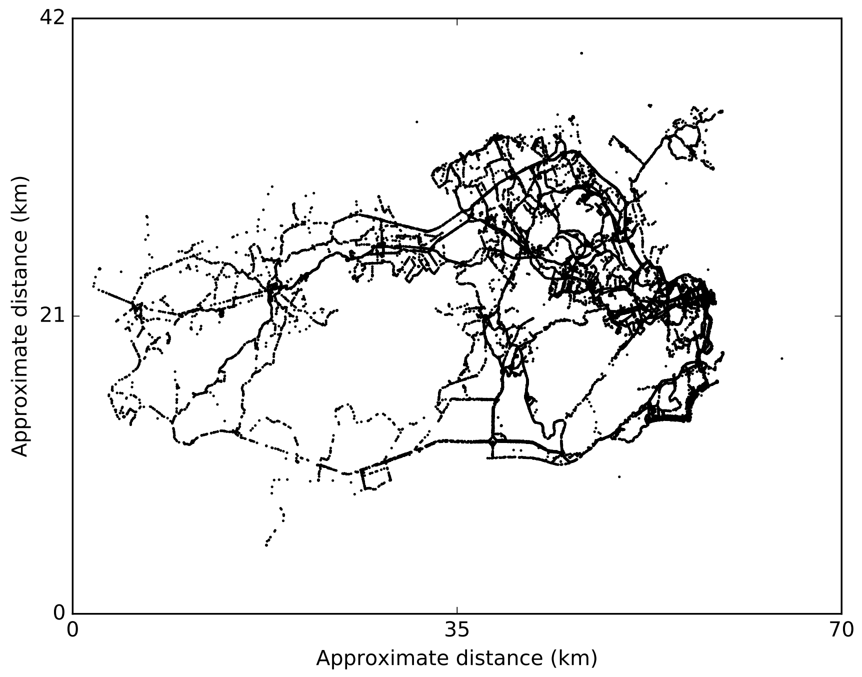


Figure 3.3: The footprint of the entire fleet.

and forests forces traffic to be distributed along thin corridors that connect other less dense but still central local areas. Figure 3.4 merges Rio's topology with one footprint example to better ilustration the situation.

Having this geographic prior knowledge in mind, a more symbolic representation can be constructed from the initial grid image. By readjusting its cells sizes, more pixels can be given to regions that might deserve more attention, less to ones that might not.

In order to achieve this goal, the pixels of the previous binary image are re-encoded, making a new image out of new division of the initial spatial boundary. This new spatial division is supported by a kd-tree [76] division of the spatial limits, where every pixel of the previous image will certainly fall into a leaf node of this kd-tree. Each example is converted such that all leaf nodes with pixels inside gets a '1' in the new image and '0' for the opposite. The input vector is then transformed to the size of the grid being used (the number of leaves). Figure 3.5 and Figure 3.6 show an grid example generated out of a kd-tree with a footprint example and a footprint example only after the reencoding process, respectively. Figure 3.7 once merges Rio's topology with one re-encoded footprint example.

The spatial division process starts with kd-tree using on single node with the city lat/long boundaries. The kd-tree is then grown (nodes are divided) based on three criteria:

- Stopping criterion: when to stop the division of nodes, like reaching a certain

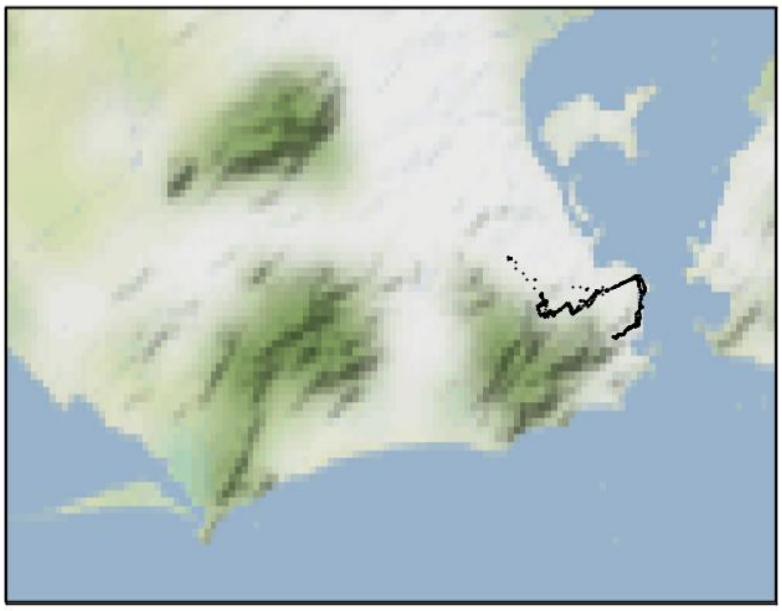


Figure 3.4: Footprint example with Rio de Janeiro's topology as background: green areas are mountains or forests with low circulation of people. Black marks are GPS measures.

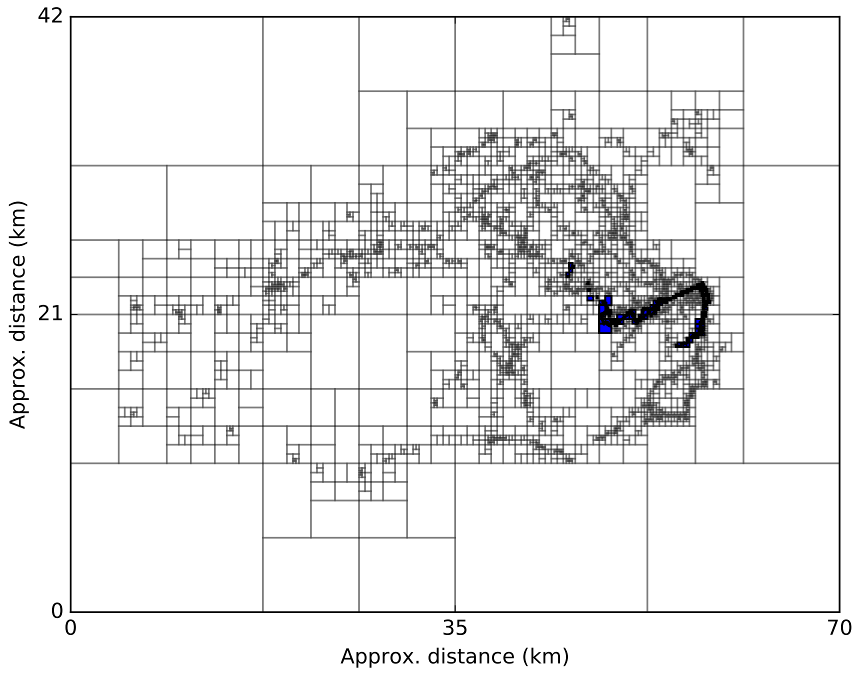


Figure 3.5: A footprint example after the grid preprocessing (with grid).

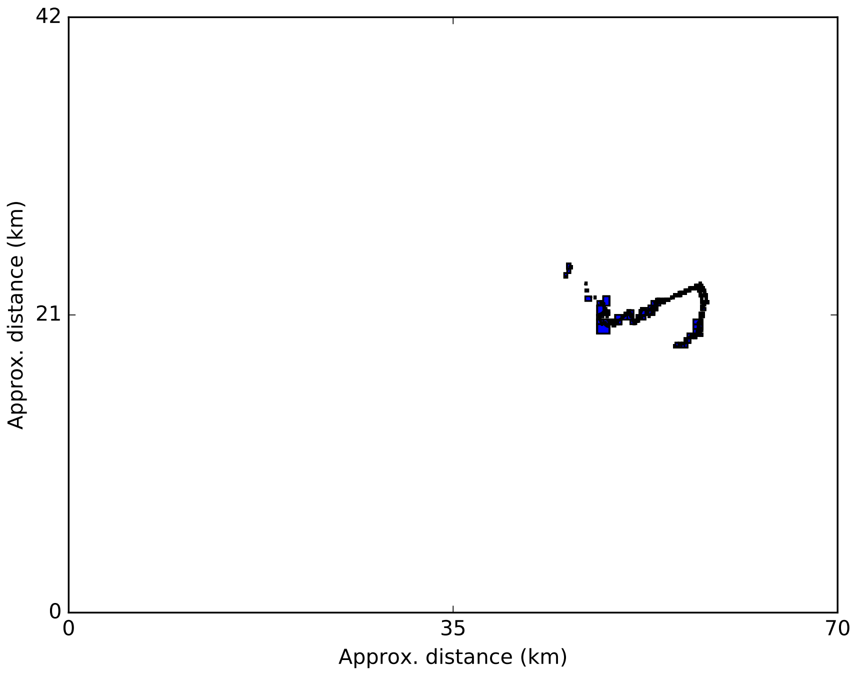


Figure 3.6: A footprint example after the grid preprocessing as itself, cell resize occurs in most of cases.

