Learning Preferences for Personalisation in a Pervasive Environment

Sarah Gallacher

Submitted for the Degree of Doctor of Philosophy

School of Mathematical and Computer Sciences
Heriot-Watt University
Edinburgh, UK

August, 2011

Copyright 2011 by Sarah Gallacher

The copyright in this thesis is owned by the author. Any quotation from the thesis or use of any of the information contained in it must acknowledge this thesis as the source of the quotation or information.

ABSTRACT

With ever increasing accessibility to technological devices, services and applications there is also an increasing burden on the end user to manage and configure such resources. This burden will continue to increase as the vision of pervasive environments, with ubiquitous access to a plethora of resources, continues to become a reality. It is key that appropriate mechanisms to relieve the user of such burdens are developed and provided. These mechanisms include personalisation systems that can adapt resources on behalf of the user in an appropriate way based on the user's current context and goals. The key knowledge base of many personalisation systems is the set of user preferences that indicate what adaptations should be performed under which contextual situations.

This thesis investigates the challenges of developing a system that can learn such preferences by monitoring user behaviour within a pervasive environment. Based on the findings of related works and experience from EU project research, several key design requirements for such a system are identified. These requirements are used to drive the design of a system that can learn accurate and up to date preferences for personalisation in a pervasive environment. A standalone prototype of the preference learning system has been developed. In addition the preference learning system has been integrated into a pervasive platform developed through an EU research project. The preference learning system is fully evaluated in terms of its machine learning performance and also its utility in a pervasive environment with real end users.

ACKNOWLEDGEMENTS

This thesis was made possible by the support and encouragement of many people, to all of whom I am very grateful.

My academic supervisor, Professor Nicholas K. Taylor, has been a continuous source of inspiration, support and encouragement throughout the entire six year period of my thesis research. I am deeply grateful for the many ideas, sources of information, resources and knowledge that he has shared. I have been inspired by and enjoyed the countless hours of discussions and debates regarding the design and implementation challenges of this research work. I am also very appreciative of the time taken to read numerous versions of this thesis and for the constructive and useful feedback given.

My colleague, Professor M. Howard Williams, has shown much support and encouragement for which I am very grateful. His knowledge and wisdom from a very rich and successful research career has been a significant factor in my development as a researcher.

My husband, Stephen Gallacher, has given me invaluable support in countless ways with regards to the thesis and my life. He has been a continuous source of support, relief, practicality and encouragement over the past six years.

My colleague, Eliza Papadopoulou, has been a true friend as well as a supportive and encouraging colleague whom I thoroughly enjoy working beside.

My family and friends, who have always supported my endeavours and helped me to become the person I am today.

The many research project partners who I have worked with on various EU projects over the past six years. They have been an invaluable source of knowledge and support.

DECLARATION

Replace with:

Research Thesis Submission form!!

CONTENTS

1 INTRODUCTION

1.1 Background

It is interesting to consider how many computational devices the average person owns. Just over a decade ago the list was likely restricted to a family desktop PC. Nowadays it is common for any single person to own a combination of devices such as laptops, desktop PCs, smart phones/PDAs and mobile phones to name a few. In addition, the current versions of such devices are much more sophisticated than their earlier counterparts providing enhanced connectivity options as well as multiple services and applications. Further, devices previously considered as "dumb" such as washing machines, fridges and even toasters are becoming increasingly sophisticated in terms of the computational and communication power they have. It is clear that computational technology is filtering into our everyday lives to a greater and greater degree. This is the view of the world that Weiser envisaged over 20 years ago and what he termed as *ubiquitous (or pervasive) environments*. As technology continues along this trend we must consider the impact on end users in such complex, resource rich environments.

To ensure the continued acceptance of additional computational technology, mechanisms must be provided to shield the user from complexity and aid them in resource management tasks. Such mechanisms should configure resources on behalf of the user, where possible, in line with user needs and goals. Recently, location aware applications have arrived on the market with the ability to adapt beneficially based on user location. Additionally many of the latest smart phones include sensors such as digital compasses, proximity sensors, ambient light sensors, etc. that allow applications to respond to other user context information. This is a step towards resource management aids; however, basing adaptations on context alone can only aid the individual user to a certain degree. For example, one individual may want their smart phone to use the cheapest network when they are at home whereas another user may want their smart phone to use the network with best Quality of Service (QoS). It is not possible to correctly manage resources on behalf of the user without personal information.

In pervasive environments, *personalisation* mechanisms typically use *preference rules*, or *preferences* for short, to manage resources on an individual basis. The preferences outline what an individual user prefers in some situation. Personalisation mechanisms can then configure or adapt resources based on the preferences so that resources may appear differently to different people, or to the same person in different situations. The individual user may define preferences manually; however, to reduce this burden on the end user many personalisation mechanisms utilise *machine learning* techniques to gather and manage preferences on behalf of the end user.

A typical approach is to monitor how the user interacts with resources in the current situation and then mine preferences from this monitored data. For example, if an individual user always selects the cheapest network when at home, this behaviour would be mined as a preference that states *"when the user is at home, always select the cheapest network"*. However, there are many issues related to learning preferences for personalisation in pervasive environments. A suitable learning algorithm is key. The algorithm should be able to produce accurate, human understandable preferences from the available input data, respond rapidly to new user behaviours to ensure an up to date preference set and handle an ever changing problem domain where resources come and go.

1.2 Aims and Objectives

The central research question to be answered in this thesis is:

How can a system learn and provide accurate and up to date preferences for personalisation in a pervasive environment?

The main aim of this thesis is to provide a solution to this question through the development of a preference learning system that can be deployed in a pervasive environment. The system should provide and maintain a set of accurate and up to date preferences to drive personalised adaptations for an individual user within the environment. The learning system should also fulfil other requirements of the pervasive problem domain.

The derivation of this requirements list is a key objective of this research work. It is necessary to identify how preference learning systems are typically implemented in pervasive environments in order to assess their ability to learn accurate and up to date preferences and identify areas for improvement. A review of related literature highlights that batch learning algorithms are the technique of choice for preference learning despite scheduled executions and batch processing constraints in such an incremental and real time problem domain. Notably, batch algorithms can provide accurate preferences but their scheduled executions often mean that preferences are not always up to date. Therefore, this thesis questions the utility of batch algorithms for preference learning and aims to identify if incremental learning techniques provide a more efficient solution for preference learning in pervasive environments.

The solution presented and evaluated in this body of work is the DIANNE (Dynamic Incremental Associative Neural NEtwork). The DIANNE is a single layer, feed forward neural network that aims to learn accurate and up to date preferences. The design of the DIANNE topology and learning algorithm are driven by the identified set of requirements. Learning is incremental with inputs processed as they occur in the environment rather than during scheduled batch executions. The preference related outputs from the DIANNE are used to drive the personalised configuration and adaptation of resources on behalf of the user depending on their current situation.

The DIANNE assumes a temporal relationship between situations and preference related behaviours. Therefore the longer that a behaviour prevails in some situation, the stronger the association between the behaviour and the situation. An objective of this research is to investigate this assumption to see if such a relationship holds in reality and hence if it is a good basis for DIANNE learning.

A key challenge faced by a preference learning system is that of concept drift. This occurs when user behaviour changes in such a way that it is no longer in line with the learnt preferences. When this occurs the preference learning system must rapidly respond to update the learnt preferences and bring them back in line with current user behaviours. For an incremental learning system a significant issue is the rate at which such responses are implemented. On the one hand the learner should not take an unsatisfactorily long time to update but on the other hand single instance updates are also undesirable. The aim is to provide a solution that performs such updates appropriately and in line with the expectations of end users. The proposal and evaluation of such a solution is an important objective of this work.

Finally, it is the aim of this research work to assess whether the provided solution (i.e. the DIANNE) satisfies all the necessary requirements and ultimately whether the provided solution answers the initial research question. To accomplish this the implementation of a sufficient testing and evaluation strategy is key. The aim is to analyse the solution both in terms of its performance as a learning algorithm and also in terms of its utility as a preference learner in a pervasive environment.

1.3 Key Contributions

The main contributions of this work are:

1. *Identification of the key requirements for the provision of an efficient solution for preference learning in a pervasive environment.*

The work presented in this thesis draws on many influences from both background literature and previous experience on research projects. Key challenges have been identified from both influences based on the consideration of past works by third parties (discussed in Chapter 2) as well as lessons learnt from personal experiences (discussed in Chapter 3). In particular, the Personalisation system developed in the EU FP7 DAIDALOS project has been cited as a prototype from which issues have been identified and lessons learnt. Based on the identified challenges, a requirements set for preference learning systems is identified. Two key outcomes include:

- the identification that an incremental learning approach is the most natural and flexible way to handle incremental tasks, such as learning user preferences.
- the identification that the temporal duration in which a preference prevails is an important piece of information that is often overlooked by preference learning systems. Utilising this information enables challenges such as negative preference learning to be overcome.

2. *Design of Dynamic Incremental Associative Neural NEtwork (DIANNE) topology.*

The DIANNE is a single layer, feed forward network that has been designed specifically to meet the identified requirements related to learning preferences for personalisation in a pervasive environment. The DIANNE represents associations between context vectors and preference vectors as linear connections. The use of a single layer neural network allows for rapid and non-complex updating of internal knowledge and hence is an ideal topology for an incremental learning solution. It is shown that a single layer topology is sufficient to represent the learning problems of the pervasive domain. Due to the absence of hidden layers it is also possible to translate internal network knowledge into human understandable form. This is a necessity in such a user centric problem domain.

3. Design of DIANNE temporal learning algorithm.

The DIANNE algorithm is an incremental learning algorithm based on temporal reinforcements. The algorithm design has introduced some novel aspects:

- The DIANNE algorithm is an incremental preference learning algorithm. Inputs (monitored user behaviour) are processed as they occur in real time. This allows for rapid response to changes in user behaviour. There is no need to retain stores of past behaviour as the DIANNE does not need to re-process past data.
- The DIANNE algorithm implements continuous learning through temporal reinforcements. Weight updates occur in a temporal fashion based on the amount of time that an active input renders an active output (i.e. the amount of time that a preference prevails in some context). This is in contrast with traditional weight update methods where updates are often error driven or based on the number of occurrences of an active input rendering an active output.
- Two learning rules are utilised. Hebbian/anti-Hebbian is used for continuous temporal learning. An error reduction approach is used for learning under conflict conditions when network output conflicts with the real world situation.
- Due to continuous temporal learning a dynamic squashing function has been designed and implemented to stop the occurrence of saturation.
- The incremental nature of the DIANNE has led to the design of an incremental conflict resolution strategy that can resolve conflicts at one instance in time based on current knowledge. Two heuristics for incremental conflict resolution are proposed in line with end user expectations and the notion of preference time constants.

4. Implementation of DIANNE as standalone system.

The DIANNE has been implemented as a standalone learning system. In this sense it can be applied to other problem domains if required. Benchmark testing and analysis of the DIANNE, described in Chapter 7, utilises the DIANNE as a standalone learning system.

5. *Implementation of DIANNE as preference learning system in PSS platform*.

The DIANNE is utilised in the EU FP7 PERSIST project. It is implemented as a preference learning system within the Personal Smart Space (PSS) platform, developed by the project. Chapter 6 describes how the DIANNE is integrated into the platform and also how it was utilised and demonstrated during the final project review. User testing of the DIANNE, described in Chapter 7, utilises the DIANNE within the PSS platform.

6. *Testing and analysis of the DIANNE as an incremental learning system.*

The DIANNE is tested and evaluated in two ways. Firstly, the DIANNE is analysed in terms of its accuracy against other benchmark learning algorithms (both batch and incremental) on well cited, real world datasets. Secondly, the DIANNE is analysed in terms of its utility as a system to learn user preferences for personalisation in a pervasive environment. This process involves user trials to determine how the DIANNE performs as a preference learner in a live pervasive environment. As part of this the performance of the DIANNE is compared with the performance of a benchmark batch learning algorithm also applied to the live environment.

1.4 Thesis Overview

Chapter 2 presents and discusses related work in three key areas: pervasive computing, personalisation and machine learning. Pervasive computing is the problem domain of this thesis and a general overview is given. The review of personalisation is focussed towards personalisation in the pervasive computing problem domain while the review of machine learning is focussed towards the problem of learning preferences for personalisation in the pervasive computing problem domain.

Chapter 3 documents and discusses the DAIDALOS Personalisation system developed during the EU FP6 DAIDALOS project. The chapter details how it was designed and implemented over two project phases and how the second phase prototype built upon and was an improvement over the first. Design and implementation of the first phase prototype was completed prior to the author's personal involvement. The work presented in this thesis is mostly related to personal experience of the design and implementation of the second phase Personalisation system. This work was undertaken in conjunction with colleagues (listed as co-authors on all related and referenced papers).

Chapter 4 outlines the requirements that should be met to successfully meet the challenges of preference learning for personalisation in a pervasive domain. The requirements are drawn from observations on previous works as well as lessons learnt from the DAIDALOS prototype.

Chapter 5 presents the Dynamic Incremental Associative Neural NEtwork (DIANNE). The network topology is outlined and analysed for capacity, stability and convergence.

Chapter 6 presents the DIANNE temporal learning algorithm and outlines how the DIANNE has been successfully implemented in the EU FP7 PERSIST project.

Chapter 7 describes the testing and analysis that has been performed on the DIANNE. Firstly, the benchmark testing phase is presented. Results outline DIANNE performance on a number of well cited real world datasets. Comparisons are drawn between DIANNE performance and that of other benchmark learning algorithms. Secondly, the user testing phase is presented. Results outline the user centric view of how well the DIANNE learns user preferences for personalisation in the pervasive problem domain.

Chapter 8 concludes the work and findings in this thesis. Key concepts, assumptions and findings are discussed and several suggestions are put forward for further work and extensions.

2 Related Work

The work related to this thesis has branched across three main research areas; pervasive computing, personalisation and machine learning. Pervasive computing is the trend towards computational devices anywhere and everywhere around us with the ability to communicate with each other, self-improve and behave in an "intelligent" manner to aid us in our everyday lives. Personalisation aims to tailor some entity to a specific individual so it looks or acts differently for different individuals or for the same individual in a different situation. Machine learning is the discipline concerned with the design and implementation of algorithms that automatically improve their performance at some task with experience.

Although individual fields in their own right, they have been brought together in this context with the common goal of learning preferences for personalisation in a pervasive environment. Pervasive computing provides the problem domain, learning preferences for personalisation is the challenge and machine learning provides possible solutions. Therefore all three areas are discussed in terms of related work with a view towards the common goal.

To illustrate how the connections between these three research areas have materialised the rest of the chapter is structured as follows. Firstly a literature review is presented for pervasive computing. This includes an introduction into the general concepts of pervasive computing and the past and ongoing innovations that are helping to realise pervasive environments. Secondly a literature review is presented for personalisation in terms of pervasive systems. It focuses on pervasive systems that utilise personalisation concepts to aid individuals in pervasive environments through the personalised management of resource and personalised adaptations. A distinction is drawn between those projects that require a user to input personalisation rules (i.e. preferences) manually and those projects that attempt to identify preferences using behaviour monitoring techniques and machine learning algorithms. Finally a literature review is presented for machine learning with a focus on those algorithms used for preference learning in pervasive systems.

2.1 Pervasive Computing

Over the past five decades computing trends have changed dramatically. In the early days, computational technology was inaccessible to the majority of the population due to the size and cost of machines as well as their complexity. Users were mainly of some scientific affiliation and literate in computational languages. Large mainframe computers served these users who shared the resource in a *many to one* relationship. As computer technology progressed, the cost and size of machines decreased. This fact, coupled with advances in peripherals and more user friendly operating systems, made computational technology more accessible to the general public. The Personal Computer (PC) promoted a *one to one* relationship between computational technology and the end user. A member of the general public could now own a computational device that was personal to them, containing their information and performing processes specific to their needs.

Already we have moved to the next natural progression defined by a *one to many* relationship between users and computational technology. In the current climate a single user will now often own multiple devices such as laptops, PDAs, mobile phones, etc. and as mobile and network technologies advance there is great potential for this one to many relationship to grow. Such device intensive environments and the challenges they raise were described by Mark Weiser in his 1991 seminal paper [1]. He outlined a new research field to tackle such challenges which he labelled 'Ubiquitous Computing'. Since then several alternative labels such as ‘Ambient Intelligence’ and ‘Pervasive Computing’ are commonly used. Regardless of the label, the fundamental concept is that computational technology is ‘weaved into the fabric of everyday life’ until indistinguishable to end users. In other words, computational technology exists in everyday objects throughout the user’s environment and interactions with such powerful technology are natural and unobtrusive to the user, rendering it invisible. This is in direct contrast with current trends where the focus of the user's attention is firmly on the computational technology.

It is difficult to provide a short succinct definition of pervasive computing since it is a complex and multi-faceted domain. Many authors have attempted to define it in terms of fundamental principles [2, 3, 4]. Although this list often differs from source to source, there are several common properties that repeatedly appear in the literature.

- Networked, distributed and transparently accessible computational technology – In an environment where pervasive computing is implemented, all devices throughout should be able to communicate and share information with one another anytime and anywhere.
- Invisibility and simple human-computer interactions – Interactions between users and the computational devices within a pervasive environment should be simple and natural, minimising distraction. In contrast to PCs the user's attention should not be focussed on the technology. Rather, computational technology should be invisible in the environment.
- Context-aware adaptation – The devices within a pervasive environment should be context-aware both in terms of their physical environment and the user. Based on current contextual information, environments should adapt to meet the needs of individual users within.

Pervasive services are also an important aspect of pervasive computing. Although not always explicitly listed as a key principal, pervasive services take advantage of the principals listed above to deliver the pervasive experience to the end user.

At the time when the vision of pervasive computing was initially conceived, many limiting factors restricted the realisation of such principles. This was largely due to the infancy of research and technology in areas upon which pervasive computing depends. Since then, advances in several key fields are bringing us closer to a pervasive world that conforms to the above principles.

2.1.1 Connectivity

One key property of pervasive systems is ubiquitous connectivity and access to all components of the pervasive system. When Weiser initially outlined his ubiquitous vision the world was a very different place, where the World Wide Web was very much in its infancy and WiFi technologies did not yet exist. In fact widespread internet usage was a substantial step forward in pervasive computing terms. The idea of ubiquitous connectivity now did not seem so farfetched with the global adoption of Internet Protocol based networking.

The mid 90's saw the introduction of the IPv6 internet protocol [5] which has further enhanced the prospects of pervasive computing by providing enough unique IP addresses for every object in the world to communicate. The potential for such an all encompassing network has lead to a new concept, defined in 1999 at MIT, and termed the *Internet of Things* [6]. This refers to a network of objects such as everyday household appliances. Such innovations make it possible to saturate everyday environments with networked computational technology.

The introduction of wireless networking and the 802.11 protocols have also been a huge boost towards pervasive ideals. Devices are free from physical wired boundaries allowing greater potential for mobility and the ability to add new devices to the network in an unobtrusive way. As well as WiFi protocols, short range solutions such as Bluetooth are now common in many mobile devices for data exchange across short distances. Research continues in this field and technologies such as WiMax [7] provide the potential for greater widespread wireless connectivity. Already several trials have brought WiMax to cities throughout the UK with the promise of widespread coverage and free internet access for residents. At the same time, telecommunications research is also tackling the issue of ubiquitous connectivity with advances in 3G networking. Coupled with very portable devices, telecoms companies have spotted their potential to impact on the pervasive market.

2.1.2 Mobility

Advances in mobility have followed different streams. On the one hand we have advances in portable devices such as laptops allowing us to take substantial computational power with us wherever we go. Another step along the mobility scale brings us to handheld devices such as PDAs and smart phones which, although computationally less powerful than laptops, are becoming increasingly feature laden with many devices such as the iPhone [8] now acting as a telephone, MP3 player, camera and video recorder to name a few. Although this means the end user requires less individual devices, to stay in line with pervasive ideals such devices must remain intuitive and simple to interact with. Overloading with features may have an anti-pervasive outcome.

Another mobility stream is *embedded technology*. Research in this field has enabled everyday objects to benefit from added computational intelligence and communication. This is an important step towards making devices, and hence the environments they reside in, more pervasive. As Weiser was communicating his vision of ubiquitous computing, across the Atlantic Roy Want's team at the Olivetti Research Labs in Cambridge University were already attempting to realise a subset of pervasive computing through embedded devices. The initial *Active Badge* project [9] embedded infrared technology in an identification badge that could be used to locate an employee within the building for the purpose of routing telephone calls. The *Active Floor* project [10] was the successor embedding sensors into the floor of the building so employees could be located and identified by various features such as their gait without the need to carry an identification token.

Indeed various projects have experimented with a range of everyday artefacts, adding computational and network technology to create pervasive devices. Another example is the Kenko Toware toilet [11]. Sensors and network technology have been added to this everyday, dumb device to allow biometric information to be obtained (through sensors in the toilet seat) indicating the user's pulse, blood pressure and body fat. This information can then be communicated directly to the user's doctor if so desired. Although perhaps a crude example, it illustrates how the most mundane artefacts can be used as pervasive devices. More commonly, white goods, vending machines, ATMs and automobiles are examples of artefacts regularly enhanced with networking and computational technology.

Another area of mobility research which often overlaps with embedded technology is *wearable computing*. One of the pioneers of wearables, Steve Mann, began investigations in the 1970's using head mounted displays to record personal visual memories [12]. From the rather bulky and un-elegant beginnings, wearable computing has advanced towards smaller, lighter and cheaper devices. In early 2009 Pattie Maes from the MIT Things That Think (TTT) consortium [13] presented her vision of sixth sense computing [14] which centred on a small wearable device that resembled a necklace. It enabled users to retrieve useful information depending on their current situation and display the information using any suitable surface in the physical world.

A more commercial trend in wearable devices has been towards the integration of computational technology into clothing rendering it less visible. This branch of wearables has been most successful in the commercial world to date. In 2006 Motorola teamed up with Burton, the skiwear company to introduce the Audex ski jacket [15] with integrated Bluetooth and MP3 players complete with control pads at the wrists allowing the wearer easy access to controls while skiing. In other sports, Adidas have produced the *Adidas_1* running shoe [16] complete with sensors and actuators allowing the shoe to adapt to the recorded biometrics of the runner.

2.1.3 Human Computer Interaction (HCI)

Pervasive systems will result in a very different relationship between humans and computers. Where current PC devices and traditional WIMPS (Windows, Icons, Menu, Pointer device) interfaces demand the full attention of the user, pervasive systems will offer a multitude of less obtrusive and less attention-intense computer interfaces. This will be necessary due to the numerous computational devices in the user's pervasive environment.

To interact with such an array of devices, interfaces must fully utilise all aspects of human communication media such as movement, gestures, speech and touch. Technologies specific to each of these media are already advanced. Vision systems couple cameras with artificial intelligence to learn and recognise gestures associated to some command. These can include the movement of hands and arms, head movements and facial expressions or full body movements. This form of interaction has recently become popular for interfacing with games consoles such as the Nintendo Wii [17]. The natural gestures used to control games have opened the gaming world up to a new, less technical, audience.

Speech interfaces have the advantage of allowing users to interact with computers in a hands-free manner. As with vision systems, artificial intelligence is applied to learn and recognise spoken words. Speech and voice recognition software is already widely used in many domains ranging from the military to health care. At a more commercial level such software is commonly available in mobile phones (supporting voice dialling) and word processors (supporting speech-to-text processing).

The Tangible Media Group [18] at MIT has adopted another interesting human-computer interaction concept by bringing together the ‘bits’ of cyberspace and the ‘atoms’ of the real world to give physical form to digital information. In other words users can interact with information systems by manipulating physical objects. Early projects such as MetaDesk [19] used *phicons* (physical icons) to allow users to interact with a campus map by moving a physical object representing some landmark. More recently at Lancaster University the Cubicle project [20] implemented a cube shaped object that could control applications. In one scenario a radio tuner could be controlled by turning the cube so that the face representing the required command (play, stop, volume up, volume down, channel up, channel down) was facing upwards.

Other projects explore a similar thread of *changing interfaces* where interfaces physically change shape to change function. The Speak Cup [21] is a cup shaped object that acts as a voice recorder and play-back device however instead of providing buttons for interaction, the user can control the cup functionality by changing the shape of the cup. When the user wishes to record themselves they talk into the cup, ‘filling’ it with the voice recording. To play the recording back the user simply turns the cup inside out, ‘spilling’ its contents.

Although physical manipulation brings artefacts into the foreground of attention the physical manipulations greatly reduce the attention required for interactions allowing the user to divide attention between tasks. Weiser referred to this concept as *Calm Technology* [22] where users did not have to focus full attention on computation and could therefore be more aware of their environment and perform other tasks. A well cited example of calm technology is the Live Wire [23] created by artist Natalie Jeremijenko. A long wire strand was attached to a motor which in turn was attached to an Ethernet connection. When a packet of data passed through the network the motor twitched making the wire move around. Therefore it was easy to see how much traffic there was on the network without devoting full attention to the information medium.

Research at the Pervasive Computing Institute in Switzerland is developing the idea of *Informative Art* as a potential mechanism for information provision in an ambient and less obtrusive way [24]. What seem like ordinary pictures hanging on the wall can provide information to the user by adapting the content of the picture. For example, if there are three people in the house, a picture of a fruit bowl may represent this information by displaying three oranges in the bowl.

Indeed, there are many natural and innovative ways in which humans can interact with computational devices. In a pervasive world permeated with technology, multiple interfacing modalities will be necessary to reduce attention requirements and increase productivity. Providing control over multi-modal interfaces has become a challenge in itself warranting research effort from recent projects such as OPEN [25]. The significant challenges are the persistent transfer of content from device to device and the adaptation of content to the most appropriate form for a given device.

2.1.4 Context-Aware Adaptation

The field of context-aware adaptation consists of two main streams; the gathering of contextual information, and the usage of contextual information for adaptation purposes. In a pervasive world, where the environment should adapt to meet user needs, context is vital. The more context information available, the more informed and appropriate system adaptation will be.

Gathering Context Information

Unfortunately the term ‘context’ is rather all-encompassing so it is a challenge to know what should be gathered as context. Dey’s well accepted definition of context describes it as

> *“…any information that can be used to characterize the* situation of an entity. *An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.”* [26]

This rather vague description hints that any information can be relevant for specific adaptation decisions and therefore does not give much direction as to what information should be gathered as context.

However in a pervasive environment, with much emphasis on mobility, information regarding the user’s location is usually relevant to most decisions. Gathering such information has various challenges depending on whether the user is indoors or outdoors. Since the early 70’s the US military have been developing the Global

Positioning System (GPS) based on 24-32 satellites. Used for many applications such as land surveying and mapping, GPS is most familiar commercially for navigation purposes due to the popularity of Sat Nav technology. Although GPS provides more accurate results than other outdoor positioning methods such as mobile network cells, unfortunately it is not appropriate for indoor positioning. Instead, alternative methods are under development such as the use of short-range positioning beacons. This method utilises various technologies such as WLAN, Bluetooth and RFIDs to calculate indoor positions based on signal strength. To improve accuracy such an approach is often used in conjunction with accelerometers and other wearable sensors.

Although important, it is unlikely adaptation decisions can be accurately made on location information alone. Wireless sensor networks (WSN) lend themselves well to pervasive computing with their ability to gather physical and environmental context information (such as temperature, sound, vibration, motion, etc.) in a distributed and autonomous fashion. One thread of research focuses on reducing the size of individual sensor nodes with the goal of achieving microscopic dimensions. The SpeckNet [27] consortium in Scotland is developing the concept of *speckled computing* which comprises networks of small programmable devices called *specks* (smaller than a two pence coin), each equipped with sensors, a processor and wireless networking. In 2001 Kristofer Pister of UC Berkeley introduced the concept of *smart dust* [28] described as a sensor network consisting of tiny devices or *motes* that would be no larger than a dust particle. As yet this concept is not fully realised as no microscopic functional motes exist. However, such small devices would be perfect for use in the pervasive domain and the development of WSN specific operating systems such as TinyOS [29] is already promoting their use in the commercial world for applications ranging from industrial process monitoring to traffic control.

Low-level context information from sensors (such as temperatures or coordinate vectors) may be useful for some decision making; however, it is often the case that higher level context information is required such as the current task the user is performing. Gathering this type of context information often requires input from a range of different sensors which must then be processed to *infer* the higher level information. The Multi-Sensor Wearable project [30] has utilised WSN technology to create a wearable sensor network, the output of which can be processed to determine user activity or other such high-level context.

Utilising Context Information for Adaptations

One fundamental requirement of a pervasive system is the management and control of the component technologies and devices within an environment in order to adapt the environment to provide a beneficial and enhanced user experience. As mentioned above context information is key to this process. For example, if the system knows that the user has entered a dark room, the system can switch on the lights for the user. Various research teams have attempted to create pervasive systems that utilise many of the hardware innovations mentioned above as well as context-awareness to provide enhanced user experiences in pervasive environments.

One of the earliest efforts was Microsoft's EasyLiving project [31] that promised "better living through geometry" within the home environment. Great effort was invested in context-awareness and the gathering of context information to drive environment adaptation. The position and orientation of devices and users within the environment were implicitly gathered through various sensors to enable the system to beneficially adapt the environment. For example, if the user moved from their PC to the living-room sofa, their session would follow transferring to a device (such as the TV) within the user's line of sight.

The University of Sejong [32] followed a similar approach to context-aware adaptation within the home environment. However, biometric information such as pulse, body temperature and facial expressions were also gathered to enhance the accuracy of environment adaptation. User location was used for device selection while biometrics were used to infer user activity and drive service selection within selected devices. Other pervasive aspects such as invisibility of technology and simplistic interactions were not so strongly considered by either project.

The Intelligent Room [33] is a component project of the larger Project Oxygen initiative at MIT [34] with a focus on context-awareness and adaptation in an office or home environment. As with EasyLiving it aims to provide beneficial environment adaptations based on implicitly gathered contextual information regarding the environment and user tasks. However, interactions between the user and the smart space are more natural, utilising speech and vision technology to recognise voice commands and gestures.

As well as context-aware pervasive systems, many groups have experimented with context-aware artefacts, often taking everyday objects and making them context-aware. The Lover's Cups [35] are pairs of context-aware cups that glow and adapt depending on the context of the partner cup, allowing users in different locations to know when the other user is drinking. A more useful example is the context-aware pill bottle [36] that reminds users to take their medication especially if it has not been lifted off its stand for some time. In fact, the list of context enhanced artefacts is lengthy, ranging from wheelchairs [37] and children's toys [38] to kitchen sinks [39].

2.1.5 Conclusion

Since the original conception of the pervasive computing idea, many of the hardware and software components required have gone from non-existent to global reality. In fact in the current climate all the constituent parts required for a pervasive system are available in some form or another (although most are still advancing). This begs the question why pervasive systems have not become a widespread reality already. With their potential to provide a better user experience one would assume that their uptake would have been more rapid.

There are several possible reasons why this is not the case. The first reason is the giant leap between the current computational climate and a pervasive environment. Throughout all phases of computing from mainframe to present day the user has been the intelligence and the computer is the slave device that responds to user requests. Pervasive computing aims to upset that trend. Pervasive devices will be intelligent entities, sharing information with each other and using that information to adapt the environment on behalf of the user. This raises two issues that must be successfully dealt with before mass uptake is likely. Firstly, users must trust the system to make decisions and perform actions on their behalf. Additionally they must trust the system to handle their information with care by not divulging embarrassing or sensitive information. Secondly, the user does not want to lose their feeling of control over the devices in their environment. This is how human-computer relationships have traditionally existed. We are expectant of control over devices and become anxious at a lack of it.

On these issues, very often pervasive computing has been badly portrayed in the media. The potential benefits are usually omitted and instead a negative spin highlights dark scenarios of intelligent devices learning our behaviour and taking control. Therefore it is vital for pervasive computing to find the balance between automation and user choice, surveillance and privacy.

The second reason for the slow commercial movement towards pervasive systems is perhaps the radical infrastructure updates and additions necessary to make pervasive environments a reality in our everyday world. For a start, our everyday devices must be replaced with pervasive versions equipped with computational and network technologies. Manufacturers will be reluctant to focus effort on such products until demand increases. As well as pervasive devices, sensors and actuators must be incorporated into the everyday environment along with network technologies to support communication between all devices. Several research institutes and commercial offices have gone some way to providing pervasive environments within their buildings but so far this has often been research related and has not expanded into the wider world.

2.2 Personalisation in Pervasive Environments

The term ‘personalisation’ has become a buzzword in many different communities. In the commercial world, and more recently the e-commerce world, the benefits of personalisation have been recognised for some time. By tailoring goods and services to the needs of individual users, vendors can gain and retain a loyal customer base. In the media community, websites [40] and TV channels [41] are developing personalised content retrieval and delivery to provide a tailored service in line with individual user interests. In highly adaptable pervasive environments, the application of personalisation techniques can also bring great benefit to end users. With ubiquitous access to pervasive services, networks and devices personalisation can help to tailor such resources to best meet the needs of the individual.

As mentioned above, context information can be used to drive adaptations such as device selection for session transfer or service configuration e.g. turning on the lights when the user enters a dark room. Projects such as Easy Living used context-dependent automation rules to dictate how the environment should be adapted depending on the current environmental state. However, the exclusively context-aware adaptations are uniform for all users, no matter who is in the environment or what their individual needs are. For example, some users may prefer all the lights to illuminate when they enter a dark room whereas other users may not. Similarly, different users may prefer sessions to transfer to different devices. Without user centric information, user specific adaptations cannot be distinguished.

Satyanarayanan et al. argue that a high level of user information is crucial for system decision taking, adding that “otherwise it would be impossible to determine which system actions will help rather than hinder the user” [42]. Indeed this is true with exclusively context-aware adaptations. A context-aware adaptation that suits one user may be wholly inappropriate for another user. In the light example above, turning on all the lights would be more of a hindrance than a help to a light sensitive person. Therefore, to provide a truly pervasive experience personalisation is essential, forcing the consideration of user information when performing environment adaptations so that the pervasive environment can be personalised to the needs of each individual user.

Such user information is stored in a *user profile* and can include a wide range of user related data. In pervasive environments there are two types of user information that are most commonly stored in profiles for personalisation. They are sequential patterns of behaviour, or *user tasks*, and user specific adaptation rules, or *user preferences*. Some profile information may be facts that always hold true (e.g. disabilities such as blindness) but in such a mobile, dynamic domain, very often user needs will be related to the user’s current context. Therefore user profiles favour context-dependent user tasks and preferences and context-dependent personalisation is usually implied when referred to in a pervasive domain. A context-dependent preference may specify such behaviours as "if the user is at home, then set the heating to 25 degrees, otherwise set the heating to 15 degrees". A context-dependent task may specify such behaviour as "if the user is leaving work, then turn off the office lights (once the user has left the office) and unlock the car (once the user arrives at the car)".

Pervasive personalisation is often divided into two distinct subsets depending on how the user profile is gained and maintained. The term *explicit personalisation* refers to personalisation processes that operate on profile information that has been manually entered into the system and is manually maintained by the owner. *Implicit personalisation* refers to personalisation processes that operate on profile information that has been learnt and created by the system and is maintained by the system on behalf of the owner. The sections below describe several pervasive projects developing personalisation mechanisms, distinguishing between those that implement explicit personalisation and those that implement implicit personalisation.

2.2.1 Explicit Personalisation in Pervasive Environments

During the late 90's several pervasive projects incorporated profile information into system decision-making processes to provide some level of personalisation. The Intelligent Home project [43] focused on the intelligent management of a home environment. Environment adaptation was based on resource availability as well as user preferences, enabling a personalised user experience. It was assumed user preferences and other profile information already existed in the system since the project scope did not include how such information would be gathered or maintained. Simple preferences were pre-entered into the system for testing and demonstration purposes but unfortunately there was no simple means to view or manipulate the preference set.

The Blue Space project [44] at IBM aimed to provide a personalisable and easily configurable office workspace. Users could control several environmental aspects of the space (e.g. lights) through a touch-screen GUI. (As an aside: although HCI aspects were not the focus of this pervasive project, Mozer [45] raises an interesting argument against the use of computational control GUIs as a replacement for existing interfaces such as light switches. He suggests that such complex GUIs act as a barrier to uptake of pervasive technology as the benefits are outweighed by the effort required in understanding the GUI.) Another GUI was provided to allow the user to view and manipulate their preferences for the various personalisable aspects of the space. An active badge system allowed the space to identify the occupant and personalise using the correct profile.

Since early 2000 Project AURA [46] at Carnegie Mellon University has been developing a slightly different approach to environment adaptation with the aim of reducing the burden on user attention caused by resource management tasks. Rather than only basing adaptation decisions on context and preferences, user tasks were also considered. Users could define tasks by specifying the services needed and associated preferences. The addition of task information allowed the system to better predict what future resource requirements might be and pro-actively adapt environments to meet those needs. As with Blue Space, GUIs were provided for interactions with the user, allowing them to view and manipulate preferences and tasks. Screen shots of the GUIs show that there has been an effort to keep complexity to a minimum but the technologically challenged person may still struggle to translate their needs into appropriate input.

In general, the explicit personalisation implemented by the projects above serves a purpose for project testing and demonstration; however, from a user perspective explicit personalisation can often be more detrimental than beneficial. On the one hand, explicit personalisation puts the user in complete control of their preference and profile information. They will have a mental understanding of the information held in their profile and hence what system behaviour to expect. Dey [47] outlines the importance of this for system acceptance. Without it, unexpected behaviour will lead to confusion and it will be difficult for the user to correct profile information to achieve the desired behaviour.

On the other hand, explicit personalisation places greater mental and interaction burdens on the user. Significantly, there is the learning curve required to understand personalisation GUIs and accurately specify the required system behaviours. The research field of *end-user programming* investigates how this learning curve can be mitigated by providing visual or tangible interfaces that non-technical users can easily understand and use to create rules for environment adaptation [48]. The Pervasive-interactive-programming (PiP) paradigm [49] for end-user programming supports non-technical users in defining the required behaviours of physical and virtual devices within a digital home environment. However, with an exclusively explicit approach, even where interfaces are simple and non-technical one could argue that mental and interaction burdens have only been slightly mitigated. Such burdens could be mitigated to a greater degree with the added support of intelligence and autonomy.

2.2.2 Implicit Personalisation in Pervasive Environments

If a system employs *implicit* personalisation, it contains mechanisms such as behaviour monitoring and machine learning to create and maintain a user profile on behalf of the user to drive future personalisation tasks. This reduces the information management responsibilities placed on the user by explicit techniques and the passive nature of implicit personalisation helps fulfil the goal of pervasive computing by rendering the underlying personalisation mechanisms less distinguishable to the user.

The concept of implicitly gathering and maintaining a user profile is not specific to pervasive environments. It has long been associated with HCI (Human-Computer Interaction) as a driver for user adapted interaction and the personalisation of interfaces [50], although here it is referred to as *user modeling*. Other common applications are personalised adaptive hypermedia, web personalisation and e-Learning. Profiling standards exist, such as CCPP [51] and ETSI [52], outlining the various types of information that constitute a profile. However, often these standards do not fulfil the needs of profiling for pervasive personalisation. CC/PP does not consider profile content such as context-dependent intentions and preferences. ETSI goes much further by proposing the use of implicit profile management mechanisms such as monitoring and learning for the management of context-dependent preferences. However, it is unclear how ETSI's hierarchical, multi-profile structure would cope in a highly dynamic pervasive environment.

A large percentage of pervasive projects implementing personalised adaptation choose a *smart space* as their domain. (A smart space is a physically bounded environment, such as a home or office, enhanced with pervasive computing technology such as that mentioned in section 2.1). Perhaps this trend is due to commercial focus and application opportunities but the finite scope of services, devices and possible contexts within a smart home or office also provides useful boundaries for development. Since 2000 the GAIA project [53] has been developing a middleware infrastructure for smart homes and offices which it terms *active spaces*. In a process typically employed by most implicit personalisation systems, the GAIA system monitors user behaviour within the environment, storing it with related context state information as training data. Once enough training data is collected, learning techniques are executed to produce context dependent preferences indicating what actions the user performs in a given context. Agents can then use these preferences to automatically adapt the environment appropriately as the context state changes.

Also since the early 00's the MavHome project [54] has been utilising a different type of profile content. This project focussed on the prediction of future user tasks to drive adaptations within a home environment. This technique is similar to the task based explicit personalisation provided by AURA; however, the MavHome system automatically learns and maintains user tasks rather than depending on manual entry and management by the user. Several machine learning techniques such as sequential pattern discovery and Markov chains are used to identify commonly occurring patterns of behaviour from stores of monitored historic behaviour data. An incremental prediction algorithm (Active-LeZi) [55] is used to predict future behaviour in real-time (e.g. in a given context, when the user switches on the VCR, they then switch on the TV).

The Adaptive Home (or Neural Network Home) project [56] constructed a prototype pervasive system in an actual residence in 1997. It utilises reinforcement learning and neural network techniques to learn the intentions of inhabitants within the smart home environment. The aim is to balance user requirements and energy conservation. To achieve this the Adaptive Home goes a step beyond other projects by employing learning techniques to build models of future context states for future context prediction (e.g. future occupancy of an area or future hot water usage). User tasks are then analysed against predicted future context states to pro-actively adapt the home appropriately in terms of future user and energy requirements.

The projects above place autonomy as a key goal with the intention of mitigating user interaction. However, the Synapse project [57], which began in the mid 00's, heeds the advice of Barkhuus [58] who argues that users want to enjoy autonomous behaviour to a moderate degree without losing control. Therefore, the Synapse personalisation system performs environment adaptations under two modes; *active* and *passive*. Bayesian Networks are employed to learn preferences dictating the relationships between context states and service usage behaviour. This learnt knowledge is then applied to personalise the user's environment through service provision. If a preference has a probability above some threshold, personalisation operates in active mode and the service is started automatically. If the preference has a probability below some threshold, personalisation operates in passive mode and the top five potential services are presented to the user for manual selection. This approach aims to minimise incorrect personalisation in uncertain situations while at the same time provide automation when appropriate.

In fact, finding the correct balance between automation and user control is a challenging personalisation issue. If personalisation is incorrect due to unsatisfactory learning techniques or a change in user behaviour then it is desirable that the system provides mechanisms to identify and rectify the problem in an acceptable time frame to mitigate user involvement. User feedback can provide some assistance but equally implicit preference learning processes should be able to accommodate new patterns of behaviour into the preference set in an acceptable time frame. It is undesirable for the user to have to continually override incorrect automatic behaviours.

It is suggested that the adoption of *batch* learning algorithms for preference learning in personalisation systems can compound this issue. The nature of such algorithms prevents any natural quick response to changes in preference related user behaviour since learning only occurs at certain intervals between which a training dataset (monitored user behaviour) is gathered. Further, batch learning algorithms are dependent on a priori training datasets before any learning can proceed at initial system usage. Hence there exists a lag period where no learnt profile content is available, between initial system usage and the initial learning execution (once enough user behaviour has been monitored for an initial training dataset). During this period personalisation may be limited to default tasks or preferences (if any are available).

A simple solution is the provision of a GUI through which the user can manually update the incorrect profile information. However, as mentioned above, without the appropriate mental picture of the user profile this process can be challenging for users, as well as increasing the interaction requirements and burden on the user. A more desirable approach is the implementation of an implicit rapid response solution.

Also beginning in the mid 00's, the Ubisec project [59] implements mechanisms to enable the quick accommodation of new information into its customisation (or user) profile. If a conflicting behaviour occurs (e.g. the device volume is set to mute by the system but the user un-mutes the volume), the real-time profile evolution process analyses the differences between the customisation profile and the device status profile. A recommendation profile is generated from the differences and used to update the customisation profile subject to manual user approval. In this way the required updates are implicitly gathered from monitored manual device re-configuration and do not require the user to understand complex profile GUIs.

Other projects such as SPICE [60] and its predecessor Mobilife [61] also implement real-time response techniques. Regular behaviour model learning uses a batch algorithm but between algorithm executions the user profile is updated in real-time based on user feedback received as a consequence of undesired personalisation. This helps to keep the user profile up to date between learning cycles.