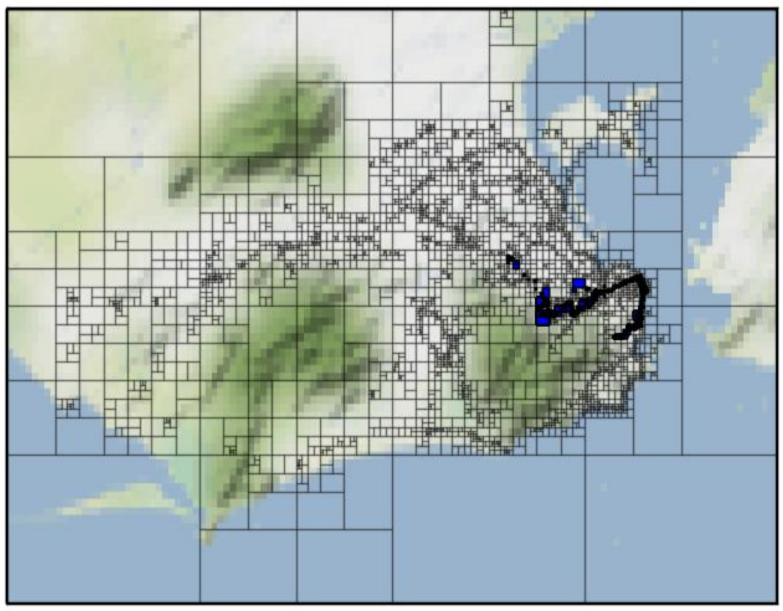


Figure 3.7: Re-encoded footprint example with Rio de Janeiro's topology as background: green areas are mountains or forests with low circulation of people.

number of leaves;

- Splitting criterion: where to divide a node, such as divide it in a middle point;
- Next node criterion: which is the next leaf node that will be split, for instance, greatest number of points within node lat/long boundaries.

Algorithm 1 gives an overview of this kd-tree induction. In a nutshell, the key idea, like any other kd-tree, is to divide a space into k dimensions, alternately, each dimension at a time. In our case, we have only two dimensions (latitude and longitude), so the algorithm alternates splits between these two dimensions, always observing the criteria mentioned earlier.

This approach can be thought as an enhancement of the previous approach, with some sort of region-based clustering (as seen in [5]), since the pixels (or attributes) from the previous approach are merged based on regions of interest. It can also be transformed into a density-based clustering (as used in [77]) for feature extraction, since density can be used to decide the next node to be divided.

Algorithm 1: A greedy 'kdtree' based spatial division algorithm

```
input  : NextNodeCriterion, SplittingCriterion, StopCriteria, tree
output: tree
while not StopCriteria() do
    nextNodeToSplit ← GetNextNodeToSplit(tree,NextNodeCriterion);
    dimension ← GetNextDimensionToSplit(nextNodeToSplit);
    SplitBy(tree,nextNodeToSplit,dimension,SplittingCriterion)
end
```

3.3 Combining WiSARD layers - WiSARD-DDAG

Going further on what [78] would define as a neuro-symbolic path, the third and final approach proposed is an ensemble of WiSARD classifiers. WiSARD classifiers are used as decision nodes in a decision directed acyclic graph - DDAG [79], a strategy for multiclass problems. Using this approach, there is an attempt to encode logic rules that hold between the same input represented in different grid sizes. In that sense, a example is not only represented by a single gridded image but by a combination of the same input encoded with different grid sizes.

In a more formal way, suppose that a grid $\mathbf{G_1}$, built using the aforementioned kd-tree approach, is composed of some disjoint cells whose union represent the whole space (like in Figure 3.8a). Consider also a second grid $\mathbf{G_2}$, also built using the same strategy with exactly same parameters and data. Its cells' union also represent the whole space, but it has more cells than $\mathbf{G_1}$ (see an example in Figure 3.8b). Then each cell in $\mathbf{G_1}$ is the union of 1 or more $\mathbf{G_2}$ cells. If that holds, the following rule applies: a trajectory happening in some subset $\mathbf{S_1}$ of $\mathbf{G_1}$ will happen in some subset $\mathbf{S_2}$ of $\mathbf{G_2}$ whose cells are also contained in $\mathbf{S_1}$. In practical terms, it is conjectured that $\mathbf{G_2}$ is a granularity refinement or zoom in over $\mathbf{G_1}$. Figures in 3.9 try to illustrate this idea.

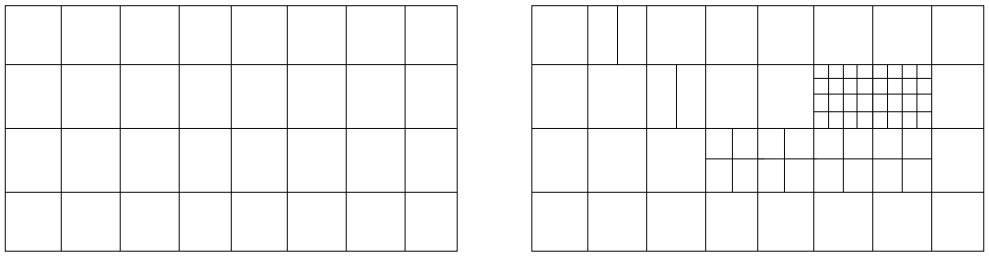

(a) Grid G_1 (b) Grid G_2

Figure 3.8: Hypothetical grids

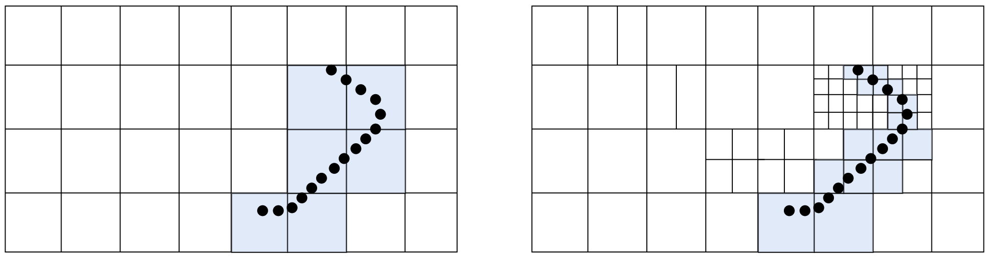

(a) Grid G_1 with a hypothetical footprint (b) Grid G_2 with the same hypothetical footprint

Figure 3.9: Hypothetical grids with a hypothetical (and same) footprint. In light blue, reader can see the different granularity between grids

This idea leads to a tie breaking policy: if two WiSARD discriminators are in a tie situation using $\mathbf{G_1}$ as input, that tie might not happen when using a more fine-grained input such as $\mathbf{G_2}$.

The proposed WiSARD-DDAG approach works as follows: a set $W = \{W_1, W_2, ..., W_n\}$ of WiSARD classifiers is trained with a combined dataset $D = \{D_1, D_2, ..., D_n\}$ and arranged in n sequential classification layers. Every D_i has the same trajectory examples, but each one is encoded with a different grid, such that D_i grid size is smaller than D_{i+1} grid size. Dataset D_i is used to train WiSARD W_i.

Output class is given by testing the input data at each every layer iteratively, starting with W_1, with three possible paths:

- No class identified: terminate execution with 'UNKNOWN' answer or output previous layer WiSARD output if there is one;

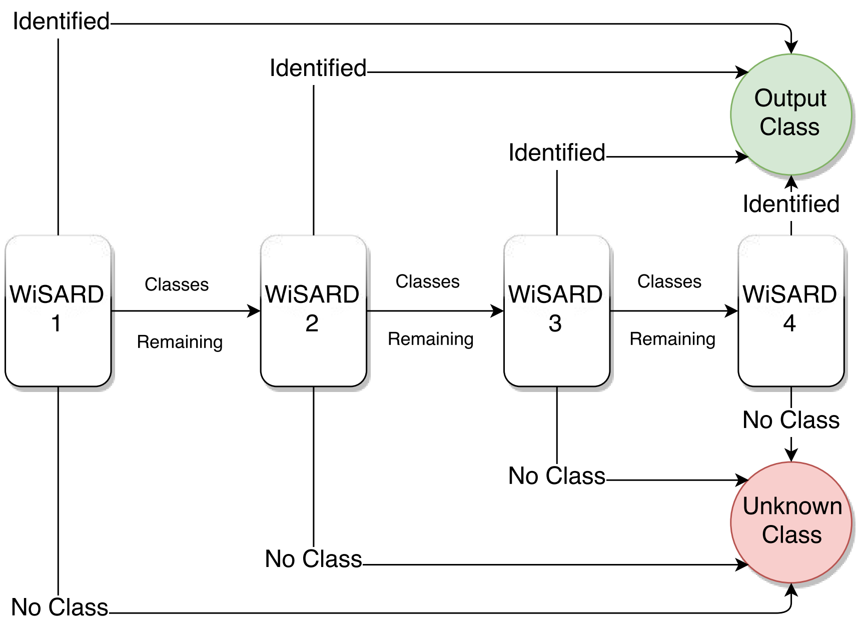


Figure 3.10: The decision directed acyclic graph approach with 4 WiSARD nodes/layers.

- Class identified: terminate execution and output class;
- Tie: either proceed to next layer or if at last layer, output last WiSARD response.

When proceeding to a next layer, only classes among the previously tied classes are considered as valid output classes. Algorithm 2 tries to put this strategy into picture together with Figure 3.10.

Algorithm 2: The Decision DAG Algorithm

```
input  : layers, testSample
output: output
filteredPartialResult ←null;
output ←NOTFOUND;
foreach layer l of pretrained layers L do
    partialResult ← PredictClass(l,testSample);
    filteredPartialResult ←
     FilterOnlyPreviouslyPredictedClasses(partialResult);
    if filteredPartialResult has only one class then
        if filteredPartialResult[0]== NOTFOUND then
            break;
        else
            output ←filteredPartialResult[0]
        end
    else if filteredPartialResult has more classes then
        output ← BreakTie(filteredPartialResult)
    else
        break;
    end
end
```

Chapter 4

Experiments, results and discussion

The methodology and proposals presented above were tested with some experiments. All these experiments were run having the same goal: identify buses routes given vehicles' trajectory data.

4.1 Dataset definition & Problem parameters

In these experiments, it was collected a month of data between June and July of 2015. This data was then filtered and pre-processed, resulting in the datasets where the classifiers were applied. In order to filter the data, measures without route tag (to train a supervised classifier) and measures in regard to routes that are not necessarily are part of the current system[1] were removed.

Concerning the pre-processing step, initial images were generated out of the measures of a vehicle (id) on a single day, following the *footprint* strategy (as presented in Section 3.2.1). Plotting parameters were default parameters given by *python* library *Matplotlib*[80], producing black and white images of 620x480 pixels. Those images were converted to binary vectors of same size or, in other words, 297600 bits.

This initial dataset D_1 had 240192 samples, involving 517 different classes (bus routes), with a highly imbalanced distribution among them (see 4.1 for an illustration). Dataset D_1 size is up to 67GB when stored using an ASCII encoding format.

Several datasets were then produced out of D_1, using the strategy mentioned in Section 3.2.2, based on a kd-tree spatial division. Several criteria were tested to generate these datasets. In respect to the *Next node* criteria, there were three evaluated:

[1]Although it may seem strange, there are several measures presenting route tags that make no sense to the bus system. Others have a special prefix, representing experimental routes

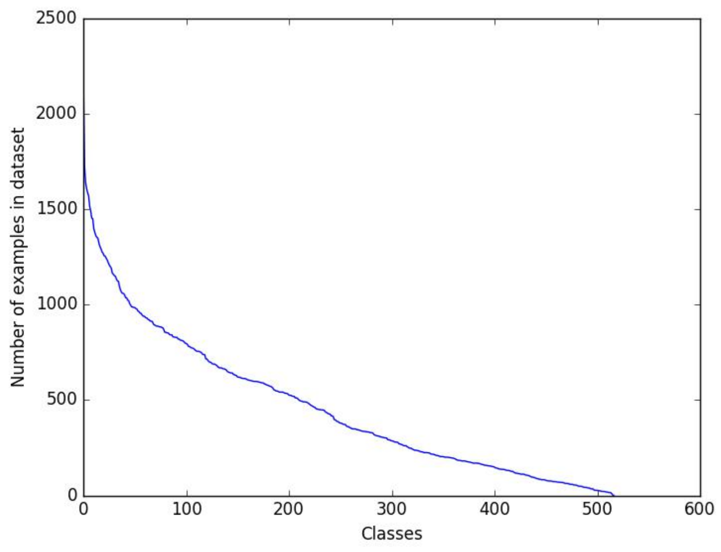


Figure 4.1: Class distribution in dataset D_1: almost a power law.

- **Size**: Largest size of node or how many points inside a node (S);
- **Density**: Greatest density of a node (D) - with density being defined as number of points divided by the lat/long boundary of a node;
- **Difference to Mean Density**: Largest absolute difference to the mean density of the current node set (DMD).

Two *Splitting criteria* were also tested:

- **Median** (M): split the set at the median of the measures inside a node (latitude or longitude - depending on the dimension being divided);
- **Half** (H): split the set at the middle point of node boundaries (latitude or longitude - also depending on the dimension being divided).

Yet, for every combination of the two above-mentioned criteria, four different grid/kd-tree sizes (**256**, **1024**, **4096** and **16384**) were tried, which were consequently the number of bits of each input vector. The grid size (or number of leaves) was used as *Stopping* criteria for the kd-tree growth algorithm as well.

Multiple WiSARD configurations were considered for the simple footprint approach and the kd-tree grid reencoded image. Once again, it is important to note that the number of neurons given a fixed-size retina determined also the number of bits used as addresses. In other words, every time X neurons are used, if a input vector of B bits is used, this leads to the definition of X tuples of n bits, where $n = B/X$ (assuming that $B \mid X$).

Solution	Grid Size	# Neurons	F1 score
Original WiSARD	297600 (620x420)	64	0.7815 ± 0.004
Best WiSARD-kdtree (Density/Half)	16384	32	0.7896 ± 0.0009
2nd Best WiSARD-kdtree (Difference to Mean Density/Half)	16384	64	0.7843 ± 0.0010
Best WiSARD-DDAG (Size/Half)	256/1024/4096/16384	4/16/32/64	0.7590 ± 0.0006

Table 4.1: Summary of each WiSARD strategy and their best results for the first experiment

In order to evaluate the WiSARD-DDAG approach (from Section 3.3), a graph with 4 classifier nodes was considered (one for each grid size we tested before – 256, 1024, 4096, 16384). In addition, for each kd-tree generation combination one WiSARD-DDAG was experimented, yielding a final test with six WiSARD-DDAGs. Each classifier node (layer) was built using the best individual result and parameters. In other words, for a given strategy and a given grid size, if WiSARD W_i had the best result using n_j neurons, it was used with n_j neurons in the referred DDAG.

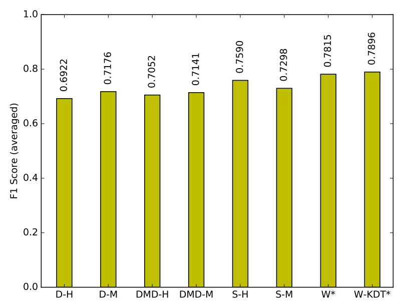
The performed experimental procedure was composed of 5 rounds of a 10-fold cross-validation. As an example, if a twenty thousand samples dataset were used, each of its ten (sub)rounds of training and testing would have 18 thousand observations used as training data, while the remaining 2 thousand examples would be used to assess classification effectiveness of the tested alternatives. Also in this regard, the macro-averaged F1 score [81] was chosen as performance measure due to the imbalanced nature of the data set. All experiments were ran on an Intel(R) Xeon(R) server, with an E5-2630 2.40GHz CPU and 32GB RAM.

4.2 First Experiment using full dataset

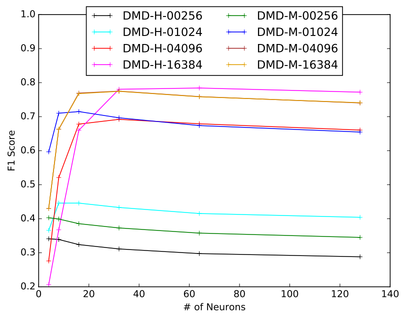
An initial experiment was conducted using the D_1 full dataset for the standard WiSARD model and the respective full dataset for each grid size and strategy combination (Size-Median, Size-Half, DMD-Median, DMD-Half, Density-Half and Density-Median) for the two proposed approaches (WiSARD-kd-tree and WiSARD-DDAG). Each dataset was tested using from 4 to 128 neurons in each discriminator.

Table 4.1 presents the winners for each approach. Figure 4.2 shows the performance evolution when the numbers of neurons increases. In general, performance starts increasing until a peak performance is reached and after that the model starts to deteriorate.

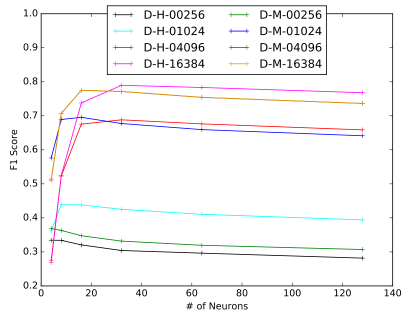
As the reader can see, the best overall result was obtained using WiSARD-kd-tree, with the "Density-Half"grid formation strategy and sixty four neurons. It is interesting to notice that the best model using WiSARD-DDAG did not perform better than the original grid, thus giving no f1-score increase over it.



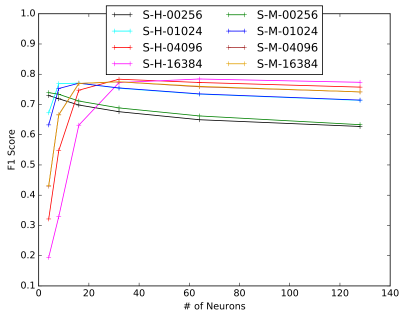
(a) WiSARD-DDAG approach results with others WiSARD for comparison - best WiSARD (W*) and best WiSARD-kd-tree (W-KDT*)



(b) WiSARD-kd-tree ('DMD') results



(c) WiSARD-kd-tree ('Density') results



(d) WiSARD-kd-tree ('Size') results

Figure 4.2: Results of the first experiment

4.3 Second Experiment - A Comparison with other classifiers

A second experiment was conducted using subsets of twenty thousand samples of D_1 and the others respective datasets for each grid size and strategy combination (Size-Median, Size-Half, DMD- Median, DMD- Half, Density-Half and Density-Median) for the two proposed approaches (WiSARD-kd-tree and WiSARD-DDAG). Each dataset was tested using from 4 to 256 neurons in each discriminator.

Also, three widely used classifiers were chosen as baselines for our experiment: Random Forests, Extremely Randomized Trees and Naive Bayes (here adapted for a black-and-white input). All these comparisons were performed using implementations obtained from the Scikit Learn [55] package with their default configurations and parameters.

Table 4.2 highlights the top result for each WiSARD approach (out of 181 combinations) as well the baselines compared (Extra Trees, Random Forest and Naive Bayes) ordered by F1-Score[2]. The best overall result was obtained using the kd-tree

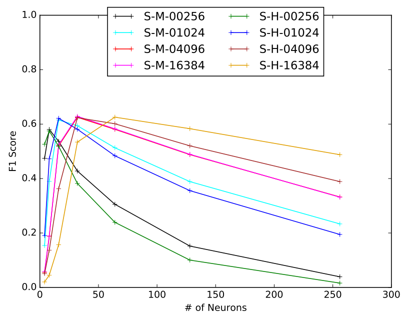
[2]A complete table with all experiments is in appendix A

Solution	Grid Size	# Neurons	F1 score
WiSARD-kd-tree	16384	64	0.6324 ± 0.0043
WiSARD-DDAG	256/1024/4096/16384	8/16/32/64	0.6209 ± 0.0035
WiSARD	297600 (620x480)	64	0.5989 ± 0.0042
Ext. Rand. Trees	297600 (620x480)	N/A	0.5983 ± 0.0070
Random Forests	297600 (620x480)	N/A	0.5825 ± 0.0067
Naive Bayes	297600 (620x480)	N/A	0.0269 ± 0.0031

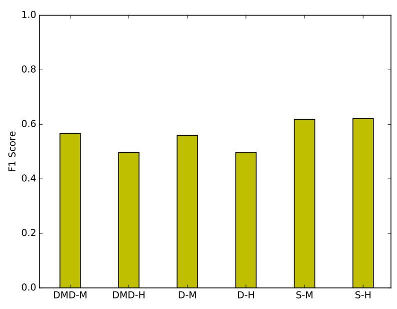
Table 4.2: Ranking of WiSARD and its derived methods, as well as the baseline alternatives. The reported results regard the best configuration of each method, found by grid-searching through their possible settings. A total of 184 different instances of all classifiers was evaluated.

approach, using a grid with 16384 cells and using the policy to split at the half the node with the greatest density.