The personalisation system implemented within the iDorm at Essex University [62] removes all user awareness during the rapid updating of profiles. Once a preference set has been established during the initialisation period, preferences can be modified, added or deleted when user behaviour changes. At such times, a non-intrusive cycle is entered where new or changing user behaviour is specifically monitored and new preferences learnt. Additionally, in a life-long learning phase the worst performing preferences are periodically replaced by new ones to preserve system performance. These rapid response mechanisms are often termed 'incremental' by the projects that employ them due to the way in which they update the user profile in real time.

2.2.3 Conclusion

As personalisation has been adopted by new fields its scope has grown. From initial use as a marketing tool to its current use in pervasive systems it has expanded to include disciplines such as behaviour monitoring and machine learning. Its success is the ability to tailor some entity to the needs of individuals giving each user an enhanced and personal experience whether it be in an e-commerce site or in a smart home.

Two approaches to personalisation were presented; explicit and implicit. Both approaches have advantages and disadvantages in terms of enabling user control over system behaviours and mitigating user involvement in system decision making processes. As Ball et al argue [63, 64], a balance must be found between the two approaches with the most beneficial solution likely employing explicit and implicit techniques in a unified way. Cole et al reflect this argument based on their analysis of the relationship between the increasing demand for user control and the changing role and expectations of intelligent buildings. They describe the problem as a reconciliation challenge between human and automated intelligence [65].

The pervasive domain has thrown up many new and interesting challenges that personalisation must tackle but the success of pervasiveness and the advancement of personalisation go hand in hand. As the hardware and software components required for pervasive computing continue to fall into place perhaps one of the biggest barriers to widespread commercial use is finding a solution to the seamless integration of component technologies with each other and their implicit management rendering them invisible to the end user. Tasks such as environment adaptations, information and device management and scalability issues must be intelligently handled by the system to meet individual user needs and minimise interaction requirements. Personalisation provides an answer; however, issues such as automation vs. user control and implicit profile management require further investigation.

Adam Greenfield suggests that one reason why pervasive computing is not currently more widespread is due to the fact that we are not very good at ‘doing smart’ and perhaps we may never reach the required level [11]. Perhaps the issue is with personalisation system design or perhaps it is related to the more fundamental issues of machine learning and artificial intelligence techniques that many implicit personalisation systems have adopted.

2.3 Machine Learning for Pervasive Personalisation

The aim of machine learning research is to create programs that automatically improve their performance at some task with experience. To achieve such a goal the field of machine learning has been influenced by many disciplines such as mathematics, psychology, biology and philosophy to name a few. This has resulted in the development of a wide range of techniques and algorithms throughout the years. Much early machine learning research was theoretical due to computational limitations and initially took inspiration from biological systems. In 1943 McCulloch and Pitts [66] applied symbolic logic to the modelling of nervous systems laying the groundwork for future connectionist innovations such as Rosenblatt’s Perceptron [67]. However, initial expectations for such neural systems were not realised and later research by Minsky and Papert [68] highlighted their limitations. The result was an exodus from neural modelling and a shift of focus towards more symbolic, concept-acquisition techniques.

Indeed, interest in symbolic techniques had been growing since the early 60’s and continued to enjoy much research focus through the 70's. The machine learning community drew inspiration from psychology and, in particular, research on models of human concept acquisition. Numerical and statistical methods were abandoned for logic-based techniques with Michalski's work on inductive learning programs [69] and Mitchell's work on Version Spaces [70].

The 80’s saw a real revival in the entire machine learning field with an explosion of new and influential techniques such as the ID3 decision tree algorithm [71] and incremental learning methods like the COBWEB [72] clustering system. A major development was Rumelhart, Hinton and William's back-propagation algorithm [73] that overcame many of the limitations of the perceptron and sparked a renewed interest in connectionist techniques. Evolutionary techniques, originally conceived in the 60's, also enjoyed renewed interest at this time; popularised by Dawkin's book entitled "The Blind Watchmaker" [74].

In the 90's, the field of fuzzy logic finally started to gain support after it had been ignored by many researchers since the 60's. The field acquired a wide literature and the concept of fuzzy logic was applied within many commercial products and control systems. In 1997, Diettrich [75] summarised how the machine learning field was following several directions; learning ensembles of classifiers to improve accuracy in supervised learning, scalable supervised learning to improve efficiency over large training datasets, reinforcement learning for online processing and agent control and learning stochastic models that incorporate prior knowledge (such as Bayesian networks).

At this time Mitchell reviewed the success of machine learning and considered its utility in real world applications [76]. As well as data-mining for knowledge discovery and difficult to program applications such as face recognition he also identified the potential machine learning offered to customizable applications that could adapt to individual users. He concluded that such complex problem domains are notoriously difficult to overcome through manual programming. Indeed, since the early 90's, pervasive projects have utilised machine learning for personalisation with much greater success than other approaches.

However, many different machine learning techniques are available with each being specific to some problem domain and no one approach consistently out-performing all others. Wolpert and Macready termed this the No Free Lunch theorem [77] stating that

"…any two algorithms are equivalent when their performance is averaged across all possible problems."

Therefore, each learning task has a small number of suitable machine learning techniques that will give optimal results in the problem domain when compared to other techniques.

In the pervasive computing domain, different machine learning techniques are used depending on project goals and the required profile content. However, there are several approaches that are more commonly used for implicit personalisation purposes across projects. Table 1 below maps the various techniques and algorithms to the projects that employ them. The rest of this section considers the techniques and algorithms in more detail.

ML Technique	Algorithm	Project
Artificial Neural Networks (ANNs)	Sparse Network of Winnows (SNoW)	GAIA
	Multi-layer perceptron	Adaptive Home
Stochastic Models	Hierarchical Hidden Markov Models (HHMM)	MavHome
	Hidden Markov Models (HMM)	Synapse
	Naïve Bayesian Classifier	Mobilife
Fuzzy Logic	Fuzzy rules	iDorm
		Ubisec
Rule Learning	Ripper algorithm	Mobilife
	Apriori algorithm	SPICE
Reinforcement Learning	Q Learning	Adaptive Home

Table 1. Machine learning techniques and algorithms employed for implicit personalisation by pervasive projects

2.3.1 Artificial Neural Networks

With regard to learning algorithms the most natural place to look for inspiration is biological systems such as the human brain with its extremely powerful learning capabilities. As scientists understand more about its internal workings, computer scientists have attempted to replicate these biological structures to process information in a similar way. The observation that the brain is built from very complex webs of interconnected neurons has inspired the study of Artificial Neural Networks (ANNs) which are networks of processing units (representing neurons) interconnected by weighted connections (representing synapses).

One of the most basic ANN systems is based on the perceptron processing unit. The perceptron sums weighted inputs and uses a linear threshold function to determine activation. It learns by adapting weights based on the *perceptron training rule* where weight manipulations are dependent on the error at the output unit. Unfortunately the perceptron is not capable of learning non-linearly separable functions such as XOR, greatly limiting its power. Multilayer Perceptrons (MLPs) overcome this issue by introducing extra (hidden) layers of processing units that utilise a sigmoid threshold function to determine activation [78]. Learning is often performed using variations of the Backpropagation algorithm [73] based on gradient descent. This is the ANN topology implemented within the Adaptive House project. A feedforward MLP is used to learn the target function relating context-based inputs to preference-based outputs.

GAIA makes use of a Sparse Network of Winnows (SNoW) [79], a learning architecture developed at the University of Illinois. It is essentially a network of perceptrons with linear thresholds, each making predictions over the feature space. Weights can be updated using a variety of learning rules such as the perceptron rule or naïve Bayes but the WINNOW rule is most successful. The WINNOW rule uses promotion (weight multiplied by some fixed parameter $\alpha > 1$) and demotion (weight set to 0) steps to update weights and only operates on weights related to incorrect outputs. SNoW is specifically tailored for domains with large numbers of features that may not be known a priori and hence is well suited to pervasive domains where the set of context features is large and ever-changing.

Generally, ANNs have a remarkable ability to derive meaning from complicated or imprecise data and can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. Unlike many logical machine learning methods, ANNs can handle real-valued inputs and are robust against errors in training examples. For real world applications such as implicit personalisation such properties are desirable where user context can contain continuous values (from sensors) as well as errors.

Due to their algorithmic similarities, Support Vector Machines (SVMs) [80] have been found to perform well in the same problem domains as neural networks (e.g. pattern recognition and data mining applications). Although not used for implicit personalisation, the University of Sejong utilises SVMs in their smart home environment for inference [32]. SVMs are an enhanced version of optimal linear classifiers that employ kernel methods to overcome linear classification issues. Just as MLPs overcame the linear issues of perceptrons, kernel methods allow SVMs to overcome the linear issues of optimal linear classifiers by adding more dimensions to the feature space to allow linear separation of different classes. Unlike neural networks the SVM algorithm can be decoupled from the application domain meaning that the neural network problem of pre-configuration is not relevant to SVMs. However, this problem is replaced by the problem of deciding on a suitable kernel for the SVM.

One constraint of such learning methods, with complex internal knowledge representations, is that the learning process is often difficult to interpret or to explain in human-understandable terms. In many common application domains such as vision systems such limitations are rarely an issue. However, in a pervasive environment users will be keen to view and understand what the system has learnt about them and hence what system behaviours to expect. Users may also want to manipulate this information in some way to adjust system behaviour. Therefore, when employing ANNs for implicit personalisation, such issues must be considered.

2.3.2 Stochastic Models

In the 1990's a trend towards learning complex stochastic models emerged in the machine learning field. It had been identified that more general learning techniques such as ANNs could not easily incorporate prior knowledge and were often difficult to interpret. In contrast stochastic models describe the real world process in which the data was observed allowing for easier interpretation as well as the incorporation of prior knowledge. Such models are typically probabilistic graphs depicting the probabilistic dependencies among variables. Once a stochastic model has been learned over a set of training data, probabilistic inference can be carried out for prediction or classification of future inputs. Indeed, such techniques are often used for context inference in pervasive environments (e.g. Mobilife). As can be seen from Table 1 stochastic techniques are also the most favoured for implicit personalisation systems.

Bayesian Networks [81] are directed graphical models representing causal relationships between random variables. The underlying paradigm is Bayes' theorem of conditional probabilities from the field of statistical mathematics. A Bayesian network consists of nodes, each representing a random variable. The nodes are connected such that a directed arc (with an assigned probability value) is drawn from node A to node B if B is dependent on A. By constructing a network of feature nodes and target nodes, the most probable hypothesis can be inferred given the input data.

Many different variations of Bayesian networks exist. The Bayes Optimal Classifier determines the most probable classification for a new instance by combining the predictions of all hypotheses weighted by their posterior probabilities. A more efficient method is the Naïve Bayesian Classifier. This approach is implemented by the Mobilife project for the classification of recommendations based on context inputs [82]. It makes the assumption that the presence of a feature is unrelated to the presence of any other feature. For example, an animal may be classed as a fish if it has the features: *lives in water, has gills* and *has scales*. A naïve Bayes classifier will consider each feature independently even though one depends on the existence of others. The advantage is the ability to learn parameters for classifications on a small training set.

Markov models [81] are a subclass of Bayesian networks often referred to as dynamic Bayesian networks. The distinction is that Markov techniques model the probabilistic dependencies between actions in a linear sequence with the assumption that an action can cause another action in the future. At the simplest level a first order Markov chain models the probability of an action x_n given the occurrence of its immediate predecessor x_{n-1}. Scaling up to the Mth order, Hidden Markov Models (HMM) model the probability $p(x_n \mid x_{n-M},...,x_{n-1})$ as a state transition diagram. This ability to model dependencies in sequential data has led to their widespread use in personalisation systems that operate on task-based user profile data where sequential patterns of behaviour are considered.

The Synapse project utilises HMMs to model relations between context and service usage [83]. An initial learning phase generates the HMM from context and service parameters. A consequent execution phase computes the occurrence probability of services based on the learned parameters and new context situations. Services with the highest occurrence probability are either executed or suggested to the user.

The MavHome project utilises Hierarchical Hidden Markov Models (HHMMs) for the modeling of sequential behaviour patterns [84]. In a HHMM each state is a probabilistic model in itself rendering the HHMM a recursive model of sub-HHMMs. In contrast to a HMM each state returns a sequence rather than a single observation. When a state is activated it processes its own probabilistic model activating states which in turn activate their own models and so on. This makes HHMMs the obvious choice for task modelling where tasks can take a hierarchical form, recursively consisting of sub-tasks. However, one potential drawback of Markov models is their over-simplified assumption that each state is only dependent on its predecessors. When applied in the pervasive domain, care must be taken to ensure the correct timing and context for the application of predicted future actions. For example, the application of action C might be dependent on the time interval between actions A and B. Similarly the application of some action A (e.g. 'unlock car') might be dependent on user context (e.g. the proximity of the user to their car).

To utilise most stochastic learning techniques a priori knowledge is required of the necessary model structure and parameters. This is due to the task specific nature of stochastic models. This lack of generality may be an issue in pervasive domains. The use of stochastic models for context inference is favoured as a priori knowledge of necessary parameters such as sensor inputs is possible. However, for personalisation purposes, if new behaviours appear due to the availability of new services it may not be known what context parameters influence the new behaviour. Devising effective algorithms to implement the learning of required structures and parameters is complex and the focus of ongoing research.

2.3.3 Fuzzy Logic

In contrast to *crisp* logic with precise values (e.g. 0 and 1) fuzzy logic is a multi-valued logic that can represent the degree of truth of a statement (i.e. a value between 0 and 1). It attempts to mimic human thinking by removing the need to specify strict binary thresholds. For example, human concepts such as 'hot' and 'cold' have no strict temperature thresholds as what one person might conceive as hot might not be the same for another person. Further, if we were to derive a threshold temperature of 20°C as hot, we would not consider the temperature 19.9999°C to be cold.

In comparison to probabilistic models, fuzzy logic does not represent the likelihood of some event. Instead it operates on *degrees of truth* representing the membership of some entity in vaguely defined sets. Its ability to handle uncertainties has proved successful in complex systems and application areas such as automatic transmissions, air conditioners and elevators to name a few. Equally it has proved successful in implicit personalisation where uncertainties exist due to unstable environments and changing user behaviours.

The Ubisec project employs fuzzy logic decisions such as device selection [85]. This seemingly trivial action becomes increasingly complex as human factors such as vision are considered. For example, the range of human sight varies between short-sighted and long sighted so a device selection that may suit one user will be inappropriate for another. iDorm also reflects on the uncertainties introduced by human factors. They identify that behaviours can change through time, e.g. due to seasonal variations. Further, concept definitions may also change. Consider how the term 'warm' can vary in meaning between summer and winter.

iDorm utilizes a multi-step learning process using fuzzy membership functions (MFs) [86]. User behaviour is monitored and MFs are extracted to form a fuzzy logic controller (FLC) that models the user's behaviour. The FLC is used to adapt the user's environment and incremental adaptations are applied to the FLC in real-time based on user feedback. This adaptation process continues until uncertainties increase above some threshold (due to changing behaviours or environmental aspects). At this point, monitoring is re-triggered and MFs are extracted to create new FLCs that better model current user behaviour. This is repeated each time uncertainties reach some threshold. In this way, the learning process provides life-long learning with a rapid response to changes between re-learning of FLCs (similar to rapid response concepts implemented in SPICE and MobiLife).

Another advantage of fuzzy logic is the ability to translate the MFs into human-readable IF-THEN rules. This enables the user to view, understand and manipulate their profile information. As mentioned above, in a pervasive environment it is desirable that learnt profile information is available to the user in this way to ensure both a sense of control and a mental picture of expected system behaviour.

2.3.4 Rule Learning

Rule learning falls within the large family of symbolic (logic-based) machine learning techniques. The common aim of symbolic techniques is to determine a knowledge representation that describes the relationship between the features and the classes. This knowledge is represented by decision trees or logic based rules and therefore is naturally human-understandable.

Decision Tree algorithms construct a decision tree as the knowledge representation. If required this can be easily parsed into rules for better human-readability. Each node in the tree is an attribute test and the leaf nodes represent the classifications. Quinlan's ID3 [71] and C4.5 [87] algorithms are the most widely used tree building techniques. The shape of the tree is determined by the *information gain* of attributes where attributes with higher information gain better separate the training examples according to the target classification and hence should be closer to the root. Once the tree is constructed over all the training examples, new, previously unseen instances are processed down the tree at each attribute test node until some classifying leaf node is reached. Over-fitting is a problem that is especially relevant to decision tree learning. However this problem may be prevented through pruning techniques, such as *reduced-error pruning* [88] where sections of the tree that provide little classification power are removed.

Inductive learning or concept learning, constructs a set of logic-based rules as a knowledge representation. Research has often focussed on learning general concepts and categories, reminiscent of how humans continually perform such tasks. A common aim is to 'find' some hypothesis that approximates the target function over the set of training examples given, with the assumption that this hypothesis will also approximate the target function well over other unobserved examples. Therefore it can be viewed as a search problem in that the hypothesis which best satisfies the training examples must be found within the space of all hypotheses.

One major issue with symbolic learning is that algorithms often scale badly in terms of computational efficiency as the size of the dataset increases, increasing the hypotheses space and hence requiring a longer search time. Mitchell showed how version spaces can be used to minimise the search by bounding the hypotheses space to contain only those hypotheses that conform to the training data. His Candidate-Elimination algorithm [89] used general (G) and specific (S) limits to bound a subset of the hypotheses space. However, the algorithm is not robust to errors and noise in training examples. To overcome this issue, several inductive rule learners have utilised adapted decision tree pruning techniques such as Quinlan's reduced-error pruning.

The Ripper algorithm [90] used for preference learning in the Mobilife project is based on the reduced-error pruning (REP) for rules technique. When compared with less scalable learning algorithms such as C4.5, the Ripper algorithm is much more efficient on large datasets while achieving comparable error rates. This is desirable in a real world environment where datasets can be very large, especially when monitoring user behaviour over weeks and months for implicit personalisation.

The SPICE project utilises a very different rule learning technique due to specific project goals of providing recommendations [91]. For this, they have implemented association rule learning, a technique commonly used for basket analysis. The chosen algorithm is the Apriori algorithm [92] that identifies commonly co-occurring *items* in a list of *transactions* (e.g. 86% of people who bought milk also bought eggs). The knowledge representation contains association rules of the form (X, Y) that holds based on the percentage of transactions within the transaction list that contain X and then also contain Y. Such rules are generated from *frequent itemsets* identified during algorithm runtime. Starting with itemsets of length 1, the itemsets are grown with each pass through the transaction list by generating candidate itemsets until the maximal length itemsets with sufficient support are found. As with other rule learning approaches scalability is an issue. However, extensions such as AprioriHybrid scale linearly with the number of transactions.

2.3.5 Reinforcement Learning

Reinforcement learning [78] is based on psychological studies and observations of biological learning processes with the basic assumption that actions accompanied or closely followed by satisfaction will be more likely to re-occur should the situation re-occur, whereas actions which are accompanied or closely followed by discomfort will be less likely to re-occur should the situation re-occur. It is a technique most commonly used for agent or robot learning where the agents exist in an environment with a given state and each agent has a set of actions which it can perform to alter the state of the environment.

Several issues are commonly associated with this learning approach. Firstly, the agent must overcome the problem of *temporal credit assignment*. The goal is to learn a control policy which will indicate an action sequence that will maximise its cumulative reward from any given state. However the agent must determine what actions within the sequence should be credited with producing the final reward. Secondly, the agent must overcome the problem of *exploration*. The agent must somehow determine when to exploit known actions with known rewards and when to explore unknown actions with unknown rewards.

The Adaptive Home project utilises reinforcement learning in a less traditional application area (but with successful results) in a smart, energy efficient house [45]. The Q-learning algorithm is employed to find an optimal control policy that will balance energy costs against the discomfort costs of inhabitants. Bias is initially towards mitigating energy costs; therefore initially the system will opt to keep lights off and heating to a minimum. As the user over-rides such automated decisions the system learns by increasing the discomfort costs which in turn alter the optimal control policy. The system also exploits the explorative nature of reinforcement learning by occasionally selecting a reduced energy setting (e.g. lower heating temperature), unbeknown to the inhabitant. If the inhabitant does not complain, the energy and discomfort costs are updated appropriately to reflect this new control policy.

2.3.6 Incremental Learning Algorithms

It is interesting to note that many pervasive projects implement *batch* learning algorithms for implicit personalisation in the pervasive domain. Some projects (e.g. SPICE) employ incremental profile updating mechanisms for rapid response to behaviour changes but such mechanisms are temporary, still relying on batch learning executions to provide a more stable user profile. Although these projects have produced encouraging results, perhaps incremental learning algorithms could provide a more natural and successful way to create and maintain user profiles in pervasive systems.

Within a pervasive system, examples of user behaviour will typically become available and hence be monitored by the system one at a time, over time (as the user interacts with their environment). It would be desirable to process examples as they occur, altering the target function in real-time without the need to retain large stores of behaviour examples or re-process past examples when new examples become available. Indeed, Giraud-Carrier [93] states that the process of building and maintaining a user profile is essentially an incremental task and convincingly argues that although incremental tasks can be handled by non-incremental algorithms, the most natural and flexible way to handle an incremental task is with an incremental algorithm. In machine learning literature there is no single definition of the term 'incremental' when referring to the behaviour of an algorithm. Different algorithms see different properties as their defining incremental feature and why the algorithm itself should be termed incremental. Hence, there is huge variation among incremental algorithms. Therefore the incremental algorithms reviewed in this section will be discussed based on their defining incremental properties of which the most common are described below:

Property 1: Process input one at a time over time (no a priori training data)

Incremental in this sense means that the algorithm should not be reliant on receiving an entire training dataset before the learning process begins. Instead learning should be continuous through time on a case by case basis as new examples are presented. Since no a priori dataset is available, no dataset sampling or re-ordering can be performed as is sometimes the case to improve the convergence rate and accuracy of algorithms. In addition it should be possible to retrieve a *best estimate* hypothesis of the target function after each new example has been processed.

Property 2: No re-processing of past training data

Non-incremental algorithms such as ID3 and Multi-layer Perceptrons (MLPs) can learn a target function that represents the training data but if new instances are presented the target function must be re-learned using the original training set plus the new instances. If the original training set is not presented again during the re-learning phase the new target function would no longer represent the original training set in what is often referred to as *catastrophic forgetting*. Hence a store of all past training data must be retained. This has obvious spatial and temporal consequences. As the store of past training data grows so do storage capacity requirements and the time taken to re-learn the target function. In practical terms, it is undesirable to maintain such a store ad infinitum. In contrast, incremental algorithms defined by this property are *memoryless* in that they do not require access to an explicit store of all past examples. Instead such algorithms store information that *represents* past examples within their knowledge base (e.g. as network weights).

Property 3: No A priori knowledge of the problem domain

It is the case that some algorithms require domain specific parameters to be set prior to learning. For example, neural networks often require a learning rate to be set. Finding the correct value for such parameters greatly influences the possibility of convergence and hence the success of the learner. However, identifying the optimal values for some problem domain is usually a trial and error process.

Property 4: Growing/ shrinking topology

Many algorithms are defined as incremental in terms of this property. Rather than defining the learner topology a priori to cover the training dataset, the topology grows by adding new concepts or classes to allow the proper classification of new examples. Pruning operations can also be performed to sections that provide little classification support allowing the topology to shrink as well as grow.

Property 5: Selective training data

This property is derived as a subset of *active learning*. Algorithms that employ active learning select the most useful training dataset from a list of all candidate datasets with the aim of increasing convergence success and speed. The two approaches to active learning are *selective learning* and *incremental learning*. The former selects a completely new dataset from a candidate list at each subset selection, replacing it after use. In contrast the latter selects a dataset from a candidate list at each subset selection and adds it to the training set. Therefore, when implementing incremental learning the candidate list of datasets shrinks and the training dataset grows as training continues.

The term *on-line* is often used interchangeably with the term incremental. However, this is most common when the algorithm possesses property one. For clarity, the term incremental will be used throughout this thesis. Table 2 lists several incremental algorithms and identifies their defining incremental properties. Each of the algorithms listed is discussed in more detail below. It should be noted that some properties are not mutually exclusive. For example it is not possible for an algorithm to require no a priori training data and utilise selective training data techniques.

	Process input one at a time over time (no a priori training data)	No re-processing of past training data	No a priori knowledge of the problem domain	Growing/ shrinking topology	Selective training data
Candidate-Elimination	✓	✓		**n/a**	
AQ15	✓	✓		**n/a**	
STAGGER	✓	✓		**n/a**	
COBWEB	✓	✓	✓	✓	
ID4/ID5/ID5R	✓	✓	✓	✓	
Pocket Algorithm	✓	✓			
WINNOW	✓	✓			
ART	✓	✓	✓	✓	
ITDNN	✓			✓	
IBPLN	✓			✓	
CSAILA				✓	✓
SELF				✓	✓
Grippo					✓
Learning++		✓			✓
SwiftFile	✓	✓		**n/a**	

Table 2. Incremental properties possessed by incremental algorithms

Since the late 70's many incremental concept learning algorithms have emerged. Perhaps this is due to the inspiration taken from considering how humans perform such a task or perhaps because of the unstable problem domain where new concepts can appear and existing concept definitions can change. One of the earliest incremental approaches is Mitchell's Candidate-Elimination algorithm [89]. It is based on the use of general and specific delimiters to define the version space of all candidate hypotheses and is defined in terms of a *search* problem for the hypothesis that best represents the presented positive and negative examples. As new examples are presented sequentially over time the general boundary set (G) becomes more specific and the specific boundary set (S) becomes more general to be consistent with the presented examples.

This results in a continually reducing version space of the candidate hypotheses where eventually G and S boundaries converge on the same hypothesis.

An obvious consideration is how the algorithm handles noisy data, or in relation to use in an unstable environment, how the algorithm would handle a moving target function due to possible changes in concept definition (often referred to as *concept drift*). Mitchell states that the algorithm is not robust to this situation as once an instance has been presented, all inconsistent hypotheses (including that which may be the future target hypothesis) are removed from the version space with no possibility for future consideration. In such a situation the result would be non-convergence.

However, many other algorithms have taken inspiration from Mitchell's search problem approach and provided solutions to robustness issues. The STAGGER [94] algorithm is not only robust to noise and concept drift, but even attempts to distinguish between them for better performance. Like Mitchell's solution, STAGGER is also based on a search through the hypothesis space; however, the search moves towards general (G) and specific (S) limits rather than starting at such limits. The search is guided by the type of error where a wrongly included negative example leads to a more specific search and a wrongly omitted positive example leads to a more general search.

AQ15 [95] is a descendent of the AQ1-AQ11 series of inductive learning algorithms. Here the search problem is through a space of logical expressions with the aim of determining those that represent all positive and no negative examples. It learns incrementally with *full memory* meaning that it doesn't forget any presented examples or rules it has formed. As with the two algorithms above it is incremental in terms of processing sequential inputs and a non-dependency on a priori training datasets; however, all three algorithms require a priori knowledge of the problem domain such as problem specific parameters to initialise search spaces and guide searches.

In contrast COBWEB [72] can operate successfully without such a priori information. It is an unsupervised concept clustering algorithm that learns concepts by building and adapting a concept tree as each new example is received. Again, the algorithm is based on a search problem; however, it is assumed that no 'teacher' exists to pre-classify examples. Therefore the algorithm must search for appropriate categories as well as appropriate concepts for each category. A probability function termed 'category utility' determines the tree structure and is considered when deciding if a new example should be incorporated into an existing category or whether a new category should be created. Past examples are represented in the tree as probabilities held at each node and hence are retained by the algorithm without the need to maintain an explicit store of past examples.

Due to their symbolic base, parallels have been drawn between the COBWEB approach to building a concept tree and the way in which incremental versions of ID3 build a decision tree for classification. In both techniques tree structures are determined by some utility parameter and past examples are represented at tree nodes as adaptable values. The ID4 algorithm [96] builds a decision tree on an instance by instance basis depending on the calculated information gain of attributes at each decision node. Past examples are represented at tree nodes as positive and negative counts of decision attribute values. If a new instance leads to a different decision attribute having a higher information gain at a decision node, the decision node over-writes the existing decision attribute with the new winner and the sub-tree below that node is discarded.

However in 1989, UtGoff [97], showed that the ID4 algorithm could not learn some functions that were learnable using ID3. Notably, where there was little difference in the information gain of decision attributes at some decision node a thrashing behaviour was observed where sub-trees below the decision node would be continuously discarded. His alternative solution was the ID5 (and later ID5R) algorithm that utilised the ID4 process for decision attribute selection at a decision node but rather than discarding the sub-tree, it restructured the entire tree by pulling the desired decision attribute up to the root (in a process called *pull up*). Analysis shows that ID5 can produce the same tree as ID3 but with fewer training instances.

Although concept learning has produced many incremental algorithms based on inductive learning and the search problem another common domain of incremental algorithms is in connectionist approaches. This is most likely due to the way in which neural networks naturally represent past examples as easily adjustable weights in a network structure. Perhaps the parallels between neural networks and biological learning processes also have some bearing. In any case a multitude of incremental neural networks are available, each exhibiting various incremental properties.

Gallant's pocket algorithm [98] is a perceptron-based incremental learning algorithm for neural networks. It works on the premise that the weights providing the best solution so far are kept in your 'pocket'. These pocket weights are replaced by the actual perceptron weights if the perceptron weights outperform the current pocket weights in correctly classifying new training instances. In this way the algorithm can produce the pocket solution at anytime, providing the best estimate of the target function so far without the need to refer to past examples. The original pocket algorithm requires no a priori training dataset; however, when a dataset is available an optional *ratchet* can be used to improve algorithm performance.

Like the pocket algorithm, WINNOW [99] assumes a definition of incremental learning based on no dependence on a priori training data and no dependence on a store of all past examples. Weights are updated after a new example is presented to incorporate the example into the network target function. The algorithm focuses on reducing mistakes by identifying irrelevant attributes and hence scales well to high-dimensional problem domains. However, when implementing both WINNOW and the pocket algorithm the network topology must be determined a priori and hence a priori knowledge of the problem domain is required. This is not the case for other incremental neural networks where a defining feature is a network structure that grows to accommodate new examples.

The ART (Adaptive Resonance Theory) family of neural networks utilise clustering algorithms for pattern classification. The basic principle introduced by Grossberg [100] involves two node vectors where F_A corresponds to the input pattern and F_B corresponds to the target output. Inputs are sent from F_A through weighted connections to F_B where nodes compete until only one F_B node is active. This winning node responds through weighted connections to the F_A nodes where the initial input activation is compared with the activation in response to the top-down signal from the winning F_B node. A vigilance parameter determines the similarity threshold required between the two activations. If this is met the winning F_B categorises the inputs, if not another F_B that better meets the vigilance parameter is searched for. If no such F_B exists, a new F_B node is created to categorise the input. In this way ART networks grow over time without the need for a pre-defined static topology. Inputs are processed sequentially over time and the algorithm does not rely on an a priori training dataset. Many variations to the basic algorithm exist for both unsupervised [95, 96, 97, 98] and supervised [99, 100] learning.

In addition to growing networks, there also exist networks that shrink. Incremental learning is defined in terms of the growing and pruning operations. However, several algorithms use such operations as temporary responses to new examples. Both the Incremental Time Delay Neural Network (ITDNN) [107] and the Incremental Backpropagation Learning Network (IBPLN) [108] algorithms use incremental growing and pruning adaptations as temporary solutions until a complete re-training of the network can be performed on the entire set of past examples. This is reminiscent of the rapid response mechanisms employed by various pervasive personalisation systems such as SPICE where temporary profile updates are performed incrementally (based on each instance of user feedback) in between scheduled profile re-learning cycles. However, dependence on a store of past examples is obviously in direct conflict with other common definitions of incremental learning.

The Clustering Sensitivity Analysis Incremental Learning Algorithm (CSAILA) [109] and Selective Learning with Flexible Neural Architectures (SELF) [110] algorithms conform to another definition of incremental learning as a subset of *active learning* where training data is selectively chosen from a store of possible candidate datasets that have not been previously used. Hence, by using selective training data as the defining incremental feature there will always be a dependency on a priori training datasets. The CSAILA algorithm first clusters the candidate training set. Then at each subset selection interval the most informative pattern from each cluster is selected as input to a growing neural network. The SELF algorithm is based on the observation that networks trained on *border* patterns (i.e. those that lie close to separating hyperplanes) generalise better than those trained on random patterns. Hence a *seed* training set is selected appropriately from the candidate training set and continually expanded until the network generalises over the candidate set.

Other incremental algorithms perform similar pre-processing functions on training datasets. Grippo [111] utilises sampled datasets to create copies of a neural network where suitable incremental training algorithms can be defined to reduce disagreements between network copies and hence converge on a target function. Similarly the Learn++ algorithm [112] utilises sampled datasets for the incremental training of neural network pattern classifiers. Sampling is driven by a distribution that ensures previously misclassified examples have a higher probability of being selected. An ensemble of classifiers is generated using the sample datasets and finally each hypothesis is merged in a voting process.

Unlike the algorithms mentioned above, SwiftFile [113] is a system utilising incremental learning to predict how incoming emails should be filed based on observed user behaviour. Although not strictly in the pervasive domain, SwiftFile is one example of a profiling system that utilises an incremental learning algorithm to automatically predict how incoming email should be filed. The authors identify some of the major issues of learning user profile data in the real world including no a priori training data and dynamic environments where new classes can appear and user behaviour can change.

The algorithm itself is a modified, incremental version of a Term Frequency-Inverse Document Frequency (TF-IDF) text classifier, that uses statistical methods to determine what folder an email should be filed into based on word occurrences. Initial evaluations on accuracy are not particularly positive but when coupled with appropriate user prompting the algorithm is sufficient to satisfy end user requirements. Interestingly the authors investigate whether a batch algorithm retrained on a nightly basis would fare better in this dynamic, real world domain. Their analysis shows that the incremental algorithm out-performs batch counterparts (emphasising the claims of Giraud-Carrier). They conclude that batch algorithms cannot respond rapidly enough to new classes or changes in user behaviour. Although incremental algorithm accuracy also drops in response to such changes, it recovers quickly and retains a higher average accuracy throughout testing.

2.3.7 Conclusion

The field of machine learning has enjoyed much research attention although this has shifted over the years between different learning paradigms [114]. As new application areas emerge new learning challenges must be addressed and hence machine learning continues to be a popular and important research area. Of particular note is the exploitation of learning algorithms for implicit personalisation in pervasive environments. Due to the task specific nature of most learning algorithms, a variety have been applied within pervasive systems depending on specific goals such as preference learning, task learning or other inference. These algorithms (and some alternatives) were discussed above in terms of their strengths and weaknesses with regard to implicit personalisation in the pervasive domain.

It was noted that most pervasive systems adopt a batch learning approach where learning cycles occur at certain intervals and algorithms are dependent on a priori datasets and access to stores of all past training examples. Although successful results have been achieved by such systems various literature suggests that learning profile information such as that used for personalisation is inherently an incremental task and therefore could be better handled by an incremental algorithm. Specifically, incremental processing of inputs allows for more rapid response to changes in user behaviour as processing is immediate rather than scheduled and processing does not require complete retraining on the entire set of past examples.

Several notable incremental algorithms were discussed and on reviewing the literature a trend seemed to emerge. During the 80's and early 90's incremental algorithms enjoyed much interest with various notable works. However, since then it seems incremental algorithms have suffered something of a winter in terms of research interest. This may be due to the fact that incremental tasks can be handled by non-incremental algorithms where the use of a priori training examples allow for *closed-world* systems (where the world is confined to the training examples) that are theoretically convenient in terms of evaluation, comparison and enhancement. However, this confined view is not representative of real world problem domains such as pervasive environments.

2.4 Summary

Mark Weiser's seminal paper outlined his vision of 'Ubiquitous Computing' (also termed Pervasive Computing or Ambient Intelligence) where everyday environments are filled with networked, computational technology weaved into daily life. At a time prior to significant hardware and software advances this was an ambitious goal. Since then innovations in key areas such as connectivity, mobility, and HCI are providing the component parts to realise Weiser's vision. We are already starting to face challenges caused by ubiquitous access to a multitude of resources in a climate of many computers to one user. Significantly, the user is faced with the issue of resource management. With a plethora of available services, networks and devices, managing such an array manually can place huge burdens on the user and will often fall short of user needs.

Context information is key to addressing this issue. By knowing the context of the user, systems can better decide how to configure resources. This is demonstrated in a number of projects implementing context-aware adaptation where intelligent spaces (e.g. the home) are configured according to the user's context (e.g. if a user enters a dark room the lights are turned on). Although useful, context-awareness alone cannot meet the needs of every individual user as it does not consider user specific needs. Rather, context-aware adaptation will be uniform for all users.

In contrast, personalisation incorporates the user's context information as well as other user centric information such as user preferences and tasks, to configure resources so they appear differently to different users or to the same user in different contexts. Therefore, personalisation has become a key concept in pervasive systems. Its utilisation in the pervasive domain has thrown up new and interesting challenges, not least the issue of how user centric information (held in a user profile) is created and maintained. The explicit personalisation approach relies on manual creation and maintenance by the user. In contrast the implicit personalisation approach utilises monitoring and machine learning mechanisms to create and maintain a user profile on behalf of the user. Both approaches have been adopted to differing degrees by various pervasive projects but finding the correct balance between user control and automated system behaviour remains difficult.

Although more in line with pervasive ideals, implicit personalisation introduces the challenges of learning user profile information from monitored user behaviour. Many different machine learning techniques are utilised in various pervasive systems depending on specific project goals. Each learning technique has various strengths and weaknesses when applied in the pervasive domain and solutions must consider changing user behaviours, changing environments, large datasets with many features as well as temporal efficiency and spatial issues. Although most projects implement batch learning approaches, it is suggested that learning for implicit personalisation is an incremental task and hence better addressed by an incremental learning approach. Although relatively few projects implement incremental learning algorithms, initial results from such systems seem to support the above claim.

This chapter has provided an overview of the techniques currently employed for preference learning in various personalisation systems and has raised several questions such as the use of batch learning algorithms. Chapter 3 provides a more in-depth analysis of the DAIDALOS personalisation system and highlights several key lessons learnt by the author from the first hand experience of its design and implementation. Chapter 4 combines the findings from Chapters 2 and 3 to present a set of design features and requirements for an efficient preference learning technique that is specifically tailored to personalisation in the pervasive computing domain.

3 Research Prototypes - Personalisation in the DAIDALOS Project

3.1 Introduction

The DAIDALOS (**D**esigning **A**dvanced network **I**nterfaces for the **D**elivery and **A**dministration of **L**ocation independent, **O**ptimised personal **S**ervices) project was an EU Framework 6 Integrated Project which ran over two phases from January 2004 until December 2008 [115]. With over forty partners from academia and industry it aimed to design and develop a beyond B3G (Beyond 3rd Generation) framework to support heterogeneous network and service infrastructures for the mobile user allowing seamless and ubiquitous access to pervasive services and information. The research areas covered were diverse and ranged from the network level up to service provisioning and user experience level. Covering such a broad range of research areas, the project relied on five key concepts to drive all design and development in a common direction. These were as follows:

- MARQS – Mobility Management, A4C (Authentication, Authorisation, Accounting, Auditing, Charging), Resource Management, Quality of Service and Security
- VID – Virtual Identities
- USP – Ubiquitous and Seamless Pervasiveness
- SIB – Seamless Integration of Broadcast
- Federation – allowing network and service operators to offer and receive services

These concepts permeated every level of the project although at the various levels some concepts were more applicable than others. At the service provisioning and user experience level the concept of Ubiquitous and Seamless Pervasiveness was one of the stronger factors. This proposed that all heterogeneous network technologies and services should be available ‘anytime, anywhere’ and should be seamlessly adaptable due to changing environments and user needs. To realise this concept an intelligent middleware was designed and developed to run on top of the heterogeneous network technologies.

Phase one of the project (DAIDALOS I) produced an initial middleware prototype, termed the DAIDALOS Pervasive Service Platform (PSP) [116]. Figure 1 illustrates the PSP architecture including the enabling services it provided such as context-awareness, privacy and security, service management and personalisation.

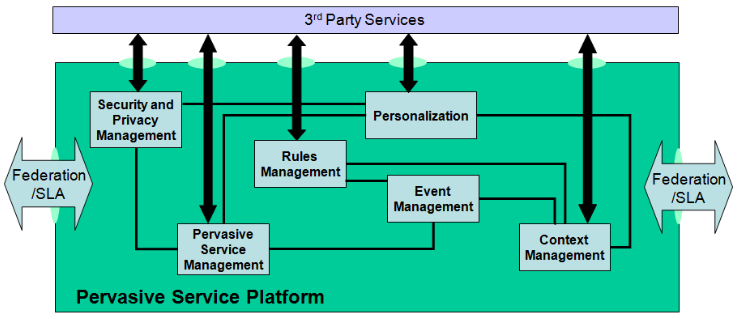


Figure 1. Architecture of the DAIDALOS I Middleware

Phase two (DAIDALOS II) produced a two-layer version of the middleware [117]. Figure 2 illustrates the two-layer architecture. 'Core' enabling services (such as service management and privacy and security) were separated into a lower Management layer while 'enhancing' enabling functionalities (such as context management and personalisation) resided in a higher User Experience layer.

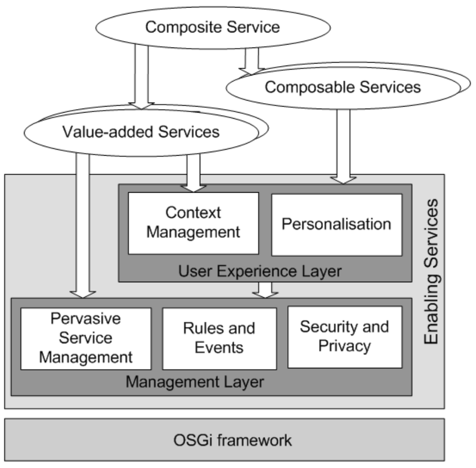


Figure 2. Architecture of the DAIDALOS II Middleware

Dependency restrictions meant that lower layers could not be dependent on upper layers. This constraint affected the design of internal systems including the Personalisation system. It became difficult to pass control from the management layer to the user experience layer and constrained the potential of the enhancing functionalities. Another constraint placed on the DAIDALOS II PSP was the prohibition of any decision making and proactive behaviour. This meant that the PSP could not automatically adapt or reconfigure services directly. Instead the service was to be notified of the required action by the PSP and would then decide whether to implement it based on other internal factors. Again this had a direct impact on the Personalisation system where its major function is the adaptation of services. The impact of these design decisions are discussed with regard to the Personalisation System (PS) throughout the following sections and in several reflective papers [112, 113].

In both prototypes the PS played a key role in realising the USP concept. The following sections describe PS prototypes produced in both project phases and their evolution from the initial phase of the project through to the final project demonstration at the end of 2008. It should be noted that the initial PS prototype for phase one was largely designed and implemented prior to the author's involvement and therefore it will not be described in full detail.

3.2 DAIDALOS I PS Prototype

The USP concept introduces two key challenges in terms of user experience. Firstly it proposes that all networks and services should be available 'anytime', 'anywhere'. This presents the user with a plethora of resources. It is essential to ensure that the user is not overwhelmed by such an array and the advertisements, requests and information generated by each resource. Secondly with so many available resources, manual (re-) configuration of each to meet current needs is no trivial task.

To provide a pervasive experience, where possible such management responsibilities should be performed on behalf of the user in line with their current context and needs. Personalisation provides mechanisms to achieve this. User needs are expressed as preferences that can be used by personalisation processes to manage resources (both in terms of access and adaptation) on behalf of the user.

3.2.1 DAIDALOS I PS Architecture

The initial PS [120] supported various personalisation tasks with regard to both third party services and enabling services internal to the PSP. Figure 3 illustrates the architecture of the first PS.

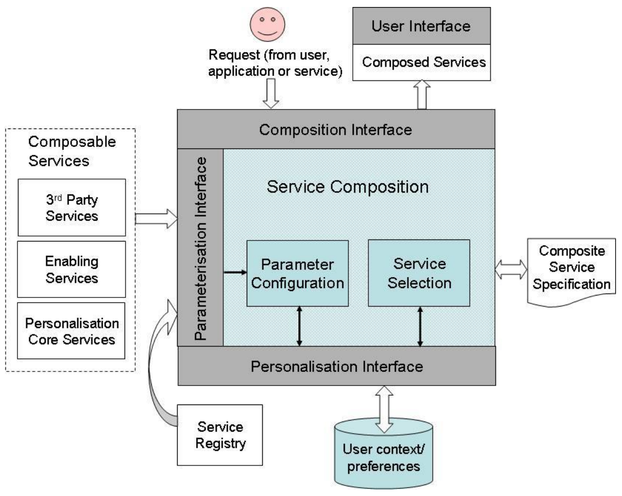


Figure 3. The DAIDALOS I PS Architecture

A preference set (stored in the Context Management database) contained a variety of preferences covering each personalisation task. With regard to third party services, the PS provided support for personalised service parameterisation. Each third party service could specify a number of ‘personalisable parameters’ (e.g. volume, font size, background colour, etc.). Such parameters were populated by the PS using *personalisable parameter preferences* during service composition and instantiation to configure the service appropriately.

With regard to enabling services (enabling functionalities within the PSP) the PS provided the following key tasks:

1) Personalised Service Parameterisation - As with third party services, enabling platform services could also have various personalisable parameters. When the enabling service was initiated the parameters were populated based on *personalisable parameter preferences*.
2) Personalised Service Selection - When a service request was received, a two-step process was triggered. Firstly, all applicable services were discovered using conventional service discovery. Secondly, the list of discovered services was ranked based on *service selection preferences* to provide the service which best met the current needs of the user [121]. For example, at work the user may prefer the service with the highest QoS regardless of price while at home they may prefer the cheapest service regardless of QoS.
3) Personalised Service Composition - In DAIDALOS I several individual services could be assembled into a single composite service. Personalisation allowed user

requirements to impact the composition process through *composition preferences* [121]. Under certain circumstances, services could be added/removed from a composition or the order in which services were placed in the composition could be altered (e.g. the user may not want to book airline tickets until accommodation is confirmed).

As illustrated in Figure 3, the DAIDALOS I PS explicitly provided functionality for each of the personalisation processes required and took full responsibility for their implementation. In other words, personalised parameterisation, service selection and composition functionality resided within the PS rather than externally in other dedicated enabling services. The PS utilised preferences to perform the functions internally. Although the PS supported all personalisation processes required, its process specific design impeded the addition of support for new personalisation tasks such as personalised redirection [116, 117] which was implemented externally to the PS in the DAIDALOS I platform.

3.2.2 User Preferences

In DAIDALOS I a preference was represented as a context-dependent IF-THEN-ELSE rule. Each preference contained various *context conditions* that represented some contextual situation which, if satisfied, indicated that the related *preference outcome* should be implemented. An example of a simple preference is illustrated below:

```
IF <location = home>                    (context condition)
        THEN [service = sports]         (preference outcome)
        ELSE [service = news]           (preference outcome)
```

This preference states that a sports service is preferred when the user is at home and a news service is preferred when the user is not at home. Besides simple assignments, preference outcomes can also be nested IF-THEN-ELSE preferences allowing for a recursive and rich structure that could model any context-dependent behaviour. As mentioned above the DAIDALOS I PS utilised a store of such preferences to drive personalisation for each user. The preference store was pre-set prior to usage but it was assumed that the preferences would be added and edited through time manually by the user. This approach conforms to accepted profiling standards that define guidelines for the management of user profiles (which include user preferences). W3C provides CCPP (Composite Capabilities/Preference Profiles) Structure and Vocabularies [51] outlining profile management guidelines within a web environment. It assumes a manually editable profile which does not change often through time.

As such, the PS provided little support for the creation and maintenance of the preference set other than an editor GUI for manual user updating. The user could view their stored preferences and manually manipulate the preference set to fit their changing needs. Such an approach is acceptable in a stable environment where updates are seldom required but the dynamic nature of pervasive environments lends itself to a more frequently updating preference set. User needs will change through time due to lifestyle changes, the availability of new resources, etc. The user's preference set must also be altered to reflect changing user needs if adequate personalisation is to be maintained. Due to the frequency of required updates purely manual creation and maintenance of a pervasive preference set quickly becomes a continuous and laborious task, eventually defeating the benefits that personalisation aims to provide.

This situation is not considered in current W3C standards which are not specifically tailored for pervasive environments. The DAIDALOS I PS confirmed that placing such a heavy burden of preference management on the user had a negative effect on personalisation. Users were reluctant to invest such effort into preference set maintenance. This led to a sparse and outdated preference set which was reflected in the quality of personalisation.

As well as preference rules, the DAIDALOS I PSP also included a *Rules Management* (RM) system that also stored and managed rules relating to the configuration and behaviour of enabling services within the PSP. There was not a clear distinction between these rules and the preferences managed by the PS. An example of a rule in the RM system was to trigger context dependent session transfers. It is not clear why this rule was not regarded as a context dependent preference and why such a separation of rule stores existed.

One possible solution relates to the design of the DAIDALOS I PS. As mentioned above, its task specific architecture rendered it very difficult to add support for extra personalisation tasks. Since no functionality existed within the PS to perform session transfers this was implemented externally in the Pervasive Service Management system utilising non preference rules.

3.2.3 Applying Personalisation

Personalisation was applied in two different ways. Preferences were always available on a request basis from the PS. At any time, a service could request preference information to personalise some aspect of itself. In addition the PS automatically configured services by applying preferences during service deployment. Once deployed, no further automatic service personalisation was supported by the PS. The onus was on the service to manage any further personalisation using the preference request mechanisms of the PS.

This placed a requirement on services to have knowledge of when re-personalisation was necessary. However, this knowledge resided in the user's context-dependent preferences which were not completely available to services. The inability to re-personalise themselves appropriately meant that service personalisation remained *static* from initial application and easily became void as the user's context (and hence preferred behaviours) continued to change. For example, user preferences indicate that an enhanced mode of some news service (costing a fee) is preferred when the user is at work while the standard (free) mode is preferred when the user is at home. If the user starts the news service at work and then travels home, the news service should be automatically re-personalised to the free mode when the user arrives at home. Provision of *dynamic personalisation* was identified as a main requirement for the enhanced PS prototype in DAIDALOS II enabling applied personalisation to change in line with changing environmental states.

Throughout DAIDALOS I a number of stand-alone demonstrations were implemented to show the innovations of the PSP including personalisation. This culminated in a final demonstrator that aimed to show the full potential of the PSP [124]. It was successfully demonstrated in early 2006 and provided a good basis for the design and realisation of an enhanced prototype during the second project phase.

3.3 DAIDALOS II PS Prototype

The shortcomings of the first PS prototype were the main input to the requirements capture for the enhanced prototype in the second project phase. A more dynamic solution was required, yet any improved system still had to provide support for the multiple personalisation tasks both internally within the PSP and externally within third party services. Further, the layer dependency and decision making constraints placed on the DAIDALOS II PSP had to be taken into consideration when designing the DAIDALOS II PS architecture. The key requirements are listed below.

Generic Personalisation

As in DAIDALOS I, it was necessary for the DAIDALOS II PS to support a variety of personalisation tasks throughout the enabling platform and third party services. Where possible the solution was to be generic to allow porting of preferences across resources. For example, if a specific preference doesn’t exist to indicate the volume of an audio service, it should be possible to use an equivalent volume preference related to another service to personalise the former.

However, it may be the case that no preference is available throughout the entire PS to personalise some service. This situation can arise when a new resource becomes available containing personalisation parameters the PS is unfamiliar with. In such situations the PS should support the creation and usage of entirely new preferences related to the previously unknown parameters. This would enable the PS to support new personalisation tasks through the provision of new preferences and hence the need for multiple rule stores in the PSP should become void. Any rules related to the configuration or behaviour of services (both enabling and third party) should be regarded as preferences and stored in a single user preference set.

Dynamic Personalisation

The internal functionality of the PS required complete re-design since the more modular and layered architecture of the DAIDALOS II PS forced the removal of management layer functionality (such as service selection, composition and parameterisation). The new PS was required to adopt a more passive role to personalisation by managing and providing preference information but not explicitly applying it to services. Instead, when a preference outcome was to be applied to a service the outcome was communicated to the service at which point the service itself made the decision as to whether or not to apply the outcome. This was due to the prohibiting of decision making and proactive behaviour within the DAIDALOS II PSP.

However, although a more passive role was adopted in terms of the outcome application strategy, the personalisation of a resource must be up to date to ensure accuracy. As the user moves through their environment their context will change, which may or may not affect how they prefer resources to be configured/behave. In the case where a context change does have an impact on personalisation, mechanisms should be in place to automatically trigger a re-personalisation of resources for the new situation. Such an approach is termed *dynamic personalisation* as the personalisation applied changes dynamically due to changing environmental states. This differs from the static personalisation provided in DAIDALOS I where applied personalisation could not alter in line with the changing environment and therefore remained static from the point of initial application.

The knowledge of when to re-personalise a resource is held in the context-dependent preferences themselves. Mechanisms are required to monitor context values that form the condition parts of the preferences. When a context value changes, all dependent preferences should be re-evaluated (under the new context) and any change in outcome communicated to the appropriate resources for re-personalisation purposes.

Support for Implicit Preference Set Maintenance

Another key requirement was the provision of mechanisms that would support implicit preference set maintenance through time to reflect changing user needs. In DAIDALOS I a preference management GUI allowed the user to view and manipulate their preference set accordingly. In other words, DAIDALOS I provided explicit personalisation. As mentioned in section 2.2.1 this approach gives the user complete control over their preference set mitigating unexplained system behaviour and the ensuing frustration.

However, as also learnt in DAIDALOS I, the frequent updates required, placed a heavy burden on the user and led to unsatisfactory personalisation. Another point to consider is the fact that the context dependent preferences themselves can potentially become large and complex in order to depict all preferred options in a variety of different contexts. The average user may have difficulty understanding or expressing such preferences, or in some cases may not even be aware of preferred behaviours e.g. the user may not realise that he/she always listens to a particular music genre at a certain time of day.

A more pervasive approach to preference set maintenance is to provide mechanisms that will create and maintain the preference set on behalf of the user. In other words, provide support for implicit personalisation where mechanisms such as behaviour monitoring and learning algorithms create and maintain the preference set automatically without the need for user involvement. Indeed, this approach to preference management is in line with ETSI standards for profile management [52]. ETSI standards refer to a mobile environment and assume a dynamic domain where preferences are context dependent and prone to frequent updates.

A system that supports implicit personalisation enables each user to have a complete and up to date preference set with minimal effort. However, a completely automated approach leaves the user out of the loop. With no way to provide any input the user may feel a loss of control. Additionally, lack of user input could lead to inaccurate preferences and possibly detrimental system behaviour. Therefore the optimum solution to providing a dynamic preference set must be a balance between implicit and explicit techniques. In this way, the user’s preference set is maintained with minimal effort but control can be passed to the user when required.

3.3.1 DAIDALOS II PS Architecture

To meet the requirements of implicit personalisation and provide an enhanced PS, an architecture was developed that divided the PS into two functional blocks as shown in Figure 4. The Preference Management subsystem supported all management activities of individual preferences including updating, evaluation and application. A new concept was added to the PS in the form of the Learning Management subsystem. It supported all (previously non-existent) activities concerned with the learning of new preferences including behaviour monitoring and data mining allowing the PS to create new preferences and maintain existing ones on behalf of the user.

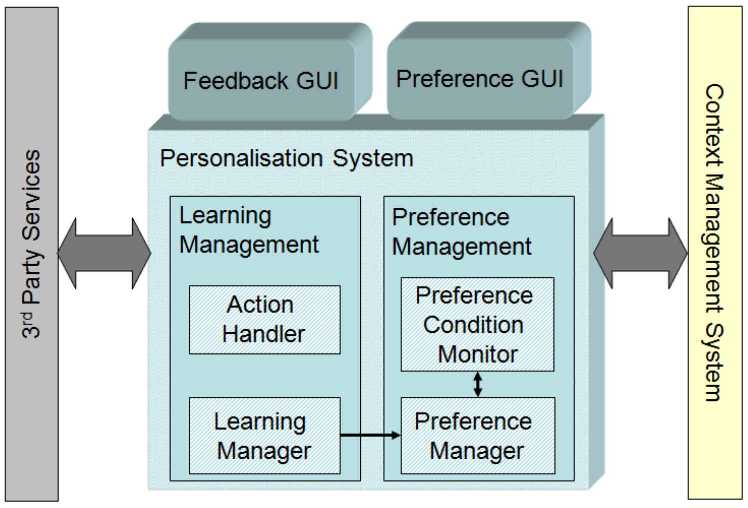


Figure 4. DAIDALOS II Personalisation System Architecture

3.3.2 Preference Management

The DAIDALOS II PS supported all the personalisation tasks provided by the DAIDALOS I PS but support was less explicit than the original prototype. The DAIDALOS II PS took on a more generic architecture where the personalisation processes themselves were moved to external services. These external services would then request preference information from the PS and use this information to personalise their own processes. For example, the external service selection process would request service selection preference information so it could rank discovered services based on user preferences and pick the service that best met the user's needs. This more generic architecture was necessary due to the layer dependency constraint but essentially it allowed the PS to easily support personalisation of other tasks as required. In DAIDALOS II this included support for personalised privacy policy and identity management [119, 120, 121, 122].

Within the Preference Management subsystem the Preference Manager (PM) component acted as the guardian of the individual preferences and as such was the only component in the platform that could access the entire set of preferences or complete preference rules. As in DAIDALOS I, preferences continued to be context dependent rules based on an IF-THEN-ELSE construct. This format was generic to all preferences in the preference set and hence preferences were portable across services (e.g. a volume preference for audio service A could be used to populate an equivalent personalisable parameter in service B). Different preferences existed within the user’s preference set to cover all key personalisation tasks that were previously supported in DAIDALOS I. The main functions of the PM were:

1. Storage/retrieval of preferences (held in the Context Management system database)
2. Evaluation of preferences against current context
3. Handling of service requests for preference information
4. Base functionality for the creation, deletion and updating of preferences as requested either manually by the user (through the Preference GUI) or due to new preferences becoming available from the Learning Management subsystem.

In DAIDALOS II the notion of *stereotypes* was investigated to provide the user with a default set of relevant preferences depending on the user’s role (e.g. ‘Doctor’). This initial set of preferences could then be expanded and refined over time either by the user or by preference learning processes [123, 124]. The use of such default preference sets meant the user was not ‘starting from scratch’ and so experienced some level of personalisation from the moment they began to use the system.

To allow for an expanding preference set it was essential that the PM could handle new preferences for new personalisable parameters. The adopted solution was to capture new attributes when a new resource first requested preference information from the PS. The PS enforced this by requiring each request for preference information to be accompanied by a default outcome. If an appropriate preference already existed, the PM evaluated the preference and returned the outcome as normal but in the situation where the request asked for a nonexistent preference, the PM returned the default outcome (sent in the request) and stored the default as a new preference for future use and refinement.

As well as handling new preferences from new resources, the PM also had to handle new preferences from the Learning Management subsystem. At various intervals the Learning Management subsystem would schedule a preference mining execution and forward the output to the PM for merging with the user’s existing preference set. In this way the user’s preference set remained up to date.

The Preference Condition Monitor (PCM) enabled the second phase PS prototype to provide dynamic personalisation within the DAIDALOS II platform and third party services [131]. It provided the functionality necessary to drive automatic re-personalisation of services at run-time due to changes in the user's context. To achieve this, the PCM held information regarding the currently running services, their related preferences and the context attributes those preferences depended on. This information allowed the PCM to identify what context attributes to monitor for changes. For example, if the following preference belonged to some running service

```
IF <location = work> AND <day = weekday>
        THEN [service = news]
        ELSE [service = sport]
```

then the PCM would monitor the context attributes 'location' and 'day' for changes. When a monitored context attribute changed, cross-referenced lists held internally enabled the PCM to trace what preferences may be affected by the change. The PCM requested a re-evaluation of all such preferences through the PM and communicated any new preferred outcomes to the appropriate services so they could re-personalise themselves.

3.3.3 Learning Management

DAIDALOS II introduced learning mechanisms to the PS to both compliment and alleviate manual preference management processes. These mechanisms supported the creation and maintenance of the user's preference set on behalf of the user with minimal effort on their part. Figure 5 shows the basic flow of the DAIDALOS II preference learning process.

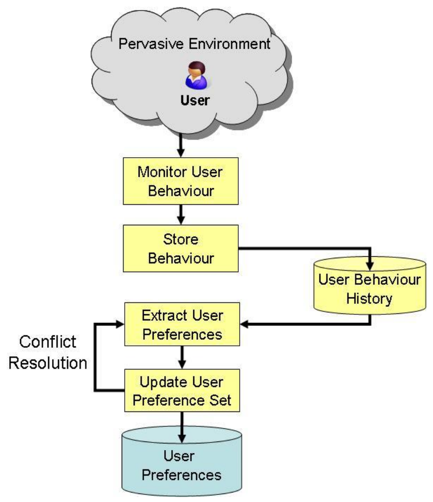


Figure 5. Flow of the Preference Learning process

Monitoring and Storing User Behaviour

As the user moves through their pervasive environment they will inevitably interact with various resources. To build a picture of user behaviour it is essential to capture information on these interactions. Mechanisms must be capable of capturing the appropriate, useful information no matter where the user is or what resource they are interacting with. Important interaction information must be distinguished from noise; for example, it may not be useful to record all keystrokes while the user is typing. DAIDALOS II monitored *user actions* defined as:

user action:	an act performed by the user that changes the personalised state of an entity where an entity may be a service, network or device.

Each entity determined a number of *personalisable parameters* that were populated by *personalisable parameter preferences* during runtime. Therefore the set of personalisable parameters dictated what could be personalised within an entity. When the user changed the value of some personalisable parameter, this changed the personalised state of the entity and therefore was monitored as a user action.

DAIDALOS II implemented passive monitoring through the Action Handler component (Figure 4). This put the onus on the resources to send user actions to the PS when the user performed them. User actions were received by the Action Handler where they were processed and stored in the User Behaviour History within the Context Management database for later use by learning procedures. Since the goal of preference learning was to produce context-dependent preferences an important step of the user action processing was the addition of an appropriate *context snapshot* to each user action. This snapshot described the situation of the user when they performed the user action.

However, as mentioned in section 2.2.2, context can be seen as anything that describes the situation of the user [26]. Such an all-encompassing definition means an entire context snapshot can potentially be huge. It is undesirable to store such a large volume of data with every monitored action. Instead, only the most relevant contextual information should be stored. DAIDALOS II provided a solution by defining a snapshot of set length containing the context attributes that were generally relevant to all actions (e.g. location). This generic snapshot was stored with each user action.

User Behaviour History

The User Behaviour History (UBH) held a complete store of all the action-snapshot pairs that occurred over some time period. The batch learning algorithms then accessed this store at intervals to process the data and extract new preferences. Most conventional approaches to storing monitored user behaviour data use a single storage method; however, the DAIDALOS UBH adopted a dual store approach which acted as a long-term store (LTS) and a short-term store (STS) [132]. Figure 6 illustrates this concept.

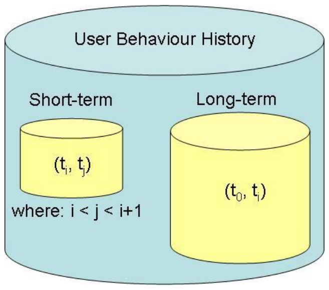


Figure 6. Dual store behaviour history illustrating the separate short-term and long-term stores

At time t_0 when the user began to use the pervasive system, the entire UBH was empty as no actions had yet been performed. As the user interacted with resources, actions were initially stored in the STS while the LTS remained empty. When the first learning cycle occurred at time t_i, the learning algorithms processed the data stored in the STS. At this point the contents of the STS were copied to the LTS and the STS was cleared ready to receive the next monitored actions. This process continued with the cyclic pattern of learning executions and subsequent STS contents being appended to the LTS.

In this way the STS only contained the actions that had occurred since the last execution of the learning algorithm at time t_i until the current time t_j. The LTS contained all actions that had occurred since the user began to use the system at time t_0 until the last execution of the learning algorithm at time t_i.

The benefits of this approach were two-fold. Firstly, since learning algorithms generally only process the contents of the STS, recent new behaviours were identified more readily since they were not inhibited by prominent behaviours from the past. Secondly, the LTS retained the entire back catalogue of user behaviour that could be called upon to aid in the resolution of conflicts between long-established preferences and newly learned preferences. Such conflicts often occurred during the preference merging process [133] when the user's preference set was automatically updated with new preferences returned from the learning process. Where the conflict was unresolvable by conventional merging techniques the learning algorithms could be applied to the combined dataset from the STS and the LTS to create a preference that represented the entire historic behaviour of the user. This preference then replaced the current one in the user's preference set.

Extraction of Preferences and Updating

With a plethora of available learning algorithms it is essential to select one suitable for the task of extracting preferences from monitored user behaviour. Each algorithm has various strengths and weaknesses and some preferred problem domain. DAIDALOS II employed several different learning algorithms for preference learning as well as context inference and reasoning. To accommodate this requirement the Learning Manager (LM) component was designed to support a library of pluggable algorithms as shown in Figure 7.

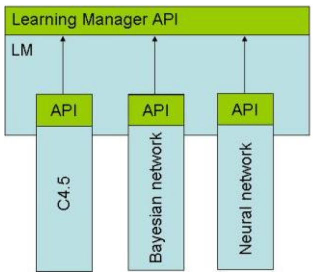


Figure 7. Pluggable library architecture of the Learning Manager

Bayesian networks, neural networks and decision tree learning approaches were implemented and utilised in DAIDALOS II although Quinlan's C4.5 tree building algorithm [87] was chosen as the primary algorithm for preference learning. This well evaluated, benchmark algorithm performed consistently well in the problem domain. The algorithm's use of Gain ratios instead of simple Gain (as in ID3) overcame problems arising from context attributes with multiple values (such as 'time' or 'date').

Further, the decision tree output generated from this algorithm mapped well to the human-readable IF-THEN-ELSE preference format and translation processes between the two formats was straightforward. The decision tree branches represented context conditions and the leaves contained the preference outcomes (i.e. the most popular user actions performed under the context conditions).

The LM component ran its own *learning cycle* thread that took full responsibility for learning algorithm execution. Therefore the PM did not need to explicitly request re-executions of the algorithm. Instead it periodically received preference updates as output from the LM when another execution of the learning algorithm completed. The frequency with which the learning cycle thread triggered executions of the C4.5 algorithm was based on the number of actions, n, received since the last algorithm execution and the time, y, that had elapsed since the last algorithm execution. A new execution was requested if the following condition held true:

$if\,((n > \alpha)OR(y > \beta))$

- where α is the maximum number of actions allowed between each execution
- where β is the maximum time interval allowed between each execution

The frequency of learning algorithm executions could be controlled by manipulating the variables α and β. When a learning algorithm execution completed, the output (i.e. list of new learned preferences) was passed to the PM. The PM then merged the new preferences with the user's existing preference set [133].

3.3.4 User Input and Control

The negative effects of denying the user input to the personalisation and learning processes include a loss of control, confusion at system behaviour and ultimately system rejection. Therefore user interaction is an essential part of the implicit personalisation process. In DAIDALOS II two GUIs were provided by the PS for user interaction purposes. The Preference Management subsystem provided the Preference GUI allowing the user to view their preference set and manually alter preferences any time they required. The Learning Management subsystem provided the Feedback GUI allowing the user to provide input during the application of preferences. This input would feed back into learning processes allowing the further refinement of the user's preference set.

When the system resolved that some preferred behaviour (preference outcome) should be implemented, system uncertainty in the proposal dictated whether the feedback GUI would be presented to the user and in what format it would appear. In the case where system uncertainty was below some threshold (i.e. the system was certain that this behaviour was correct) the proposed behaviour would be implemented automatically without consulting the user.

In the case where system uncertainty was above some threshold (i.e. the system was uncertain that this behaviour was correct) the feedback GUI could be used in explicit mode which required a specific authorisation from the user before the behaviour would be implemented. In this case the user would be prompted with an 'OK/Cancel' option requiring the user to explicitly input their decision.

In the case where system uncertainty was between upper and lower thresholds (i.e. the system was indifferent about this behaviour) the feedback GUI could be used in implicit mode where user input was only required if the user rejected the proposal. In this case the user would be prompted, for some timeout period, that the behaviour would occur. If the user did not push the 'cancel' button within the timeout period the system inferred that the user agreed with the proposal (i.e. positive feedback was gained implicitly) and the proposal was implemented.

In each case the user's response was captured by behaviour monitoring mechanisms and fed back into the UBH and learning processes. This enabled further refinement of the user's preference set as well as the creation and maintenance of negative preferences indicating what the user doesn't want in a given context.

3.4 DAIDALOS II PS Evaluation

An evaluation of the DAIDALOS II PS was carried out to capture performance statistics. An evaluation framework was implemented to test both subsystems of the PS in terms of how accurately user preferences were learnt and applied. Figure 8 shows the evaluation framework.

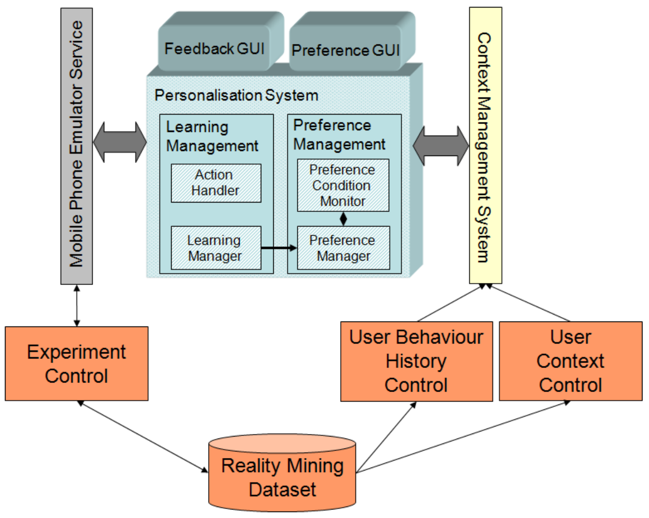


Figure 8. Evaluation Framework for the DAIDALOS II PS

The Mobile Phone Emulator Service (MPES) acted as a third party telecommunications service running on top of the DAIDALOS II platform. It contained several personalisable parameters which the Learning Management subsystem would learn preferences for and the Personalisation Management subsystem would populate dynamically. To simulate service usage, subsets of the Reality Mining Dataset [134] were used to represent historic user behaviour. The Reality Mining Dataset (created by the MIT Media Lab) contained mobile phone usage data for over 100 users across a period of 350,000 hours.

3.4.1 Evaluation Tests

During the test initialisation period a subset of usage data from one user was divided into test and training data and the training data was uploaded as user behaviour history to the Context Management System by the User Behaviour History Control. Once uploaded, the Experiment Control triggered the LM to perform a learning execution. This resulted in a set of preferences being forwarded to the PM where they were then merged into the user's preference set.

To test the accuracy of preference learning and the dynamic application of preferences, the User Context Control then used the test data to drive further context changes. After each context change the Experiment Control recorded how the MPES was dynamically re-personalised by the PCM (based on the learned preferences) and compared this with the actual behaviour in the test data set to calculate an overall percentage accuracy of personalisation for the test. This process was repeated ten times for each user using different test and training data subsets across three users. The test results [135] showed that the PS performed well with an average accuracy of 84% across all tests.

3.5 Lessons Learnt

3.5.1 Generic PS Architecture

The DAIDALOS II PS provided many enhancements to the functionality of the PS from DAIDALOS I. A significant improvement was the replacement of the task specific PS from DAIDALOS I with a more generic PS in DAIDALOS II. The role of the PS changed from personalisation task management to that of support for personalisation tasks through the management of task specific preferences. The IF-THEN-ELSE preference format from DAIDALOS I was adopted again in DAIDALOS II. All preferences, regardless of the task they related to, followed this format enabling all preferences in both DAIDALOS I and DAIDALOS II to be managed by a common mechanism. In DAIDALOS II it also enabled preferences to be ported across services, where a preference for one service could be applied for an equivalent task in another service.

Provision for dynamic application of preferences led to more up to date and accurate personalisation. Profile management went a step beyond current standards providing monitoring and learning mechanisms to support the maintenance of a frequently changing preference set that could expand and alter through time in line with user needs. This mitigated the user’s role in such laborious and time consuming tasks; however, when user control was required several GUIs allowed the user to manually view and manipulate their preference set as well as provide input to the preference application process.

Although the main requirements were fulfilled, the DAIDALOS II PS did not reach its full potential due to the top down constraints that were applied to all enabling services within the PSP. The layer dependency restriction rendered it difficult for the PS to perform necessary tasks to allow for the personalisation of enabling services residing in the management layer. Since these lower layer enabling services could not be dependent on the PS, tasks such as monitoring user behaviour, requesting user feedback and communicating preference outcomes became complex and non-standard work-arounds were necessary. Since management layer services could not send monitored actions directly to the Action Handler component, eventing was used as an alternative solution to capture user behaviour in such services. Further, management layer services could not receive new preference outcomes from the PM as this involved implementing a PM call-back method. Therefore, the PM was forced to directly apply preference outcomes to enabling services violating the decision making and proactive behaviour constraint. This restricted environment led to restricted personalisation of lower layer enabling services.

3.5.2 Negative Preferences

In DAIDALOS II environment monitoring was triggered by the occurrence of a user action. When a user performed a user action, a context snapshot was taken and stored with the action in the UBH. However, it became apparent that by only monitoring user behaviour when some user action occurred, the system became biased towards positive behaviour. In other words the system could learn positive preferences based on action occurrences, i.e. what the user *did* prefer, but could not learn negative preferences based on the non-occurrence of actions, i.e. what the user *did not* prefer.

Consider the following example. A user walks into a room twelve times. On three of the twelve entries the user turns on the light but on nine of the twelve entries the user doesn't turn on the light. The DAIDALOS II learning system would only be able to incorrectly learn that the user prefers to turn on the light when they enter the room as the user behaviour history set (of action-snapshot pairs) only contained data related to action occurrences (due to the user action-triggered environment monitoring policy). Of course the user could correct this erroneous system behaviour through feedback mechanisms but incorrect personalisation would be experienced first.

3.5.3 Pre-actions

When detailing scenarios for the DAIDALOS II demonstrations the issue of *pre-actions* was raised. A pre-action is an action performed by the user in a previous context to prepare for entrance into a new context. For example, if the user is entering a lecture theatre, they may mute their mobile phone in the corridor outside before they enter the lecture theatre. Equally, when leaving the lecture theatre, the user may un-mute their mobile phone just before they leave the lecture theatre. This issue is most specific to the problem domain of preference learning in pervasive environments where user behaviour can be pre-emptive, introducing consistent noise into the dataset.

The DAIDALOS II system would capture the user actions and store them with a snapshot of the current context. Hence the system would learn a preference stating that the user mutes their phone in the corridor and un-mutes their phone in the lecture theatre, even though the user actually prefers the reverse. As above the only solution provided by DAIDALOS II involved preference refinement from user feedback once incorrect personalisation had been experienced.

3.5.4 Preference Format

In DAIDALOS II preferences were stored internally as IF-THEN-ELSE rules. This decision followed from DAIDALOS I where the IF-THEN-ELSE internal format was sufficient since the primary goal was to display user preferences in a human-understandable format so that preference manipulations could be performed manually by the user. However, as complex processes such as preference learning and preference merging were added to the PS in DAIDALOS II, implicitly managing the preference set on behalf of the user became increasingly non-trivial due to the complex internal format of preferences. A more efficient solution would be to store preferences internally as a tree or network structure that can be more efficiently manipulated and processed. The internal format could then be translated to an IF-THEN-ELSE rule when the user wishes to view it.

3.5.5 Preference Learning Algorithm

The use of the C4.5 decision tree learning algorithm proved successful in the preference learning domain. It provided an accurate preference set and the non-complex output format allowed users to view and update their learned preferences manually. However, the slow time constant associated with batch algorithms such as C4.5 often meant that the user's preference set was not completely up to date. When the user presented a new behaviour, the learning system could not provide a rapid response to quickly update the preference set accordingly. This was due to the batch nature of the C4.5 algorithm which was only executed at various intervals. If a new behaviour presented itself just after a learning execution, it would not be accommodated into the user's preference set until the next learning execution which may not be due for some time. Therefore, in between learning executions the accuracy of the user's preference set could deteriorate. Equally, at the point of initial system usage no monitored behaviour was processed and no learnt preferences were available until the first learning execution.

This issue is also faced by other pervasive projects, all utilising batch algorithms for preference learning. For this reason similar projects such as SPICE, Mobilife and Ubisec implement rapid response mechanisms that accommodate user feedback into the user's preference set immediately when it is received. This provides a partial solution but does not address rapid response to all monitored (non-feedback) user actions.

As mentioned in Section 2.3.6 user modelling (of which preference learning is a subset) is essentially an incremental task and as such can be most efficiently handled by an incremental learning algorithm. With such an algorithm all new information would be processed immediately as it is received. The behaviour history store would become redundant and updating of the user's preference set would not be restricted to cyclic executions. Equally, learnt preference information from monitored behaviour would be available from initial service usage without the need for the lag time required by batch algorithms to acquire an initial user behaviour history. However, such an incremental approach must consider several key issues. How should new input be rapidly incorporated into the entire information store and what if the new input conflicts with what already exists in the information store?

In the DAIDALOS II PS conflicts are handled during the preference merging process when new learnt preferences are merged into the user's existing preference set. If a preference has been learnt based on a new behaviour it may be in direct conflict with a preference based on a past behaviour. Where no alternate solution is possible the PS executes learning on the entire store of user behaviour history data to generate a compromise preference that takes account of both past and recent user behaviour. This solution is not transferrable to an incremental approach where no history of user behaviour is stored. An alternative solution is required to deal with the emergence of new conflicting behaviours in an incremental system.

3.6 Summary

The DAIDALOS project made a significant contribution to pervasive personalisation research. Initial prototypes produced during the first phase of the project provided a good base for further development and innovation in the second project phase. The initial PS provided various personalised processes such as personalised service selection, personalised composition and personalised service parameterisation. Although successfully demonstrated at the end of the first project phase, key requirements were identified for an enhanced PS in the second project phase. These included a more flexible PS architecture, dynamic personalisation (i.e. re-personalisation during service runtime) and implicit personalisation involving monitoring and learning techniques.

The second PS delivered all enhancements required and an evaluation of the entire DAIDALOS II PS demonstrated its ability to implicitly create and manage an accurate set of preferences on behalf of the user that would drive dynamic personalisation. Although successful, the DAIDALOS II PS has also raised some important lessons learnt for consideration if developing a future personalisation system. It is highlighted above how the layered architecture of the DAIDALOS II middleware placed several restrictions on PS functionality. Further, the addition of implicit personalisation functionalities such as behaviour monitoring and machine learning has also highlighted several areas for improvement in terms of how and when user behaviour is monitored, how internal data should be stored and processed and what learning approaches may be better suited to the task of preference learning in a pervasive environment.

In Chapter 4 the findings presented in this chapter are combined with those presented in Chapter 2 to present a set of design features and requirements for an efficient preference learning technique that is specifically tailored to personalisation in the pervasive computing domain.

4 Design Issues and Requirements for Preference Learning in a Pervasive Environment

4.1 Problem Description

This thesis attempts to answer the following research question:

How can a system learn and provide ***accurate*** *and* ***up to date*** *preferences for personalisation in a pervasive environment?*

If we consider this statement in more detail, it raises several obvious issues such as the selection of an appropriate learning algorithm and determining an appropriate method of environment monitoring as input to learning processes. However, it also contains several less obvious issues that must equally be considered. Notably, implicit personalisation is applied in a user-centric domain and hence the user should have ultimate control. Therefore he/she should be able to view all preferences and manipulate them as required. This raises further questions as to how preferences should be represented internally within the system, externally to the user and how translations should occur between the two formats.

Section 2.2.2 outlined various pervasive projects that have attempted to address these challenges. Chapter 3 described how the DAIDALOS II PS provided support for implicit personalisation. The final evaluations of the Dadialos II PS produced satisfactory results and indeed many other pervasive projects (detailed in section 2.2.2) have enjoyed similar success. However, reflecting on the shortcomings of past projects and the lessons learnt from DAIDALOS II, it seems there are several key areas for improvement to better provide both accurate and up to date preferences for implicit personalisation. The following sections discuss various design issues related to the problem description above and determine key requirements for an efficient preference learning solution.

4.2 What to Monitor

To learn preferences for personalisation in a pervasive environment it is necessary to monitor user behaviour in context situations. However, as identified in DAIDALOS II it is important to outline the scope of terms such as user behaviour and context since they are potentially all encompassing.

As outlined in section 3.3.3 not all actions performed by the user are useful for preference learning. Behaviour monitoring in DAIDALOS II was bounded by the set of personalisable parameters identified by each service, network and device. Monitored user actions only included those actions performed by the user that altered the value of some personalisable parameter, hence altering the personalised state of the entity. The same boundaries for behaviour monitoring will be adopted here as a design decision.

User context is provided from environmental sensors and other sources. It is difficult to determine a priori, what context data will be useful for preference learning since user actions can potentially be dependent on all user context. Therefore, unlike user behaviour, it could prove detrimental to pre-determine a bounded subset of user context for monitoring. DAIDALOS II and many other implicit personalisation systems were forced to bound monitored context to a finite static set due to the storage and processing limitations resulting from the use of batch learning algorithms (and the required behaviour history stores). An optimum approach would monitor all context for consideration during preference learning; therefore the design decision is to aim towards the monitoring of all available context in an efficient manner.

4.3 Batch vs. Incremental Learning

A significant area for improvement in providing support for implicit personalisation involves the type of learning algorithm employed for the preference learning task. Although the most common approach is to employ a batch learning algorithm it has been highlighted throughout Chapter 2 that batch algorithms have several drawbacks when applied to the task of preference learning in a pervasive environment. This is also echoed in the lessons learnt from the DAIDALOS II project where the batch C4.5 algorithm was utilised for preference learning.

If we refer to the problem description above, the aim is to learn preferences that are both accurate and up to date. Evaluation of the DAIDALOS II PS showed that batch algorithms are capable of learning accurate preferences based on user history data; however, it has also been highlighted numerous times that batch algorithms cannot ensure up to date preferences due to their cyclic execution. This issue is most acute when the user changes their behaviour between cycles warranting an update to their existing preferences. A batch algorithm alone will not provide updates to the preference set until the next learning cycle execution and until that time incorrect personalisation can be manifested.

Indeed, first-hand experience of the design and implementation of the DAIDALOS II Learning Management subsystem highlighted the need for additional supporting functionalities besides the main learning algorithm to ensure a more up to date preference set. In DAIDALOS II the dual-store user history database took account of recency to more rapidly identify new behaviours in behaviour history data while in projects such as Ubisec, SPICE and MobiLife, rapid updating mechanisms (based on user feedback) immediately altered the user's preference set between learning cycles.

Giraud-Carrier [93] proposes that user modelling (of which preference learning is a subset) is essentially an incremental task, best handled by an incremental learning algorithm. Inputs will naturally occur one at a time through time and input vectors should not be static. This is reflected in the DAIDALOS II lessons learnt where one important lesson hints towards the benefits that a more incremental learning solution could provide. Not only would an incremental preference learning algorithm provide a more responsive system to changes in user behaviour but it would also remove the need for large stores of user history data and remove the need for lag periods when these stores are populated with monitored data between learning cycles.

Indeed the typical *open-world* assumption of most incremental approaches maps naturally to the real world problem domain of a pervasive environment. As users move through their environment, interacting with services, networks and devices, context and behaviour updates will occur sequentially, through time. Several incremental algorithms discussed in section 2.3.6 naturally handle input in such a continuous fashion. Additionally, several incremental algorithms can also grow/shrink their topology to accommodate changing classes and concepts such as new forms of context information and new user actions.

However, Giraud-Carrier highlights a key design issue of incremental algorithms. Essentially such algorithms follow a learning curve. Starting with very little information, an incremental algorithm may initially provide inaccurate output. Accuracy will improve over time as input increases but it may be difficult to determine when the algorithm has received sufficient input to be trusted. This could have serious consequences in terms of addressing the problem description where learnt preferences should be both up to date and accurate.

Indeed, there is some trade-off between an up to date preference set and an accurate one. However, as Webb [136] outlines, often in a user modelling domain, predictive accuracy comes second to various other factors such as CPU time. He also highlights that appropriate prompting can greatly improve the user experience where predictive accuracy is lower. These observations were also noted in the Swiftfile system where final analysis showed that incremental learning out-performed batch learning of user behaviours. Essentially, the great benefit of incremental algorithms is their ability to provide some kind of preference information from initial system usage. Even if initial output is not highly accurate, it is better than nothing and when used in conjunction with prompting mechanisms, can satisfy user expectations.

Therefore, the outcome of this fundamental design decision is to adopt an incremental approach to preference learning. The next step is to determine the requirements that an incremental algorithm must fulfil within this problem domain.

4.4 Incremental Algorithm Requirements

As detailed in section 2.3.6 the term 'incremental' is interpreted in several different ways when referring to machine learning algorithms. Current incremental algorithms vary in terms of what incremental properties they possess and hence what distinguishes them as incremental. Therefore it is necessary to consider what are the key properties of an incremental algorithm for the purposes of preference learning in a pervasive domain.

4.4.1 Incremental Properties

For preference learning in a pervasive environment, it is desirable that an incremental algorithm would possess the following properties.

1. **Process input one at a time over time**. The incremental algorithm should receive and process inputs as they occur in real-time rather than rely on an a priori training set to be presented. With regard to preference learning in a pervasive environment the inputs will be user actions and user context. User context will change as the user moves through their environment and user actions will occur as the user interacts with services. The incremental learning algorithm should process such events immediately when they occur.
2. **No re-processing of past data**. The incremental algorithm should not need to re-process past data to update its internal knowledge store or to provide output. Instead the algorithm should create a new hypothesis *h* for each new example *e* where h_{i+1} depends only on h_i and the current example e_i. Equally, at any time the algorithm should be capable of providing a best approximation, h_i, of the target so far when queried.
3. **No a priori knowledge of the problem domain**. In a highly dynamic pervasive environment an incremental algorithm should not rely on preset knowledge such as learning rates as changes in the pervasive environment could easily render such static, domain specific configurations invalid.
4. **Growing/shrinking topology**. In a pervasive environment new context sensors can become available or the user may enter a previously unknown situation (e.g. if they go on holiday). In either case new context attributes or new values of existing context attributes should be considered. Equally, the user may perform new user actions over time as new resources become available. Therefore new user actions should be considered. An incremental algorithm should be able to accommodate such changes by altering internal knowledge structures. In effect it should be an open-world system, unbounded by static input and output vectors.

It is noted that several incremental algorithms (ART, COBWEB and ID4) reviewed in section 2.3.6 possess all four properties. Considering past projects and background literature discussed in section 2.3.6, none of these algorithms have been employed for implicit personalisation in a pervasive system to date. One could question if any of these existing incremental algorithms are sufficient for preference learning in a pervasive domain. However, a suitable algorithm must also satisfy the learning properties below (Section 4.4.2) and support all the design decisions outlined in this chapter.

4.4.2 Learning Properties

As well as incremental properties, the algorithm should implement several fundamental learning properties due to the pervasive problem domain.

1. **Hetero-associative mapping mechanism**. To create context-dependent preferences the internal representation is essentially a mapping between context data and preference data. In learning terms this translates to a hetero-associative mapping since the input is very different to the mapped output. The preference learning algorithm must be able to handle the two heterogeneous data types and the hetero-associative mappings between them.
2. **Unsupervised learning**. Most learning methods can be classified into two categories: *supervised learning* and *unsupervised learning*. While supervised learning relies on an external teacher and/or global information, unsupervised learning relies only on local information to self-organise presented data and identify emergent properties. In a pervasive environment the preference learning algorithm will only be presented with context and preference data which it must self-organise to identify correlations between the two.