An aspect denoted by Figure 4.3 and Figure 4.4 is again the evolution of performance when the number of neurons increases, as all tested architectures seem to reach a peak performance followed by slightly decreases. In addition, the reader can notice some correlation between a grid size (effectively the size of the input) and the number of neurons where the best performance is achieved. Smaller grids tend to reach its peak performance using fewer neurons.



(a) WiSARD-kd-tree ('Size') results



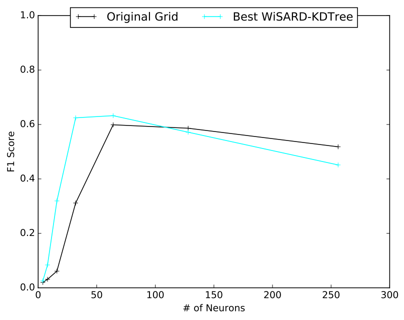
(b) WiSARD-DDAG results

Figure 4.3: Results of the second experiment

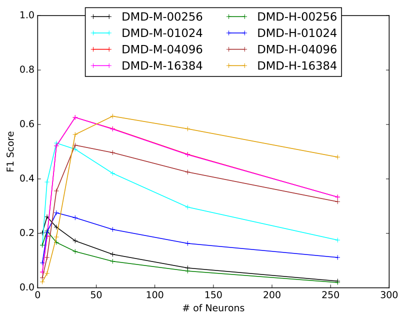
When comparing the performance on F1-Score of different classifiers, as depicted by Figure 4.5 and Figure 4.6, using the WiSARD-kd-tree approach consistently achieves better results. ExtraTrees and Random Forests have similar performance, with the latter reaching a slightly lower performance than the former and with Naive Bayes being definitely not a good option.

Moreover, it is noticeable the difference of score comparing grid sizes. Both size-driven policies perform relatively similar even using few cells when other strategies tend to show progress when increasing their grid sizes. This might indicate that size-driven grids are more representative of the problem space then other grids.

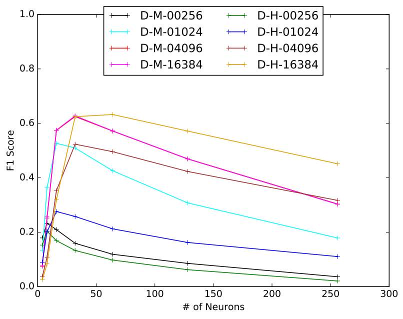
Concluding, the DDAG approach presents an impressive performance difference



(a) Original WiSARD results (no grid) w/ Best KD-Tree approach comparison



(b) WiSARD-kd-tree ('DMD') results



(c) WiSARD-kd-tree ('Density') results

Figure 4.4: Results of the second experiment (continued)

depending on the grid strategy, which might be related to the representativeness of the grids. For example, having poor grids at a small granularity ('Density/Half' and 'DMD/Half') severely decreases performance, which is not perceived when size-driven grids are used. The ensemble method was not able to surpass a single kd-tree approach, but it achieved comparable performance.

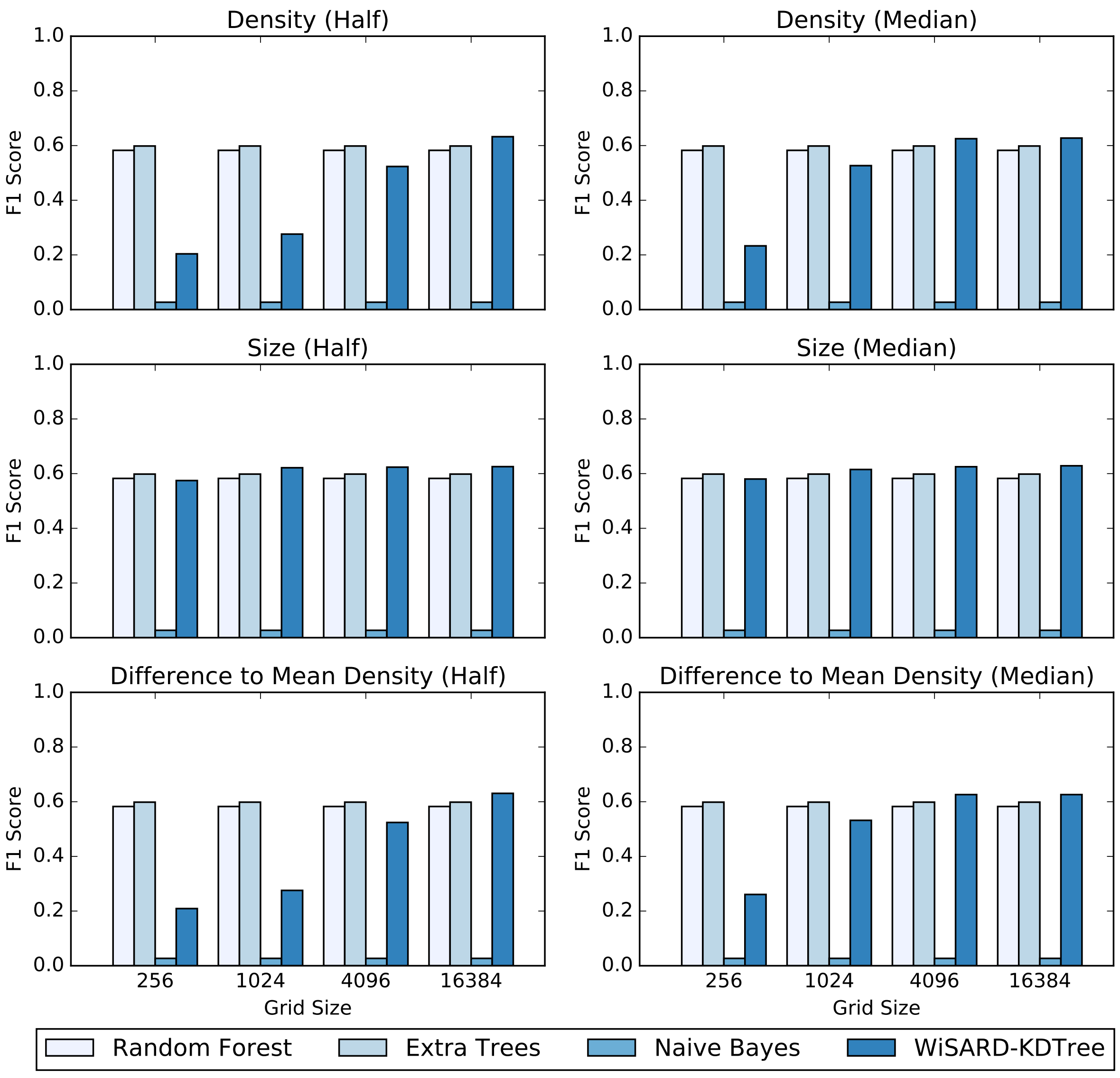


Figure 4.5: Comparing Results: WiSARD–kd-tree (all grids) vs. Extremely Randomized Trees (ExtraTrees) vs Random Forests vs Naive Bayes

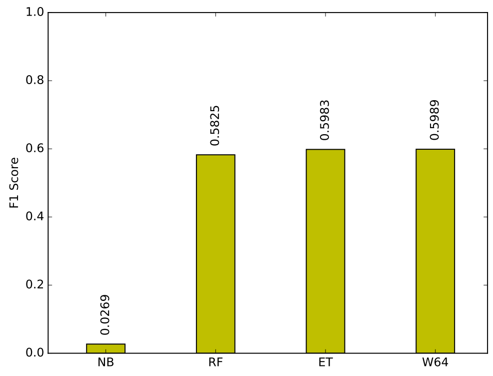


Figure 4.6: Comparing Results: WiSARD with the original grid using 128 neurons (W128) vs. Extremely Randomized Trees (ET) vs Random Forests (RF) vs Naive Bayes (NB)

Chapter 5

Conclusions and Future Works

This works aims to contribute to the literature of machine learning and knowledge discovery from geographical location data. More than ever this is an interesting and active research area given the current availability of data of this kind due to the ubiquity of sensors supporting its gathering. In this regard, our research was based on a very large collection of GPS trajectories from buses in the city of Rio de Janeiro. The city bus system is in a chaotic situation and identifying labelling inconsistencies through proper classification of bus routes could be a valuable asset for city management, specially in regard to traffic optimization and urban engineering.

In this text it was introduced a neuro-symbolic framework for classification based on GPS data. This framework can be a viable solution to the problem of identifying the route of running buses of the city of Rio de Janeiro based of their GPS data. The WiSARD artificial neural network model is the kernel of such tool, contributing with some of its patent strengths as accuracy and reduced computational cost. Different pre-processing procedures were explored, namely gridding strategies, to transform raw geolocation data into WiSARD-friendly binary inputs. Both WiSARD-based neuro-symbolic learning, as well as binarization techniques compatible with this model have been approached previously, and this work provides novel findings for both subjects.