4.5 Internal Knowledge Representation

In the DAIDALOS II pervasive platform, preferences were stored in a human-readable form in the PS. This proved to be inefficient for a number of reasons. Firstly, each time new learnt preferences were delivered, the decision tree output from the C4.5 learning algorithm had to be translated into the IF-THEN-ELSE preference format and merged with the existing preference set. Both these processes were potentially time intensive depending on the volume of output from the C4.5 algorithm. Secondly, manipulating and evaluating preferences in such a human-readable format proved non-trivial and computationally expensive.

A more efficient solution would be to hold the user preferences in some internal structure for quick manipulation rather than converting them to an external format for storing. Several personalisation systems considered in section 2.2.2 use Bayesian networks or neural networks to represent preferences internally. However, such networks can themselves become complex systems that are difficult for humans to interpret or understand. For example, most neural networks contain a hidden layer to enable them to solve non-linear problems (such as XOR). Although this makes for a more powerful network it becomes very difficult to understand why the network settles on a target function and even more difficult to extract this information into a human-understandable format.

Therefore, although neural and Bayesian networks allow for more efficient manipulation and evaluation of the preferences they represent it should not be forgotten that they are employed in a user centric domain and therefore the user should be able to take ultimate control when required. To ensure this, it must be possible to display all preferences to the user in a human-understandable format. Additionally, the user should be able to manipulate preferences in this human-understandable format and such changes should be transferrable back to internal preference representations.

One solution is to represent context-dependent preferences as a linear neural network with weighted connections between context data and preference data. With no hidden layers, it is much easier to explain internal network knowledge and in turn translate the linear connections into human-understandable rules. Consider a typical context dependent IF-THEN-ELSE preference. The context conditions consist of tuples of the form:

<context parameter, logical operator, context parameter value>

It is changes in the *context parameter value* that affects the outcome of the preference. For example, if we have the following service selection preference:

```
IF <location = home>
THEN [service_type = sport]
ELSE IF <location = work>
THEN [service_type = news]
```

we can see that the context parameter values ‘home’ and ‘work’ determine whether the implemented preference outcome is ‘sport’ or ‘news’. Therefore, individual context parameter values influence the implementation of individual preference outcomes. We can represent this as a linear network with weighted connections between the two heterogeneous vectors (context parameter values and preference outcomes) as shown in Figure 9.

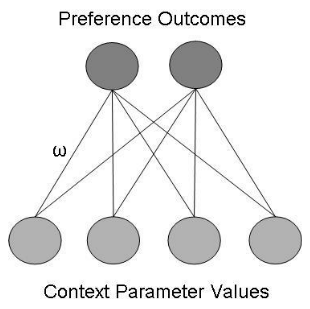


Figure 9. Linear network representing associations between context parameter values and preference outcomes

The weighted connections between layers represent the strength of the association between some context attribute value and some preference outcome. Altering such weights would alter internal network knowledge and hence the preferences represented by the network. Weight manipulations could be performed rapidly in real-time in line with the rapid response behaviour required of an incremental solution. The preferred outcomes in some context could be determined at any point by summing the weighted inputs of the preference outcome nodes and applying some activation function.

4.5.1 Dynamic Network Topology

As well as being easier to interpret and translate into a human-understandable form, implementing an incremental algorithm as a single layer neural network also makes it easier to manipulate the network topology to reflect new classes. A pervasive environment is a very dynamic space. The array of available resources continuously changes as the user moves through the environment. For example, location based services may become available or unavailable as the user moves in and out of range. As the available services change, so do the possible user actions that the user can perform in the environment. Similarly context sensors may be active or inactive through time impacting on the available context information. Therefore, any system that attempts to correlate context and user behaviour must be able to cope with the dynamic structure of these vectors for association.

Many neural systems rely on a static architecture which is defined before learning begins. The input and output vectors cannot change during the course of learning. If any changes are required the network architecture must be re-defined and learning repeated. The inhibiting factors for a dynamic topology are typically the existence of hidden layers (adding internal complexity) and the use of a global network error, hence changes to the network topology will affect the accuracy of the entire network.

However, in a linear network with no hidden layers, it becomes much easier to dynamically add/remove network nodes, especially if a local learning rule is implemented to manipulate weights on individual connections. Since there is no global network error the addition or removal of nodes will not affect entire network accuracy. Therefore as regards the internal preference format, the design decision taken is that the incremental preference learning algorithm will be implemented as a single layer neural network with a dynamic topology.

4.6 Learning Positive and Negative Preferences

Another important lesson learnt from DAIDALOS II was in relation to the inability to learn negative preferences from the outset, i.e. what the user *does not* prefer. This was due to the action-triggered monitoring process that only captured the current state of the environment when the user performed some user action. When a user action occurred, a snapshot of context was taken, appended to the user action and stored as an example in the user behaviour history store. Useful information regarding the non-occurrence of user actions in context situations was therefore omitted and hence the behaviour history dataset was biased towards positive action occurrences. Therefore the system could only learn positive preferences indicating what the user *does* prefer. Of course negative preferences could be introduced to the preference set based on negative user feedback due to incorrect personalisation but this required the user to experience and correct erroneous system behaviour.

A better approach is to continuously monitor all user actions and all context changes so both the occurrence and non-occurrence of actions can be identified in some context. If we consider the single-layer network structure proposed in Figure 9, the vectors for association (context attribute values and preference outcomes) could retain the current state of user context and implemented preference outcomes. When a context update occurs or when the user performs an action, this new input could be quickly accommodated into the network by altering the appropriate vector nodes and updating related network weights. By implementing both incrementing and decrementing processes on network weights, positive and negative associations could be identified between context and preference vectors. Hence, the incremental neural network for preference learning would be able to learn both positive and negative preferences from the outset without relying on negative user feedback.

4.7 Overcoming Pre-Actions

Another issue raised in DAIDALOS II concerns pre-actions, i.e. actions that are performed in a previous context to prepare for entry into another context. The example given detailed how a user may mute their mobile phone before entering a lecture theatre and un-mute their mobile phone before leaving the lecture theatre. A typical batch system would learn the reverse of what the user actually preferred since the monitoring process would store the context outside the lecture theatre with the mute action and the context inside the lecture theatre with the un-mute action.

Equally, an incremental algorithm could provide incorrect output if faced with such a situation. For example, with an incremental neural network it is also typical that network updates are input triggered. Therefore, when the user mutes their phone, the network vectors and weights would be adjusted to incorrectly associate the mute action with the current context outside the lecture theatre.

One may question if this issue could be handled with additional sensing and inference. With appropriate sensing and inference techniques the system could predict the user's future location (lecture theatre) for association with behaviours (muting the phone). However, even with additional sensing and future context prediction it may still be the case that the user is performing actions to prepare for entry into contexts that are more than one step ahead. Therefore it does not always solve the issue and could still result in the incorrect association of context and behaviours.

To overcome this issue an alternative solution is proposed that considers the reinforcement policy of the incremental algorithm. The reinforcement policy dictates *when* an algorithm updates its internal knowledge representation during learning processes. It is clear an input-based policy is not adequate in all circumstances, therefore another reinforcement policy must be considered.

4.7.1 Temporal Reinforcement Policy

If we reconsider the lecture theatre scenario, the user mutes their mobile phone outside the lecture theatre and then immediately enters the lecture theatre. Therefore, the 'mute' state prevails for only a short time period in the context outside the lecture theatre, but prevails for a much longer time period in the context inside the lecture theatre. The temporal duration of the co-occurring context and preference states provides important information that naturally leads one to conclude that the mute state is more strongly associated to the context inside the lecture theatre where it prevailed for a greater temporal duration.

Consider this proposal from another angle. In some context the user performs an action that sets their preferred screen background colour to blue. This state prevails for several minutes before the user performs another action to set their preferred screen background colour to yellow. This state prevails for a number of weeks. Note that the two actions only occur once in some context. With an input-based reinforcement policy both actions would be equally associated to the context state. This contrasts with the natural assumption that the second action is more strongly associated to the context due to its longer duration. Again, the temporal duration of the co-occurring context and preference states provides valuable information.

Therefore, it is proposed that the incremental preference learning algorithm will implement a *temporal reinforcement policy* to take into account temporal information regarding environmental states. Rather than manipulating weights only once when input is received, weights will be continuously manipulated through time based on current environmental states. This will enable the incremental network to learn associations based on the duration of co-occurring vector states, rather than simply on the fact that they co-occurred at one point in time. Vector states that co-occur for longer time periods will be more strongly associated than vector states that co-occur for shorter periods of time. This design decision will overcome the issue of pre-actions identified in DAIDALOS II.

4.8 Concept Drift and Conflict Resolution

Concept drift is highlighted as a major learning algorithm design issue in a number of research papers. It describes the situation where the target function and its statistical properties change through time rendering a prediction model created from past examples, inconsistent with new examples. Webb [136] outlines the challenges of concept drift from the perspective of user modelling (including preference learning). When attempting to model user behaviour, concept drift will be an inevitable issue due to changing user behaviours. Giraud-Carrier also highlights concept drift as a major design issue from the perspective of incremental learning algorithms where new inputs are continuous and ongoing.

Incremental learning algorithms have the ability to adapt quickly to concept drift in a pervasive domain as user behaviour changes but a drifting concept renders some underlying conflict between what used to be true and what is now actually true. In machine learning terms, there is some error between the algorithm's internal knowledge representation and the real world. Reducing this error essentially over-rides old information with the new conflicting information so that the learning system can adapt to new input. Batch algorithms typically minimise such errors during several epochs of training data, aided by global knowledge of the entire dataset; however, an incremental algorithm must determine how to minimise error in one instance based only on current knowledge. In other words, when a conflict arises incremental algorithms must consider how to over-ride old information with new information based on minimal global knowledge.

The issue of incremental conflict resolution has triggered much debate and received significant consideration during the course of this research. Due to the intended user centric application domain, a psychological element is introduced into the equation. It is impossible not to consider this issue from a user's perspective, questioning how a user would expect a learning system to behave in terms of accommodating changes in behaviour. For example, if a user has always preferred to use the BBC News website, what does it mean when they suddenly start to use MSN News? How quickly would the user expect the learning system to pick up this new behaviour, if at all?

Essentially, it comes down to the intended duration of the behaviour change. In the example above, if the user intends to use MSN News from now on, then the user would most likely want the learning system to accommodate the behaviour change rapidly. Equally, if the user intends to use MSN News for one session, the user would most likely want the learning system to accommodate the behaviour change slowly, so their prevailing preference for BBC News is not over-ridden.

Unfortunately attempting to determine if new conflicting input is intended for a long-term or short-term duration is non-trivial. Prompting the user for clarification is an option; however, it is undesirable to request user involvement every time they change a preferred outcome. Instead it is proposed that the temporal nature of a past behaviour can be used to infer the likely temporal nature of a new behaviour. In preference terms this is captured in two heuristics for conflict resolution:

***Heuristic 1*:** A change to a long-term preferred outcome is more likely to be a long-term change and therefore should be accommodated rapidly.

***Heuristic 2*:** A change to a short-term preferred outcome is more likely to be a short-term change and therefore should be accommodated less rapidly.

Considering these heuristics from an end user perspective they assume that a long-term preferred outcome is akin to a deeply held belief. Changes to such an outcome are likely to be the result of much contemplation and hence less likely to revert or change frequently. Equally, the heuristics assume that a short-term preferred outcome is akin to a less deeply held belief. Changes to such an outcome are likely to be more whimsical and hence more likely to revert or change frequently.

Of course, the counter of these heuristics can also be argued. There will always be situations where the heuristics hold and equally situations where they do not. This is due to the complex nature of both human behaviour and pervasive environments. There are many reasons why the user may change their preferred outcome in some context. It may be due to a context change of which the system has no knowledge, a change in lifestyle or simply a change of mind. Equally there are many reasons why the user may allow a preferred outcome to prevail for a substantial time period, other than because it is strongly preferred. Perhaps the outcome personalises some insignificant parameter or perhaps it is difficult for the user to change this preferred outcome. The key phrase in both heuristics is 'most likely'. It was felt that an investigation into psychological literature was worthwhile in an attempt to uncover support for such heuristics at a human behavioural level.

4.8.1 Psychological Investigations

Psychological literature on human behaviour tends to regard behaviour as something we learn and therefore investigates and explains human behaviour in terms of learning and memory. In this sense behaviour change is often viewed as changes in memory due to cognitive processes such as learning and forgetting. Temporal aspects are only considered in terms of the time taken to learn or forget a behaviour.

In human memory over-riding of old memories with new memories can be described by various learning and forgetting phenomena, one of which is retro-active interference [137]. It is based on the principle that if a subject learns the association A-B, later learning of the association A-C will weaken the recall of B. This is essentially what should happen when a conflict arises. The new conflicting preferred outcome should over-ride the old preferred outcome. Retro-active interference also states that the stronger the association between A and B, the longer it will take to forget. This makes sense when considering learning and forgetting as mechanical processes for adapting memories. However, if we reflect on the two heuristics above and their proposals to adapt internal knowledge at different rates, it is clear that the mechanical processes described by retro-active interference do not consider how higher-level variables may affect the rate of learning and forgetting.

One interesting memory concept does consider how external variables influence memory and recall. *Levels of processing* [138] identifies that some memories are stored and recalled more efficiently than others due to the way in which the memories are processed. The theory states that memories processed in a written or spoken format are *shallow* memories and will be quickly forgotten or over-ridden by new information. Conversely memories processed in a semantic way are *deep* memories, not easily forgotten or over-ridden by new information. Reflecting on the heuristics short-term behaviour changes are essentially shallow memories, with little impact on internal knowledge, whereas long-term behaviour changes are essentially deep memories with a bigger impact on internal knowledge.

Unfortunately, the processing concept that defines a memory as deep or shallow cannot be mapped to the preference learning problem domain. Preference learning inputs will all come from similar sources (context management system and services). Of course one could argue that some services may be more critical than others (e.g. a service managing the disclosure of your personal details) and hence new behaviours related to these services should be accommodated more rapidly. This could prove a useful distinction but it also abstracts the conflict resolution issue to a higher level and binds incremental conflict resolution processes to service attributes.

Although investigation of the psychology literature uncovered some interesting cognitive concepts it provided little support for the heuristics but equally it did not invalidate them. Therefore focus shifted towards more domain specific aspects. The temporal nature of behaviours was further investigated in terms of *preference time* constants which give an insight into the past temporal trends of a preference.

4.8.2 Preference Time Constants

The time constant of a preference relates to how frequently the preferred outcome changes in some context. Some preferences will have rarely changing preferred outcomes (e.g. font size preference) whereas other preferences will have frequently changing preferred outcomes (e.g. volume preference). In a single layer network the time constant of a preference can be identified by comparing the association strengths of the preference's outcomes with the current context.

If all the outcomes of a preference have similar connection strengths this indicates that the preference frequently fluctuates between preferred outcomes (i.e. the user is indifferent about their preferred outcome). We can say the preference has a short time constant. Alternatively if a preference has one outcome with a significantly higher connection strength than all the others, this indicates that the preference rarely fluctuates between preferred outcomes (i.e. the user has an obvious preferred outcome). We can say the preference has a long time constant.

Essentially, the time constant of a preference is a record of its temporal history and hence it is reasonable to base future predictions of a preference's time constant on past trends. Therefore one could assume that a preference with a short time constant will most likely continue to have a short time constant in the future while a preference with a long time constant will most likely continue to have a long time constant in the future. In other words, a change to a preference with a short time constant will most likely be for the short-term while a change to a preference with a long time constant will most likely be for the long-term. This is in line with the conflict resolution heuristics detailed above.

Consider this further in terms of network error reduction. At the time of conflict the incremental neural network will hold the connection strength of the old preferred outcome to the current context and the connection strength of the new preferred outcome to the current context. The error is the difference between them with the new preferred outcome initially always having a lower connection strength than the old preferred outcome. Minimising such error over-writes the old information with the new information until eventually the new preferred outcome is more strongly associated with the current context.

If a conflict arises in a preference with a long time constant the error will be large since the old preferred outcome will have a significantly stronger connection to the current context than all other outcomes (including the conflicting outcome). Since the preference has a long time constant the large error should be reduced rapidly.

Equally if a conflict arises in a preference with a short time constant the error will be small since all preference outcomes will have a similar connection strength to the current context. Since the preference has a short time constant the small error should be reduced slowly.

Therefore the design decision regarding conflict resolution is to implement resolution strategies based on the heuristics proposed above. In terms of network error, larger errors (occurring in preferences with long time constants) will be reduced rapidly while smaller errors (occurring in preferences with short time constants) will be reduced less rapidly.

4.9 Summary

The problem description was presented at the start of this chapter outlining the main goal of the research associated with this thesis. As well as the obvious issues it raises, several less obvious issues were highlighted for consideration. Section 2.2.2 outlined how various pervasive projects have attempted to achieve this goal. In considering their solutions, several shortcomings were identified. Additionally, section 3.5 outlined how first-hand experience of providing an implicit personalisation system for DAIDALOS II has highlighted several key areas for improvement.

Most notably, the use of batch learning algorithms for preference learning does not completely meet requirements in a dynamic pervasive environment. The cyclic nature of batch learning means preference sets can become out-of-date between learning cycles. The store of historical user behaviour can also contain bias and errors that are reflected in the learnt preferences. Giraud suggests such issues can occur when a non-incremental algorithm is used for an essentially incremental task. However, incremental algorithms can also suffer problems.

Essentially there is a trade off between up to date output and accuracy. This is most critical in the early stages of incremental learning when initial prediction accuracy is often low until sufficient inputs have been received. Even so, there is evidence to suggest that prediction accuracy is not the most essential requirement in a user modelling domain and mechanisms such as prompting can provide support when required. Therefore it was proposed that an incremental algorithm should be used for the preference learning process in an implicit personalisation system. The desirable incremental and learning properties of such an algorithm were outlined.

The learner's internal knowledge representation is another area for consideration. The rule based approach adopted in DAIDALOS II proved too inefficient for internal processes and hence it was proposed that a neural network structure be adopted.

Although utilised for preference representation in other pervasive projects, it is important to consider how one would understand or alter such data. It was noted that single layer neural networks are simpler to understand and manipulate. They are also successfully implemented in the GAIA project where incrementing and decrementing weight processes enable the networks to handle non-linear problems.

As well as the fundamental design decisions mentioned above, several other lessons learnt from DAIDALOS II were considered. Firstly, it was proposed that environment monitoring should be continuous, capturing both context changes and user actions as they occur. In contrast to the DAIDALOS II action-triggered approach, this would enable learning of both positive and negative preferences from the outset. Secondly, it was proposed that the network should adopt a temporal reinforcement policy. By updating weights continually through time, associations can be made between context and preferences based on their temporal longevity rather than their co-occurrence at one instance in time. This will allow the network to overcome the problem of pre-actions.

Finally, a key issue related to user modelling was discussed. Concept drift is an inevitable phenomenon in a real world environment where user behaviour can change over time. This renders a moving target function which in preference learning terms means that what the user preferred in the past may not be what they prefer in the present. In other words there is a conflict between past behaviour and present behaviour which manifests itself as error in the learning system. Unlike batch algorithms that can reduce error over a number of epochs, the incremental algorithm must appropriately reduce error in one instance. This becomes even more complex when we consider human behaviour and expectations. Ideally, long term changes should be accommodated rapidly while short term changes should be accommodated less rapidly. However, identifying whether a behaviour change is most likely long-term or short-term is non-trivial.

Two conflict resolution heuristics were proposed indicating that change to a long-term preferred outcome is most likely long-term and change to a short-term preferred outcome is most likely short-term. Although the heuristics will not hold in all cases, in the absence of deep knowledge they can indicate the most likely scenario. An investigation into the psychological literature was carried out to seek support for the heuristics at a human behavioural level.

Little information was uncovered in the psychological literature that either supported or refuted the heuristics. Therefore focus shifted back to the known temporal nature of past behaviours indicated through preference time constants. Preference time constants can give an indication to the past temporal trends of a preference and hence can provide a basis for the prediction of future trends. This approach is in line with the heuristics. At a network error level, following such an approach would reduce network error in direct relation to the size of the error.

Although several existing incremental algorithms a) satisfy many of the fundamental incremental and learning properties, b) have a neural network based internal knowledge representation and c) support dynamic topologies, they do not utilise temporal information for weight reinforcement or conflict resolution. Based on the design decisions and requirements outlined in this chapter a novel incremental learning algorithm has been developed to provide an efficient and tailored solution to the problem description.

5 Dynamic Incremental Associative Neural NEtwork (DIANNE)

5.1 Introduction

The DIANNE is a **D**ynamic **I**ncremental **A**ssociative **N**eural **NE**twork with the primary goal of learning associations between two heterogeneous vectors. In the pervasive preference learning domain our aim is to learn context-dependent preferences for a user where the preferred outcome is the most commonly performed action in some context situation. Therefore, the two vectors for association are the user's context and the user's behaviour (i.e. the actions the user performs when interacting with pervasive services). By learning associations between these two vectors the network can learn and store context dependent preferences indicating preferred outcomes for service adaptation and usage in a given context.

The network implements several key concepts. Firstly, the DIANNE implements an incremental approach to learning and hence does not rely on stores of user behaviour history as input. Instead data received from services and context is processed immediately and incorporated into the network. In this way the DIANNE can provide output to personalise services within the pervasive environment from initial system usage. This is a great advantage over batch learning approaches where an initial lag period is required to capture a user behaviour history. During this time, no input is processed and hence no learnt preferences are available for personalisation. Equally, between batch algorithm executions, no new data is processed and hence no new learnt preferences are generated. This incremental approach allows the DIANNE to provide rapid response to changes in user behaviour rendering the network highly responsive to change and constantly up to date.

Secondly, the DIANNE executes a temporal reinforcement policy that continuously alters the strength of associations between actions and context over time. This policy implements the hypothesis that the time a behaviour endures in some context is just as important as the fact that the behaviour was observable in the context. Therefore the strength of associations learned by the DIANNE is not only based on the simultaneous occurrence of action and context states but also the period of time that the simultaneous occurrence of action and context states endured. This enables the DIANNE to overcome the issue of pre-actions.

Finally the DIANNE implements a dynamic architecture. This allows the vectors for association to change structure during network learning. New nodes can be added to vectors as new resources and context information become available. In such a dynamic environment where the array of available resources frequently changes, such an approach is a great asset ensuring continuous learning. The sections below describe the topology of the DIANNE.

5.2 Network Topology

Figure 10 shows a high-level view of the DIANNE infrastructure. In a similar manner to the DAIDALOS PS the DIANNE resides between some Context Management System and the pervasive services which provide input to the DIANNE.

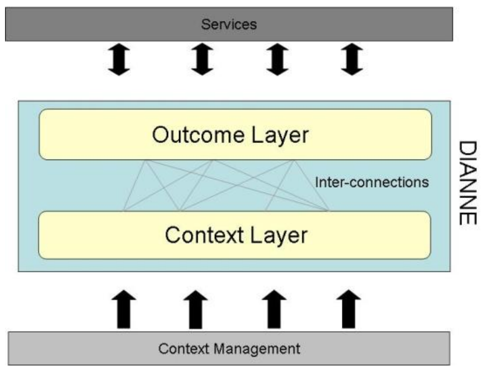


Figure 10. DIANNE Infrastructure

The DIANNE is a single-layer network with no hidden layer although for ease it is described in terms of two layers; the *context layer* and the *outcome layer*. The context layer represents the context parameters and values related to the user in their real world environment. The outcome layer represents the preferred preference outcomes for the adaptation and usage of the pervasive services in the user's real world environment. As the user moves through their environment interacting with various services, the services capture user actions (e.g. selecting a service, changing the volume of a service, etc.) and pass them through to the DIANNE where they are associated with the current context of the user (provided by the Context Management System). These associations allow the DIANNE to predict future adaptations of services on behalf of the user depending on the user's context.

The context layer is an input only layer and does not return any output to the environment. Conversely the outcome layer acts as both an input and an output layer. It receives input from the services in the form of user actions. As in DAIDALOS, after user actions are associated with context (i.e. become part of a context dependent preference) they are referred to as preference outcomes. Therefore the outcome layer provides output in the form of preference outcomes to the services, indicating what the network has learnt the user will do in the current context. This allows the services to implement the preferred outcomes before explicit user request thereby personalising themselves to that particular user.

Within the layers there are two different types of network node. The context layer contains *context nodes* and the outcome layer contains *outcome nodes*. Each context node is connected to every outcome node (and vice versa) and a synapse exists on each connection. Figure 11 shows the structure of a network in a very simple state.

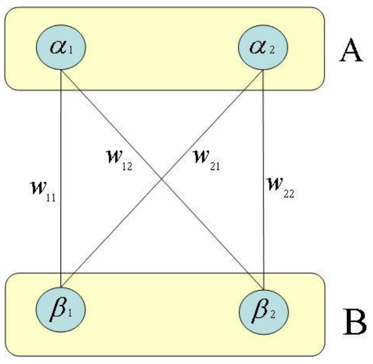


Figure 11. DIANNE Network Structure

A is the set of all outcome nodes $\mathrm{A} = \{\alpha_1, ..., \alpha_m\}$, B is the set of all context nodes $\mathrm{B} = \{\beta_1, ..., \beta_n\}$ and W is the set of all synapses $W = \{w_{11}, ..., w_{mn}\}$. This hetero-associative network stores the pattern pairs (B, A) associating a context state with preferred outcomes.

5.2.1 Context Nodes

A context node represents one value of some context parameter in the user's real world environment (e.g. a context node may represent the value 'home' of the context parameter 'location). This differs from traditional neural networks where nodes often represent a range of values for some parameter. Each context node can be in one of two states; *active* or *inactive*. The *activity* of each context node refers to whether the node is in the active or inactive state and is therefore binary. The activity of a context node is determined by the truth of the related value in the real world environment.

As the user moves through their environment their context will continuously change. The network receives context updates from the context management system. When an update is received the appropriate context nodes are made active and inactive to represent the new real world context of the user. If a context node is active it means that the related real world context value is true, whereas an inactive context node means that the related real world context value is false. Therefore the DIANNE context layer provides a pseudo-representation of the user's real world context.

Due to the explicit, single value structure it may be the case that several context nodes exist in the network, all relating to the same overall context parameter (e.g. there may be multiple context nodes each representing a different value of the context parameter 'location'). It is assumed that a user's context model can only have one true value for each parameter at a time. Therefore this should be reflected in the network by allowing only one node related to the same context parameter to be active at a time.

To implement such a constraint, the context nodes relating to the same context parameter are grouped together as shown in Figure 12. $C = \{c_1, ..., c_i\}$ is the set of all context node groups in the context layer where each group c_i represents some context parameter (e.g. location) and is a unique set of context nodes $\{\beta_n\}$. Any context node that exists in c_i cannot exist in any other context node group.

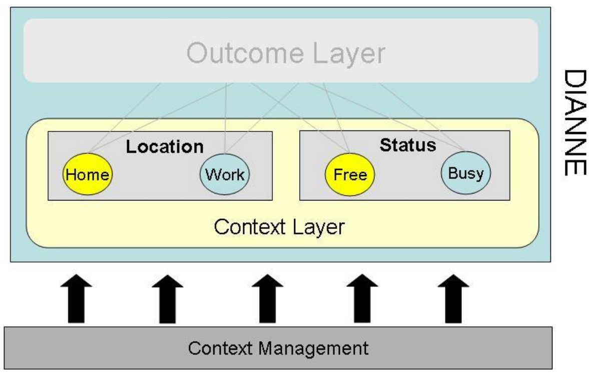


Figure 12. Context Node Groupings in the Context Layer

The activity of members within a context group is mutually exclusive (Figure 12 illustrates active nodes as yellow and inactive nodes as blue) and the active node at any time is the node that represents the true real world value of the context parameter. In the diagram above the network indicates that the user's location is 'home' and their status is 'free' in the real world.

Each context node has an associated *input potential*. This is binary depending on the activity of the node. The input potential is the value that the context node pushes into the network. The input potential of context node β_i is defined as:

$$ip(\beta_i) = \begin{cases} 1 & \text{if } \beta_i \text{ is active} \\ 0 & \text{if } \beta_i \text{ is inactive.} \end{cases} \qquad \textbf{(1)}$$

As the activity of context nodes change so do their input potentials and hence the overall input to the network.

5.2.2 Outcome Nodes

An outcome node represents some possible outcome of a preference in the user's real world environment (e.g. an outcome node may represent the outcome 'mute' of a 'volume' preference). As with context nodes, outcome nodes can also be in one of two states; *active* or *inactive* and therefore the *activity* of an outcome node is also binary referring to what state each outcome node is in. The activity of an outcome node once again reflects the truth of the related value in the real world environment.

As the user interacts with pervasive services in their environment they will perform user actions that change the personalisable aspects of services. The DIANNE receives updates from the services regarding the state of their personalisable parameters (i.e. what preference outcomes are implemented). When an update is received the appropriate outcome nodes are made active and inactive to represent the implemented preference outcomes in the real world. If an outcome node is active it means that the related preference outcome is implemented in the real world, whereas an inactive outcome node means that the related preference outcome is not implemented in the real world. Therefore the DIANNE outcome layer provides a pseudo-representation of the preference outcomes that are actually implemented.

In a similar fashion to the context layer, nodes in the outcome layer can relate to the same preference. Again it is assumed that only one preferred outcome can hold true in the real world for each preference at a time; therefore only one outcome node related to a preference should be active at a time. To implement this constraint outcome nodes are also grouped depending on the preference they relate to as shown in Figure 13. $O = \{o_1, ..., o_i\}$ is the set of all outcome node groups in the outcome layer. Each outcome group o_i represents some preference (e.g. 'volume') and is a unique set of outcome nodes $\{\alpha_m\}$ relating to the various preference outcomes of the preference. Any outcome node that belongs to o_i cannot belong to any other outcome node groups.

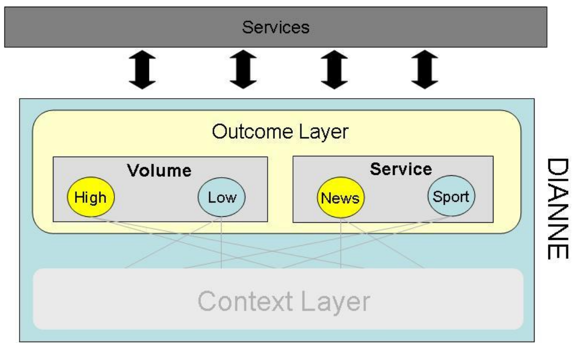


Figure 13. Outcome Node Grouping in the Outcome Layer

As with context groups, the activity of members within an outcome group is mutually exclusive (Figure 13 illustrates active nodes as yellow and inactive nodes as blue) and the active node at any time is the node that reflects the implemented preference outcome in the real world. In the diagram above the network indicates that the volume of some service is set to 'high' and the current selected service is 'news' in the real world.

However, as well as receiving and reflecting input from pervasive services, the DIANNE outcome layer can also provide output to services altering the service to reflect the state of the network. Therefore the activity of an outcome node can also depend on internal network factors and processing.

Each outcome node has an output potential which indicates how strongly the DIANNE believes that this outcome node should be active given the current context. The higher the potential the stronger the belief that the node should be active. Each outcome node has some number *n* of inputs where *n* = |B|. The output potential of an outcome node is the sum of its inputs; therefore the output potential of α_j at time *t* is defined as:

$$
\begin{aligned}
op(\alpha^t{}_j) &= \sigma\left(\sum_{i=0}^{n} w^t{}_{ji}\beta^t{}_i\right) \\
&= \sigma\left(w^t{}_{j1}\beta^t{}_1 + w^t{}_{j2}\beta^t{}_2 + \ldots + w^t{}_{jn}\beta^t{}_n\right) \qquad \textbf{(2)}
\end{aligned}
$$

where σ is the dynamic squashing function that maps the output potential from the possibly very large range of values to a finite range of values between -1 and +1. It should be noted than when the input potential of $\beta^t{}_i$ is zero (i.e. context node β_i is not active at time *t*) the input along that connection received by the outcome node α_j will also be zero since $w^t{}_{ji}0 = 0$.

The outcome node with the greatest potential in each outcome node group is known as the *winner* node. This is the node that the DIANNE believes should be active and implemented in the real world. In the majority of cases the winner node will already be active; however, sometimes this may not be the case. This situation indicates a change in user behaviour from what has been observed and learnt in the past. For example, the DIANNE may identify the 'mute' outcome of some volume preference as the winner node but the user may have manually changed the volume to 'unmute'. The DIANNE must determine how to deal with the conflict and provide appropriate output to the environment based on both internal network variables and external environmental input. This entire process, including conflict resolution is described as part of the DIANNE temporal learning algorithm in section 6.2.

5.2.3 Dynamic Squashing Function

Section 4.8.2 introduced the notion of preference time constants where the time constant of a preference gives some indication of how frequently the preferred outcome of the preference has changed through time. Some preferences will have short time constants while others will have long time constants. This must be considered when determining how normalisation should be applied.

The DIANNE implements a squashing function σ to map the output potentials of outcome nodes from the possibly very large range of values to a finite range between -1 and +1. If the gradient of σ is too steep, behaviour changes will be reinforced too rapidly due to temporal reinforcement (almost as a manifestation of one instance learning). The normalised output potentials would quickly reach upper and lower saturation points and over-ride past information. Of course it is possible to implement σ with a very shallow gradient (hence saturation would take longer to occur) but inevitably preferences with the longest time constants will saturate due to temporal reinforcement, rendering further reinforcements useless.

To overcome this problem, σ is implemented as a dynamic squashing function with a variable gradient. Since output potential comparisons only occur between nodes within the same outcome node group, each group has its own squashing function σ with a dynamic gradient that alters independently in relation to the preference time constant of the group. Therefore the DIANNE implements multiple dynamic squashing functions, one for each outcome group.

For each outcome group, the gradient of σ is identified based on the output potentials of the group's winner node α_{win} (i.e. the node with the highest output potential in the group) and the group's loser node α_{lose} (i.e. the node with the lowest output potential in the group). The function σ is defined as:

$$\sigma = \begin{Bmatrix} 1 & :\text{if } \mathrm{op}(\alpha_{win}) \geq \text{highlimit} \\ grad \bullet op(\alpha^{t}{}_{j}) & \\ -1 & :\text{if } \mathrm{op}(\alpha_{lose}) \leq \text{lowlimit} \end{Bmatrix}$$

where *highlimit* and *lowlimit* dictate the dynamically adjustable positive and negative saturation points of σ. The $grad$ variable is the gradient of the slope of σ defined as:

$$\begin{aligned} grad &= \frac{y_2 - y_1}{x_2 - x_1} \\ &= \frac{1-(-1)}{\text{highlimit} - \text{lowlimit}} \\ &= \frac{2}{\text{highlimit} - \text{lowlimit}} \qquad \textbf{(3)} \end{aligned}$$

Figure 14 below shows a graphical representation of σ with highlimit = +5 and lowlimit = -5.

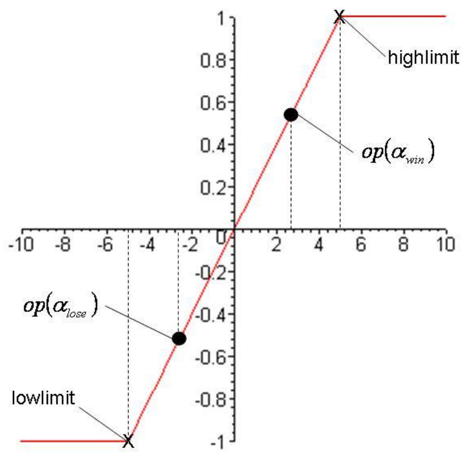


Figure 14. Example graph of squashing function

The highlimit and lowlimit variables influence the value of $grad$ as well as dictating the saturation points of σ. The variable highlimit dictates the positive saturation point while lowlimit dictates the negative saturation point. At time t_0 highlimit is initialised to $+\phi$ and lowlimit is initialised to $-\phi$ where ϕ is come constant greater than zero. In current DIANNE implementations ϕ is initialised to 10. The value of ϕ only affects the frequency with which the gradient of the dynamic squashing function changes.

During DIANNE operation if $op(\alpha_{win}) \geq \text{highlimit}$ or $op(\alpha_{lose}) \leq \text{lowlimit}$ the highlimit and lowlimit variables are updated. The variable highlimit is increased by ϕ while lowlimit is decreased by ϕ. The new value of $grad$ is calculated based on equation **(3)** above. This allows the ramped squashing function to alter over time relative to the time constant of the preference represented by the outcome group. It should be noted that saturation will therefore not occur in the DIANNE due to this approach. This ensures that no valuable temporal reinforcement information is lost as would be the case if saturation were allowed.

Once the gradient of the squashing function is decreased, there is no function to increase it to a steeper gradient should the output potentials of α_{win} and α_{lose} approach closer to zero again. This decision is based on the assumption that if the preference's time constant led to this gradient in the past then it is very possible such a gradient will be required again in the future. In real world terms an outcome group with a shallow gradient reflects the fact that the user consistently selected the same preference outcome in some context over an extended period of time. Even if recent user behaviour is much more changeable in this context it is always possible that the user will revert back to their significantly preferred outcome for an extended period of time again in the future.

The decision to use a linear ramped squashing function is based on several factors. Firstly, the ramped function does not introduce any bias into the system as is the case with sigmoid functions. Using a sigmoid curve such as tanh would favour inputs closer to zero over those further away. Secondly, the ramped function has distinct saturation points compared to a sigmoid curve where no distinct saturation points exist. However, as the sigmoid curve levels off, near-saturation points are reached as further changes to raw node potentials render miniscule changes to the normalised output potential. The lack of distinct saturation points renders it more difficult to dictate thresholds for altering the gradient of the curve.

5.2.4 Network Weights

Each connection between a context node β (pre-node) and an outcome node α (post-node) in the DIANNE has a synapse with an associated weight value. The weight value determines the strength of the connection between β and α (and hence the strength of the association between the real world values they represent). It is the manipulation of these weights that allows the DIANNE to learn. The plasticity of weights is dependent on the activity of pre and post nodes and follows Hebbian and Anti-Hebbian learning policies. A weight will increase if the positive activity of β leads to the positive activity of α (in line with Hebbian learning), decrease if the positive activity of β leads to the negative activity of α (in line with Anti-Hebbian learning) and stay the same if β is not active.

The weight of synapse w_{mn} at time T is defined as:

$$\begin{aligned} {w_{mn}}^{T} &= \left({w_{mn}}^{T-1} + activity^{T}(\alpha_m, \beta_n)\right) \\ &= \left(\sum_{t=0}^{T} activity^{t}(\alpha_m, \beta_n)\right) \end{aligned} \qquad \textbf{(4)}$$

Where:

$$activity^{t}(x,y) = \begin{cases} +1 & : \text{if } \beta_n \text{ is active at time } t \text{ and } \alpha_m \text{ is active at time } t; \\ -1 & : \text{if } \beta_n \text{ is active at time } t \text{ and } \alpha_m \text{ is not active at time } t; \\ 0 & : \text{if } \beta_n \text{ is not active at time } t \end{cases}$$

It should be noted that no squashing function is applied to $w_{mn}{}^{T}$. Initially a sigmoid function (tanh) was implemented to map the very large range of possible weight values to between limits of ± 1. However, the sigmoid curve introduced a bias to the influence of each weight on the output potential of the related outcome node. It was observed that when weight value $w_{mn}{}^{T}$ was closer to zero, tanh($w_{mn}{}^{T}$) was closer to the ± 1 limits compared to a weight value that was further away from zero. Linear squashing functions are a possibility but saturation issues must be dealt with uniformly across all weights. The outcome would be greater overhead and no obvious benefit since weight values are not compared with one another.

5.2.4.1 Weight Initialisation

In many artificial neural networks weights are initialised randomly and modified to converge on some target function during the course of the training process. However, due to the 'learning from scratch', incremental nature of the DIANNE, it is natural to initialise the weights to zero giving them an initial state that is neither excitory nor inhibitory. Unlike non-incremental networks, the weights will not converge on some definitive target value. Some may converge towards upper or lower bounds ($\pm\infty$) for periods of time but equally others may not.

5.3 Network Analysis

There are many performance vectors with which neural networks can be analysed and compared. Apart from encoding and recall mechanisms we can also consider network generality, capacity, stability and convergence.

5.3.1 Generality

Non-Linear Problems

A well documented constraint of single layer neural networks is their inability to handle non-linear problems such as XOR. As a single layer network, the DIANNE will not be able to represent XOR; however, in this problem domain (context-dependent preference learning) the DIANNE will never need to solve the XOR problem. Consider the four XOR states shown in Figure 15.

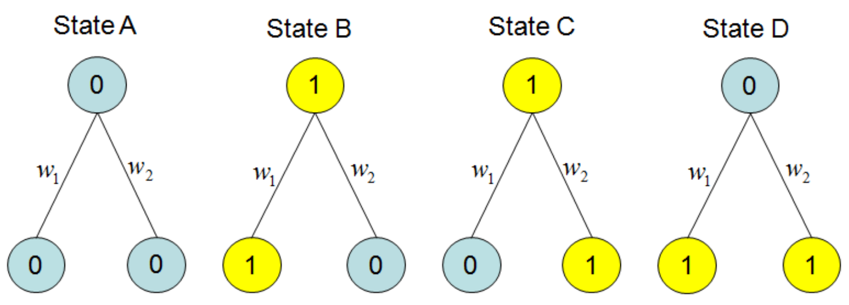


Figure 15. XOR Network States

It is possible to represent state A in the DIANNE. When all context nodes are inactive the DIANNE can make no prediction so it will not return an outcome node in these circumstances. States B and C can also be represented by the DIANNE. When input is available (i.e. some context nodes are active) the DIANNE will always return the most likely outcome node for each node group. If there is only one outcome node in a group, that outcome node will always be returned in response to input. With that in mind, state D will never need to be represented by the DIANNE as when input is available output will always be provided.

Continuous Inputs

The DIANNE has been designed to handle discrete inputs. Due to the fact that each network node represents only one value, continuous input would result in the creation of numerous network nodes and would decrease performance significantly. Since many context attributes can relate to continuous sensor inputs we must consider how this data is handled in the DIANNE.

As is the case with many other learning algorithms it is necessary to discretise inputs before they are presented to the DIANNE. Rather than utilising raw sensor data the DIANNE utilises discrete values that are inferred from one or more raw sensor inputs. For example, the DIANNE utilises inferred symbolic location names rather than continuous GPS coordinates.

5.3.2 Capacity

It is important to understand how many associations between input vectors and output vectors the DIANNE can store. The capacity of a network is often expressed as a function of *N*, the number of nodes the network contains. However, the DIANNE groups nodes into mutually exclusive sets. Therefore we must define the capacity of the DIANNE in terms of both network nodes and groups.

First we will consider the number of possible input patterns P_{con} the network can handle. A context group $c_i \in C$ contains some n number of context nodes. Since the activation of each node in c_i is binary the total number of possible node activation patterns in c_i is 2^n. However, we must discount any activation patterns that violate the mutually exclusive constraint on the activation of nodes within the same group. Therefore, if c_i contains 1 node the possible activation patterns are:

Node A
on
off

Both activation patterns are legal and do not violate mutually exclusive node activation. Therefore when c_i contains only 1 node it has 2 possible activation patterns. If c_i contains 2 nodes the possible activation patterns are:

Node A	Node B
on	on
on	off
off	on
off	off

The first pattern is not possible as it violates mutually exclusive node activation. Therefore, when c_i contains 2 nodes it has 3 possible node activation patterns, or more generally if c_i contains n nodes the number of possible node activation patterns for c_i is:

$$(n+1)_i$$

Therefore if we have k context node groups then:

$$\begin{aligned} P_{con} &= \left((n+1)_1 \times (n+1)_2 \times \ldots \times (n+1)_k\right) \\ &= \prod_{i=1}^{k} (n+1)_i \end{aligned}$$

Linear connections, binary node activations and the single layer architecture of the DIANNE mitigate against internal hidden complexity. Therefore each input pattern $P_{con}{}^{i}$ has one associated output pattern $P_{out}{}^{i}$. In other words:

$$P_{out} = P_{con}$$

Therefore the DIANNE storage capacity is equal to the number of possible input patterns (i.e. the number of possible context situations) P_{con}.

One interesting point to note is the relationship between the network capacity and the number and size of context node groups. Let's assume we have 2 networks each containing 32 context nodes but grouped in different ways. Network 1 has 3 context node groups containing 10, 13 and 9 nodes respectively. The capacity of this network is:

$$= (10+1)\times(13+1)\times(9+1)$$
$$= 1170$$

Network 2 has 12 context node groups containing 2, 4, 3, 3, 2, 2, 3, 4, 3, 2, 2 and 2 context nodes respectively. The capacity of this network is:

$$= (2+1)\times(4+1)\times(3+1)\times(3+1)\times(2+1)\times(2+1)\times(3+1)\times(4+1)\times(3+1)\times(2+1)\times(2+1)\times(2+1)$$
$$= 82944$$

It is clear that in terms of network capacity the number of context node groups is the influential factor, not the number of context nodes.

5.3.3 Stability

Network stability is realised if output is only slightly disturbed by deviant input. It is often formally expressed through the notion of Lyapunov stability [139] which conceptually states that solutions starting 'close enough' to the equilibrium remain 'close enough' forever.

The domain of all possible raw outcome node potentials is infinite; however, a dynamic squashing function σ maps the raw node potentials from their infinite domain to within the limits:

$$\lim_{x\to\infty} y = 1 \qquad \lim_{x\to-\infty} y = -1$$

The dynamic nature of σ ensures that saturation will not happen. However, we must ensure that the dynamically changing gradient of σ does not have a negative effect on the stability of the network. The gradient of σ is global for all outcome nodes in the same group. This means that different outcome groups can have different global gradients related to σ. However, the output potentials of the outcome nodes are only compared between nodes within the same outcome group. Therefore, different gradients in different outcome groups will not affect the output of the network. Within each outcome group the changing gradient should not cause any detrimental effects to outcome group processing (such as determining the winner node) since the output potentials of nodes within the group are always normalised with the same gradient and are therefore comparable.

One potential problem the network may face is *flicker*. This occurs when small environmental changes have large effects on network output. For example, consider the situation where temperature sensors (providing continuous input) are context sources. If the user prefers to set the volume of some multimedia service to ‘low’ because it is after 9pm, it should not be the case that the continually changing temperature sensor inputs affect this setting, causing the volume to frequently change.

In most cases the natural network learning will filter such network noise. A variable that remains constant in some environment will become more strongly associated with other, less changeable, environmental variables due to reinforcement over time. Variables that change frequently will be very weakly associated with other environmental variables and therefore have little influence over network output. Other external approaches such as context inference can be used to infer higher level discrete context values from multiple continuous context sources (such as sensors) helping to reduce the possibility of network flicker from rapidly changing inputs.

5.3.4 Convergence

Convergence is related to the learning in a system. We tend to say a system has captured presented data correctly if it converges to some fixed value or some fixed set with minimal error. However, the incremental and temporal nature of DIANNE learning means we will not observe convergence in the traditional way. A fixed set with minimal error will never be reached although the network will tend towards it. If we assume no conflicts occur in the network, convergence can be observed at outcome node level and outcome node group level.

Generally, convergence is defined as:

$$\lim_{n \to \infty} x_n = x$$

If we assume that n is equal to t (a unit of time over which the DIANNE operates) and x is equal to the potential of some outcome node α we can rewrite the equation to represent the convergence of an outcome node.

$$\lim_{t \to \infty} \alpha_t = 1 \, (\text{if } \alpha \text{ is active})$$

$$\lim_{t \to \infty} \alpha_t = -1 \, (\text{if } \alpha \text{ is not active})$$

Notice that the outcome node has two limits at positive and negative 1. The convergence limit at time t depends on the activity of the outcome node. If the outcome node is active it will converge towards $+1$ otherwise it will converge towards -1. The rate of convergence towards these upper and lower limits depends on the learning rate of the DIANNE and the frequency with which temporal reinforcements are applied.

The convergence of an outcome node group is observable by considering the delta Δ between the output potential of the winner node and the output potentials of other nodes in some context. This will alter through time due to temporal reinforcements based on the activity of the group nodes.

In a *stable* outcome group the same outcome node will remain the active winner for a significant period of time (given some context input vector). In this situation Δ will tend towards its maximum value 2 as the output potential of the active winner node tends towards +1 and the output potentials of the other group nodes tend towards -1 in this context. This reflects the fact that the user has settled on a significantly preferred outcome in this context and hence the network converges on this outcome accordingly.

In an *unstable* outcome group different outcome nodes will be the active winner for short periods of time (given some context input vector). In this situation Δ will remain close to its minimum value 0 i.e. the output potentials of all the outcome nodes in this outcome node group will remain very similar. This reflects the fact that the user has not settled on a preferred outcome in this context and regularly changes their mind. In such a situation the network cannot converge on a single outcome. However, in this situation another type of convergence is observable.

In an unstable outcome group the output potential of all the nodes in the group will tend towards -1. The rate of this convergence is dependent on the number of nodes in the outcome group. Consider the network state shown in Figure 16.

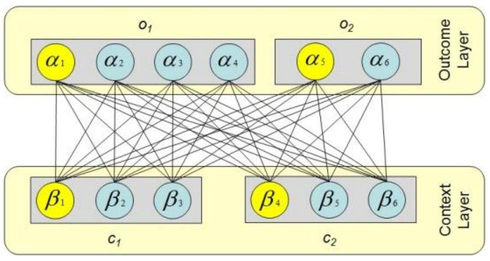


Figure 16. Example network state

Assume that both outcome groups o_1 and o_2 are unstable. Each outcome node in group o_1 will, on average, be the active winner (with an increasing potential) 25% (1/4) of the total time and inactive (with a decreasing potential) 75% (3/4) of the total time in some context. In comparison the outcome nodes in group o_2 will, on average, be the active winner (with an increasing potential) 50% (1/2) of the total time and inactive (with a decreasing potential) 50% (1/2) of the total time in some context. With a greater chance of inactivity (decreasing potential) than activity (increasing potential), the output potentials of the nodes in group o_1 will tend towards -1 at a greater rate than the output potentials of the nodes in group o_2. Therefore the greater the number of outcome nodes in an unstable outcome node group, the more rapidly the output potentials of the group nodes will converge towards -1.

This convergence of output potentials to -1 in an unstable outcome group is expected behaviour. Since the behaviour of the user is uncertain and changeable it is correct for this to be reflected through greater uncertainty in the network. If this continues for a significant period of time the potentials of the (unstable) outcome group nodes should tend towards their greatest uncertainty value of -1.

5.4 Summary

The DIANNE network is a dynamic, incremental, associative neural network which aims to learn associations between heterogeneous vectors in an incremental and time dependent fashion. When learning user preferences for the adaptation and usage of pervasive services the two vectors for association are the user's context and the actions they perform related to preference outcomes when interacting with services. By associating these two vectors the DIANNE can learn and store context-dependent preferences indicating what preference outcomes the user prefers to implement in a given context. The dynamic nature of the DIANNE also allows input and output vectors to adapt during the learning process without the need to re-define the network and re-run learning.

The linear structure of the DIANNE was presented and each of its component parts described. Two node layers (the context layer and the outcome layer) provide pseudo-representations of user context and implemented preference outcomes in the real world. Single value nodes, node groupings and a combination of Hebbian and Anti-Hebbian weight manipulations allow the network to represent all situations that it will be required to handle in the context-dependent preference learning domain. This does not include non-linear problems such as XOR. It is shown that in this problem domain the DIANNE will never need to represent the XOR problem.

The use of a dynamic squashing function at outcome node potentials ensures that saturation points are not reached in this temporally reinforced network. The issues of capacity, stability and convergence were discussed with regard to the DIANNE including the impact of nodes and node groups on network capacity and observed trends towards negative output potentials in outcome groups related to preferences with short time constants.

6 DIANNE Operation and Application

6.1 Introduction

The DIANNE is essentially a feed-forward, single layer neural network. Figure 17 summarises the topology of the DIANNE as presented in Chapter 5 highlighting some of the significant features. Outcome node potentials are determined by the sum of their weighted inputs. Normalisation is applied by multiple independent dynamic squashing functions, one for each outcome node group. Outcome node activations are mutually exclusive within node groups and are based on both external input from the environment and internal variables such as the output potential of each outcome node. Therefore the activation of outcome nodes is boolean even though their output potentials are continuous.

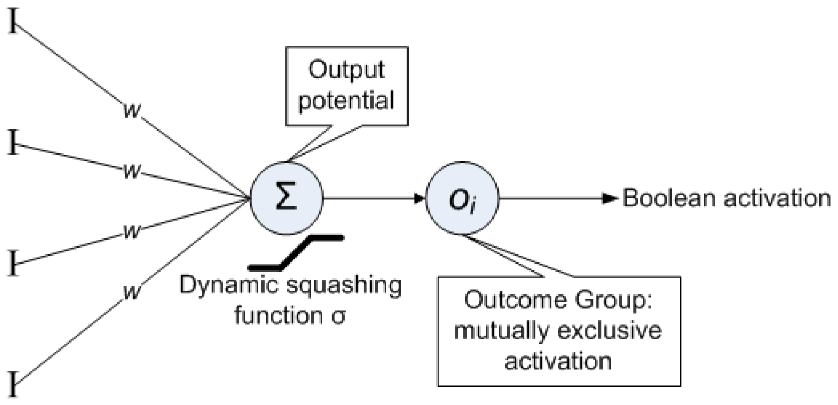

I
I
I
I
w
w
w
w
Output potential
Σ
o_i
Boolean activation
Dynamic squashing function σ
Outcome Group: mutually exclusive activation

Figure 17. Summary of DIANNE Topology

The single layer topology of the DIANNE has many advantages when utilised for incremental learning purposes. Indeed a single layer neural network has also been the topology of choice for several incremental learning algorithms detailed in section 2.3.6. However, it is the temporal reinforcement policy of the DIANNE learning algorithm (outlined in section 4.7.1) that sets it apart from other incremental approaches. This section details the algorithm and its various sub-processes before going on to describe how the DIANNE was applied within the EU FP7 PERSIST project.

6.2 The DIANNE Temporal Learning Algorithm

Unlike most conventional neural systems, the DIANNE learns associations between vectors based on the duration of vector state co-occurrences rather than the fact that they co-occurred at one instance in time. To do this the DIANNE learning algorithm is temporal, iterating in a continuous cycle in real-time. The iteration frequency is controlled by a *network update rate (nur* where *nur>0)* variable which is set to some unit of time. Therefore the network is updated and the DIANNE learns, strengthening and weakening its associations, on a temporal basis.

Initial DIANNE algorithm designs tended towards an asynchronous approach with each network node operating in its own thread, updating itself depending on asynchronous inputs from other nodes. However, synchronisation issues quickly appeared and network accuracy suffered as a result. Therefore a more synchronous approach has been adopted where all network updating occurs in an ordered and discrete fashion.

However, the adoption of a synchronised implementation opens the DIANNE to sampling which can add bias to the system. For example, it is possible that between cycles, some context attribute value could change from X to Y and back to X again. During the next cycle the network will learn based on the X value being active, overlooking the occurrence of the Y value. Therefore it is essential that the nur is sufficiently small to capture and learn upon most inputs. Set to 1 second, the nur is sufficient to handle discrete context attributes such as 'day of the week' where values typically change at a much lower frequency. Although context attributes related to sensor input can potentially update at a frequency greater than 1 second, one could argue that context states that endure for durations shorter than 1 second will have minimal effect on preference outcomes and hence little significance on network output. Therefore in the example above the occurrence of Y for less than 1 second will have little significance on the preferred outcomes.

The DIANNE can overcome two other bias issues: capturing the non-occurrence of actions for negative preference learning (as discussed in section 4.6) and handling pre-actions (as discussed in section 4.7). By capturing and processing a snapshot of the entire environment every second, both the occurrence and non-occurrence of actions in some context are captured and learnt upon. By learning associations based on the duration of vector co-occurrence the DIANNE does not incorrectly associate actions with the current context when in fact the actions were performed prematurely in preparation for a future context.

Figure 18 illustrates the temporal DIANNE learning algorithm. Note that the algorithm is cyclic, repeating at a frequency dictated by the *nur*. The algorithm can be split into two major processes. The *layer update process* is concerned with processing input (if any) received from the environment since the previous iteration. The *learning process* is concerned with updating the DIANNE weights and providing output (if any) to the environment. Each major step and its sub-processes are described in detail below.

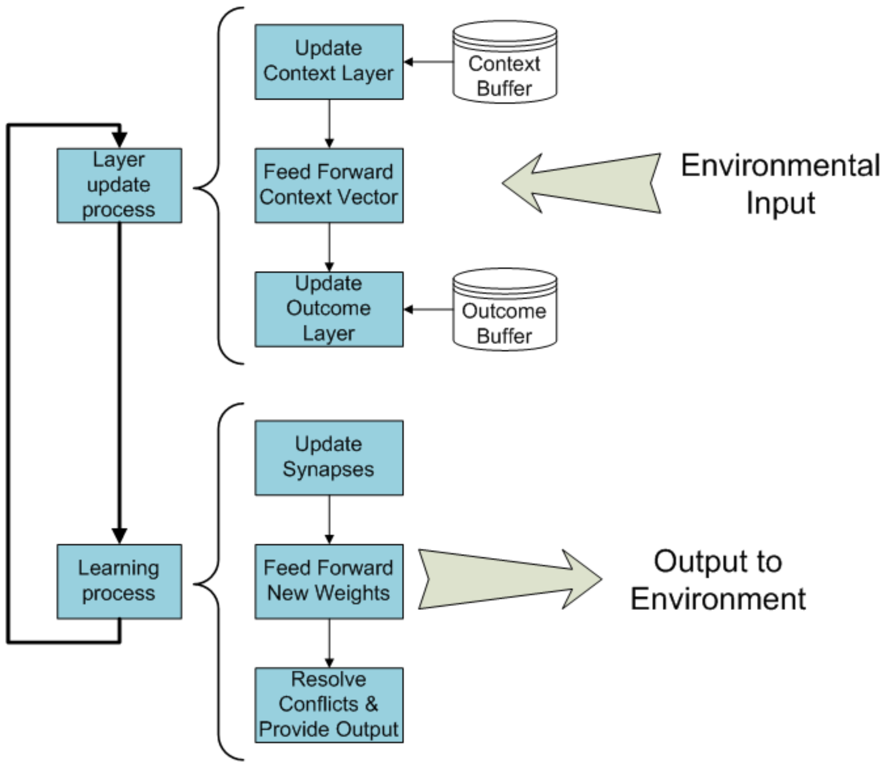


Figure 18. Illustration of the main processes involved in the DIANNE temporal learning algorithm.

6.2.1 The Layer Update Process

The main purpose of this process is to ensure that the network nodes correctly reflect the environmental state before any learning updates occur. The DIANNE context vector should reflect the user's current context and the DIANNE outcome vector should reflect the user's currently preferred preference outcomes. Failure to perform this process correctly will result in the network associating incorrect context and preference vectors. Between iterations, all changes to the user's context are stored in a *context buffer* and all changes to the user's preferred preference outcomes are stored in an *outcome buffer*. The buffers then provide input to the three sub-processes involved in the layer update process.

Update Context Layer

The first sub-process involves updating the context layer based on the contents of the context buffer. This ensures that the context layer reflects the user's current contextual state (for example, since the last algorithm iteration, the user may have changed location). Figure 19 shows the pseudocode and flow diagram for the process of updating the context layer. Each item in the context buffer is a context attribute-value pair of the form:

$$<c_m, \beta_i>$$

where c_m is the context attribute and corresponds to the name of a context group in the DIANNE. β_i is the context attribute value and corresponds to the name of a context node within that group. If the group and node are located inside the DIANNE the node is activated and all others in the group de-activated. However, since the DIANNE has a dynamic architecture that grows over time as new context sources and preferences arise, it may be the case that a related node or even group does not yet exist in the DIANNE. If no group exists, a new group is created with a name equal to the context attribute and a new node is created inside the group with a name equal to the context attribute value. The node is then activated. If the group already exists but the node does not, a new node is created inside the existing group with a name equal to the context attribute value and activated. Each new context node is connected to all nodes in the outcome layer with initial connection strengths (weight values) of 0.

```
if $c_m \in C$ where $c_m$ corresponds to the context attribute of the input
    if $\beta_i \in c_m$ where $\beta_i$ corresponds to the context attribute value of the input
        set activity of $\beta_i$ to active;
        set activity of all other nodes in $c_m$ to inactive
    else
        create new context node $\beta_{new}$ in $c_m$
        set activity of $\beta_{new}$ to active
        set activity of all other nodes in $c_m$ to inactive
else
    create new context node group $c_{new}$
    create new context node $\beta_{new}$ in $c_{new}$
    set activity of $\beta_{new}$ to active
```

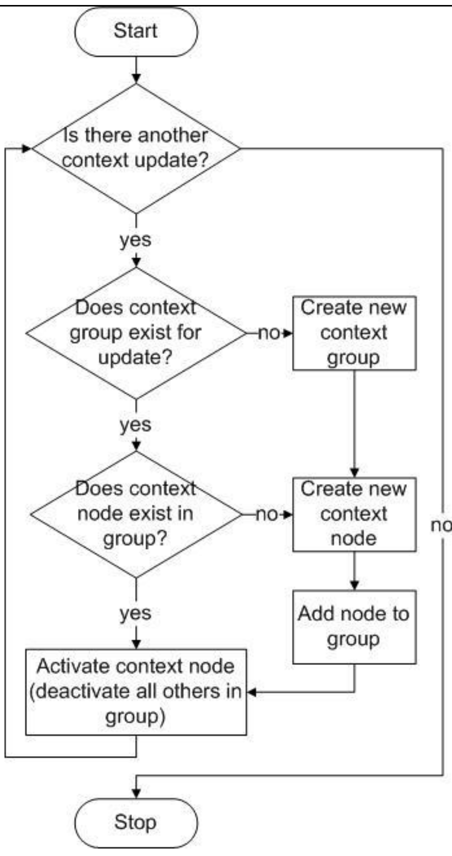

Start
Is there another context update?
yes
Does context group exist for update?
no
Create new context group
yes
Does context node exist in group?
no
Create new context node
Add node to group
yes
Activate context node (deactivate all others in group)
no
Stop

Figure 19. Pseudocode and flow diagram illustrating the process of updating the context layer.

Feed Forward Context Vector

The second sub-process feeds the new context vector forward through the network to the outcome layer where the outcome vector is updated accordingly. The outcome vector now represents what the network believes to be the preferred preference outcomes in this context. Figure 20 illustrates the pseudocode and flow diagram for the process of feeding forward the new context vector to the outcome layer.

The potentials of the outcome nodes are re-calculated and the node with the highest potential in each outcome group is activated. At this point, if any potentials are reaching the saturation point for the group, the gradient of the dynamic squashing function is decreased for the group to ensure saturation does not occur. The outcome layer now represents what the network believes to be true in this context. It is notable that the DIANNE produces no output at this point even though the outcome vector may have changed due to a new context vector. Output is only returned to the environment during the learning process after associations have been updated.

```
for each outcome node group $o_n$ :
    for each outcome node $\alpha_j \in o_n$ :
        calculate output potential based on equation (2); [Section 5.2.2]
    activate node with highest output potential in $o_m$

Update group gradient based on equation (3); [Section 5.2.3]
```

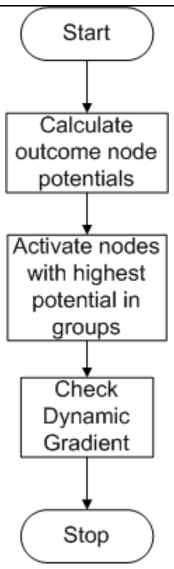

Start
Calculate outcome node potentials
Activate nodes with highest potential in groups
Check Dynamic Gradient
Stop

Figure 20. Pseudocode and flow diagram illustrating the process of feeding forward the new context vector to the outcome layer.

Update Outcome Layer

The final sub-process involves updating the outcome layer based on the contents of the outcome buffer. The contents of the outcome buffer are the result of user actions (for example, the user may have changed their preferred news website since the last algorithm iteration). In the previous sub-process the outcome vector was updated based on new context input to reflect what the network believes to be true in this context. In contrast, this sub-process updates the outcome vector based on new user input to reflect any changes in preferred outcomes made by the user. Therefore after the completion of this sub-process, the DIANNE outcome vector will reflect both what the network believes to be the preferred preference outcomes in this context and also what the user has actually implemented as their preferred preference outcomes in this context.

Figure 21 illustrates the pseudocode and flow diagram for the process of updating the outcome layer. This process is equivalent to the process of updating the context layer. Input from the outcome buffer is processed one at a time. Each item in the outcome buffer is a preference name-outcome pair of the form:

$$<o_n, \alpha_j>$$

where o_n is the preference name and corresponds to the name of an outcome group in the DIANNE. α_j is the preference outcome and corresponds to the name of an outcome node within that group. If the group and node are located inside the DIANNE, the node is activated and all others in the group de-activated. However, as with the context layer it may be the case that the input corresponds to some new preference or some new preference outcome warranting the creation of a new outcome group or outcome node. When a new node is created inside a new or existing group, it is activated. It is also connected to every node in the context layer with initial connection strengths of 0. Therefore the initial potential of any new outcome node is also 0.

```
if $o_n \in O$ where $o_n$ corresponds to the preference name of input
    if $\alpha_j \in o_n$ where $\alpha_j$ corresponds to the preference outcome of input
        set activity of $\alpha_j$ to active;
        set activity of all other nodes in $o_n$ to inactive
    else
        create new outcome node $\alpha_{new}$ in $o_n$
        set activity of $\alpha_{new}$ to active
        set activity of all other nodes in $o_n$ to inactive
else
    create new outcome node group $o_{new}$
    create new outcome node $\alpha_{new}$ in $o_{new}$
    set activity of $\alpha_{new}$ to active
```

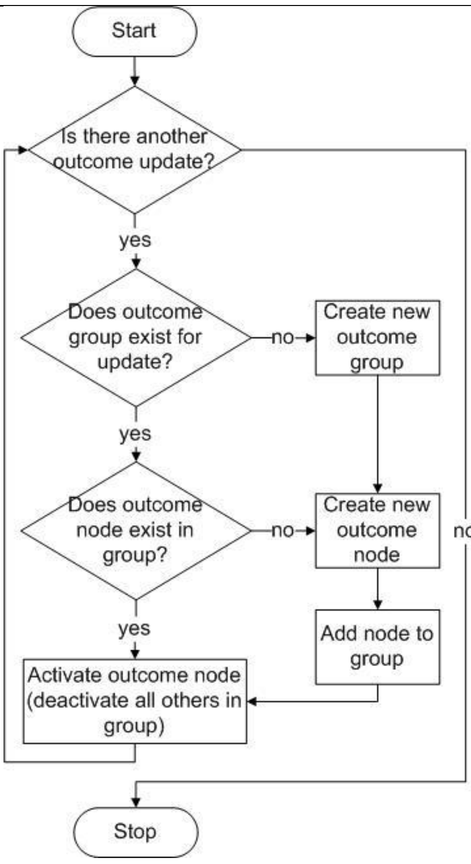

Start
Is there another outcome update?
yes
no
Does outcome group exist for update?
no
Create new outcome group
yes
Does outcome node exist in group?
no
Create new outcome node
yes
Add node to group
Activate outcome node (deactivate all others in group)
Stop

Figure 21. Pseudocode and flow diagram illustrating the process of updating the outcome layer.

6.2.2 The Learning Process

Now that the DIANNE vectors reflect the current real world state of context and preference outcomes, the learning process can execute to strengthen and weaken associations between the two vectors enabling the DIANNE to learn. Within the learning process, two learning rules are utilised. Firstly, the Hebbian learning rule is used for normal, temporally driven updating of the synapse weights. This learning occurs always in each iteration of the DIANNE algorithm, incrementally increasing and decreasing network weights based on pre and post synaptic node activations. Secondly, an error driven policy is utilised to update synapse weights during conflict resolution. This learning only occurs when conflicts exist between network knowledge and real world states and is based on the potentials of the conflicting network nodes. There are three sub-processes involved in the overall learning process. Each is described in detail below.

Update Synapses

Firstly, the synapse on each network connection is updated based on the Hebbian learning rule (including anti-Hebbian). Figure 22 illustrates the pseudocode and flow diagram for the process of updating all synapses. Each update is dependent on the activity of the pre and post synaptic nodes i.e. the activity of the context and outcome nodes attached to the synapse. As outlined in section 5.2.4, if both the pre-node and the post-node are active, the weight at the synapse is increased. If the pre-node is active and the post-node is not active, the weight at the synapse is decreased. If the pre-node is not active the weight is not manipulated. Once this sub-process is complete the DIANNE must now respond to reflect the new weight values in the outcome vector.

```
for each connection $(\alpha_j \beta_i)$
        for each weight $w_{ji}$
                update value of $w_{ji}$ based on
                equation (4); [Section 5.2.4]
```

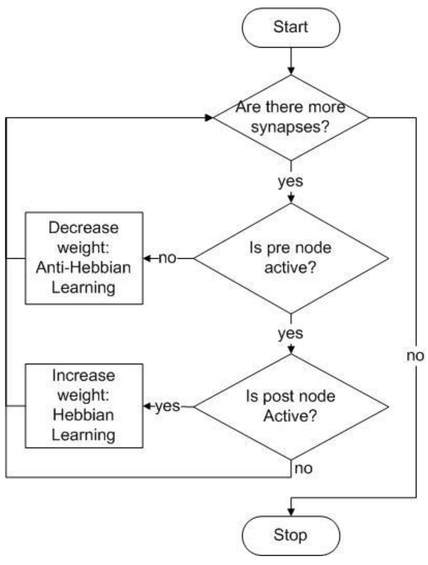


Figure 22. Pseudocode and flow diagram illustrating the process of updating the network synapses.

Feed Forward Context Vector

As in the layer update process, a feed-forward sub-process is now required to feed the new synaptic weights forward to the outcome layer allowing the outcome vector to reflect network output, post learning. This sub-process is similar to the feed forward sub-process in the layer update process as illustrated in Figure 23. The context vector is fed through the network to the outcome nodes, where their potentials are re-calculated based on the updated synapse weight values. However, this time no outcome nodes are activated. Instead the winner node (the node with the greatest potential) in each outcome group is identified. Therefore in terms of active nodes, the outcome vector still reflects its state after the layer update process. This is necessary for the final sub-process where conflicts are identified and resolved and output is provided.

```
for each outcome node group $o_n$ :
    for each outcome node $\alpha_j \in o_n$ :
        calculate output potential based
        on equation (2); [Section 5.2.2]
    identify the winner node in $o_n$ ;

Update group gradient based on equation (3);
[Section 5.2.3]
```

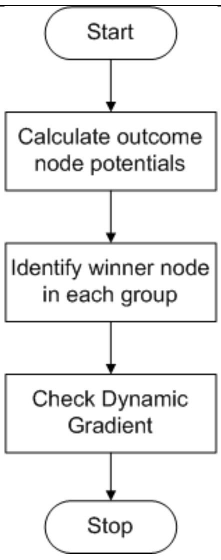

Start
Calculate outcome node potentials
Identify winner node in each group
Check Dynamic Gradient
Stop

Figure 23. Pseudocode and flow diagram illustrating the process of feeding forward the context vector (and new network weights) to the outcome vector.

Resolve Conflicts & Provide Output

At this stage the DIANNE can now provide output to the environment as appropriate and resolve conflicts within the network. Figure 24 illustrates the pseudocode and flow diagram for the process of resolving conflicts and providing output to the environment.

At this point in the algorithm cycle, each outcome group has a winner node and an active node. The winner node indicates the preference outcome that the network believes should be implemented while the active node indicates the preference outcome that has actually been implemented by the user. If the winner node and active node are the same node we can say that what the DIANNE believes to be true is actually true in the real world, hence there is no conflict.

In such a situation the DIANNE can now provide output to the environment if appropriate. The current winner node of this outcome group is checked against the winner node of the previous algorithm iteration. If they are different this means that a new winner node has been identified for this context and must be communicated to the environment. In this case the DIANNE broadcasts the new winner node as output. If the winner node is the same as the previous iteration, the environment already knows about and conforms to this winner node and hence the DIANNE does not need to communicate the winner node again.

A conflict occurs within the network when the winner node is not the same node as the active node i.e. what the network believes should be implemented is not what is actually implemented. This usually occurs when the user changes their preferred preference outcome in some context.

```
for each outcome node group $o_n$ :
    if winner node is active
        if winner node is a new winner node
            send output to service
    else
        resolve conflict between network and user
```

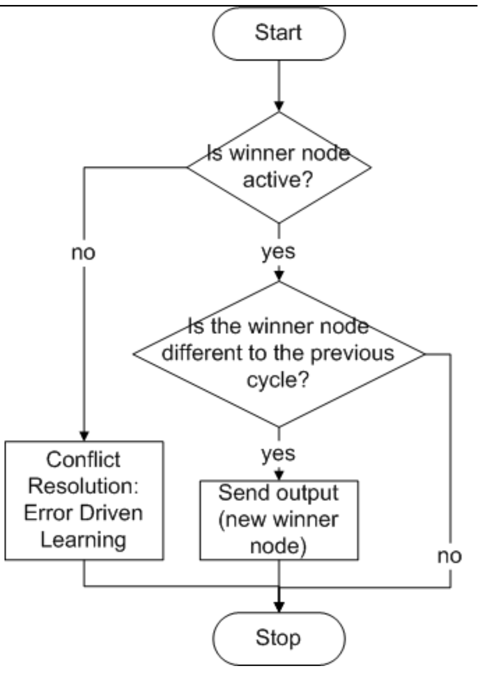

Start
Is winner node active?
no
yes
Is the winner node different to the previous cycle?
yes
no
Conflict Resolution: Error Driven Learning
Send output (new winner node)
Stop

Figure 24. Pseudocode and flow diagram illustrating the process of resolving conflicts and providing output to the environment.

Section 4.8 discussed aspects of incremental conflict resolution reflecting on the problem domain of learning user preferences. Psychological aspects (such as user expectation of learning rates and the intended longevity of behaviour changes) come into play when considering a user centric domain and hence DIANNE conflict resolution is considered as more than a typical error reduction process.

For this reason, the DIANNE cannot use the Hebbian learning rule (used for temporal DIANNE learning) to resolve conflicts. The Hebbian learning rule could not always reduce network error in a sufficient time frame as would be deemed acceptable by the user. As the DIANNE learns through time, the potentials of nodes within the same node group can differ greatly, increasing and decreasing to upper and lower limits. For example, if a user has preferred to use the BBC News website for a year the potential of the BBC News node will have been reinforced towards the upper limit for a year during which time the potential of the MSN News node will have been reinforced towards the lower limit. Figure 25 illustrates this scenario showing the two outcome node potentials on the dynamic squashing function.

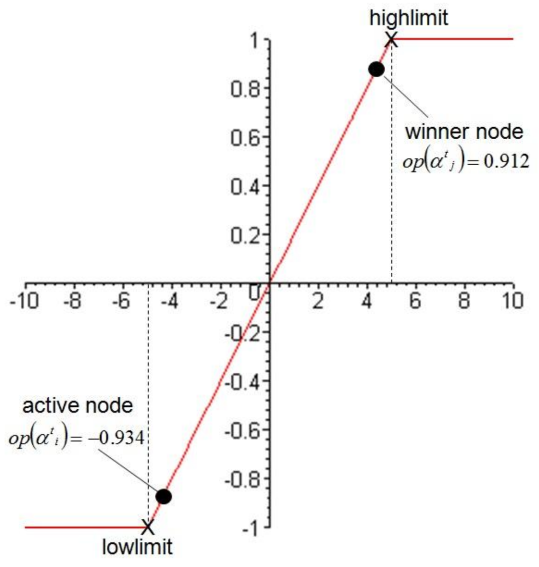


Figure 25. A possible scenario involving the potentials of 2 competing outcome nodes.

If a conflict now arises, Hebbian learning alone would take an unacceptably long time to accommodate the behaviour change, reducing the error in small incremental steps. Therefore, the DIANNE utilises a different learning policy for error driven learning. The DIANNE error driven learning policy is based on the heuristics proposed in section 4.8.

Any change to a long-term preferred outcome is viewed as being a long-term change and therefore accommodated rapidly into the network. Section 4.8 explained how in network terms, a long-term preferred outcome will have a much higher connection strength to the current context than all other outcomes. Therefore, when network error is greater, the error will be reduced more rapidly. Equally, any change to a short-term preferred outcome is viewed as being a short-term change and therefore accommodated less rapidly into the network. A short-term preferred outcome will have a connection strength to the current context that is similar to all other outcomes. Therefore, when network error is smaller, the error will be reduced less rapidly.

To reduce error the DIANNE boosts the potential of the active node by some degree, allowing the active node to better compete with the winner node in an acceptable time frame. Based on the statements above, the rate of error reduction is proportional to the size of the error. In other words the DIANNE boosts the active node potential $op(\alpha_{active})$ by some value Δ which is a factor of the error such that the new potential of the active node is

$$op(\alpha_{active})' = op(\alpha_{active}) + \Delta$$

To determine the boost value we must specify the error as a percentage of the entire possible error. The error is the difference between the active node potential $op(\alpha_{active})$ and the winner node potential $op(\alpha_{win})$.

$$error = op(\alpha_{win}) - op(\alpha_{active})$$

The maximum possible error is 2 due to the dynamic squashing function normalising all outcome node potentials between limits of -1 and +1. Therefore the difference between the potentials of the two conflicting nodes must be identified as a percentage of 2.

$$percentage\ error = \frac{error}{2} \times 100$$

The active node potential $op(\alpha_{active})$ is then boosted by the percentage error. The adaptation value Δ becomes

$$\Delta = \frac{percentage\ error\ \times\ error}{100}$$

Expanding this equation into its original values we can simplify it as follows:

$$\Delta = \left(\left(\frac{\left(op(\alpha_{win}) - op(\alpha_{active})\right)}{2} \times 100 \right) \times \left(\frac{\left(op(\alpha_{win}) - op(\alpha_{active})\right)}{100} \right) \right)$$

$$= \frac{100\left(op(\alpha_{win}) - op(\alpha_{active})\right)}{2} \times \frac{\left(op(\alpha_{win}) - op(\alpha_{active})\right)}{100}$$

$$= \frac{\left(op(\alpha_{win}) - op(\alpha_{active})\right)}{2} \times \left(op(\alpha_{win}) - op(\alpha_{active})\right)$$

$$= \frac{\left(op(\alpha_{win}) - op(\alpha_{active})\right)^2}{2}$$

Interestingly, the simplified equation is equivalent to the *Stochastic Gradient Descent* learning rule (also called the *Incremental Gradient Descent* learning rule*)* used by other incremental algorithms such as WINNOW and the Pocket Algorithm. This result provides encouraging support for the heuristics proposed in section 4.8 as the heuristics appear to be in line with standard error driven learning policies utilised in existing incremental algorithms.

As the potential of the active node is now boosted, the change must be reflected down through the synapses on the connections between this node and the currently activated context nodes otherwise the sum of the inputs to the active node will no longer match the node's potential. The weights of all active synapses (where context pre-node is activated) should also be boosted by some equal value. This value is identified by dividing the active node boost value Δ by the number of active synapses. Note that only active synapses are updated as inactive synapses are not currently contributing to the potential of the active node.

When a conflict is identified, boosting of the active node potential only occurs once. Boosting the active node potential on subsequent algorithm iterations would reduce error extremely rapidly simulating undesirable one instance learning. Instead the boost operation is only performed when the conflict is initially discovered. After that Hebbian learning is applied as usual on subsequent algorithm iterations allowing temporal reinforcement policies to regain dominance.

6.3 DIANNE Application - The PERSIST Project

The DIANNE has been successfully implemented and deployed within the PSS platform, developed as part of the EU Framework 7 PERSIST project [140]. The PERSIST (PERsonal Self-Improving SmarT spaces) project ran over two and a half years from April 2008 to September 2010 involving 10 partners from academia and industry. The goal of the project was to define and develop a pervasive system platform based on the concept of a *Personal Smart Space* (PSS) [135, 136].

6.3.1 The Personal Smart Space (PSS)

Pervasive computing research often centres on the development of pervasive systems within a local or a global domain. When pervasive technology is applied to a local domain we refer to this as a *smart space*. This is a bounded physical environment filled with adaptive devices (such as lights, window shutters, etc.) that can be automatically managed to meet the needs of individual users. In a global domain the goal is to provide mobile users with devices, networks and services to meet their needs wherever they may be. This is often provided by some telecommunications provider allowing the mobile user to access and use services on the provider's network wherever they are.

These two research tracks have tended to remain independent of one another, resulting in islands of pervasiveness separated by voids in which the support for pervasiveness is limited. For example, in a local domain users will not experience pervasive behaviour besides smart spaces and in a global domain pervasive services provided are often restricted to some provider with the result that the user cannot make use of other services in their environment. The PSS approach integrates these two by unifying local, or fixed, smart spaces (associated with buildings) and global pervasiveness, mediated via mobile ad hoc networks (associated with users).

A PSS is based on a personal area network constructed from a variety of devices ranging from static resources (e.g. printer) to smaller mobile and wearable devices to minute devices such as smart dust. It can be *mobile* in which case its physical boundary moves with the PSS owner (who may be a person or legal entity) or *fixed* in that it is implemented within a static structure. Figure 26 shows examples of what a mobile PSS associated with a person might look like and what a fixed PSS implemented inside an office building might look like.

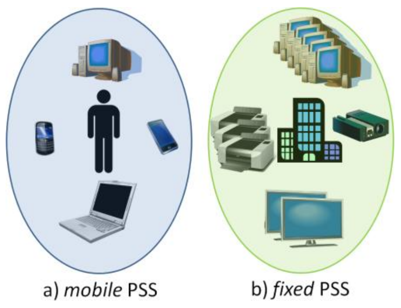


Figure 26. Example PSSs. a) A mobile PSS, typically associated with an individual; b) A fixed PSS, implemented inside a static structure.

One key feature of the PSS is its ability to interoperate with other PSSs utilising ad-hoc network technology. This allows one PSS to share information and services with other PSSs. Each PSS broadcasts it existence to the world so that other PSSs can see it. The broadcast messages also include advertisements for services that the PSS owner has decided to share with others.

6.3.2 The PSS Platform

To make a networked device *PSS enabled*, it is simply a matter of downloading and running an instance of the PSS platform in the device. PSS enabled devices can then be linked to other PSS enabled devices (i.e. added to an existing PSS) or the device can constitute a new PSS. The PSS platform developed within the PERSIST project has a layered architecture with each layer providing various functionalities to enable pervasive behaviour. Figure 27 shows a high-level view of the PSS Platform architecture.

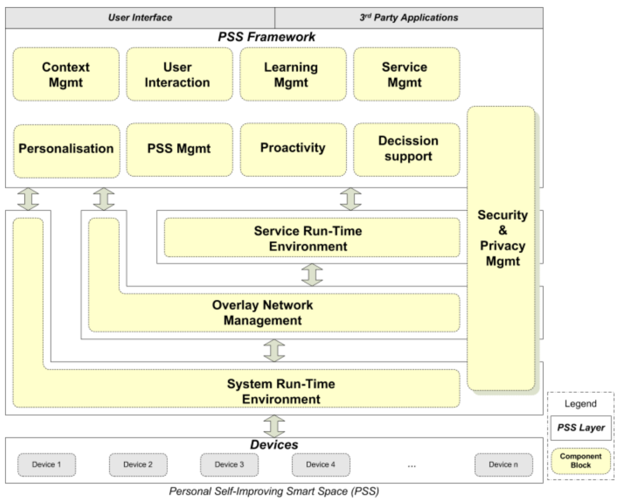


Figure 27. High-level view of the PSS Platform layered architecture.

The lower layers of the platform manage ad-hoc networking, PSS advertisements, inter/intra PSS messaging and service runtime. The upper layer provides intelligence and self-improvement within the PSS, enabling it to take decisions on behalf of the user and learn from previous behaviours as well as interactions with other PSSs. Privacy and security is paramount within the PSS architecture and is provided across all layers. The platform provides an administration GUI through which the PSS owner can manage the behaviour of the platform, the devices within their PSS and the sharing and consuming of services. The platform also supports the development of pervasive third party services which can run on top of the platform and utilise the functionality of the components within.

6.3.3 DIANNE Implementation with the PSS Platform

The DIANNE is implemented within the Learning Management system in the upper layer of the PSS platform. In terms of inputs, context updates are received from the Context Management system and user actions are received from the User Interaction system. In terms of outputs, all DIANNE output is forwarded to the Proactivity component which handles the application of all behaviours based on platform intelligence, such as user preferences and user intent models.

When implemented within the PSS, the DIANNE topology and algorithm does not change from that outlined in previous sections. However, in some cases it is necessary to adhere to PSS concepts. Significantly, within the PSS platform the DIANNE does not communicate directly with services. Instead all communications between the DIANNE and services are wrapped by other components of the platform. The reasons for this are two-fold. Firstly, the service input required by the DIANNE is also required by other learning processes within the platform. Hence service input is processed by the User Interaction system and then disseminated to the appropriate processes throughout the platform. Secondly, the PSS platform adopts a proactivity policy that all intelligent systems adhere to. Where possible, the PSS platform implements behaviours proactively on behalf of the user but final control is always passed to the user when required. This policy is realised by the functionality within the proactivity system. As well as receiving input from the DIANNE, the proactivity system also receives input from a Preference Management subsystem and a User Intent subsystem. The Preference Management subsystem creates and maintains user preferences using a batch decision tree learning algorithm (provided by the Learning Management system). The User Intent subsystem creates and maintains user intent models using a pattern discovery algorithm (also provided by the Learning Management system).

All three intelligent sources forward all outputs to the proactivity system where any conflicts are resolved. Following this, accepted behaviours are applied proactively using the appropriate user control/feedback mechanism. User control mechanisms range from requiring no user input, to requiring input only if the user does not approve of the behaviour, to requiring explicit user input before the behaviour is implemented. All feedback is returned to the intelligent sources allowing them to accommodate the information into their internal knowledge.

As well as proactive behaviour, at all times the privacy of the user is paramount as illustrated by the vertical Security & Privacy Management system that spans all layers of the architecture. Another major policy adopted within the PSS is that of *multiple identities*. This allows the user to masquerade behind a number of different identities with each one potentially disclosing different information about the user.

For example, the user may have one identity which they use to interact with a highly trusted banking service. This identity may contain sensitive personal information such as transaction details, credit card numbers, etc. The user may also have another identity which they use to interact with a less trusted restaurant finder service. This identity may only contain information such as favourite food, current city, etc. It is important that the information under the two identities is managed and disclosed correctly.

This includes behaviours and preferences that may be learnt when the user is assuming one of the identities. If a preference for adult bars is identified under the more sensitive identity (based on transaction information), this preference should not be available to the restaurant finder service used under another identity. Therefore, what the DIANNE learns under the one identity should remain under that identity alone and not be transferable to another identity. To ensure this is the case, additional functionality is required to make the DIANNE identity aware.

To respect the multiple identities concept a DIANNE manager is implemented to hold a one-to-one mapping between DIANNEs and identities, meaning that there is one DIANNE for every identity of the user. The DIANNE Manager listens for all context updates and user actions and directs them to the appropriate DIANNE depending on the identity that the context update or user action originated from. This allows all DIANNE learnt preferences to remain completely identity dependent and stops transfer of learnt behaviours across multiple identities.

6.3.4 DIANNE Demonstration

During the PERSIST project final review (in October 2010), proactivity was demonstrated based on each of the three input sources mentioned above (Preference Management, User Intent and the DIANNE). The DIANNE was demonstrated in a Disaster Management scenario driven by German project partners from the Deutsches Zentrum für Luft und Raumfahrt (DLR).

The scenario described a disaster situation where relief workers were using PSSs to orchestrate aid and rescue operations. Over time the PSSs self-improve by learning the behaviours and needs of the relief workers. In one scenario scene the PSS must learn what views and information a relief worker would like to share with his colleagues. The demonstration of this scene took place in real-time utilising two PSSs, one for the relief worker and one for his colleague.