The experimental evaluation of the methods proposed showed that its performance is generally at least as good compared to that of other also lightweight, widely used and state-of-the-art classifiers. That is, the WiSARD-based solution has the best overall F1-score, but it is precipitate to declare it as an overall winner.

Since using a kdtree grid led to the best performances, testing other tree structures in this regard is an interesting follow-up of this work. R-trees are also widely used in the database world, especially when dealing with geospatial queries. Other classifiers might benefit from using a kd-tree quantization instead of a raw image approach as well. Although there is some evidence, is was out of the scope of this work to evaluate this, since it needs to be better investigated.

Also, the presented framework can be an alternative when dealing with large data masses and it can be adapted to solve other similar problems such as rogue vehicle detection. The footprint image and its grid variants can still be explored, since parameters like grid size and observation window were not exhausted in this work. Moreover, to further investigate the neuro-symbolic, the DDAG-based learning process is another interesting future work, possibly considering the addition of parameters such as a confidence threshold, more nodes and different grid granularities.

This work is just a step towards a complete solution. The focus was constrained to a closed class set classification, but there is still room to look at the problem through an open-set classification perspective. Also, online classification of running buses is desirable, whenever it is related to real time classification of the API responses or to the ability to learn new and forget unused routes automatically, as a human-independent solution.

Bibliography

[1] ZHENG, Y. "Trajectory Data Mining: An Overview", *ACM Trans. Intell. Syst. Technol.*, v. 6, n. 3, pp. 29:1–29:41, maio 2015. ISSN: 2157-6904. 1, 14, 19, 20

[2] FENG, Z., ZHU, Y. "A survey on trajectory data mining: Techniques and applications", *IEEE Access*, v. 4, pp. 2056–2067, 2016. 1

[3] ZHENG, Y., LIU, L., WANG, L., et al. "Learning transportation mode from raw gps data for geographic applications on the web". In: *Proceedings of the 17th international conference on World Wide Web*, pp. 247–256. ACM, 2008. 1

[4] XIAO, Z., WANG, Y., FU, K., et al. "Identifying Different Transportation Modes from Trajectory Data Using Tree-Based Ensemble Classifiers", *ISPRS International Journal of Geo-Information*, v. 6, n. 2, pp. 57, 2017. 1, 3

[5] LEE, J.-G., HAN, J., LI, X., et al. "TraClass: trajectory classification using hierarchical region-based and trajectory-based clustering", *Proceedings of the VLDB Endowment*, v. 1, n. 1, pp. 1081–1094, 2008. 1, 25

[6] PATEL, D., SHENG, C., HSU, W., et al. "Incorporating duration information for trajectory classification". In: *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, pp. 1132–1143. IEEE, 2012. 1

[7] PATTERSON, D. J., LIAO, L., FOX, D., et al. "Inferring high-level behavior from low-level sensors". In: *UbiComp*, pp. 73–89. Springer, 2003. 1

[8] BOLBOL, A., CHENG, T., TSAPAKIS, I., et al. "Inferring hybrid transportation modes from sparse GPS data using a moving window SVM classification", *Computers, Environment and Urban Systems*, v. 36, n. 6, pp. 526–537, 2012. 1

[9] MENDES, T. "Ônibus do Rio passam a ser monitorados em tempo real", *Jornal O GLOBO*, Sep 2013. Disponível em: <`https://oglobo.globo.com/rio/`

onibus-do-rio-passam-ser-monitorados-em-tempo-real-9785747>. Accessed 24 January 2018. 1

[10] "PROJETO DE LEI Nº 303/2013". jun. 2013. Disponível em: <http://mail.camara.rj.gov.br/APL/Legislativos/scpro1316.nsf/249cb321f17965260325775900523a42/9a0143dcbd61a91e03257b87004bbc73?OpenDocument>. Accessed 24 January 2018 (In Portuguese). 1

[11] "LEI Nº 5550/2013". jan. 2013. Disponível em: <http://mail.camara.rj.gov.br/APL/Legislativos/contlei.nsf/50ad008247b8f030032579ea0073d588/d5607fc3fc01cbc703257c5f0055b039?OpenDocument&Highlight=0,PROGRAMA,PLURIANUAL>. Accessed 24 January 2018 (In Portuguese). 1

[12] PADRÓN, G., CRISTÓBAL, T., ALAYÓN, F., et al. "System Proposal for Mass Transit Service Quality Control Based on GPS Data", *Sensors*, v. 17, n. 6, pp. 1412, 2017. 2

[13] TONG, C., CHEN, H., XUAN, Q., et al. "A Framework for Bus Trajectory Extraction and Missing Data Recovery for Data Sampled from the Internet", *Sensors*, v. 17, n. 2, pp. 342, 2017.

[14] FIORI, A., MIGNONE, A., ROSPO, G. "DeCoClu: Density consensus clustering approach for public transport data", *Information Sciences*, v. 328, pp. 378–388, 2016. 2

[15] DO AMARAL, B. G., NASSER, R., CASANOVA, M. A., et al. "BusesinRio: Buses as mobile traffic sensors: Managing the bus GPS data in the city of Rio de Janeiro". In: *Mobile Data Management (MDM), 2016 17th IEEE International Conference on*, v. 1, pp. 369–372. IEEE, 2016. 2

[16] RODRIGUEZ, K., CASANOVA, M. A., LEME, L. A. P. P., et al. "On the Design of a Traffic Observatory Application based on Bus Trajectories." In: *ICEIS (1)*, pp. 215–222, 2016. 2

[17] ALEKSANDER, I., THOMAS, W., BOWDEN, P. "WISARD a radical step forward in image recognition", *Sensor review*, v. 4, n. 3, pp. 120–124, 1984. 2, 10

[18] CARNEIRO, H. C., PEDREIRA, C. E., FRANÇA, F. M., et al. "A universal multilingual weightless neural network tagger via quantitative linguistics", *Neural Networks*, v. 91, pp. 85 – 101, 2017. 2

[19] CARDOSO, D. O., FRANÇA, F. M. G., GAMA, J. "WCDS: A Two-Phase Weightless Neural System for Data Stream Clustering", *New Generation Computing*, pp. Accepted, to be published, online available, 2017.

[20] GREGORIO, M. D., GIORDANO, M. "Background estimation by weightless neural networks", *Pattern Recognition Letters*, v. 96, n. Supplement C, pp. 55 – 65, 2017. ISSN: 0167-8655. doi: https://doi.org/10.1016/j.patrec.2017.05.029. Disponível em: <`http://www.sciencedirect.com/science/article/pii/S0167865517301927`>. Scene Background Modeling and Initialization. 2

[21] HAMMER, B., HITZLER, P. *Perspectives of neural-symbolic integration*, v. 8. Springer Heidelberg:, 2007. 3

[22] BARBOSA, R., CARVALHO, D., CARDOSO, D., et al. "A neuro-symbolic approach to GPS trajectory classification". In: *ESANN*, 2017. 3

[23] BARBOSA, R., CARVALHO, D., CARDOSO, D., et al. "Weightless neuro-symbolic GPS trajectory classification", *Neurocomputing*, 2018. 3

[24] LIMA, P. M. "A goal-driven neural propositional interpreter", *International Journal of Neural Systems*, v. 11, n. 03, pp. 311–322, 2001. 3

[25] CORDEIRO, M. C., PINTO, N. L., CARVALHO, D. "DESENVOLVIMENTO DE UMA FERRAMENTA DE AVALIAÇÃO DA FROTA DE ÔNIBUS DISPONÍVEL PARA ATENDIMENTO DAS DEMANDAS POPULACIONAIS". In: *XXXVI ENCONTRO NACIONAL DE ENGENHARIA DE PRODUCÃO*, 2016. 4, 6

[26] "Projeções dos modais". ago. 2014. Disponível em: <`http://www.fetranspor.com.br/wp\penalty\@M\hskip\z@skip\discretionary{-}{}{}\penalty\@M\hskip\z@skipcontent/uploads/2014/08/transporte_coletivo.pdf(Accessedon24January2018)`>. Accessed 24 January 2018 (In Portuguese). 4

[27] "Dados operacionais mensais do Município do Rio de Janeiro". maio 2017. Disponível em: <`http://www.fetranspordocs.com.br/downloads/TB4_DadosoperacionaismensaisdomunicipiodoRio_anode2016.xls`>. Accessed 24 January 2018 (In Portuguese). 4

[28] "CONCORRÊNCIA N° CO 10/2010". jun. 2010. Disponível em: <`http://www.rio.rj.gov.br/dlstatic/10112/4800832/4128521/EDITAL.pdf`>. Accessed 24 January 2018 (In Portuguese). 5

[29] "Página principal". ago. 2014. Disponível em: <http://www.rioonibus.com/rio-onibus/consorcios-e-empresas/>. Accessed 24 January 2018 (In Portuguese). 5

[30] "CONCORRÊNCIA Nº CO 10/2010 - ANEXO I". jun. 2010. Disponível em: <http://www.rio.rj.gov.br/dlstatic/10112/4800832/4128532/ANEXOI.pdf>. Accessed 24 January 2018 (In Portuguese). 6

[31] "GPS Overview". 2018. Disponível em: <https://www.gps.gov/systems/gps/>. Accessed 24 January 2018. 7, 8

[32] WOODEN, W. H. "Navstar Global Positioning System: 1985". In: *Proceedings of the First International Symposium on Precise Positioning with the Global Positioning System*, pp. 23–32, 1985. 8

[33] HOFMANN-WELLENHOF, B., LICHTENEGGER, H., WASLE, E. *GNSS–global navigation satellite systems: GPS, GLONASS, Galileo, and more.* Springer Science & Business Media, 2007. 8, 9