The demonstration successfully showed how the DIANNE could incrementally learn user behaviours in real-time from initial system usage. Through the course of the demonstration the DIANNE accurately learnt what views and information to share with the colleague. It was also possible to show how the DIANNE could pick up new behaviours in real-time should the relief worker change the views and information they preferred to share.

The success of the DIANNE within the PSS platform has guaranteed its adoption within the recently started EU FP7 SOCIETIES project [143] which will utilise and build upon many of the successful aspects of the PSS. The latest open source version of the PSS platform including the DIANNE is available to download from the PERSIST Sourceforge website [144].

6.4 Summary

Several other incremental learning algorithms employ single layer neural networks. Their linear structure and lack of hidden layers reduces complexity allowing local updates to be performed in an incremental fashion without the possibility of affecting global network stability. The DIANNE topology includes several novel features including a dynamic squashing function but its main defining feature is the use of a temporal learning algorithm that iterates continuously allowing the network to learn associations based on the duration of vector state co-occurrences.

This enables the DIANNE to overcome several of the sampling issues outlined in Chapter 4 some of which are specific to the problem domain of preference learning in a pervasive environment. The DIANNE temporal algorithm includes two major steps. Firstly, the DIANNE is updated to reflect any environmental changes that have occurred since the last iteration. Secondly the DIANNE updates synaptic weights to learn associations between current context and outcome vectors.

However, it is possible that what the DIANNE believes to be the most preferred preference outcome is not in line with what the user has actually implemented. In such cases conflicts are resolved by reducing network error. The DIANNE temporal learning policy is not sufficient for this purpose as it could potentially take an unacceptably long period of time to reduce errors sufficiently. Therefore an error driven learning policy is utilised.

The DIANNE error driven learning policy is based on the heuristics proposed in section 4.8 but on further investigation, the final equation for DIANNE error reduction matches the Stochastic/Incremental Gradient Descent rule used for error reduction within other incremental neural networks such as WINNOW or the Pocket Algorithm. However, such algorithms are often solely error driven, differing from the DIANNE where two learning policies are utilised. The DIANNE utilises the Hebbian (and Anti Hebbian) learning rule for temporally driven learning and also utilises the Stochastic Gradient Descent rule for error driven learning.

An implementation of the DIANNE has been successfully integrated into the PSS platform within the EU Framework 7 PERSIST project. The DIANNE topology and algorithm remained consistent and did not require adjustment within this real world pervasive architecture but additional management functionality was required to respect PSS concepts such as multiple identities.

The PSS platform was presented and demonstrated at the final project review in October 2010 during which the DIANNE took a lead role in a disaster management demonstration driven by German partners from DLR. The DIANNE was successfully showcased and the demonstration illustrated how the DIANNE could identify user preferences from initial system usage and also adapt preferences in real-time as user behaviour changed. The success of the DIANNE implementation within PERSIST has ensured its adoption in the EU FP7 SOCIETIES project.

7 DIANNE Evaluation and Testing

7.1 Introduction

The aim of the testing and evaluation process was to analyse the DIANNE both in terms of performance (specifically accuracy) as well as the DIANNEs utility as a preference learning tool for use in pervasive environments. Therefore the testing and evaluation process was divided into two parts; benchmark testing and user evaluations.

Firstly the DIANNE was applied to several commonly cited datasets. The goal of this process was to determine DIANNE accuracy over such datasets and investigate how additional training data affects accuracy figures. The benchmark datasets chosen for DIANNE testing have been used to evaluate many algorithms (both batch and incremental) in the past and hence it was possible to draw comparisons with DIANNE accuracy figures.

Secondly the DIANNE was evaluated in live user trials conducted within a pervasive environment. The goal of this process was to analyse DIANNE performance as a preference learner in the pervasive personalisation domain. The user trials provided real time, temporal inputs which were not available from existing datasets. In this domain, the DIANNE's temporal reinforcement algorithm and conflict resolution mechanisms (for handling concept drift) were fully tested as was the ability of the DIANNE to drive personalised adaptations, based on continuously changing internal knowledge in a real world pervasive environment.

Another key goal of the user trials was to gather subjective views of DIANNE performance from end users. In a user centric domain the accuracy of the learner is only one of several key factors for acceptance alongside others such as processing and storage speeds, recall speed, learning rate and concept drift response rate. All factors were implicitly evaluated through the user's subjective view of their trial experience.

7.2 DIANNE Benchmark Testing

7.2.1 Datasets

All datasets for the benchmark testing process were sourced from the UCI Machine Learning Repository [145]. An initial search was performed for classification datasets with integer or categorical attribute and class values. From the results, six classification datasets were selected due to their frequent citations and previous usage to evaluate learning algorithms. This allows us to compare the DIANNE's performance with these other algorithms. The selected datasets are:

- Breast Cancer (CANCER) - Provided by the University Medical Centre, Ljubljana, Slovenia for the problem of predicting the reoccurrence of breast cancer five years after the removal of a tumour.
- Breast Cancer Wisconsin (CANCERW) - Provided by the University of Wisconsin Hospitals for the problem of predicting whether a lump is cancerous.
- Congressional Voting (VOTE) - Provided by the Congressional Quarterly Almanac for the problem of predicting whether a member of Congress will vote democrat or republican.
- Lymphography (LYMPH) - Provided by the University Medical Centre, Ljubljana, Slovenia for the problem of determining the type of cancer in lymphography.
- Primary Tumor (TUMOUR) - Provided by the University Medical Centre, Ljubljana, Slovenia for the problem of locating the primary tumour in patients with metastases.
- SPECT Heart (HEART)- Provided by the Medical College of Ohio for the problem of diagnosing cardiac SPECT images.

Table 3 gives an overview of the main characteristics of each dataset.

Dataset Name	Completeness	# Instances	# Attributes	# Class Values	Entropy
CANCER	Incomplete	286	9	2	0.73
CANCERW	Complete	699	10	2	0.93
VOTE	Incomplete	435	16	2	0.96
LYMPH	Complete	148	18	4	1.28
TUMOUR	Incomplete	339	17	22	3.89
HEART	Complete	267	22	2	0.73

Table 3. Datasets used for DIANNE performance testing

A mixture of complete and incomplete datasets were used to identify how the DIANNE performs under each. The datasets also vary in size ranging from 148 instances to 699 with differing numbers of attributes. Most of the datasets have a low number of class values with the exception of the TUMOUR dataset with 22 class values. The entropy of each dataset was also calculated to observe how the DIANNE handled datasets with higher entropy.

7.2.2 Evaluation Harness

A test harness was created to control the DIANNE performance tests. It interacts with the DIANNE providing inputs and collecting outputs for comparison as illustrated in Figure 28.

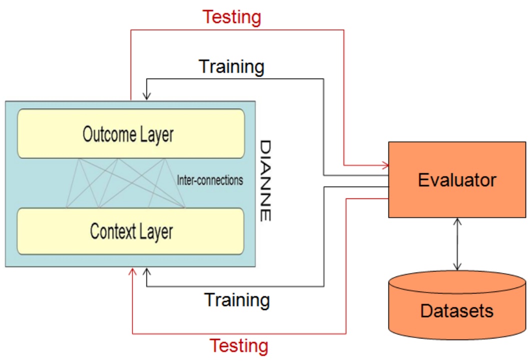


Figure 28. DIANNE performance test harness

The datasets were stored as individual scripts which were then read into the Evaluator one at a time. When a dataset was fed into the Evaluator it was divided into training and testing subsets. The training set was then fed into the DIANNE one instance at a time with the attributes feeding into the context layer and the class value feeding into the outcome layer. The DIANNE processed each instance and updated internal knowledge accordingly.

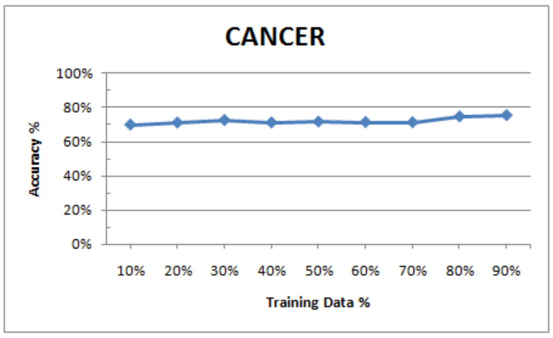
Once all the training instances were fed into the DIANNE, the Evaluator began to feed the testing set into the DIANNE one instance at a time. The attributes were fed into the context layer and based on this new vector the DIANNE returned a class value as output to the Evaluator. The Evaluator then checked the output against the correct class value for that instance and kept a tally of the number of correct and incorrect outputs received from the DIANNE. This gave a percentage accuracy over the testing set.

The first variable that was defined was the way in which the dataset was split into training and testing proportions. The most common proportions are a 70/30 split of training to testing data [140, 141, 142, 89] although 80/20 splits [149] and 67/33 splits [150] have also been used. However, for a more complete evaluation it was useful to investigate how DIANNE performance improved as the proportion of training data increased. Indeed this was the approach adopted by Syed et al [151] when evaluating an incremental SVM algorithm.

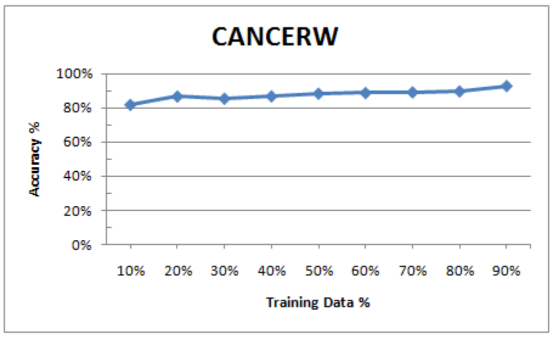
For DIANNE testing, the training data proportion was initialised at 10% and increased in increments of 10% to a maximum of 90%. For each training data proportion a test was repeated ten times. Firstly, the training subset was randomly selected from the dataset leaving the rest as the testing subset. Secondly, the training and testing subsets were applied to the DIANNE to give an accuracy value. The ten accuracy values from the ten tests are then averaged to give an average accuracy for the DIANNE on each training data proportion. The DIANNE was reset between each test for every data proportion so that each test (and its result) was independent of any other.

7.2.3 Results and Discussion

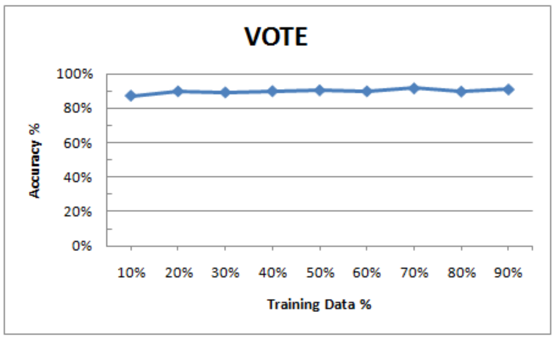
The DIANNE accuracy results for each dataset are illustrated in graphs (a) - (f) in Figure 29.



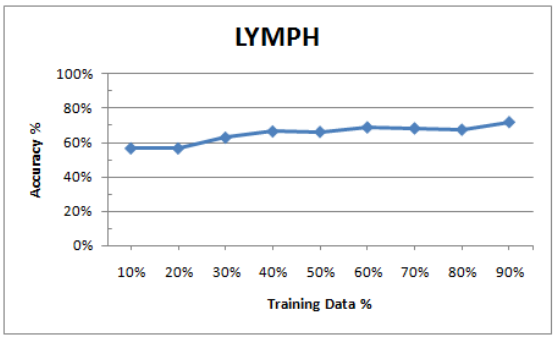
(a)



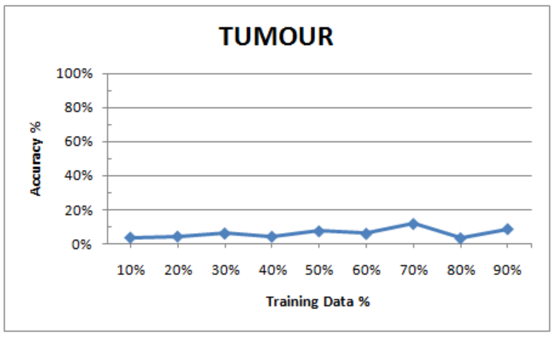
(b)



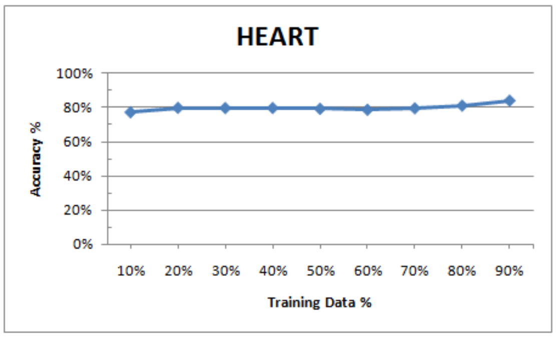
(c)



(d)



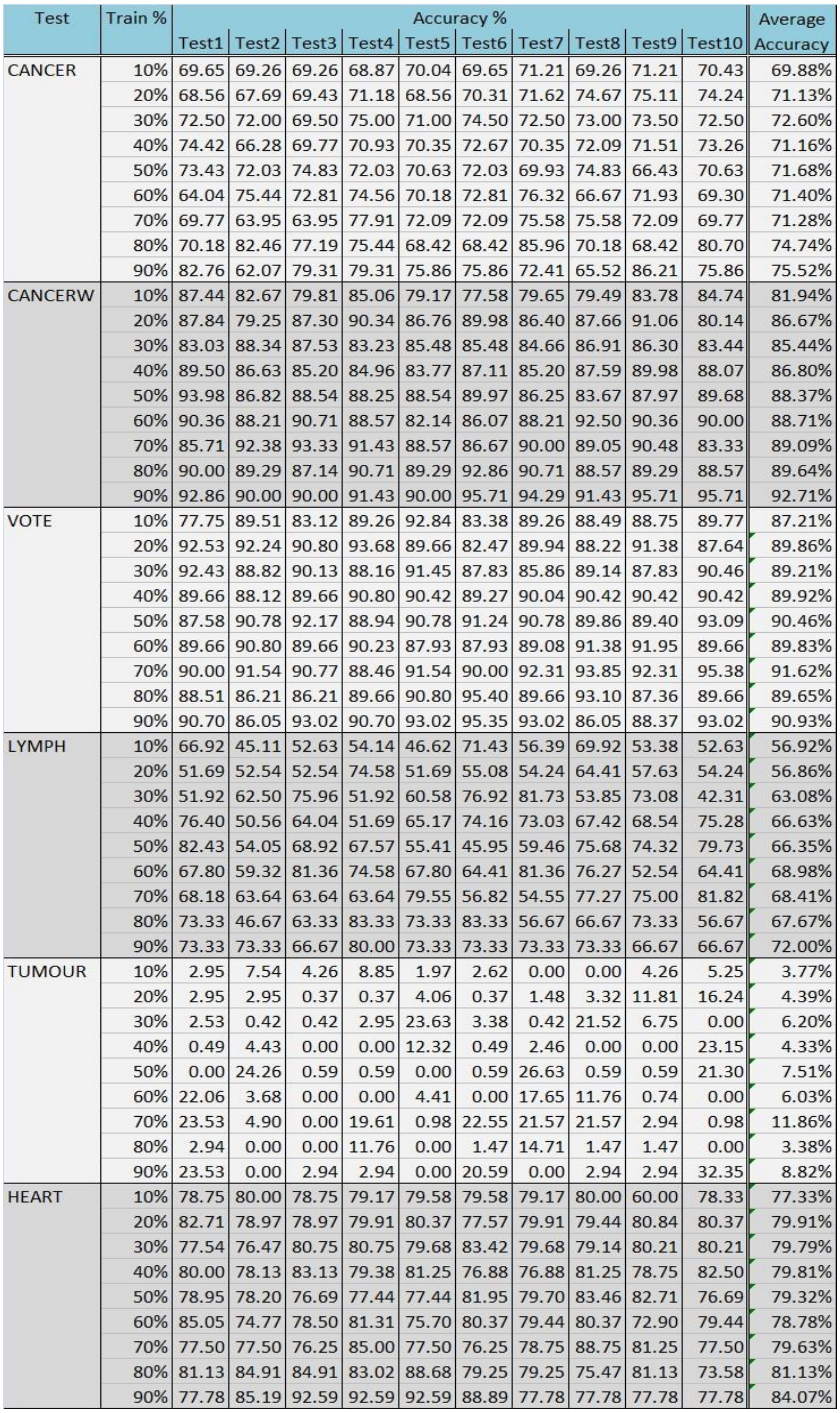
(e)



(f)

Figure 29. Graphs illustrating DIANNE accuracy on datasets as the proportion of training data increases

One of the most notable outcomes is that all graphs present relatively straight lines which are not representative of the curves that one would expect to see due to improvement with increased training data. In almost all cases the improvement in accuracy from 10% training data to 90% training data is less than 10%. Initial assumptions about the cause of such results were towards an error in the testing procedure such as insufficient resetting processes between tests, enabling tests on minimal training data to benefit from latent network connections left behind after tests on maximal training data. However, this does not appear to be the case. Table 4 below shows the results from all individual tests on all data proportions.



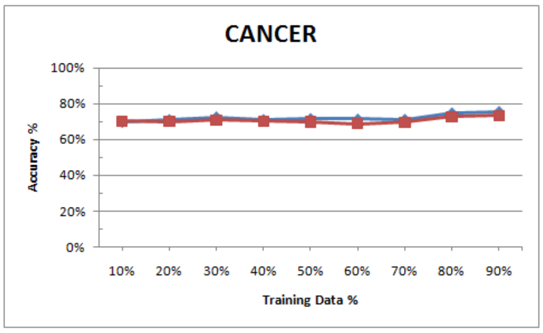
Test	Train %	Accuracy %										Average
		Test1	Test2	Test3	Test4	Test5	Test6	Test7	Test8	Test9	Test10	Accuracy
CANCER	10%	69.65	69.26	69.26	68.87	70.04	69.65	71.21	69.26	71.21	70.43	69.88%
	20%	68.56	67.69	69.43	71.18	68.56	70.31	71.62	74.67	75.11	74.24	71.13%
	30%	72.50	72.00	69.50	75.00	71.00	74.50	72.50	73.00	73.50	72.50	72.60%
	40%	74.42	66.28	69.77	70.93	70.35	72.67	70.35	72.09	71.51	73.26	71.16%
	50%	73.43	72.03	74.83	72.03	70.63	72.03	69.93	74.83	66.43	70.63	71.68%
	60%	64.04	75.44	72.81	74.56	70.18	72.81	76.32	66.67	71.93	69.30	71.40%
	70%	69.77	63.95	63.95	77.91	72.09	72.09	75.58	75.58	72.09	69.77	71.28%
	80%	70.18	82.46	77.19	75.44	68.42	68.42	85.96	70.18	68.42	80.70	74.74%
	90%	82.76	62.07	79.31	79.31	75.86	75.86	72.41	65.52	86.21	75.86	75.52%
CANCERW	10%	87.44	82.67	79.81	85.06	79.17	77.58	79.65	79.49	83.78	84.74	81.94%
	20%	87.84	79.25	87.30	90.34	86.76	89.98	86.40	87.66	91.06	80.14	86.67%
	30%	83.03	88.34	87.53	83.23	85.48	85.48	84.66	86.91	86.30	83.44	85.44%
	40%	89.50	86.63	85.20	84.96	83.77	87.11	85.20	87.59	89.98	88.07	86.80%
	50%	93.98	86.82	88.54	88.25	88.54	89.97	86.25	83.67	87.97	89.68	88.37%
	60%	90.36	88.21	90.71	88.57	82.14	86.07	88.21	92.50	90.36	90.00	88.71%
	70%	85.71	92.38	93.33	91.43	88.57	86.67	90.00	89.05	90.48	83.33	89.09%
	80%	90.00	89.29	87.14	90.71	89.29	92.86	90.71	88.57	89.29	88.57	89.64%
	90%	92.86	90.00	90.00	91.43	90.00	95.71	94.29	91.43	95.71	95.71	92.71%
VOTE	10%	77.75	89.51	83.12	89.26	92.84	83.38	89.26	88.49	88.75	89.77	87.21%
	20%	92.53	92.24	90.80	93.68	89.66	82.47	89.94	88.22	91.38	87.64	89.86%
	30%	92.43	88.82	90.13	88.16	91.45	87.83	85.86	89.14	87.83	90.46	89.21%
	40%	89.66	88.12	89.66	90.80	90.42	89.27	90.04	90.42	90.42	90.42	89.92%
	50%	87.58	90.78	92.17	88.94	90.78	91.24	90.78	89.86	89.40	93.09	90.46%
	60%	89.66	90.80	89.66	90.23	87.93	87.93	89.08	91.38	91.95	89.66	89.83%
	70%	90.00	91.54	90.77	88.46	91.54	90.00	92.31	93.85	92.31	95.38	91.62%
	80%	88.51	86.21	86.21	89.66	90.80	95.40	89.66	93.10	87.36	89.66	89.65%
	90%	90.70	86.05	93.02	90.70	93.02	95.35	93.02	86.05	88.37	93.02	90.93%
LYMPH	10%	66.92	45.11	52.63	54.14	46.62	71.43	56.39	69.92	53.38	52.63	56.92%
	20%	51.69	52.54	52.54	74.58	51.69	55.08	54.24	64.41	57.63	54.24	56.86%
	30%	51.92	62.50	75.96	51.92	60.58	76.92	81.73	53.85	73.08	42.31	63.08%
	40%	76.40	50.56	64.04	51.69	65.17	74.16	73.03	67.42	68.54	75.28	66.63%
	50%	82.43	54.05	68.92	67.57	55.41	45.95	59.46	75.68	74.32	79.73	66.35%
	60%	67.80	59.32	81.36	74.58	67.80	64.41	81.36	76.27	52.54	64.41	68.98%
	70%	68.18	63.64	63.64	63.64	79.55	56.82	54.55	77.27	75.00	81.82	68.41%
	80%	73.33	46.67	63.33	83.33	73.33	83.33	56.67	66.67	73.33	56.67	67.67%
	90%	73.33	73.33	66.67	80.00	73.33	73.33	73.33	73.33	66.67	66.67	72.00%
TUMOUR	10%	2.95	7.54	4.26	8.85	1.97	2.62	0.00	0.00	4.26	5.25	3.77%
	20%	2.95	2.95	0.37	0.37	4.06	0.37	1.48	3.32	11.81	16.24	4.39%
	30%	2.53	0.42	0.42	2.95	23.63	3.38	0.42	21.52	6.75	0.00	6.20%
	40%	0.49	4.43	0.00	0.00	12.32	0.49	2.46	0.00	0.00	23.15	4.33%
	50%	0.00	24.26	0.59	0.59	0.00	0.59	26.63	0.59	0.59	21.30	7.51%
	60%	22.06	3.68	0.00	0.00	4.41	0.00	17.65	11.76	0.74	0.00	6.03%
	70%	23.53	4.90	0.00	19.61	0.98	22.55	21.57	21.57	2.94	0.98	11.86%
	80%	2.94	0.00	0.00	11.76	0.00	1.47	14.71	1.47	1.47	0.00	3.38%
	90%	23.53	0.00	2.94	2.94	0.00	20.59	0.00	2.94	2.94	32.35	8.82%
HEART	10%	78.75	80.00	78.75	79.17	79.58	79.58	79.17	80.00	60.00	78.33	77.33%
	20%	82.71	78.97	78.97	79.91	80.37	77.57	79.91	79.44	80.84	80.37	79.91%
	30%	77.54	76.47	80.75	80.75	79.68	83.42	79.68	79.14	80.21	80.21	79.79%
	40%	80.00	78.13	83.13	79.38	81.25	76.88	76.88	81.25	78.75	82.50	79.81%
	50%	78.95	78.20	76.69	77.44	77.44	81.95	79.70	83.46	82.71	76.69	79.32%
	60%	85.05	74.77	78.50	81.31	75.70	80.37	79.44	80.37	72.90	79.44	78.78%
	70%	77.50	77.50	76.25	85.00	77.50	76.25	78.75	88.75	81.25	77.50	79.63%
	80%	81.13	84.91	84.91	83.02	88.68	79.25	79.25	75.47	81.13	73.58	81.13%
	90%	77.78	85.19	92.59	92.59	92.59	88.89	77.78	77.78	77.78	77.78	84.07%

Table 4. Individual results for each of the ten tests per training proportion on all six datasets.

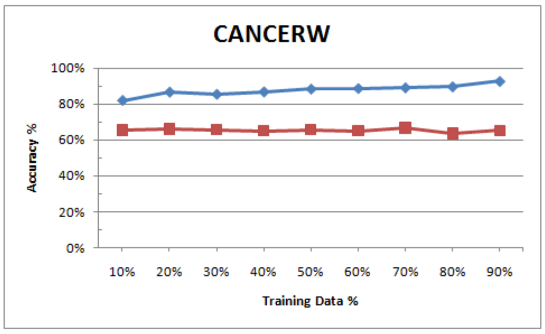
The table shows that in some cases there is much variation between the individual test results for each training data proportion. This indicates that the DIANNE was properly reset between each test as there are no inexplicable patterns in the results table. It should also be noted that for each dataset the testing proportions began at 10% increasing to 90% so there was no possibility that the tests on 10% training data could have benefitted from network connections that had not been properly reset after tests on 90% training data.

After ensuring that no errors had been made in the testing procedure further investigations were carried out to identify an explanation. This began by looking at the results of other researchers who have performed similar evaluations, using equivalent datasets, with incrementing training set sizes. Graphs presented by Syed et. al [151] and Ratanamahatana et. al [149] convey learning curves but in both works the graphical scale used to present the results is zoomed into a range that accentuates the slight curves. When viewed at a graphical scale of 0% to 100% their results reflect the relatively straight lines shown in the graphs above. Michalski [95] also makes reference to strong patterns within the CANCER and LYMPH datasets, used to evaluate his AQ15 algorithm.

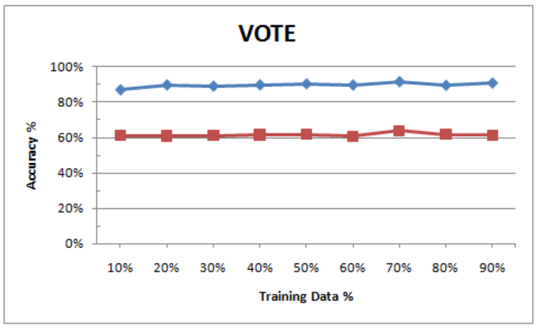
Considering that the CANCER, CANCERW, VOTE and HEART datasets only have two possible class values it may not be so surprising that such high accuracy figures are achievable with minimal training data. Randomly selecting a classification for each testing instance could potentially give around 50% accuracy. Graphs (a) - (f) in Figure 30 compare the accuracies achieved by applying a simple default rule (shown in red) with the DIANNE accuracies (shown in blue). The default rule uses the most common class value from the training subset as the predicted class value for all testing instances regardless of their attributes.



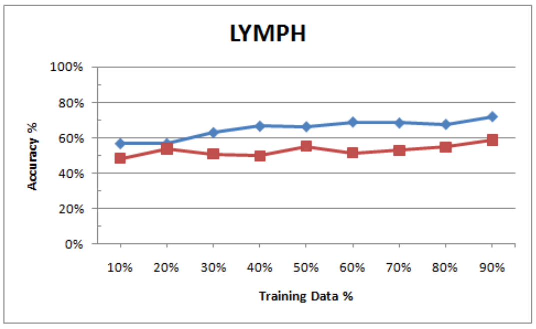
(a)



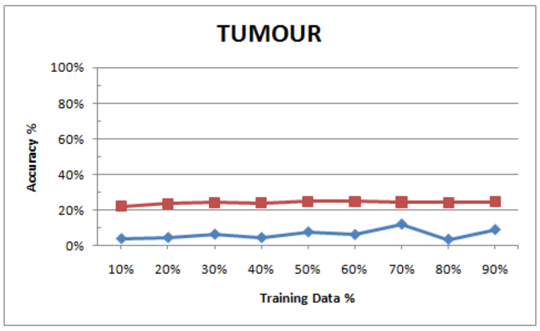
(b)



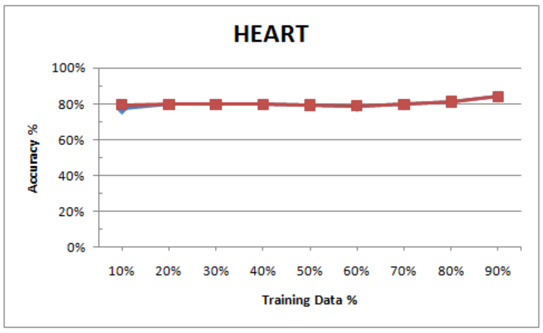
(c)



(d)



(e)



(f)

Figure 30. Graphs illustrating the accuracies of the DIANNE (blue) and the default rule (red) over the datasets

The acceptable results achieved by the default rule on most of the datasets illustrates that in many cases a small percentage of the dataset is representative of the entire dataset. This also appears to be the case for the LYMPH dataset with four possible class values. Indeed, when the C45 algorithm was run on the LYMPH dataset with increasing training set sizes from 10% to 90% a similarly straight line was achieved with accuracies between 60% and 75% (see Annex A). However, accuracy drops significantly for the TUMOUR dataset with a total of twenty two class values. Graph (e) shows that the default rule achieves low accuracies of just above 20% on the TUMOUR dataset; however, these accuracies are higher than the accuracies achieved by the DIANNE. Indeed, the DIANNE achieves high accuracy values on all datasets apart from TUMOUR.

Looking more closely at the TUMOUR dataset several factors are identified as probable causes for the poor accuracies achieved. Firstly, the TUMOUR dataset appears to be relatively noisy. This suggestion is reinforced by considering historic accuracy figures obtained from the UCI repository. Other algorithms have typically performed unsatisfactorily on this dataset achieving accuracies ranging from 29% to 51%. Human experts only achieve an accuracy of 42%. Michalski also shows that there are very few strong patterns in this dataset by illustrating that decision rule complexes typically only cover two instances from the dataset.

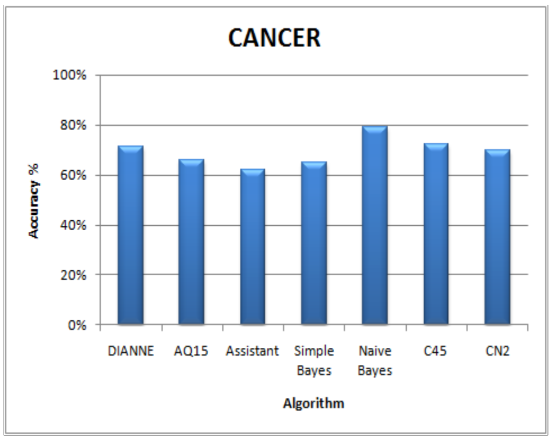
Secondly, there is a significantly large number of class values (22 in all) in the TUMOUR dataset compared to the other datasets. Since DIANNE training data is not processed or re-ordered in any way prior to the training phase it cannot be guaranteed that instances related to each of the class values are in the training dataset. In the situation where a class isn't represented in the training dataset the DIANNE will not represent such instances in the testing phase and therefore cannot provide the correct classification of such instances. Batch algorithms can improve performance to some degree when confronted by such datasets as it is possible to be selective over training data. However, since the DIANNE is designed as an incremental network this pre-processing step is not an option.

Thirdly, the combination of a noisy dataset and a large number of class values can also potentially cause issues when adding new nodes into the DIANNE during the training phase. In an unstable outcome node group where the winner node regularly changes the potentials of all nodes will tend towards -1. In such a situation, if a new outcome node is introduced with an initial potential of zero, this new outcome will over-ride all others for a period of time during which incorrect classifications are likely. The factors leading to this situation are strongly apparent in the TUMOUR dataset.

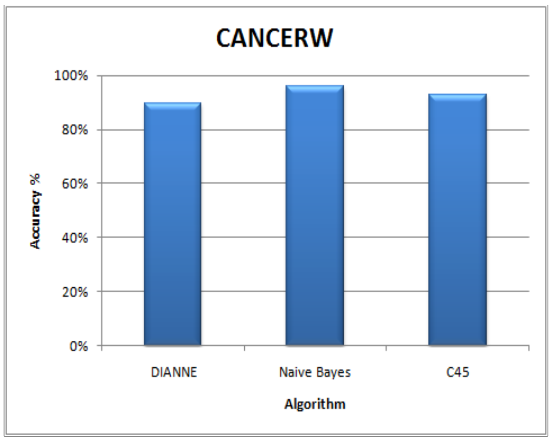
Benchmark Comprisons

As well as investigating the DIANNEs performance in terms of accuracy across a variety of different datasets, further testing was also performed to compare DIANNE accuracy against the accuracy of other machine learning algorithms (both batch and incremental) on the benchmark datasets. Fortunately, many of the datasets in the UCI repository contain details of past usage, citing several algorithms that have been evaluated with the dataset and the accuracy they achieved.

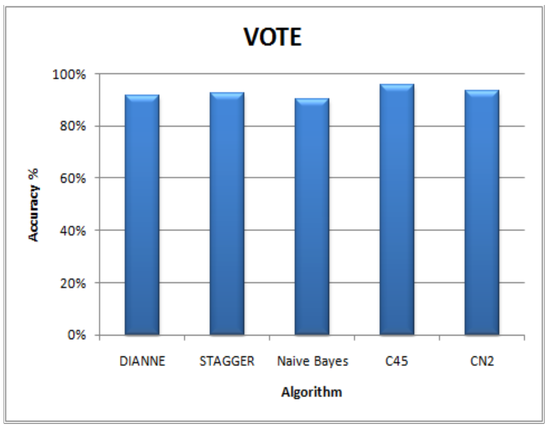
Since many of the algorithms are evaluated with a set proportion of training and testing data (i.e. no increases are made to the training dataset proportion), comparisons between the DIANNE and each algorithm were made with equivalent training/testing data proportions. Figure 31 shows the comparison graphs illustrating the accuracies of various algorithms (where an accuracy figure could be obtained) on each dataset.



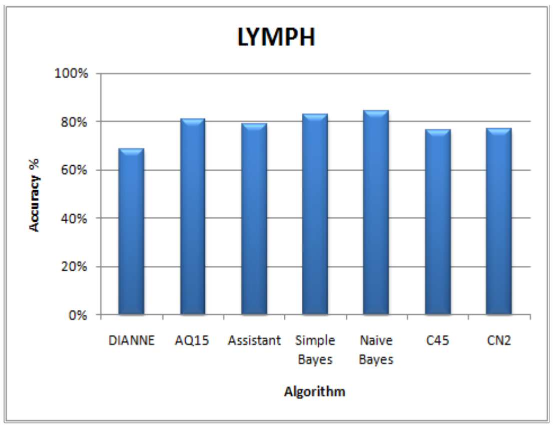
(a)



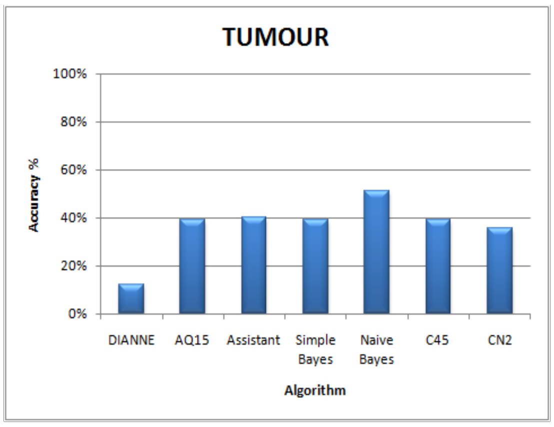
(b)

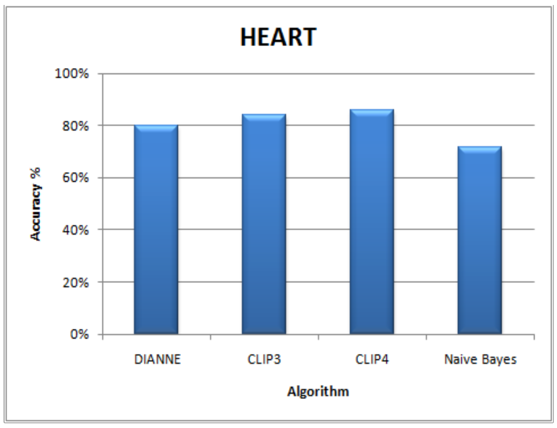


(c)



(d)





(e) (f)

Figure 31. Comparison graphs illustrating the accuracies of various algorithms on the datasets

Accuracy values could not be obtained for all the algorithms on all the datasets. The graphs above compare the DIANNE against algorithms for which accuracy values were available on that particular dataset. As can be seen, many well recognised machine learning algorithms, both batch and incremental in nature are available for comparison. A primary observation is that all algorithms (for which results were obtainable) showed unsatisfactory results when applied to the TUMOUR dataset. However, the results in graph (e) show that DIANNE accuracy on this dataset is much lower than that achieved by other algorithms.

Notably, over the other five datasets the DIANNE achieves accuracy figures as good, if not better, than other algorithms. Compared to batch algorithms, the DIANNE performs comparably with C45 and outperforms CN2, Simple Bayes and Assistant on the CANCER dataset. The Naive Bayes algorithm is outperformed on the HEART and VOTE datasets. These are encouraging results since the DIANNE does not have a priori knowledge of the entire dataset and cannot re-process past training data.

Compared to incremental algorithms, the DIANNE outperforms AQ15 on the CANCER dataset and achieves accuracies comparable to that of the STAGGER algorithm on the VOTE dataset. Encouragingly the graphs show that the DIANNE is able to compete with current algorithms (both incremental and batch) in most instances.

7.3 User Trials

In addition to the benchmark tests, DIANNE performance was also evaluated in a real world pervasive environment with end users. This was important for several reasons.

Firstly, the datasets used for the benchmark tests do not include any temporal information and hence the DIANNE could not fully perform as intended. Although the results achieved on benchmark datasets are encouraging it was important to investigate how the DIANNE incrementally learns on temporal data. No appropriate temporal datasets were found to exist and therefore it was necessary to create them by capturing behavioural and contextual inputs in real-time during a trial situation with an end user.

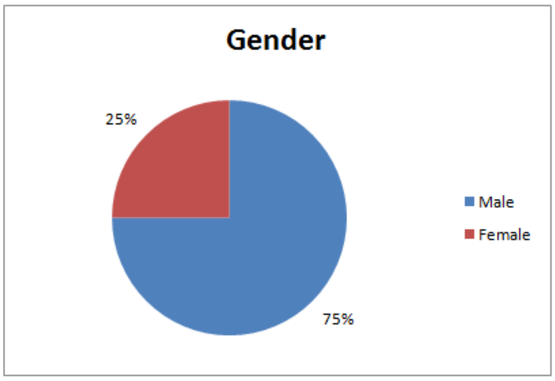
Secondly, the pervasive personalisation domain for which the DIANNE is intended is user centric. Hence it was necessary to obtain feedback on the user experience of system behaviour driven by DIANNE learning. Most significantly, it was important to understand from a user perspective, how appropriately the DIANNE learnt preferences based on user behaviours and how appropriately the DIANNE was perceived to adapt learnt preferences when the user changed their behaviour.

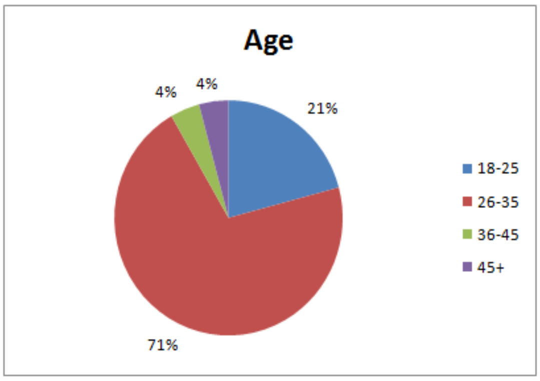
The user trials were based on a personalised television experience where the DIANNE would incrementally learn viewing preferences and apply them to drive personalised adaptations. Trial participants were asked to choose channels to watch on various screens placed in different locations around a building. The learning challenge for the DIANNE was to incrementally associate location context input with channel selection input to identify what channel the trial participant preferred to watch on which screen. Based on this information the DIANNE could drive personalised adaptations to show the correct channel to the user on each screen.

During the trial, participants were also asked to reconsider their channel selections allowing them to change their viewing behaviours if desired. The learning challenge for the DIANNE then became one of incrementally adapting internal knowledge to appropriately learn any new over-riding behaviours (i.e. incrementally handle concept drift).

7.3.1 User Group

The user trials took place over a two week period from Monday 11th April 2011 until Friday 22nd April 2011. A total of 24 people took part in the trials. Figure 32 illustrates the gender and age demographics of the user trial group. (Note: percentages are rounded to the nearest whole number).

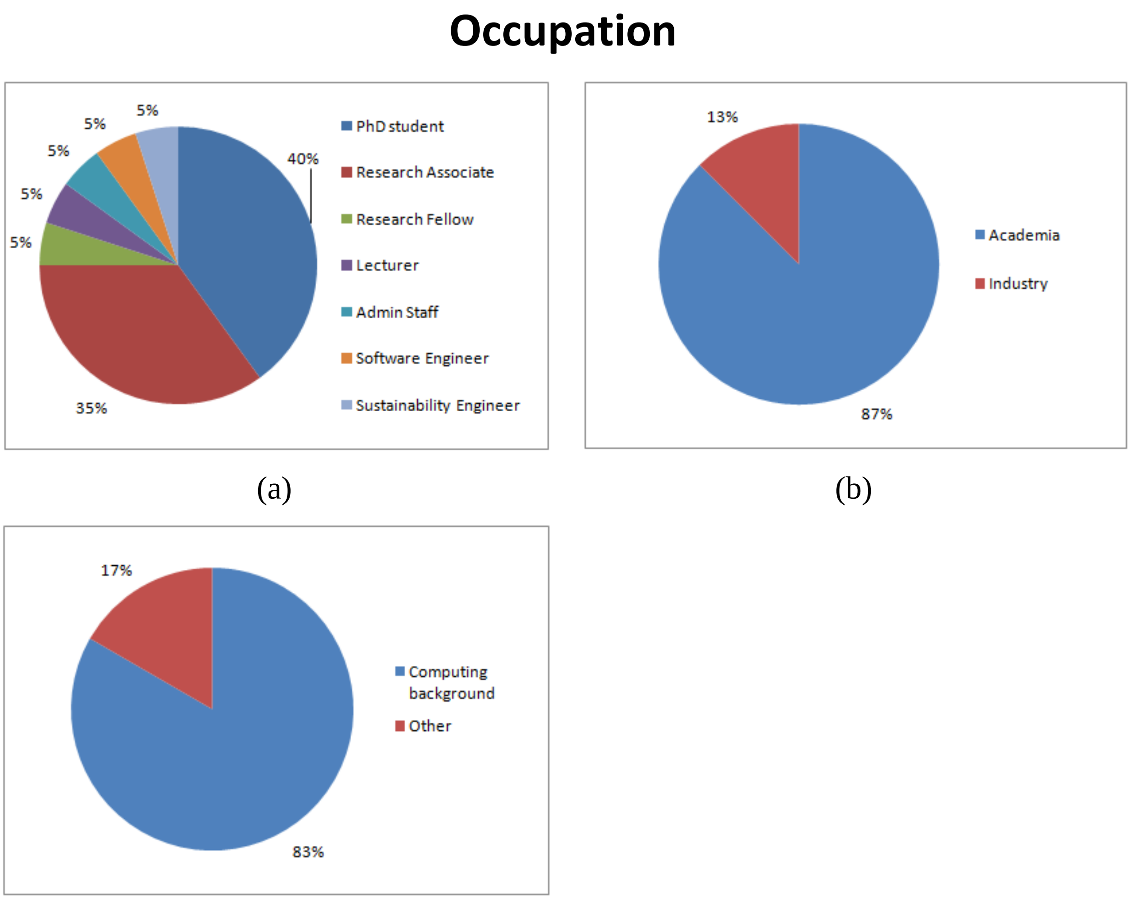




(a) (b)

Figure 32. Pie charts illustrating the (a) Gender and (b) Age ratios of user trial participants

The majority of the participants were male and aged between 26 and 35. Figure 33 illustrates the occupation of the participants from several viewpoints.



(a) (b)

(c)

Figure 33. Pie charts illustrating the (a) occupation, (b) academic background and (c) the computing background of the user trial participants

The most common occupation among trial subjects was PhD student, closely followed by research associate. Indeed most of the trial participants were related to academia in some way with a small minority from an industrial background. We can also see that the majority of trial subjects have an occupation related to computer science. These demographics reflect the population of individuals who received an invitation to take part. Due to accessibility constraints, the majority of invitation emails were sent to colleagues and students within the School of Mathematical and Computer Sciences (MACS) at Heriot-Watt University.

7.3.2 The Test Environment

A test environment was set up within the MACS department at Heriot-Watt University. Figure 34 shows the network diagram of the test environment.

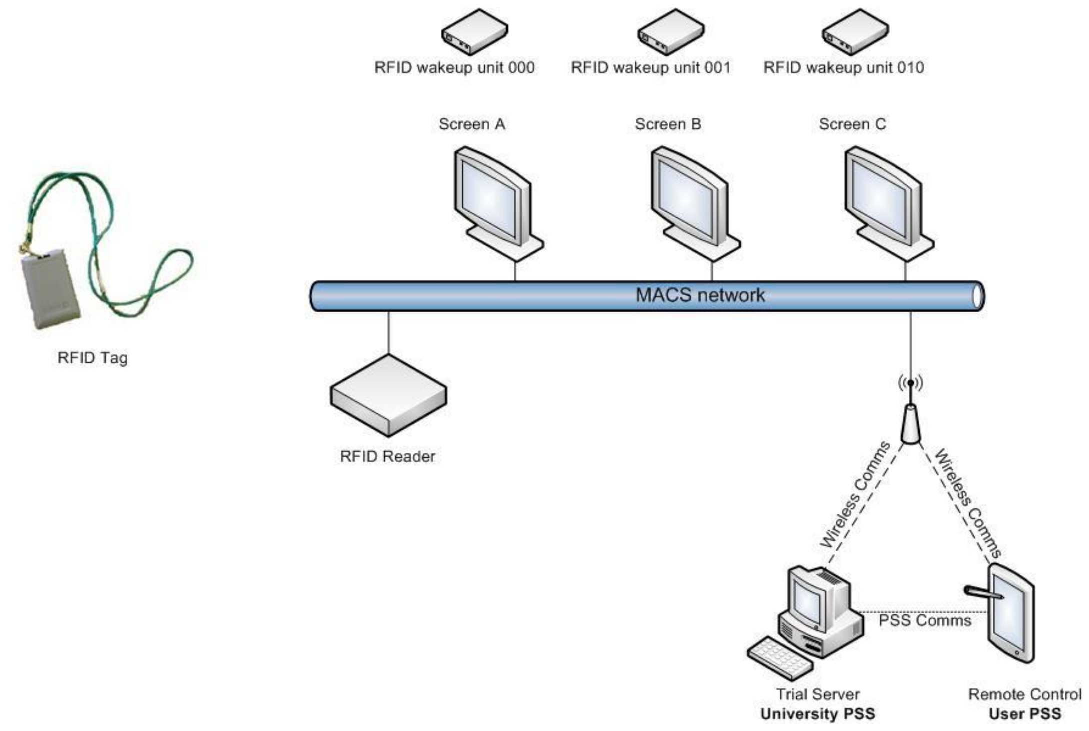


Figure 34. Test environment network diagram.

Screens and Content

Three screens, A, B and C (each attached to a PC), were positioned at different locations in the MACS department building and connected to the MACS ethernet network. Each screen could display the same three channels and a default channel. Channel 1 showed several University promotional videos. Channel 2 showed a number of 3D animations that were the results of student coursework. Channel 3 showed several research project videos. All three channels played on loop so the trial participant did not watch each channel from the beginning every time it was selected. The default channel was a splash page which was displayed by a screen when no other channels were selected.

The Trial Server

The trial server was a PC connected to the MACS network via a wireless access point. It ran the PSS (Personal Smart Space) platform developed in the PERSIST project and acted as the 'University PSS'. The PSS platform provided service sharing, context management and pro-activity features; all of which are required for this trial. The DIANNE is also fully integrated into the PSS platform. The University PSS used the PSS platform's service sharing mechanisms to provide University related services for other PSSs to use. One such service was the 'DisplayScreen Service' which gave users control over the three screens. The service communicated directly with each screen via sockets sending channel selection commands. These commands were received and implemented by a socket listener running on each screen. The University PSS advertised itself throughout the MACS network so other PSSs could discover it, interact with it and use its services.

The Remote Control

The remote control was a wireless tablet device illustrated in Figure 35.

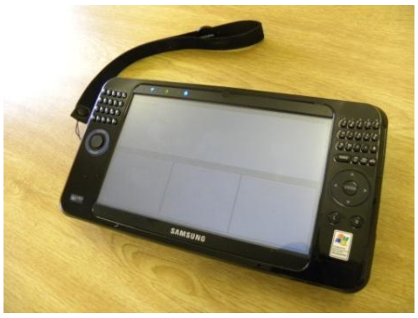


Figure 35. Remote Control device

It also ran the PSS platform and acted as the 'User PSS' of the trial participant. The User PSS advertised itself through the MACS network via the wireless access point so other PSSs could discover it and interact with it. The University PSS and the User PSS discovered each other and the DisplayScreen service (hosted by the University PSS) was executed on the User PSS. The User PSS (and the trial participant using this device) could now control the display screens.

The DisplayScreen service was context-dependent, only allowing control over the screen that the trial participant is close to, if any. In other words, the trial participant could only control a screen if they were beside it. If the trial participant was not beside any of the screens then they could not control any of the screens. The DisplayScreen service provided a GUI that displayed location information to the trial participant indicating if they were beside (and hence had control of) any of the screens.

Indoor Location using RFID Technology

An RFID system was utilised to provide the indoor location of the trial participant. The system consisted of several elements. Three RFID wakeup units were positioned, one at each screen. This created a 'hotspot' with a 5 metre radius around each screen. The trial participant was given an RFID tag to wear around their neck during the trial. When the trial participant entered a hotspot, the RFID wakeup unit instructed the RFID tag to begin broadcasting its location to the RFID reader (connected directly to the MACS network). On receiving location updates from a tag, the RFID reader forwarded this information through the MACS network to the remote control device (and the DisplayScreen service running there) so it knew what screen, if any, to control.

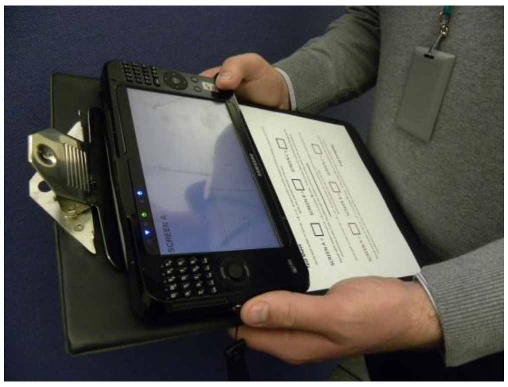
The DIANNE in the Test Environment

The DIANNE is integrated into the PSS platform hence DIANNE learning was available on the trial server and the remote control device. For these trials it was only utilised on the remote control device where the participant interacted with the DisplayScreen service and where the RFID reader sent location updates. The DisplayScreen service provided outcome updates that are processed by the DIANNE's outcome layer in real time. The RFID reader provides location updates to the PSS Context Management system on the remote control device. These updates are then propagated to the DIANNE where they are processed in real time by the DIANNE's context layer. In this way the DIANNE could perform incremental learning based on live, temporal inputs. During the trials, all other learning and intelligent behaviour in the PSS platform was disabled to ensure adaptations were only driven by the DIANNE.

7.3.3 Trial Format

Throughout the user evaluation process care was taken to reduce bias and ensure that all trials were equivalent. The following steps were followed for each trial:

1. When the participant arrived for the trial they were taken to an introduction area where they were seated. They were given an information sheet (see Annex B) to read which outlined the format of the trial and then introduced the participant to important aspects such as the screens, their locations, the different channels and the remote control device. Once the participant had read the information sheet the tester answered any questions.
2. The participant was given an RFID tag to wear around their neck. They were also given a clipboard with a trial sheet (see Annex C) and the remote control device so they could control the screens (Figure 36 (a)). The tester shadowed the participant during the trial to give ensure the trial format was adhered to, discretely note observations, answer any questions and handle any technical issues if they occurred (Figure 36 (b)). The tester did not interfere with the participant in any way during the course of the trial and apart from answering questions or solving technical issues, tester/participant interactions were kept to a minimum of scripted instructions.


(a)

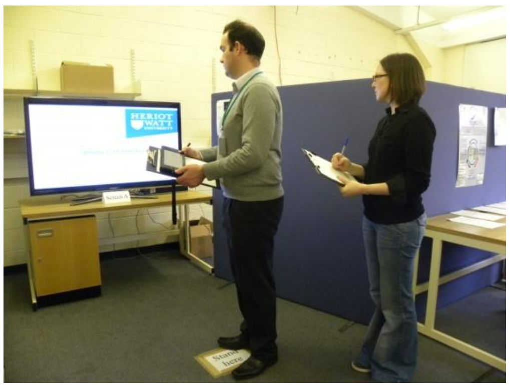

(b)

Figure 36. Images of (a) the participant's trial equipment and (b) the participant and shadowing tester during a trial

3. The participant was instructed to visit each screen in turn and pick a channel for each screen. The participant was made aware that channel selection was completely at their discretion. They were also made aware that how they go about selecting a channel at each screen was also up to them; they could browse the channels or simply select one and move on.
4. ***Primary Selection Circuit*** - The participant visited each screen, selected a channel and wrote the channel number in a box beside the screen name on their trial sheet.

5. ***Primary Test Circuits (PT1, PT2, PT3)*** - The participant was then instructed to complete a further three circuits of the screens. During the circuits, if a screen didn't show the correct channel (i.e. the channel that the participant selected and wrote in the box beside that screen name) the participant had to correct the screen using the remote control to set the screen to the correct channel. If a screen did show the correct channel then the participant was not required to perform any action.
6. Once the primary selection and primary test circuits were complete the participant was instructed to visit each screen again in turn where they could revise their channel selection if desired. The participant was made aware that the decision of whether or not to revise their original channel selections was at their discretion. Channel selections were explicitly unscripted to encourage a wide range of behaviours across all trials, allowing for richer analysis of DIANNE performance.
7. ***Secondary Selection Circuit*** - The participant visited each screen, selected a channel (which may be the channel they originally selected for that screen) and wrote the channel number in a box beside the screen name on their trial sheet.
8. ***Secondary Test Circuits (ST1, ST2, ST3, ST4, ST5)*** - The participant was then instructed to complete a further five circuits of the screens (extra circuits were added to ensure the learning of new over-riding behaviours was observed). As with the primary test circuits (step 5), if the screen did not show the correct channel the participant had to the screen using the remote control. If the screen did show the correct channel no further action was required.
9. Once the secondary selection and test circuits were completed the tester retrieved the trial sheet, remote control and RFID tag from the participant.
10. The participant completed a short questionnaire (see Annex D) about their trial experience.

7.3.4 Generated Datasets

During each trial a number of datasets were generated for later analysis and processing. Table 5 describes the datasets that were generated during each trial.

Dataset	Description
Tester Observations (TO)	During the primary and secondary test circuits the tester notes what channel is automatically presented each time the participant comes into proximity of a screen and whether or not this is the correct channel.
Monitored Behaviour (MB)	All interactions between the participant and the DisplayScreen service are captured. When the participant interacts with the DisplayScreen service to select a channel on some screen the channel number is stored with the current location of the participant (provided by the RFID reader). This creates a list of channel-location pairs representing the channel selections made on each screen. This dataset is typical of the monitored behaviour histories gathered for preference learning in many pervasive systems.
Temporal Monitored Behaviour (TMB)	This dataset is an extension of the Monitored Behaviour dataset. Temporal information is included to represent the duration that each channel-location pair prevails. To achieve this the latest channel-location pair is duplicated in the dataset for every second that it prevails. Therefore this dataset represents the channel selections made on each screen and how long the participant watches the various channels on each screen.
Questionnaire Results (QR)	The questionnaire answers provided by each participant help to capture qualitative data regarding DIANNE learning and the subsequent driving of automatic adaptations.

Table 5. Generated Dataset Descriptions

7.3.5 Results and Discussion

Four datasets were collected during each trial so that different aspects of DIANNE learning could be analysed. Firstly the Tester Observation (TO) datasets were analysed to investigate how accurately and rapidly the DIANNE learnt channel preferences as well as how accurately and rapidly it updated internal knowledge to handle concept drift. Secondly the Monitored Behaviour (MB) and Temporal Monitored Behaviour (TMB) datasets were applied to the benchmark C45 tree building algorithm (utilised for preference learning in both the DAIDALOS and PERSIST projects) to compare the performance of a batch algorithm with that of the DIANNE in this problem domain. Thirdly the Questionnaire Results (QR) datasets were analysed to investigate the end user experience of adaptations driven by the DIANNE.

DIANNE Performance

The TO datasets were analysed to investigate the accuracy and learning rate of the DIANNE over the primary and secondary test circuits. At the beginning of each trial the DIANNE had no internal knowledge and was essentially starting from scratch. The DIANNE's internal knowledge was initially generated as the participant selected different channels and visited different screens on the primary selection circuit. The DIANNE continued to update and reinforce internal knowledge as the participant revisited, and where necessary corrected, the screens during the primary test circuits. On each primary test circuit the tester noted if the DIANNE drove the presentation of the correct channel to the participant at each screen. Looking at these figures over 24 trials we can identify a percentage accuracy for the DIANNE on each primary test circuit. Figure 37 illustrates that the DIANNE retains a high percentage accuracy over the three primary test circuits (PT1, PT2 and PT3) and also shows a slight increase in accuracy as the DIANNE improves internal knowledge over time with each circuit.

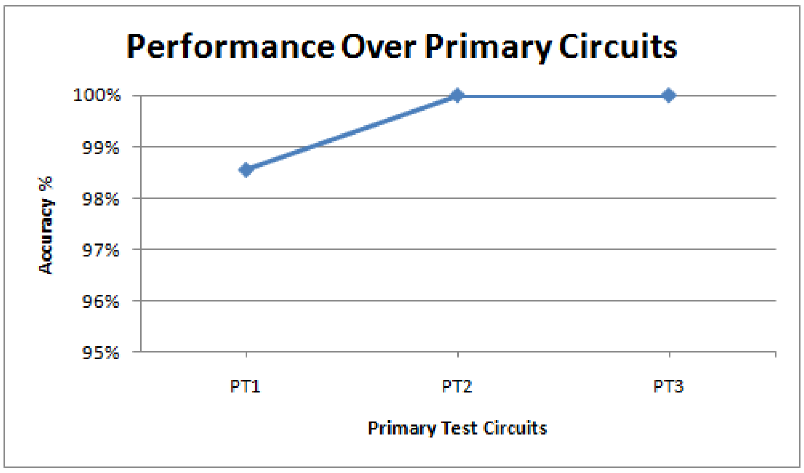


Figure 37. Graph illustrating the percentage accuracy of the DIANNE over the three primary test circuits

After the primary selection and test circuits the DIANNE had an internal knowledge that tended towards the initial viewing preferences of the participant. During the secondary selection circuit the participant could change their viewing preferences if they wished by selecting a different channel to view on a screen. The DIANNE updated internal knowledge as the participant made their secondary channel selections. Over the course of the five secondary test circuits (ST1, ST2, ST3, ST4 and ST5) the DIANNE continued to update and reinforce internal knowledge as the participant revisited, and where necessary corrected, the screens.

As before, during the secondary test circuits the tester noted if the DIANNE drove the presentation of the correct channel to the participant at each screen. Taking these figures over 24 trials we can identify a percentage accuracy for the DIANNE on each secondary test circuit. Figure 38 illustrates that DIANNE accuracy is below 10% on ST1 but increases rapidly to an accuracy above 95% by ST5.

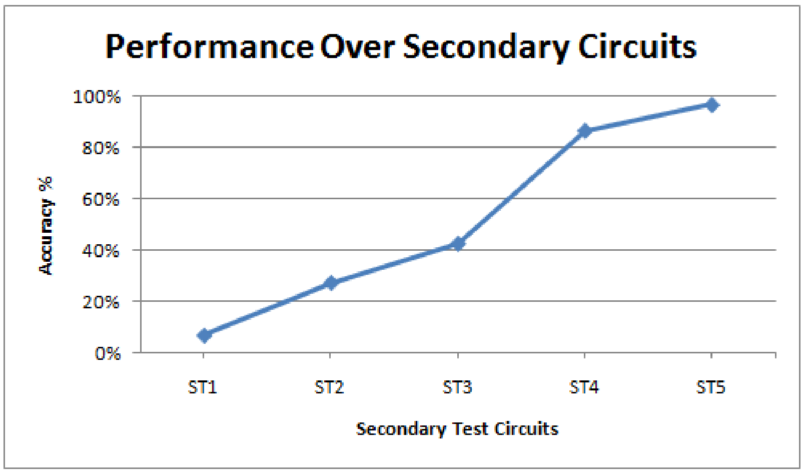


Figure 38. Graph illustrating the percentage accuracy of the DIANNE over the five secondary test circuits

Such a curve is what one would expect to see from an incremental learning system and shows that the DIANNE continued to incrementally update internal knowledge on each circuit, accommodating new behaviours to learn new, over-riding preferences. The graph shows that the accuracy on ST1 is not 0%. All instances where the participant did not change their preferred channel on the second selection circuit have been removed from the results. Therefore, this shows that for some participants the DIANNE has rapidly accommodated the new behaviours, learning over-riding preferences simply from the behaviours exhibited on the secondary selection circuit. Indeed the rate at which the DIANNE updates internal knowledge is heavily dependent on the behaviour of each individual. It is useful to investigate further how participant behaviour affects DIANNE preference learning and handling of concept drift.

During the selection circuits, each participant had flexibility in terms of how many times they could switch between channels and how long they could spend watching channels at each screen. This led to two distinct categories of participant; *browsers* and *non-browsers*. Browsers tended to spend more time viewing and switching between channels during the selection circuits. Non-browsers tended to select a channel based on a pre-meditated decision and hence did not spend much time viewing or switching between channels during the selection circuits.

This is relevant from a DIANNE perspective since channel switches translate to conflicts within the DIANNE invoking the conflict resolution strategy. Additionally long viewing times translate to greater temporal reinforcements while short viewing times translate to fewer temporal reinforcements. Therefore it is useful to investigate how the DIANNE handles learning and concept drift for the two different categories of user and hence under different conflict and temporal reinforcement conditions.

From the 24 trial participants, 16 have been identified as browsers and 8 as non-browsers. Figure 39 shows how the average number of channel switches made at each screen in the selection circuits differs between the two groups.

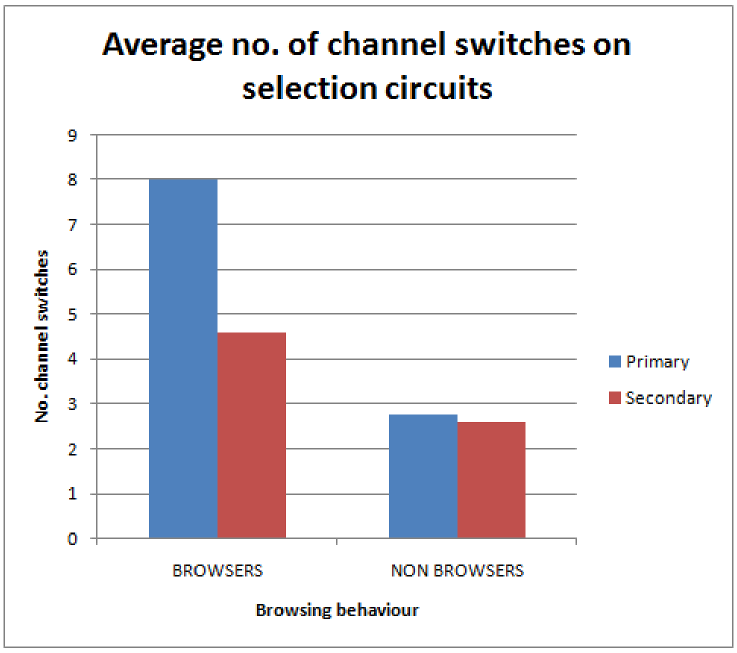


Figure 39. Graph illustrating the average number of channel switches made by browsers and non-browsers during the selection circuits

On the primary selection circuit, the browsers performed almost three times as many channel switches compared with non-browsers who performed less than three switches on the primary selection circuit (averaging less than one channel switch per screen). The trend continued on the secondary selection circuit with the browsers again performing significantly more channel switches than the non-browsers. Comparing primary and secondary circuits, the browsers performed nearly double the channel switches on their primary selection circuit compared to their secondary whereas the non-browsers performed almost exactly the same number of channel switches on their primary and secondary selection circuits.

Figure 40 shows the time that browsers and non-browsers spent viewing channels during the primary and secondary selection circuits.

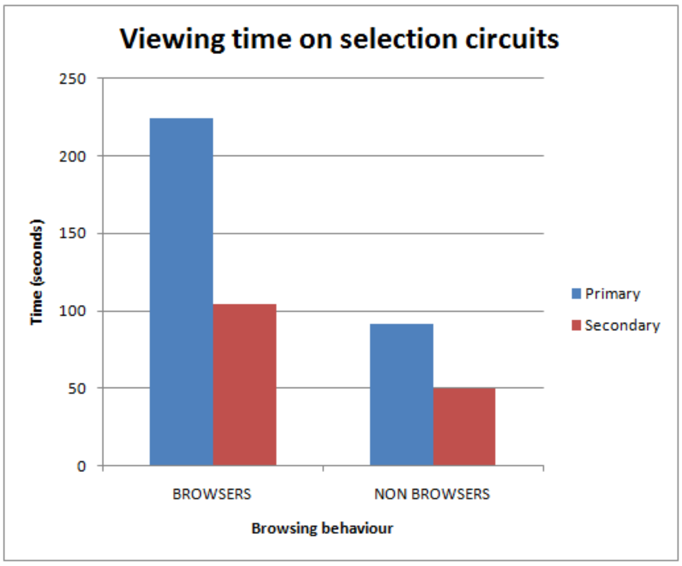


Figure 40. Graph illustrating the time that browsers and non-browsers spent viewing channels during the primary and secondary selection circuits

Browsers spent twice as long viewing channels as the non-browsers on the primary selection circuit. This trend continued in the secondary selection circuit with the browsers again spending twice as long viewing channels as the non-browsers. Looking at the difference between primary and secondary circuits, both browsers and non-browsers spent roughly twice as long viewing channels on their primary selection circuit compared with their secondary. However, looking at how much time browsers and non-browsers spent viewing channels on the primary and secondary *test* circuits the results are very different as illustrated in Figure 41.

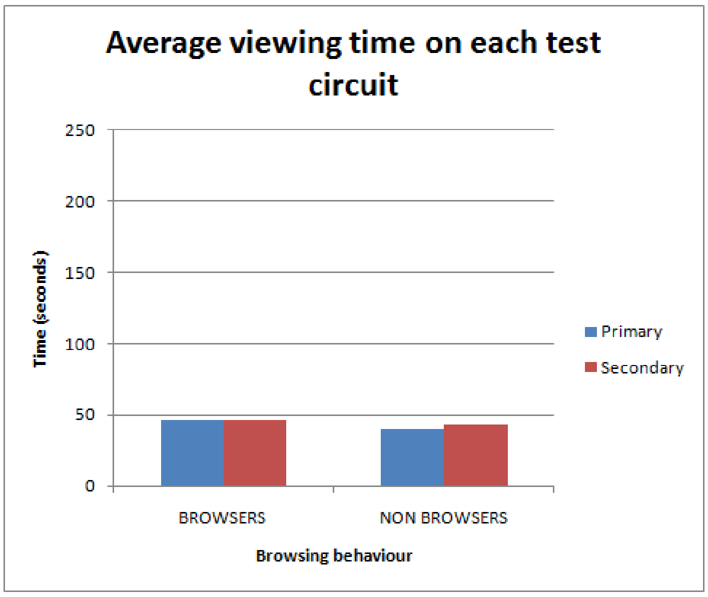
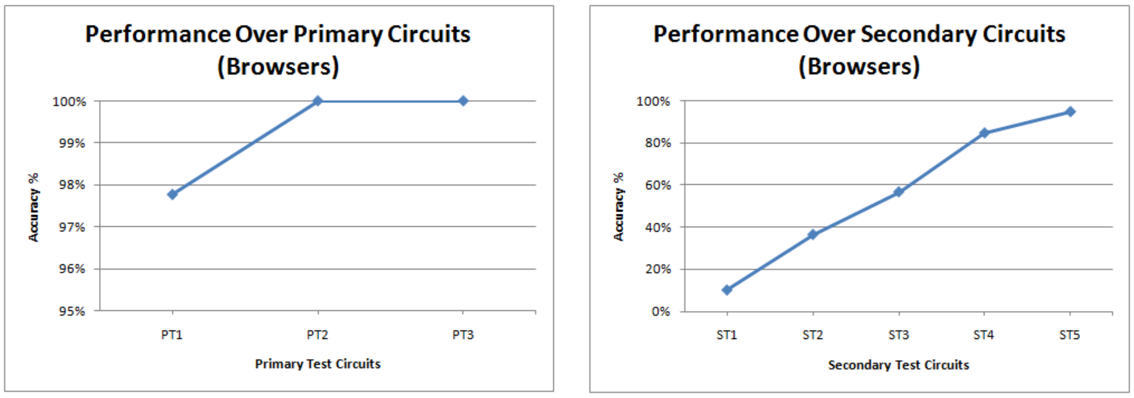


Figure 41. Graph illustrating the average time that browsers and non-browsers spent viewing channels on each primary and secondary test circuit

This graph shows that both browsers and non-browsers spent a comparable time viewing channels on each primary and secondary test circuit. For browsers, this was significantly less time than that spent viewing channels on the selection circuits. For non-browsers this is also less time but the difference is less significant. During the primary and secondary test circuits, typical participant behaviour was to approach a screen and immediately move away again after observing the automatic channel change and correcting it if necessary. Neither browsers nor non-browsers spent any significant time viewing channels during the primary and secondary test circuits. Therefore the main behavioural differences between the two groups were observed on the primary and secondary selection circuits.

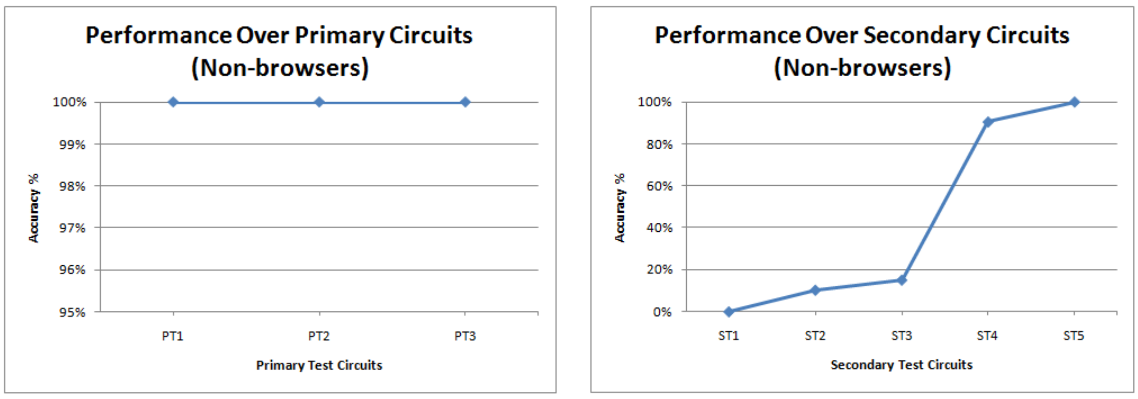
By splitting the Tester Observations (TO) datasets into two groups for browsers and non browsers we can compare DIANNE performance for the two groups. Figure 42 illustrates DIANNE accuracy over the primary and secondary test circuits for the sixteen browsers.



(a) (b)

Figure 42. Graphs illustrating DIANNE accuracy over (a) primary and (b) secondary test circuits for Browsers

Graph (a) shows that DIANNE accuracy is very high over the three primary test circuits marginally improving from around 98% to 100% as the circuits progress. Graph (b) shows that initial accuracy is around 10%, steadily increasing to around 97% during the course of the five secondary test circuits. Figure 43 illustrates DIANNE accuracy over the primary and secondary test circuits for the eight non-browsers.



(a) (b)

Figure 43. Graphs illustrating DIANNE performance over (a) primary and (b) secondary test circuits for Non-browsers

Graph (a) shows that for non-browsers, the DIANNE was 100% accurate over the three primary test circuits. Graph (b) shows that over the course of the secondary test circuits, DIANNE accuracy rose from 0% on ST1 to 100% on ST5 taking a significant jump from below 20% accuracy to above 90% accuracy between ST3 and ST4.

Comparing the two sets of graphs raises some interesting points. In comparing the two primary circuit graphs; Figure 42 (a) and Figure 43 (a), we can see that the DIANNE performs better for non-browsers than browsers, achieving 100% accuracy over all non-browser primary test circuits. This is likely because the DIANNE often faces a more complex learning challenge with browsers as described below.

During the primary selection circuit non-browsers tend to only watch one channel at each screen for a short duration. This leads to a simple network structure and a simpler association challenge for the DIANNE compared to browsers who tend to watch numerous channels at each screen for longer periods of time. By the end of the primary selection circuit browsers will often have larger, more complex networks than non-browsers and hence the association challenge for the DIANNE will be more complex.

In the situations where the DIANNE did not predict the correct channel during the primary test circuits it was usually the case that the participant had trouble choosing a preferred channel on the selection circuit, switching between channels numerous times and watching channels for comparable durations before finally selecting a preferred channel and immediately moving away from the screen. Based on conflicts and viewing durations the DIANNE occasionally predicted the unselected channel in error. However, this was quickly rectified on subsequent primary test circuits as the participant corrected the screen causing the DIANNE to perform conflict resolution and correct internal knowledge.

In conclusion one can say that increased channel switches and longer viewing times (exhibited by browsers) does not greatly affect the accuracy of the DIANNE when learning participant viewing preferences from scratch. In network terms one can conclude that the DIANNE remains stable and accurate when learning from scratch in situations with both numerous and few conflicts and both greater and fewer temporal reinforcements.

By comparing the two secondary circuit graphs; Figure 42 (b) and Figure 43 (b) we can see that DIANNE accuracy on ST1 is better for browsers (around 10%) than non-browsers (0%). In contrast at ST5 the DIANNE achieves a better accuracy for non-browsers (100%) than browsers (around 97%). Consider the accuracies at ST1 where for browsers this is around 10%. During the secondary selection circuit browsers often make numerous channel switches and watch channels for long periods of time. This combination of increased conflicts (caused by channel switches) and greater temporal reinforcements (caused by long viewing times) can lead to the over-riding of the old preferred channel with the new preferred channel during the secondary selection circuit. Hence in some situations the DIANNE can accurately predict the new preferred channel by ST1.

In contrast, non-browsers typically select one channel during the secondary selection circuit and immediately move away from the screen with little time spent viewing the newly selected channel. Therefore, with minimal conflicts and minimal temporal reinforcements it is very unlikely that the old preferred channel will be over-ridden by the new preferred channel during the secondary selection circuit. This is reflected in the 0% accuracy for non-browsers during ST1.

Now consider the accuracies at ST5 where for browsers this is around 97% compared with 100% for non-browsers. This discrepancy is minimal showing that by ST5 the DIANNE can handle concept drift equally well under situations with both numerous and few conflicts and both greater and fewer temporal reinforcements.

Looking closer at the learning curves in Figure 42 (b) and Figure 43 (b) one can see that the learning curve for the browser group is much smoother and more gradual than for the non-browser group where a significant leap between 20% and 90% accuracy occurs between ST3 and ST4. This leap is likely due to the uniform behaviour displayed by non-browsing participants. Since the majority of non-browsers only selected one channel during the selection circuits and spent minimal time viewing the channels the DIANNE learnt the new preferred channel at roughly the same rate for non-browsers (i.e. between ST3 and ST4). In contrast, the varied behaviours displayed by the browsers meant that the DIANNE learnt the new preferred channel at different rates, affected by the number of channel switches and the time spent viewing channels during the selection circuits.

DIANNE Comparison

The Monitored Behaviour (MB) datasets are a list of action-context pairs that can be applied to other learning algorithms to give a comparison with DIANNE performance. In this instance the C45 tree building algorithm has been chosen as the algorithm for comparison as it has been utilised for preference learning in several research projects such as DAIDALOS and PERSIST due to both its accuracy and tree based output which can be translated into human readable form.

At the end of each trial the DIANNE and C45 algorithms were tested to see if they could correctly predict the participant's secondary viewing preferences (i.e. the channel selections that the participant made during their secondary selection circuit). The MB dataset was applied to the C45 algorithm and from the tree based output an IF-THEN-ELSE preference rule was generated indicating a channel number for each location. This was compared with the participant's secondary viewing preferences to give an accuracy figure for the C45 algorithm. The preferences held in the DIANNE's final state were also compared with the participant's secondary viewing preferences to give a final accuracy figure for the DIANNE.

In addition the Temporal Monitored Behaviour (TMB) dataset was also applied to the C45 algorithm at the end of each trial. The TMB dataset includes extra context-action pairs extending the MB dataset with temporal information. The TMB dataset replicates DIANNE input allowing the C45 algorithm to take advantage of extra environmental data. The reason for this additional dataset is to ensure that the C45 algorithm is not hampered by less inputs or environmental updates than the DIANNE. As with the MB dataset, the TMB dataset was applied to the C45 algorithm and from the tree based output an IF-THEN-ELSE preference rule was generated indicating a channel number for each location. This was compared with the participant's secondary viewing preferences to give an accuracy figure for the C45 algorithm operating on a temporal dataset. Taking the accuracies of the DIANNE, C45(MB) and C45(TMB) over all 24 trials gives an average accuracy for each algorithm illustrated in Figure 44.

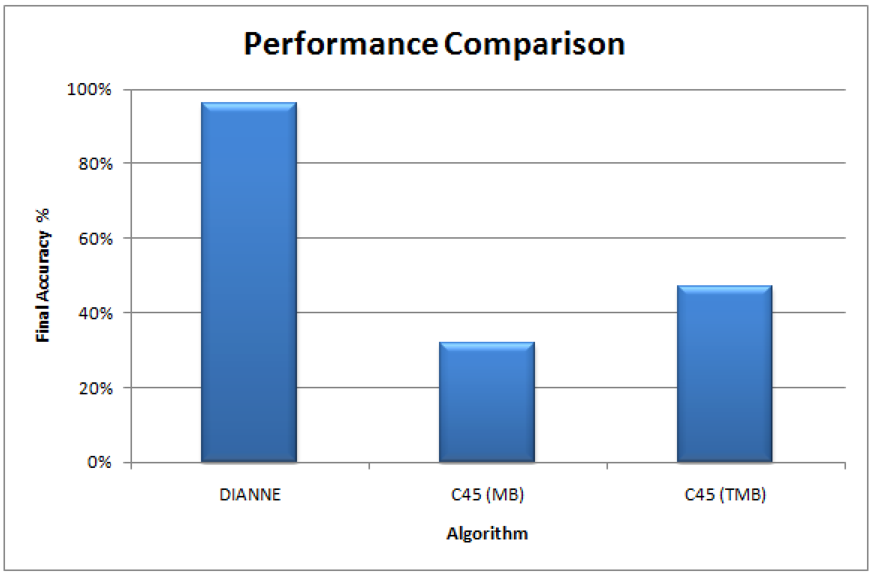


Figure 44. Graph illustrating the final accuracy of the DIANNE, the C45 algorithm on a non-temporal dataset (MB) and the C45 algorithm on a temporal dataset (TMB) over all 24 trials

The graph shows that the DIANNE is over three times more accurate at learning the participant's secondary viewing preferences as the C45 algorithm on the MB dataset. The graph also shows that there is some improvement in the accuracy of the C45 algorithm on the TMB dataset suggesting that extra environmental input capturing temporal information is of benefit. However, even with added temporal information the C45 algorithm still only achieves an accuracy of less than 50%, roughly half that of the DIANNE at 96%. In the case of the TMB datasets both algorithms have essentially received the same input except the DIANNE processes inputs in real time as they occur throughout the trials whereas the C45 algorithm processes all inputs in batch after the trial is completed.

Looking more closely at the IF-THEN-ELSE preferences generated by the C45 algorithm on the MB and TMB datasets it can be seen that they often portray the participant's primary viewing preferences (i.e. the channels they selected during the primary selection circuit). This is understandable since most participants spent significantly longer selecting channels on their first primary selection circuit than on their second, meaning that their primary channel selections often appear more frequently in the MB and TMB datasets.

It is understood that these test results do not provide a like for like comparison as the C45 algorithm is not designed for changing rules, but rather for static situations. Recent data is not weighed more heavily than less recent data. However, the MB and TMB datasets illustrate a typical preference learning situation. User preferences are not static and often change over time for various reasons. What this comparison confirms is that algorithms such as C45 are not best suited to the preference learning problem domain. In comparison the DIANNE can respond more rapidly to changes in behaviour (or concept drift in learning terms), returning to a high performance accuracy in an acceptable time frame. This reflects the findings of Segal et al when comparing incremental and batch learning techniques for Swiftfile [113].

The DIANNE can also match many non-network based algorithms in terms of its translatability into human understandable form. This is highly important in the pervasive domain where internal knowledge should be available for presentation to end users.

Questionnaire results

At the end of each trial the participant was asked to complete a short survey related to their trial experience. There are nine multiple choice questions in total, covering aspects such as automatic behaviours, DIANNE learning rates, user monitoring and prediction. The first two questions, presented in Figures 45 and 46, query user satisfaction and annoyance at correct and incorrect automatic behaviours driven by the DIANNE.

1. *On a scale of 1 to 5, how pleasing did you find it when the screens automatically changed to the correct channel?*

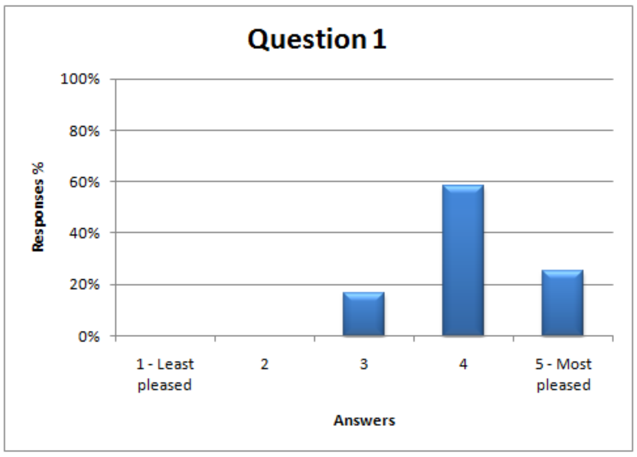


Figure 45. Question 1 text and responses

2. *On a scale of 1 to 5, how annoying did you find it when the screens automatically changed to the incorrect channel?*

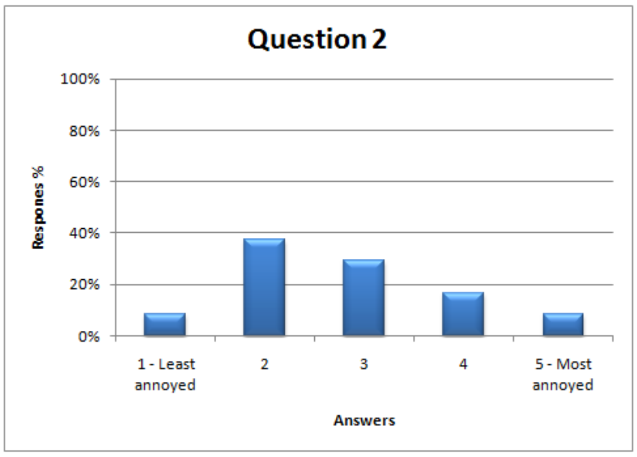


Figure 46. Question 2 text and responses

In this particular instance where the automatic behaviour corresponds to channel selection Figure 45 shows that the majority of participants were very pleased when the automatic behaviour was correct with no participants rating their pleasure at 2 or below. Additionally, Figure 46 shows that annoyance at incorrect automatic behaviours is relatively low with the majority of participants rating their annoyance in the middle of the scale at 2 or 3. Of course it can be argued that pleasure and annoyance ratings will depend on the type of automatic behaviour with some causing more pleasure/annoyance than others.

Identifying whether or not a participant would wish to have such technology in their own home is often a good test of its acceptability. This was queried in question 3 presented in Figure 47.

3. *Would you use such functionality in your own home if it were freely available?*

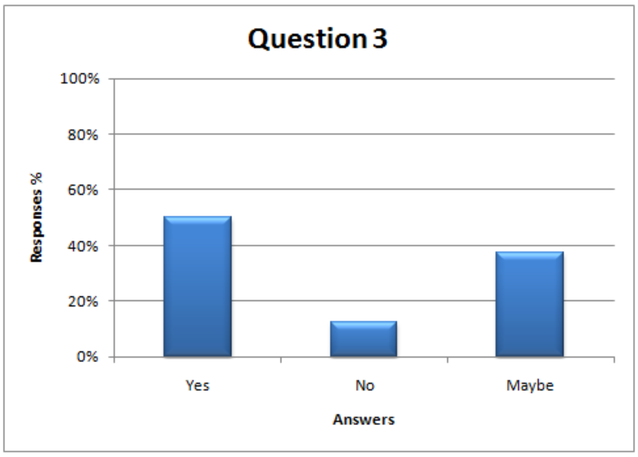


Figure 47. Question 3 text and responses

Only 12% of participants stated that they would not wish to have such technology in their own home. In most cases the reason was a dislike of not being in control. Some participants were also concerned about how multiple occupant situations would be handled. However, the significant majority of participants answered 'yes' or 'maybe' suggesting that for most participants such automatic behaviours do not provoke negative feelings.

The next question, presented in Figure 48, relates to the learning rate of the DIANNE throughout the trial. Since the DIANNE is not a single instance learner it is useful to investigate the expectations of participants regarding the rate at which their viewing preferences are learnt.

4. *Did you feel that the system learnt your channels too slowly?*

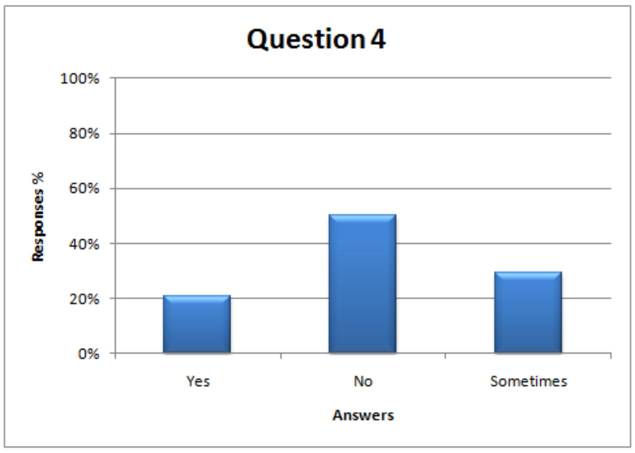


Figure 48. Question 4 text and responses

The graph shows that half of the participants do not regard DIANNE learning as too slow. 29% of participants found it to be too slow sometimes and only 21% of participants felt it was too slow throughout the trial. This suggests that from an end user perspective the DIANNE both learns preferences and handles concept drift within an acceptable time frame.

The next two questions, presented in Figures 49 and 50, relate to the monitoring of behaviour and the predicting of behaviour. They aim to identify how comfortable participants are with having their behaviour monitored and predicted throughout the trial.