[34] VAN DIGGELEN, F., ENGE, P. "The worlds first gps mooc and worldwide laboratory using smartphones". In: *Proceedings of the 28th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2015)*, pp. 361–369, 2015. 8

[35] "GPS Overview". 2018. Disponível em: <https://www.gps.gov/multimedia/poster/poster-web.pdf>. Accessed 24 January 2018. 8

[36] "Cartopy projections list". 2018. Disponível em: <http://scitools.org.uk/cartopy/docs/v0.15/crs/projections.html>. Accessed 24 January 2018. 10

[37] FRANÇA, F., DE GREGORIO, M., LIMA, P., et al. "Advances in Weightless Neural Systems". In: *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, pp. 497–504, 2014. 10, 12, 15

[38] MCCULLOCH, W. S., PITTS, W. "A logical calculus of the ideas immanent in nervous activity", *The bulletin of mathematical biophysics*, v. 5, n. 4, pp. 115–133, Dec 1943. doi: 10.1007/BF02478259. Disponível em: <https://doi.org/10.1007/BF02478259>. 10

[39] BLEDSOE, W. W., BROWNING, I. "Pattern Recognition and Reading by Machine". In: *Proceedings of Eastern Joint Computer Conference*, pp. 225–232, 1959. 11

[40] ABU-MOSTAFA, Y. S., MAGDON-ISMAIL, M., LIN, H.-T. *Learning from data*, v. 4. AMLBook New York, NY, USA:, 2012. 11

[41] CARDOSO, D. O., GAMA, J., FRANÇA, F. M. G. "Weightless neural networks for open set recognition", *Machine Learning*, pp. Accepted, to be published, online available, Jul 2017. 11, 14

[42] LINNEBERG, C., JORGENSEN, T. M. "Discretization methods for encoding of continuous input variables for Boolean neural networks". In: *Neural Networks, 1999. IJCNN'99. International Joint Conference on*, v. 2, pp. 1219–1224. IEEE, 1999. 12

[43] KOLCZ, A., ALLINSON, N. "Application of the CMAC input encoding scheme in the N-tuple approximation network", *IEE Proceedings-Computers and Digital Techniques*, v. 141, n. 3, pp. 177–183, 1994.

[44] KAPPAUN, A., CAMARGO, K., RANGEL, F., et al. "Evaluating Binary Encoding Techniques for WiSARD". In: *Intelligent Systems (BRACIS), 2016 5th Brazilian Conference on*, pp. 103–108. IEEE, 2016. 12, 13

[45] CHAKI, N., SHAIKH, S. H., SAEED, K. *Exploring Image Binarization Techniques*, v. 560. Springer, 2014. 12

[46] GUYON, I., GUNN, S., NIKRAVESH, M., et al. *Feature extraction: foundations and applications*, v. 207. Springer, 2008. 14, 21

[47] GRIECO, B. P., LIMA, P. M., DE GREGORIO, M., et al. "Producing pattern examples from "mental" images", *Neurocomputing*, v. 73, n. 7, pp. 1057–1064, 2010. 14

[48] SOARES, C. M., DA SILVA, C. L. F., GREGORIO, M. D., et al. "Uma implementação em software do classificador WISARD". In: *V Simpósio Brasileiro de Redes Neurais*, v. 2, pp. 225—229, 1998. 14

[49] CARVALHO, D. S., CARNEIRO, H. C., FRANÇA, F. M., et al. "B-bleaching: Agile Overtraining Avoidance in the WiSARD Weightless Neural Classifier." In: *ESANN*, 2013. 14

[50] SAFAVIAN, S. R., LANDGREBE, D. "A survey of decision tree classifier methodology", *IEEE Transactions on Systems, Man, and Cybernetics*, v. 21, n. 3, pp. 660–674, May 1991. ISSN: 0018-9472. doi: 10.1109/21.97458. 15

[51] WU, X., KUMAR, V., QUINLAN, J. R., et al. "Top 10 algorithms in data mining", *Knowledge and information systems*, v. 14, n. 1, pp. 1–37, 2008. 15, 16

[52] TAN, P.-N., STEINBACH, M., KUMAR, V. "Introduction to data mining. 1st". 2005. 15, 16, 17, 18

[53] BREIMAN, L., FRIEDMAN, J., STONE, C. J., et al. *Classification and regression trees*. CRC press, 1984. 16

[54] QUINLAN, R. J. "C4. 5: Programs for Machine Learning". 1993. 16

[55] PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., et al. "Scikit-learn: Machine Learning in Python", *Journal of Machine Learning Research*, v. 12, pp. 2825–2830, 2011. 16, 32

[56] HALL, M., FRANK, E., HOLMES, G., et al. "The WEKA data mining software: an update", *ACM SIGKDD explorations newsletter*, v. 11, n. 1, pp. 10–18, 2009. 16

[57] HYAFIL, L., RIVEST, R. L. "Constructing optimal binary decision trees is NP-complete", *Information processing letters*, v. 5, n. 1, pp. 15–17, 1976. 16

[58] HO, T. K. "Random decision forests". In: *Proceedings of 3rd International Conference on Document Analysis and Recognition*, v. 1, pp. 278–282 vol.1, Aug 1995. doi: 10.1109/ICDAR.1995.598994. 17

[59] HO, T. K. "The random subspace method for constructing decision forests", *IEEE transactions on pattern analysis and machine intelligence*, v. 20, n. 8, pp. 832–844, 1998. 17

[60] BREIMAN, L. "Random forests", *Machine Learning*, v. 45, n. 1, pp. 5–32, 2001. 17

[61] BREIMAN, L. "Bagging predictors", *Machine learning*, v. 24, n. 2, pp. 123–140, 1996. 17

[62] RIIHIJARVI, J., MAHONEN, P. "Machine Learning for Performance Prediction in Mobile Cellular Networks", *IEEE Computational Intelligence Magazine*, v. 13, n. 1, pp. 51–60, Feb 2018. ISSN: 1556-603X. doi: 10.1109/MCI.2017.2773824. 17

[63] BOSCH, A., ZISSERMAN, A., MUNOZ, X. "Image Classification using Random Forests and Ferns". In: *2007 IEEE 11th International Conference on Computer Vision*, pp. 1–8, Oct 2007. doi: 10.1109/ICCV.2007.4409066. 17

[64] GEURTS, P., ERNST, D., WEHENKEL, L. "Extremely randomized trees", *Machine Learning*, v. 63, n. 1, pp. 3–42, 2006. 17

[65] AMOR, N. B., BENFERHAT, S., ELOUEDI, Z. "Naive bayes vs decision trees in intrusion detection systems". In: *Proceedings of the 2004 ACM symposium on Applied computing*, pp. 420–424. ACM, 2004. 18

[66] LEWIS, D. "Naive (Bayes) at forty: The independence assumption in information retrieval", *Machine Learning: ECML-98*, pp. 4–15, 1998. 18

[67] DE SOUZA, D. F. P. *TIME-SERIES CLASSIFICATION WITH KERNEL-CANVAS AND WISARD.* Tese de Mestrado, Universidade Federal do Rio de Janeiro, 2015. 20

[68] WANG, L., ZHENG, Y., XIE, X., et al. "A flexible spatio-temporal indexing scheme for large-scale GPS track retrieval". In: *Mobile Data Management, 2008. MDM'08. 9th International Conference on*, pp. 1–8. IEEE, 2008. 20

[69] GE, Y., XIONG, H., ZHOU, Z.-H., et al. "Top-eye: Top-k evolving trajectory outlier detection". In: *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pp. 1733–1736. ACM, 2010. 20

[70] JOHNSON JR, P. D., HARRIS, G. A., HANKERSON, D. *Introduction to information theory and data compression.* CRC press, 2003. 20

[71] GUYON, I., ELISSEEFF, A. "An introduction to variable and feature selection", *Journal of machine learning research*, v. 3, n. Mar, pp. 1157–1182, 2003. 21

[72] MANNING, C. D., RAGHAVAN, P., SCHÜTZE, H., et al. *Introduction to information retrieval*, v. 1. Cambridge university press Cambridge, 2008. 21

[73] JOLLIFFE, I. T. "Principal Component Analysis and Factor Analysis". In: *Principal Component Analysis*, pp. 115–128, New York, NY, Springer New York, 1986. ISBN: 978-1-4757-1904-8. doi: 10.1007/978-1-4757-1904-8_7. Disponível em: <`https://doi.org/10.1007/978-1-4757-1904-8_7`>. 21

[74] GOLUB, G. H., REINSCH, C. "Singular value decomposition and least squares solutions", *Numerische mathematik*, v. 14, n. 5, pp. 403–420, 1970. 21

[75] LOWE, D. G. "Distinctive Image Features from Scale-Invariant Keypoints", *International Journal of Computer Vision*, v. 60, n. 2, pp. 91–110, Nov 2004. ISSN: 1573-1405. doi: 10.1023/B:VISI.0000029664.99615.94. Disponível em: <`https://doi.org/10.1023/B:VISI.0000029664.99615.94`>. 21

[76] BENTLEY, J. L. "Multidimensional binary search trees used for associative searching", *Communications of the ACM*, v. 18, n. 9, pp. 509–517, 1975. 22

[77] ESTER, M., KRIEGEL, H.-P., SANDER, J., et al. "A density-based algorithm for discovering clusters in large spatial databases with noise." *Kdd*, v. 96, n. 34, pp. 226–231, 1996. 25

[78] PINKAS, G., LIMA, P., COHEN, S. "Representing, binding, retrieving and unifying relational knowledge using pools of neural binders", *Biologically Inspired Cognitive Architectures*, v. 6, pp. 87 – 95, 2013. 25

[79] PLATT, J. C., CRISTIANINI, N., SHAWE-TAYLOR, J. "Large Margin DAGs for Multiclass Classification." In: *nips*, v. 12, pp. 547–553, 1999. 25

[80] HUNTER, J. D. "Matplotlib: A 2D graphics environment", *Computing In Science & Engineering*, v. 9, n. 3, pp. 90–95, 2007. doi: 10.1109/MCSE.2007.55. 29

[81] SOKOLOVA, M., LAPALME, G. "A systematic analysis of performance measures for classification tasks", *Inf. Process. Manage.*, v. 45, n. 4, pp. 427–437, 2009. 31

Appendix A

Second Experiment Full Results

Rank	Method	# Neurons	F1 Score
1	WiSARD-KDTree (Density/Half/16384)	64	0.6324 ± 0.0043
2	WiSARD-KDTree (Diff. to Mean Density/Half/16384)	64	0.6305 ± 0.0039
3	WiSARD-KDTree (Size/Median/16384)	32	0.6287 ± 0.0034
4	WiSARD-KDTree (Density/Median/16384)	32	0.6274 ± 0.0023
5	WiSARD-KDTree (Diff. to Mean Density/Median/04096)	32	0.6260 ± 0.0036
6	WiSARD-KDTree (Diff. to Mean Density/Median/16384)	32	0.6260 ± 0.0040
7	WiSARD-KDTree (Size/Half/16384)	64	0.6256 ± 0.0038
8	WiSARD-KDTree (Size/Median/04096)	32	0.6252 ± 0.0040
9	WiSARD-KDTree (Density/Median/04096)	32	0.6250 ± 0.0030
10	WiSARD-KDTree (Density/Half/16384)	32	0.6245 ± 0.0041
11	WiSARD-KDTree (Size/Half/04096)	32	0.6237 ± 0.0044
12	WiSARD-KDTree (Size/Half/01024)	16	0.6215 ± 0.0030
13	WiSARD-DDAG (Size/Half)	8/16/32/64	0.6209 ± 0.0035
14	WiSARD-DDAG (Size/Median)	8/16/32/32	0.6180 ± 0.0020
15	WiSARD-KDTree (Size/Median/01024)	16	0.6151 ± 0.0041
16	WiSARD-KDTree (Size/Half/04096)	64	0.6013 ± 0.0032
17	WiSARD	64	0.5989 ± 0.0042
18	Ext. Rand. Trees	N/A	0.5983 ± 0.0070
19	WiSARD-KDTree (Size/Median/01024)	32	0.5941 ± 0.0047
20	WiSARD	128	0.5862 ± 0.0042
21	WiSARD-KDTree (Diff. to Mean Density/Median/04096)	64	0.5843 ± 0.0028
22	WiSARD-KDTree (Diff. to Mean Density/Half/16384)	128	0.5841 ± 0.0020
23	WiSARD-KDTree (Size/Half/16384)	128	0.5833 ± 0.0026
24	WiSARD-KDTree (Diff. to Mean Density/Median/16384)	64	0.5829 ± 0.0039
25	WiSARD-KDTree (Size/Median/16384)	64	0.5828 ± 0.0027
26	Random Forests	N/A	0.5825 ± 0.0067
27	WiSARD-KDTree (Size/Half/01024)	32	0.5812 ± 0.0026
28	WiSARD-KDTree (Size/Median/04096)	64	0.5809 ± 0.0029
29	WiSARD-KDTree (Size/Median/00256)	8	0.5803 ± 0.0029
30	WiSARD-KDTree (Density/Median/16384)	16	0.5750 ± 0.0018

Table A.1: Final ranking (part 1 of 4): Showing from rank 1 to 30. The number of experiments was 184.

Rank	Method	# Neurons	F1 Score
31	WiSARD-KDTree (Size/Half/00256)	8	0.5747 ± 0.0023
32	WiSARD-KDTree (Density/Median/04096)	16	0.5741 ± 0.0049
33	WiSARD-KDTree (Density/Median/16384)	64	0.5720 ± 0.0012
34	WiSARD-KDTree (Density/Median/04096)	64	0.5716 ± 0.0017
35	WiSARD-KDTree (Density/Half/16384)	128	0.5716 ± 0.0028
36	WiSARD-DDAG (Diff. to Mean Density/Median)	8/16/64/32	0.5668 ± 0.0031
37	WiSARD-KDTree (Diff. to Mean Density/Half/16384)	32	0.5631 ± 0.0053
38	WiSARD-DDAG (Density/Median)	8/16/32/32	0.5593 ± 0.0029
39	WiSARD-KDTree (Size/Median/00256)	16	0.5369 ± 0.0033
40	WiSARD-KDTree (Size/Half/16384)	32	0.5340 ± 0.0041
41	WiSARD-KDTree (Diff. to Mean Density/Median/01024)	16	0.5318 ± 0.0031
42	WiSARD-KDTree (Density/Median/01024)	16	0.5265 ± 0.0009
43	WiSARD-KDTree (Size/Half/00256)	4	0.5262 ± 0.0035
44	WiSARD-KDTree (Diff. to Mean Density/Half/04096)	32	0.5240 ± 0.0024
45	WiSARD-KDTree (Density/Half/04096)	32	0.5234 ± 0.0018
46	WiSARD-KDTree (Size/Median/16384)	16	0.5233 ± 0.0071
47	WiSARD-KDTree (Diff. to Mean Density/Median/16384)	16	0.5217 ± 0.0033
48	WiSARD-KDTree (Diff. to Mean Density/Median/04096)	16	0.5215 ± 0.0054
49	WiSARD-KDTree (Size/Half/04096)	128	0.5203 ± 0.0029
50	WiSARD-KDTree (Size/Median/04096)	16	0.5202 ± 0.0045
51	WiSARD-KDTree (Size/Half/00256)	16	0.5188 ± 0.0033
52	WiSARD	256	0.5180 ± 0.0040
53	WiSARD-KDTree (Size/Median/01024)	64	0.5131 ± 0.0043
54	WiSARD-KDTree (Density/Median/01024)	32	0.5094 ± 0.0024
55	WiSARD-KDTree (Diff. to Mean Density/Median/01024)	32	0.5091 ± 0.0026
56	WiSARD-DDAG (Density/Half)	8/16/32/64	0.4974 ± 0.0052
57	WiSARD-DDAG (Diff. to Mean Density/Half)	8/16/32/64	0.4970 ± 0.0014
58	WiSARD-KDTree (Diff. to Mean Density/Half/04096)	64	0.4966 ± 0.0014
59	WiSARD-KDTree (Density/Half/04096)	64	0.4958 ± 0.0021
60	WiSARD-KDTree (Diff. to Mean Density/Median/04096)	128	0.4904 ± 0.0017
61	WiSARD-KDTree (Size/Median/16384)	128	0.4891 ± 0.0029
62	WiSARD-KDTree (Diff. to Mean Density/Median/16384)	128	0.4890 ± 0.0040
63	WiSARD-KDTree (Size/Median/04096)	128	0.4879 ± 0.0022
64	WiSARD-KDTree (Size/Half/16384)	256	0.4877 ± 0.0042
65	WiSARD-KDTree (Size/Half/01024)	64	0.4837 ± 0.0034
66	WiSARD-KDTree (Diff. to Mean Density/Half/16384)	256	0.4803 ± 0.0040
67	WiSARD-KDTree (Size/Median/00256)	4	0.4743 ± 0.0050
68	WiSARD-KDTree (Size/Half/01024)	8	0.4725 ± 0.0023
69	WiSARD-KDTree (Density/Median/04096)	128	0.4696 ± 0.0031
70	WiSARD-KDTree (Density/Median/16384)	128	0.4692 ± 0.0027
71	WiSARD-KDTree (Density/Half/16384)	256	0.4514 ± 0.0041
72	WiSARD-KDTree (Size/Median/00256)	32	0.4277 ± 0.0023
73	WiSARD-KDTree (Density/Median/01024)	64	0.4258 ± 0.0021
74	WiSARD-KDTree (Diff. to Mean Density/Half/04096)	128	0.4253 ± 0.0021
75	WiSARD-KDTree (Density/Half/04096)	128	0.4232 ± 0.0015
76	WiSARD-KDTree (Diff. to Mean Density/Median/01024)	64	0.4207 ± 0.0042
77	WiSARD-KDTree (Size/Median/01024)	8	0.3896 ± 0.0059
78	WiSARD-KDTree (Size/Half/04096)	256	0.3889 ± 0.0039
79	WiSARD-KDTree (Size/Median/01024)	128	0.3888 ± 0.0038
80	WiSARD-KDTree (Diff. to Mean Density/Median/01024)	8	0.3886 ± 0.0025

Table A.2: Final ranking (part 2 of 4): Showing from rank 31 to 80. The number of experiments was 184.

Rank	Method	# Neurons	F1 Score
81	WiSARD-KDTree (Size/Half/00256)	32	0.3820 ± 0.0037
82	WiSARD-KDTree (Density/Median/01024)	8	0.3636 ± 0.0054
83	WiSARD-KDTree (Size/Half/04096)	16	0.3626 ± 0.0058
84	WiSARD-KDTree (Diff. to Mean Density/Half/04096)	16	0.3558 ± 0.0022
85	WiSARD-KDTree (Size/Half/01024)	128	0.3556 ± 0.0087
86	WiSARD-KDTree (Density/Half/04096)	16	0.3529 ± 0.0023
87	WiSARD-KDTree (Diff. to Mean Density/Median/04096)	256	0.3335 ± 0.0040
88	WiSARD-KDTree (Diff. to Mean Density/Median/16384)	256	0.3333 ± 0.0033
89	WiSARD-KDTree (Size/Median/04096)	256	0.3324 ± 0.0056
90	WiSARD-KDTree (Size/Median/16384)	256	0.3316 ± 0.0057
91	WiSARD-KDTree (Density/Half/16384)	16	0.3192 ± 0.0085
92	WiSARD-KDTree (Density/Half/04096)	256	0.3169 ± 0.0049
93	WiSARD-KDTree (Diff. to Mean Density/Half/04096)	256	0.3163 ± 0.0038
94	WiSARD	32	0.3115 ± 0.0027
95	WiSARD-KDTree (Density/Median/01024)	128	0.3077 ± 0.0050
96	WiSARD-KDTree (Size/Median/00256)	64	0.3056 ± 0.0069
97	WiSARD-KDTree (Density/Median/04096)	256	0.3036 ± 0.0038
98	WiSARD-KDTree (Density/Median/16384)	256	0.3027 ± 0.0043
99	WiSARD-KDTree (Diff. to Mean Density/Median/01024)	128	0.2965 ± 0.0028
100	WiSARD-KDTree (Density/Half/01024)	16	0.2761 ± 0.0023
101	WiSARD-KDTree (Diff. to Mean Density/Half/01024)	16	0.2756 ± 0.0043
102	WiSARD-KDTree (Diff. to Mean Density/Median/00256)	8	0.2607 ± 0.0034
103	WiSARD-KDTree (Density/Half/01024)	32	0.2577 ± 0.0028
104	WiSARD-KDTree (Diff. to Mean Density/Half/01024)	32	0.2574 ± 0.0017
105	WiSARD-KDTree (Density/Median/16384)	8	0.2565 ± 0.0060
106	WiSARD-KDTree (Density/Median/04096)	8	0.2521 ± 0.0058
107	WiSARD-KDTree (Size/Half/00256)	64	0.2392 ± 0.0032
108	WiSARD-KDTree (Size/Median/01024)	256	0.2334 ± 0.0053
109	WiSARD-KDTree (Density/Median/00256)	8	0.2331 ± 0.0021
110	WiSARD-KDTree (Diff. to Mean Density/Median/00256)	16	0.2230 ± 0.0024
111	WiSARD-KDTree (Diff. to Mean Density/Half/01024)	64	0.2144 ± 0.0024
112	WiSARD-KDTree (Density/Half/01024)	64	0.2123 ± 0.0025
113	WiSARD-KDTree (Density/Median/00256)	16	0.2095 ± 0.0026
114	WiSARD-KDTree (Diff. to Mean Density/Half/00256)	8	0.2091 ± 0.0019
115	WiSARD-KDTree (Diff. to Mean Density/Half/01024)	8	0.2042 ± 0.0034
116	WiSARD-KDTree (Density/Half/00256)	8	0.2039 ± 0.0021
117	WiSARD-KDTree (Diff. to Mean Density/Median/00256)	4	0.2014 ± 0.0023
118	WiSARD-KDTree (Density/Half/01024)	8	0.2012 ± 0.0022
119	WiSARD-KDTree (Size/Half/01024)	256	0.1946 ± 0.0066
120	WiSARD-KDTree (Diff. to Mean Density/Median/16384)	8	0.1911 ± 0.0055
121	WiSARD-KDTree (Size/Half/01024)	4	0.1907 ± 0.0039
122	WiSARD-KDTree (Size/Median/16384)	8	0.1892 ± 0.0040
123	WiSARD-KDTree (Diff. to Mean Density/Median/04096)	8	0.1883 ± 0.0047
124	WiSARD-KDTree (Size/Median/04096)	8	0.1879 ± 0.0037
125	WiSARD-KDTree (Diff. to Mean Density/Half/16384)	16	0.1866 ± 0.0032
126	WiSARD-KDTree (Density/Median/00256)	4	0.1791 ± 0.0045
127	WiSARD-KDTree (Density/Median/01024)	256	0.1788 ± 0.0033
128	WiSARD-KDTree (Diff. to Mean Density/Median/01024)	256	0.1753 ± 0.0018
129	WiSARD-KDTree (Diff. to Mean Density/Median/00256)	32	0.1726 ± 0.0009
130	WiSARD-KDTree (Density/Half/00256)	16	0.1691 ± 0.0023

Table A.3: Final ranking (part 3 of 4): Showing from rank 81 to 130. The number of experiments was 184.

Rank	Method	# Neurons	F1 Score
131	WiSARD-KDTree (Diff. to Mean Density/Half/00256)	16	0.1668 ± 0.0033
132	WiSARD-KDTree (Diff. to Mean Density/Half/01024)	128	0.1630 ± 0.0032
133	WiSARD-KDTree (Density/Half/01024)	128	0.1623 ± 0.0040
134	WiSARD-KDTree (Density/Median/00256)	32	0.1593 ± 0.0012
135	WiSARD-KDTree (Size/Half/16384)	16	0.1568 ± 0.0044
136	WiSARD-KDTree (Diff. to Mean Density/Median/01024)	4	0.1567 ± 0.0038
137	WiSARD-KDTree (Diff. to Mean Density/Half/00256)	4	0.1566 ± 0.0010
138	WiSARD-KDTree (Size/Median/01024)	4	0.1540 ± 0.0027
139	WiSARD-KDTree (Density/Half/00256)	4	0.1526 ± 0.0019
140	WiSARD-KDTree (Size/Median/00256)	128	0.1520 ± 0.0046
141	WiSARD-KDTree (Size/Half/04096)	8	0.1368 ± 0.0023
142	WiSARD-KDTree (Diff. to Mean Density/Half/00256)	32	0.1335 ± 0.0010
143	WiSARD-KDTree (Density/Half/00256)	32	0.1329 ± 0.0012
144	WiSARD-KDTree (Density/Median/01024)	4	0.1318 ± 0.0049
145	WiSARD-KDTree (Diff. to Mean Density/Median/00256)	64	0.1230 ± 0.0021
146	WiSARD-KDTree (Density/Median/00256)	64	0.1189 ± 0.0012
147	WiSARD-KDTree (Diff. to Mean Density/Half/01024)	256	0.1114 ± 0.0023
148	WiSARD-KDTree (Diff. to Mean Density/Half/04096)	8	0.1113 ± 0.0033
149	WiSARD-KDTree (Density/Half/01024)	256	0.1105 ± 0.0023
150	WiSARD-KDTree (Density/Half/04096)	8	0.1082 ± 0.0050
151	WiSARD-KDTree (Size/Half/00256)	128	0.1004 ± 0.0055
152	WiSARD-KDTree (Density/Half/00256)	64	0.0976 ± 0.0018
153	WiSARD-KDTree (Diff. to Mean Density/Half/00256)	64	0.0973 ± 0.0021
154	WiSARD-KDTree (Diff. to Mean Density/Half/01024)	4	0.0915 ± 0.0031
155	WiSARD-KDTree (Density/Half/01024)	4	0.0895 ± 0.0023
156	WiSARD-KDTree (Density/Median/00256)	128	0.0855 ± 0.0025
157	WiSARD-KDTree (Density/Half/16384)	8	0.0839 ± 0.0027
158	WiSARD-KDTree (Density/Median/16384)	4	0.0773 ± 0.0021
159	WiSARD-KDTree (Density/Median/04096)	4	0.0748 ± 0.0026
160	WiSARD-KDTree (Diff. to Mean Density/Median/00256)	128	0.0729 ± 0.0005
161	WiSARD-KDTree (Density/Half/00256)	128	0.0623 ± 0.0015
162	WiSARD-KDTree (Diff. to Mean Density/Half/00256)	128	0.0616 ± 0.0016
163	WiSARD	16	0.0613 ± 0.0012
164	WiSARD-KDTree (Size/Median/16384)	4	0.0581 ± 0.0023
165	WiSARD-KDTree (Diff. to Mean Density/Median/16384)	4	0.0579 ± 0.0038
166	WiSARD-KDTree (Diff. to Mean Density/Median/04096)	4	0.0578 ± 0.0014
167	WiSARD-KDTree (Size/Median/04096)	4	0.0550 ± 0.0016
168	WiSARD-KDTree (Diff. to Mean Density/Half/16384)	8	0.0528 ± 0.0020
169	WiSARD-KDTree (Size/Half/04096)	4	0.0508 ± 0.0012
170	WiSARD-KDTree (Size/Half/16384)	8	0.0450 ± 0.0016
171	WiSARD-KDTree (Size/Median/00256)	256	0.0391 ± 0.0023
172	WiSARD-KDTree (Diff. to Mean Density/Half/04096)	4	0.0368 ± 0.0023
173	WiSARD-KDTree (Density/Half/04096)	4	0.0367 ± 0.0014
174	WiSARD-KDTree (Density/Median/00256)	256	0.0363 ± 0.0009
175	WiSARD	8	0.0313 ± 0.0014
176	Naive Bayes	N/A	0.0269 ± 0.0031
177	WiSARD-KDTree (Density/Half/16384)	4	0.0254 ± 0.0010
178	WiSARD-KDTree (Diff. to Mean Density/Median/00256)	256	0.0245 ± 0.0017
179	WiSARD-KDTree (Diff. to Mean Density/Half/16384)	4	0.0217 ± 0.0011
180	WiSARD-KDTree (Density/Half/00256)	256	0.0208 ± 0.0026
181	WiSARD-KDTree (Size/Half/16384)	4	0.0202 ± 0.0011
182	WiSARD	4	0.0197 ± 0.0010
183	WiSARD-KDTree (Diff. to Mean Density/Half/00256)	256	0.0193 ± 0.0019
184	WiSARD-KDTree (Size/Half/00256)	256	0.0160 ± 0.0028

Table A.4: Final ranking (part 4 of 4): Showing from rank 131 to 184. The number of experiments was 184.