5. *On a scale of 1 to 5, how comfortable did you feel about the system monitoring your behaviour during the trial?*

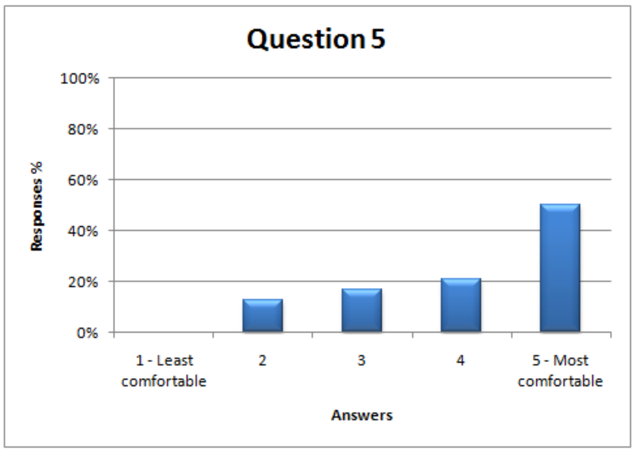


Figure 49. Question 5 text and responses

6. *On a scale of 1 to 5, how comfortable did you feel about the system predicting your behaviour during the trial?*

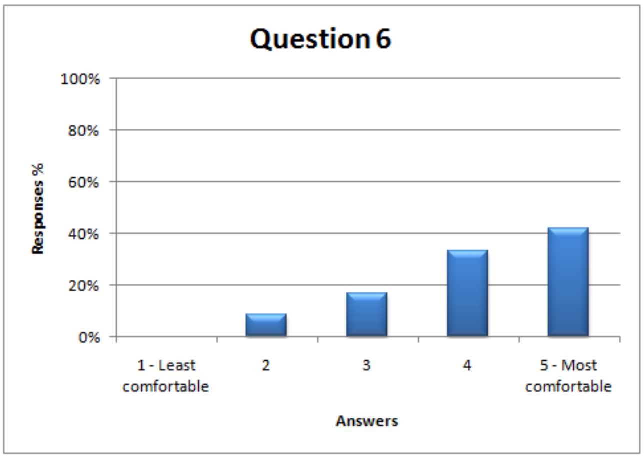


Figure 50. Question 6 text and responses

Both graphs in Figures 49 and 50 show that the majority of participants are very comfortable with the system monitoring and predicting their behaviour. 71% of participants rated their comfort level at 4 or above for behaviour monitoring and 75% of participants rated their comfort level at 4 or above for behaviour prediction. Of course in this closed trial environment privacy issues are not a concern as they would be if such monitoring and prediction was performed in the real world by a third party.

Questions 7 and 8 aim to identify any notable links between the channel that a participant personally prefers, the channel that they think they watch most often and the channel that they actually do watch most often over the course of the trials. Question 7 asks the participant to select the channel that they personally preferred. Question 8 asks the participant to select the channel that they thought they watched the most during the trial. In addition, the TMB dataset can provide the channel that the participant actually watched the most during the trial.

Comparing the responses to question 7 with the TMB datasets, the results show that 50% of participants watched their personally preferred channel for the longest time during their trial. This is surprising as one would expect to see the majority of users viewing their personally preferred channel for the longest time. The fact that the results are equally divided suggests that under trial conditions the link between preferred channel and viewing time is inconclusive. However, this result may be influenced by the trial setting itself. In particular it is noted that non-browsers appeared to select channels based on pre-meditated decisions rather than preferred content. Indeed it is difficult to manufacture a trial situation where participants can express preferences through behaviour in a completely natural way.

Comparing the responses to questions 7 and 8 with the TMB datasets highlights another interesting point. Of the 50% of participants who thought they watched their preferred channel the most, only half of them actually did according to their TMB datasets. Equally, of the 50% of participants who didn't think they watched their preferred channel the most, one third of them actually did according to their TMB datasets. This highlights the discrepancy between how people believe they behave and how they actually do behave in reality. This discrepancy can have implications on what personalised adaptations a user expects to experience. A user's belief of their preference related behaviour may not match reality and hence they may assume personalised adaptation to be incorrect when in fact it is in line with observed user behaviours.

Question 9 queried whether the participant was aware that viewing time was a key factor in how preferences were learnt throughout the trial. To reduce bias it is best if participants are not aware of how the DIANNE performs incremental learning. Only 12% of participants answered positively with the rest being completely unaware that viewing time was a key factor. On further investigation the participants who stated they were aware clarified that it was an assumption made during their trial and not something they explicitly knew prior to the trial.

7.4 Summary

The DIANNE was tested and evaluated in a two step process involving benchmark tests and user trials. The aim of the benchmark tests was to investigate performance as a machine learning algorithm. The aim of the user trials was to evaluate utility as a preference learner in the pervasive domain.

For the benchmark tests several well cited datasets were applied to the DIANNE with increasing training data proportions. The results were unexpected in that they demonstrated very shallow learning curves. For most datasets the DIANNE achieved high accuracy figures with only 10% of training data. In most cases accuracy figures do improve as additional training data is provided but improvements are typically around 10%. However, these anomalies were attributed to the characteristics of the datasets and not an error in the evaluation process. Since many datasets only contained a small number of class values, high accuracies were achievable with minimal training data.

To illustrate this, a default rule was applied to the datasets. The default rule uses the most common class in the training dataset to classify all instances in the test dataset regardless of their attributes. Results showed that the default rule could achieve high accuracies for most datasets since the most common class in the training set was also the most common class in the test set.

Although obvious learning curves were not observable, the DIANNE achieved very good accuracy figures for most of the datasets. However, accuracy figures dropped for the particularly noisy TUMOUR dataset which contains a large number of class values. Later comparisons showed that all algorithms struggle with this dataset but the poorer accuracies achieved by the DIANNE suggest that there are other driving factors. Notably, the TUMOUR dataset establishes conditions under which the addition of new nodes into the network can reduce DIANNE accuracy.

The DIANNE was then compared with several other learning algorithms, both batch and incremental, in terms of accuracy over the selected datasets. With the exception of the TUMOUR dataset, the DIANNE performed comparably with, if not better than, many of those algorithms. With regard to batch algorithms the DIANNE outperformed CN2, Assistant, Simple Bayes and Naive Bayes on various datasets despite no a priori knowledge of the problem domain and no reprocessing of past data. With regard to incremental algorithms the DIANNE performed comparably with STAGGER and outperformed AQ15 on various datasets.

The second part of the testing and evaluation process involved live user trials. The trials took place in a real world pervasive environment providing contextual and behavioural inputs that the DIANNE could process and act upon in real time. A total of 24 participants took part in the trials that centred around a personalised TV scenario. A test environment was implemented within a University department building including several personalisable screens, a remote control device and an indoor location tracking system (based on RFID technology).

For the first part of the trial, participants were asked to select a preferred channel on each screen. They were then asked to revisit each screen a number of times to see if each screen (driven by the DIANNE) would automatically switch to their preferred channel for that screen. If the screen did not switch to the correct channel, the participant corrected it, allowing the DIANNE to refine internal knowledge towards the participant's viewing preferences. For the second part of the trial participants were given the chance to revise their preferred channel on each screen. Again they were asked to revisit the screens a number of times, correcting the screens if they showed the wrong channel.

The first part of the trial evaluated the ability of the DIANNE to learn from scratch and continue to refine learning over time. The second part of the trial evaluated the ability of the DIANNE to handle concept drift due to changes in the participant's viewing behaviour. The results showed that the DIANNE was able to identify initial viewing preferences very rapidly, meaning that the participant was almost always shown the correct channel at each screen during the first part of the trial. The results also show that during the second part of the trial DIANNE accuracy dropped immediately after the user changed their viewing behaviour. However, the accuracy steadily increased to almost 100% over the course of the second part of the trial, showing that the DIANNE can rapidly respond to concept drift.

During each trial a temporal and non-temporal behaviour dataset was logged. At the end of each trial both datasets were applied to the C45 tree building algorithm in order to compare the performance of a batch algorithm with that of the DIANNE in the problem domain. The temporal dataset was equivalent to the input received by the

DIANNE during the course of the trial. The non-temporal dataset was typical of those generated by preference learning systems.

The results show that the DIANNE outperformed the C45 algorithm regardless of whether the temporal or non-temporal dataset is used. However, C45 performance is improved with the temporal dataset confirming that extra temporal information is of benefit. These results reflect the findings of several other works that concluded an incremental learning system is best suited to incremental problem domains such as preference learning in pervasive environments.

During each trial the participant was requested to complete a short questionnaire about their trial experience. The responses received highlighted several interesting points. Firstly, when asked about pleasure and annoyance of correct and incorrect automatic channel switches most participants found it very pleasing when the screen automatically changed to the correct channel and only mildly annoying when it didn't. In this case the benefits appear to outweigh any detriment. This is reflected by the fact that the majority of participants would use such technology in their own home if it were freely available.

Many people did not feel that the DIANNE learnt their preferences too slowly, even when the incorrect channel was shown a number of times on the second part of the trial. This is an important indicator of the expectations that end users may have of a pervasive system employing machine learning technology.

Regarding behaviour monitoring and prediction, the majority of people stated that they felt very comfortable having their behaviour monitored and predicted during the course of the trial. Two major issues often encountered by pervasive systems are that end users often state their unease at being monitored by technology (the "Big Brother" issue) and their unease at technology performing actions on their behalf (the "loss of control" issue). However, it is noted that the views relating to this closed trial environment will be different from a real world scenario where privacy issues are more critical. It is also noted that the demographics of the trial participants are limited and that views may differ among users from different backgrounds.

To further investigate the connection between preferences and temporal behaviour, each participant was asked to state their personally preferred channel. When compared with the channel they actually watched the most during the trial, the results are inconclusive with exactly half of participants watching their personally preferred channel for the longest time and half watching another channel the longest. It is suggested that under trial conditions it is often difficult to replicate normal preference related behaviour. However, further comparisons between the channel that the participant believed they watched the most and the channel that they actually did watch the most highlights that there is often a gulf between how users believe they behave and how they actually do behave. This can have implications on the end user's perception of what personalised adaptations should occur.

8 Conclusion

The previous chapters have demonstrated the main ideas and research of the thesis and presented a solution to learning accurate and up to date preferences for personalisation in a pervasive environment. This concluding chapter affirms that the initial research question has been answered by concluding several discussion points and summarising the main work of this thesis. Finally, several suggestions are presented for future work that builds on the outcomes and findings of this thesis.

8.1 Discussion

The main research question of this thesis is:

How can a system learn and provide accurate and up to date user preferences for personalisation in a pervasive environment?

To affirm that this question has been answered by the final solution provided, firstly the requirements stated in Chapter 4 are discussed in terms of their fulfilment and, secondly, further discussion points are revisited and concluded.

8.1.1 Fulfilment of Design Requirements

Chapter 4 outlined a list of design requirements to be met by a learning system for preference learning in a pervasive environment. The requirements were identified from related literature and personal experience gained while developing the preference learning system for the DAIDALOS project. Considering this list again allows one to confirm that the DIANNE satisfies all the requirements of a pervasive preference learning system.

The initial requirements relate to incremental properties and learning properties. In terms of incremental properties the DIANNE does process one input at a time over time, doesn't reprocess past data, doesn't require a priori knowledge of the problem domain and does support a growing topology. The concept of a shrinking topology has not been fully investigated in this body of research and is identified as relevant future work. In terms of learning properties the DIANNE is a hetero-associative neural network which supports unsupervised learning.

The third requirement states that the learning system should represent internal knowledge in a format that is easy to maintain, quick to update and translatable into human understandable form. Exporting, storing and maintaining internal knowledge as IF-THEN-ELSE rules proved to be complex and inefficient in the DAIDALOS project. The DIANNE represents internal knowledge as a single layer network allowing for easy maintenance and rapid updates. The single layer topology also allows for rule extraction on demand since complexity introduced by hidden layers is not an issue. Developing rule extraction mechanisms for the extraction of rules from the DIANNE has not been fully investigated in this body of research and is identified as relevant future work.

The fourth requirement states that the learning system should have a dynamic topology with the ability to handle continuously changing input and output vectors. This is essential in a pervasive environment where context sources and services can appear and disappear through time. The DIANNE supports a dynamic topology allowing for the addition of new context/outcome groups and nodes allowing the DIANNE to represent new context sources and preferences throughout its lifetime. The removal of context/outcome groups and nodes has not been fully investigated in this body of research and is identified as relevant future work.

Requirement five states that the learning system should be able to learn positive and negative preferences. Many preference learning systems only support the learning of positive preferences due to action-triggered environment monitoring (which only occurs when the user performs some interaction). In contrast the DIANNE holds an internal representation of the environmental state, allowing for continuous updating of internal knowledge irrespective of whether the user performs an interaction or not. This enables the DIANNE to learn both positive and negative preferences.

The next requirement states that the learning system should be able to overcome pre-actions. During development of the DAIDALOS preference learning system it was identified that sometimes users will perform some action while in context A in preparation for entering context B. This led to incorrect action-context associations. The DIANNE overcomes this issue by basing association strengths on the temporal duration that an action prevails in some context.

Finally, the seventh requirement states that the learning system should incrementally handle concept drift and resolve conflicts in an appropriate and timely manner. The DIANNE supports a conflict resolution strategy to handle concept drift driven by changes in user behaviour. The conflict resolution strategy is based on two proposed heuristics that distinguish between long-term and short-term behaviour changes and hence the rate at which network error should be reduced. No reprocessing of past data is required and all conflicts are handled at one instance in time. The results from the user trials show that the DIANNE can learn over-riding preferences in a timely manner utilising the conflict resolution strategy in conjunction with temporal reinforcements. The DIANNE does not learn over-riding preferences in a one instance manner nor does it take an unacceptably long time.

8.1.2 Incremental vs. Batch

This thesis adopts the use of an incremental learning algorithm for preference learning in a pervasive domain; however, it is interesting to consider the success of this decision and whether incremental approaches are actually superior to batch in this problem domain.

It is stated in Section 2.3.6 that incremental tasks (such as user modelling, of which preference learning is a subset) are best handled by incremental learning algorithms. It is also noted in Section 3.5.5 that batch algorithms have several constraints when utilised for preference learning, the most significant being their lack of rapid response to an ever changing environment. The scheduled executions of batch algorithms do not seem in line with the open-world, continuous and dynamic characteristics of pervasive environments. However, batch algorithms seem to be the favoured approach for preference learning in numerous research projects.

Indeed, incremental tasks such as preference learning can be handled well with batch algorithms as shown by several projects including DAIDALOS where a C45 algorithm was utilised for preference learning. System evaluations and analysis return good results in terms of the accuracy of the preferences learnt although a key issue is that such preferences can become out of date between learning executions if user behaviour changes (manifested as concept drift in learning terms). For this reason it is noted that several projects take additional measures to handle changing behaviours between executions. However, an incremental algorithm provides natural support for rapid response to behaviour changes since inputs are processed as they occur through time.

The findings of this body of work suggest that an incremental algorithm is superior to a batch algorithm at learning accurate and up to date preferences in a pervasive environment. When directly compared with a batch algorithm during user trials the DIANNE achieves significantly better accuracy figures than the batch algorithm. The batch algorithm often struggles to learn recent over-riding behaviours when prominent past behaviours exist. Results also show that the DIANNE can rapidly respond to concept drift enabling accuracy to recover in a respectable time frame.

8.1.3 DIANNE as a Preference Learning System

Through the course of the thesis research the DIANNE has been implemented as a preference learning system in the PSS platform developed by the EU FP7 project PERSIST. Within this platform the DIANNE has been successfully demonstrated at a final project review and also utilised for live user trials.

The results of the user trials show that the DIANNE is successful as a preference learning system providing advantages over the approach implemented in the DAIDALOS project. Apart from improved accuracy and response rates the DIANNE also subsumes the functionality of the entire DAIDALOS personalisation system in a lighter and more efficient way. Since preferences are represented as weights within the network, no separate preference management system is required. Since the DIANNE updates preferences in real time as inputs are received, no separate learning system is required and since the DIANNE provides real time outputs as new input is received, no separate preference condition monitoring system is required.

The DIANNE provides the learning power and efficiency of a neural network but its single layer topology also allows the extraction of human readable preference rules. This is essential in a user centric domain allowing the end user to stay informed of what preferences the network has learnt about them. Although the DIANNE cannot represent non-linear problems (such as XOR) it is explained that in the preference learning domain the DIANNE will never need to represent such problems.

The user trials confirm that the DIANNE answers the main research question of this thesis. Although the context and behaviour vectors in the trial are non-complex, the DIANNE produced high accuracy values for preference learning from scratch in a variety of learning situations driven (due to varied user behaviour across trials). The DIANNE as also capable of rapid recovery of accuracy values when behaviour changes occurred. According to user feedback, this recovery happened in an adequate time frame without demonstrating one instance learning.

8.2 Future Work

The testing and evaluation process described in Chapter 7 has highlighted some areas for possible improvement within the DIANNE algorithm. Additionally, the discussion on requirements fulfilment in section 8.1 has also highlighted several research questions for future work.

8.2.1 DIANNE Extensions and Improvements

Initial Weight Assignments

When a new node is added into the DIANNE, all weights on the newly generated connections are initialised to zero so that they are neither excitatory nor inhibitory. However, this initialisation strategy can be problematic in some instances. In an unstable outcome node group the potentials of all outcome nodes tend towards -1. If the new node is an outcome node, when added to such an unstable outcome node group, the initial zero potential of the new node (sum of all weighted inputs) may be much higher than the potentials of all other group nodes. Hence the new node will over-ride all other outcomes in a one instance learning manner. This problem manifested itself during the benchmark test when the DIANNE was applied to the TUMOUR dataset.

It is interesting to investigate if an alternative weight initialisation strategy would improve DIANNE performance. However, an alternative strategy would require one to consider how weight initialisation should be applied relative to the potentials of the already existing outcome nodes in the outcome node group. For example, should the new node be initialised with weights that sum to a potential relative to (a) the winner node potential, (b) the loser node potential or (c) the average of the winner and loser node potentials?

Forgetting

Anti-Hebbian learning is implemented in the DIANNE although there is no general *forgetting* strategy applied across the network. The research question raised is whether applying a forgetting strategy within the DIANNE will enhance DIANNE performance in terms of accuracy.

Network Pruning

Anti-Hebbian learning can reduce the potential of network nodes to -1. At present, nodes with minimal potentials remain within the DIANNE. Although such nodes are rarely or never active, by remaining in the network they continue to affect the potentials of other group nodes. The research question raised is whether pruning such nodes from the DIANNE will enhance DIANNE performance in terms of efficiency and accuracy.

Handle Continuous Inputs

The DIANNE has been designed to handle discrete inputs only and it is assumed that inputs will be discretised before presentation to the DIANNE. If continuous inputs were presented to the current configuration it would result in the creation of excessive numbers of network nodes and would ultimately degrade efficiency and accuracy. Significant redesign would be required to enable the DIANNE to appropriately handle continuous inputs.

8.2.2 DIANNE Related Research

Rule Extraction

The DIANNE is a single layer neural network with no hidden layers. It should therefore be relatively trivial to extract human readable rules from the network. Other single layer incremental networks such as the Pocket Algorithm already employ rule extraction techniques which could be transferrable to the DIANNE. This on-demand rule extraction allows end users to view and understand what knowledge the network holds regarding their preferences. However in a pervasive personalisation domain it is useful to go beyond rule extraction. As well as viewing their preferences end users may also wish to manually change them.

This will require a bi-directional translation mechanism that can translate network format into rule format and rule format into network format. The end user could view and manually change their preferences in the rule format. The alterations to the preferences would then be translated back from rule format into network format. This challenge will be investigated within the EU FP7 SOCIETIES project where the DIANNE will be implemented as a key preference learning system.

Link Between Temporal Behaviour and Preferences

The solution provided in this thesis utilises a temporal reinforcement approach assuming some connection between the temporal nature of preference related behaviours and the actual preferences of an individual. The solution is successful but user trial results regarding the existence or strength of such a link are inconclusive. This may be due to trial conditions rendering abnormal preference related behaviour but in any case further investigation into this suggested link would prove useful.

8.3 Contributions of this Thesis

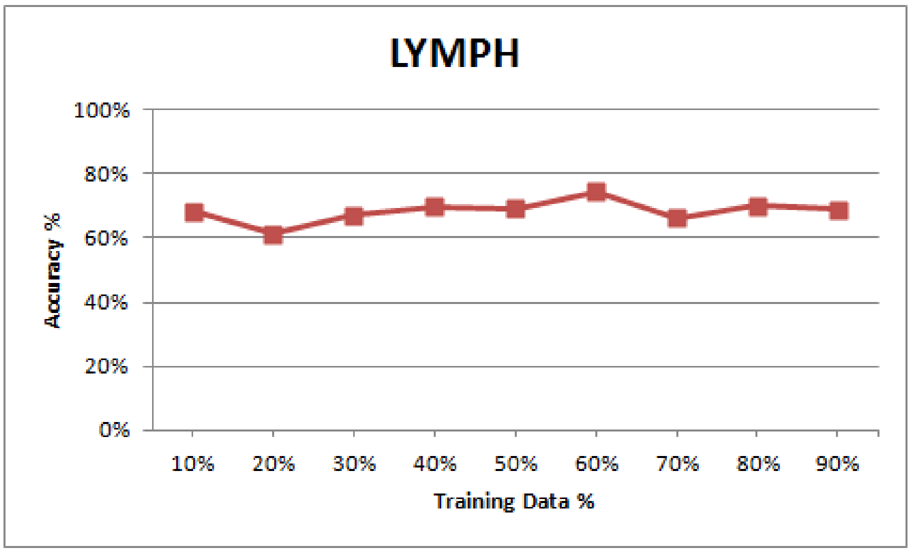
A final summary of the key contributions of this thesis is given below:

- Identification of key requirements for the provision of an efficient solution for preference learning in a pervasive environment. Most notably:
    - the identification that an incremental learning approach is the most natural and flexible way to handle incremental tasks, such as learning user preferences.
    - the identification that the temporal duration in which a preference prevails is an important piece of information. Utilising this information enables the learning system to perform negative preference learning and overcome learning issues caused by pre-actions.
- Design of Dynamic Incremental Associative Neural NEtwork (DIANNE) topology. Most notably:
    - The use of a single layer network for rapid and non-complex updating of internal knowledge. A single layer topology is shown to be sufficient for the preference learning problem domain and allows network knowledge to be translated into human understandable form for user review.
- Design of DIANNE temporal learning algorithm including several novel aspects such as:
    - an incremental approach to preference learning, processing inputs as they occur in real time.

    - continuous learning through temporal reinforcements. Weight updates occur in a temporal fashion based on the amount of time that an active input renders an active output.
    - the use of two learning rules. Hebbian/anti-Hebbian is used for continuous temporal learning. An error reduction approach is used for learning under conflict conditions when network output conflicts with the real world situation.
    - a dynamic squashing function applied to the outcome nodes to limit their potentials between -1 and +1 and stop the occurrence of saturation.
    - an incremental conflict resolution strategy that can resolve conflicts at one instance in time based on current knowledge. The strategy is based on two heuristic for incremental conflict resolution that are proposed in line with end user expectations and the notion of preference time constants.
- Implementation of DIANNE as a standalone system which has been tested on benchmark datasets and compared with other learning algorithms in terms of accuracy.
- Implementation of DIANNE as a preference learning system in the Personal Smart Space (PSS) platform. This integrated system was demonstrated to EU project reviewers and utilised during the evaluation of the DIANNE in live user trials.

Appendix A: C45 Algorithm Performance on LYMPH Dataset with 10% - 90% Training Data

This annex includes a graph showing the accuracy results achieved by the C45 algorithm when applied to the LYMPH dataset with an increasing training set size from 10% to 90%. The C45 algorithm was also applied to the other five datasets (CANCER, CANCERW, TUMOUR, HEART, VOTE) with training dataset sizes of 10-20%. These tests confirmed high accuracies with minimal training data.



Appendix B: User Trial Documents - Information Sheet

This appendix includes the information sheet given to each trial subject to read at the beginning of their trial session.

Introduction

Welcome and thanks for taking part in this user trial!

This trial consists of 2 steps:

1. Practical Trial
2. Questionnaire

Before you embark on the practical trial, this document will introduce and explain some important aspects. Please read it carefully.

The Practical Trial

During the practical trial you will be asked to visit 3 different screens which have been set up in rooms EM1.69 and EM1.70. You will select your preferred channel for each screen using the provided remote control device. Over time the screens will attempt to identify what channel you prefer to watch there and automatically change to the appropriate channel.

The practical trial involves several things that you will be introduced to here:

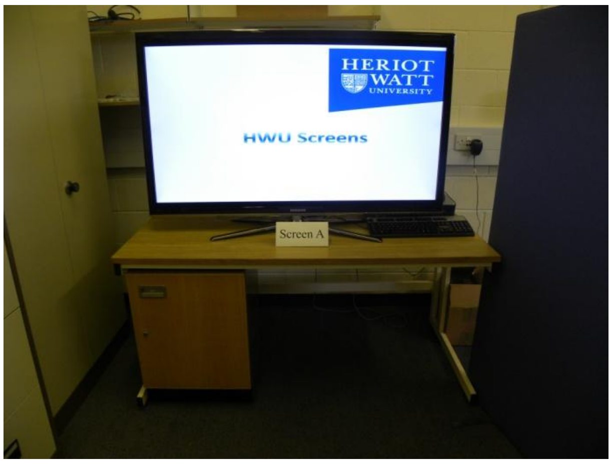
- 3 Screens
- 3 Channels
- Remote Control
- RFID Tag

The sections below will introduce you to each of these things before you start the trial.

The 3 Screens

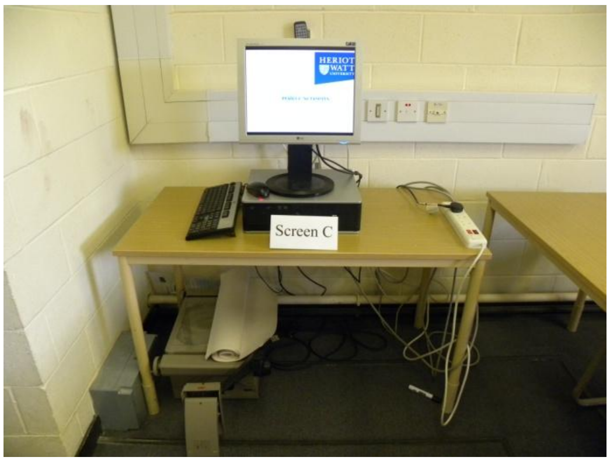
There are 3 screens set up in various rooms:

Screen A
EM1.69 (next to the door)

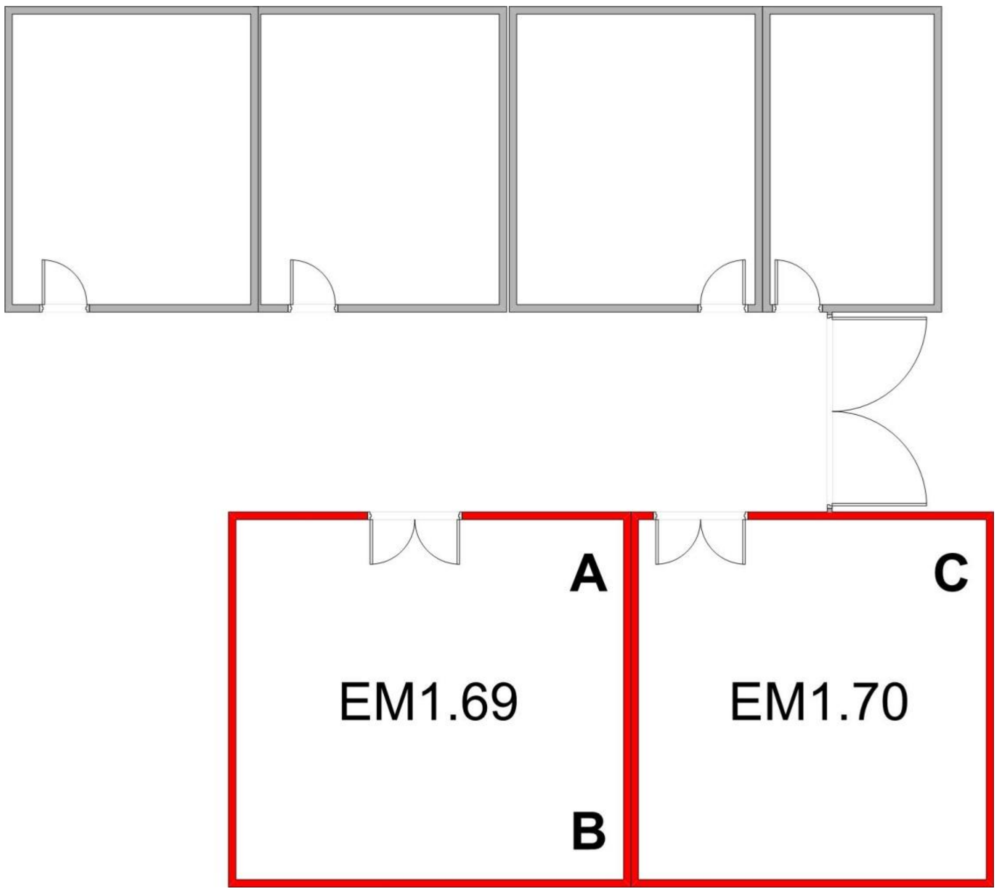


Screen B
EM1.69 (next to the window)

Screen C
EM1.70 (next door)



The map below shows the screen locations. Signs are also in place to guide you to the screens.

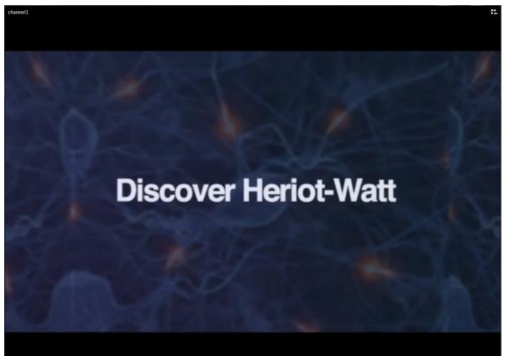

A
C
EM1.69
EM1.70
B

The 3 Channels

Each screen can play 3 different channels.

Channel 1

This channel shows HWU promotional videos.



Channel 2

This channel shows 3D animations created by MACS students for a coursework assignment

Channel 3

This channel shows promotional videos of several EU projects that the MACS department is involved in.

The Default Image

The default image is displayed on all screens at the start of the trial. During the trial, when you move away from a screen, the screen will return to displaying the default image.

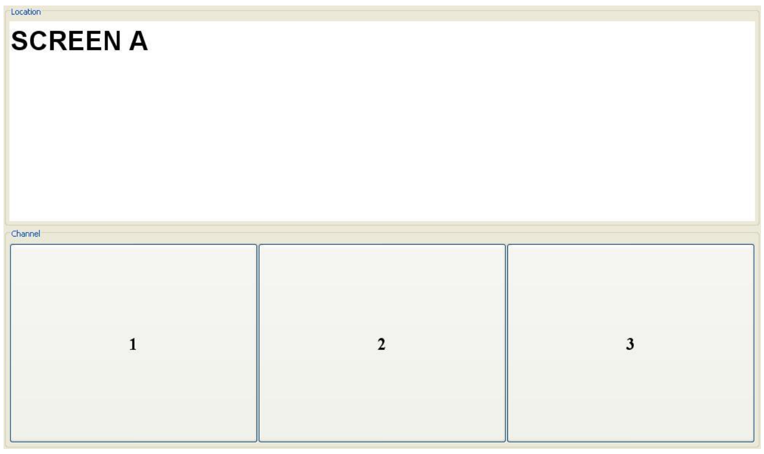
The Remote Control

You will be given a small device that will act as your remote control. The remote control looks like this:

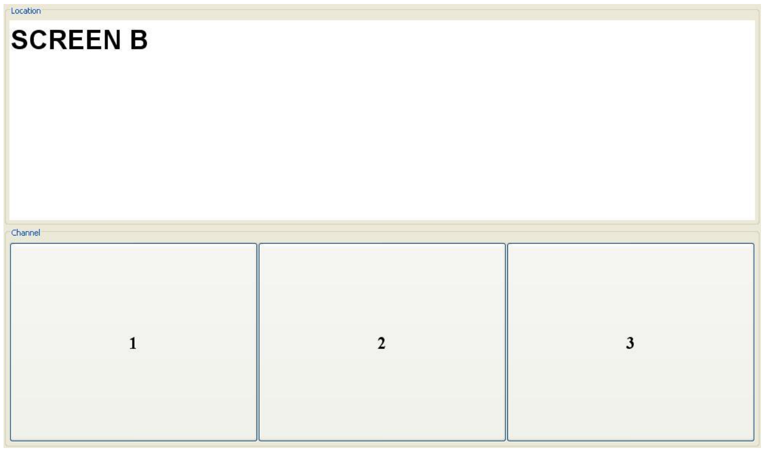
The remote control has two functions.

Firstly, the white box at the top of the remote control GUI shows the screen you are closest to.

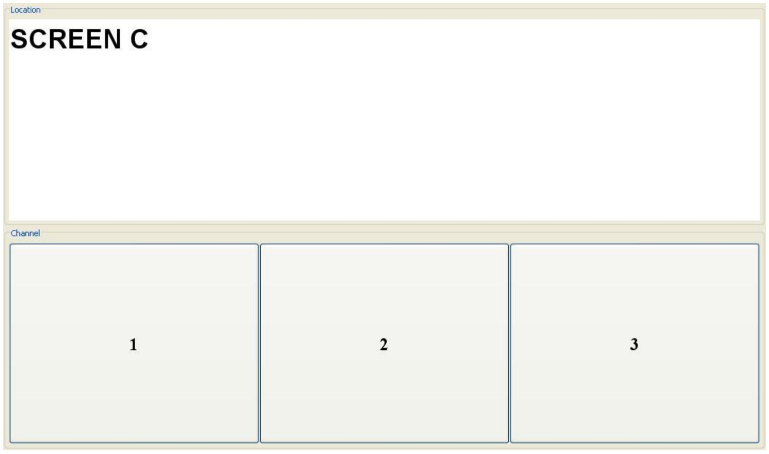
- When you are near screen 1 the remote control will display: "**Screen A**"



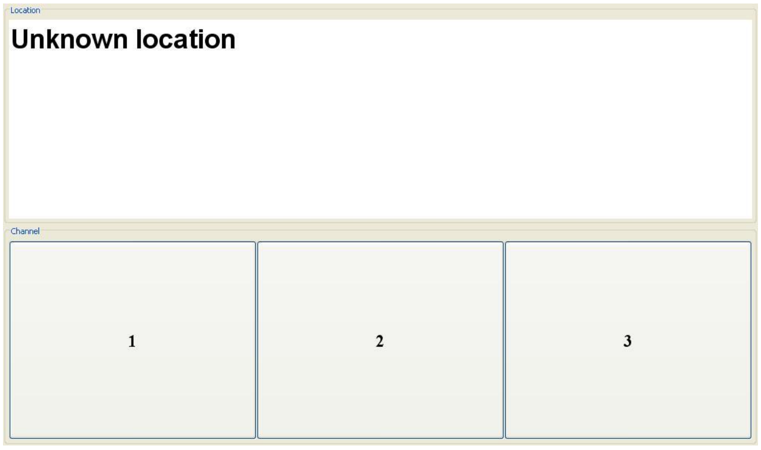
- When you are near screen 2 the remote control will display: "**Screen B**"



- When you are near screen 3 the remote control will display: "**Screen C**"



- If you are not near any screen the remote control will display: "**Unknown location**"



Secondly, the remote control has 3 buttons - one for each channel. This is how you change channel on the screens. **The remote control will change the channel on the screen that you are closest to.**

- If the remote control displays "**SCREEN A"** the remote control will change screen A
- If the remote control displays "**SCREEN B**" the remote control will change screen B

- If the remote control displays "**SCREEN C**" the remote control will change screen C
- If the remote control displays "**Unknown location**" the remote control will not change any screen.

The RFID Tag

The RFID tag is shown in the picture below:

It is a small credit card sized object that allows the remote control and the screens to know where you are. Please wear it round your neck at all times and try not to obscure it with clothing.

Please note that sometimes when you are standing in front of a screen the RFID tag will momentarily lose connection. When this happens the remote control will show "Unknown Location" and the screen will revert to the default image. Wait for a few seconds and it should rectify itself.

You are now ready to begin the practical part of the trial.

The tester will now give you:

- A clipboard with a trial instruction sheet and pen
- An RFID tag
- A remote control

If the remote control becomes unresponsive at any point during the trial or if you feel there is a problem please return to EM1.69 and alert the tester.

Thank you.

Appendix C: User Trial Documents - Trial Sheet

This appendix contains the trial sheet given to each trial subject during their trial session to note their channel selections.

Appendix D: User Trial Documents - Questionnaire

This appendix includes the questionnaire given to trial subjects to complete at the end of their trial session.

Questionnaire

1. On a scale of 1 to 5, how pleasing did you find it when the screens automatically changed to the correct channel?

5	☐	(Most pleasing)
4	☐	
3	☐	
2	☐	
1	☐	(Least pleasing)
n/a	☐	The screens never changed to the correct channel

2. On a scale of 1 to 5, how annoying did you find it when the screens automatically changed to an incorrect channel?

5	☐	(Most annoying)
4	☐	
3	☐	
2	☐	
1	☐	(Least annoying)
n/a	☐	The screens never changed to an incorrect channel

3. Would you use such functionality in your own home if it were freely available?

Yes	☐	
No	☐	If you selected NO, please explain why below.
Maybe	☐	

____

____

____

4. Did you feel that the system learnt your selected channels too slowly?

Yes	☐
No	☐
Sometimes	☐

5. On a scale of 1 to 5, how comfortable did you feel about the system monitoring your behaviour during the trial?

5	☐	(Most comfortable)
4	☐	

3	☐	
2	☐	
1	☐	(Least comfortable)

6. On a scale of 1 to 5, how comfortable did you feel about the system <u>predicting</u> your behaviour during the trial?

5	☐	(Most comfortable)
4	☐	
3	☐	
2	☐	
1	☐	(Least comfortable)

7. What channel was your personal favourite?

Channel 1	☐
Channel 2	☐
Channel 3	☐

8. What channel do you think you spent the most time watching?

Channel 1	☐
Channel 2	☐
Channel 3	☐

9. Were you aware that the system predicted your preferred channel on each screen based on the TIME you spent watching it on that screen?

Yes	☐
No	☐

REFERENCES

[1] Weiser, M.: 1991, "The Computer for the 21st Century", Scientific America

[2] Polsad, S.: 2009, "Ubiquitous Computing: Smart Devices, Environments and Interactions", Wiley.

[3] Hansmann, U., Merk, L., Nicklous, M.S., Stober, T.: 2003, "Pervasive Computing: Second Edition", Springer-Verlag.

[4] Satyanarayanan, M.: 2001, "Pervasive Computing: Vision and Challenges", IEEE Personal Communications, 8(4), pp. 10-17.

[5] Deering, S., Hinden, R.: 1995, "Internet Protocol Version 6 (IPv6) Specification", RFC 1883, Available online: http://www.ietf.org/rfc/rfc1883.txt, Accessed on 26th October 2010.

[6] Gershenfeld, N., Krikorian, R., Cohen, D.: 2004, "The Internet of Things", Scientific American, 291(4), pp. 76-81.

[7] Ghosh, A., Wolter, D.R., Andrews, J.G., Chen, R.: 2005, "Broadband Wireless Access with WiMax/802.16: Current Performance Benchmarks and Future Potential", IEEE Communications Magazine, 43(2), pp. 129-36.

[8] Apple iPhone homepage, Available online: http://www.apple.com/iphone, Accessed on 26th October 2010.

[9] Want, R., Hopper, A., Falcao, V., Gibbons, J.: 1992, "The Active Badge System", ACM Transactions on Information Systems, 10(1), pp. 91-102.

[10] Addlesee, M.D., Jones, A., Livesey, F., Samaria, F.: 1997, "The ORL Active Floor", IEEE Personal Communications, 4(5), pp. 35-41.

[11] Greenfield, A.: 2006, "Everyware: The Dawning Age of Ubiquitous Computing", New Riders.

[12] Mann, S.: 1997, "Wearable Computing: A first step toward personal imaging", Computer, 30(2), pp. 25-32.

[13] Things That Think Consortium homepage, Available online: http://ttt.media.mit.edu/, Accessed on 26th October 2010.

[14] Mistry, P., Maes, P.: 2009, "Sixth Sense: A Wearable Gestural Interface", Proc. of International Conference on Computer Graphics and Interactive Techniques, Yokohama, Japan, Article 11.

[15] Motorola/Burton Audex Jacket, Available online: http://www.gizmag.com/go/5072/, Accessed on 26th October 2010.

[16] Adidas_1 Shoe, Available online: http://www.gizmag.com/go/3810, Accessed on 26th October 2010.

[17] Wii homepage, Available online: http://www.nintendo.com/wii, Accessed on 26th October 2010.

[18] Tangible Media Group homepage, Available online: http://tangible.media.mit.edu/index.php, Accessed on 26th October 2010.

[19] Ishii, H., Ullmer, B.: 1997, "Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms", Proc. of SIGCHI Conference on Human Factors in Computing Systems (CHI '97), Atlanta, Georgia, pp. 234 - 241.

[20] Van Laerhoven, K., Villar, N., Schmidt, A., Kortuem, G., Gellersen, H.: 2003, "Using an Autonomous Cube for Basic Navigation and Input", Proc. of International Conference on Multimodal Interfaces, Vancouver, Canada, pp. 203 - 210.

[21] Zigelbaum, J., Labrune, J.B.: 2009, "Some Challenges of Designing Shape Changing Interfaces", CHI 2009, Workshop on Transitive Materials, Boston, Massachusetts.

[22] Weiser, M., Brown, J.S.: 1996, "The Coming Age of Calm Technology", In Denning, P.J., Metcalfe, R.M. (Eds.), *Beyond Calculation: The next Fifty Years of Computing*, Springer-Verlag.

[23] Live Wire, Available online: http://tech90s.walkerart.org/nj/transcript/nj_04.html, Accessed on 26th October 2010.

[24] Ferscha, A.: 2007, "Informative art display metaphors", Proc. Fourth International Conference on Universal Access in Human–Computer Interaction, LNCS 4555, Springer-Verlag, pp. 82–92.

[25] Open Project hompage, Available online: http://giove.isti.cnr.it:88/, Accessed on 26th October 2010.

[26] Dey, A.K.: 2001, "Understanding and Using Context", Personal and Ubiquitous Computing, 5(1), pp. 4 - 7.

[27] SpeckNet Consortium homepage, Available online: http://www.specknet.org/, Accessed on 26th October 2010.

[28] Warneke, B., Liebowitz, B., and Pister, K. S. J.: 2001, "Smart dust: communicating with a cubic-millimeter computer", IEEE Computer, 34(44), pp. 2 - 9.

[29] Levis, P., Madden, S., Polastre, J., Szewczyk, R., Whitehouse, K., Woo, A., Gay, D., Hill, J., Welsh, M., Brewer, E., and Culler, D.: 2004, "TinyOS: An operating system for wireless sensor networks", In Webber, W., Rabaey, J.M., Aarts, E (Eds.), *Ambient Intelligence*, Springer-Verlag, pp. 115 - 148.

[30] Van Laerhoven, K., Schmidt, A., Gellersen, H.W.: 2002, "Multi-Sensor Context-Aware Clothing", Proc. of the Sixth International Symposium on Wearable Computers (ISWC'02), Seattle, pp. 49 - 56.

[31] Brummitt, B.: 2001, "Better Living Through Geometry", Personal and Ubiquitous Computing, 5(1), pp. 42 - 45.

[32] Choi J., Shin, D.: 2006, "Intelligent Pervasive Middleware Based on Biometrics", Proc. of Third International Conference on Ubiquitous Intelligence and Computing (UIC '06), LNCS 4159, Springer-Verlag, pp. 157-165.

[33] Brooks, R.A.: 1997, "The Intelligent Room Project", Proc. of Second International Conference on Cognitive Technology: Humanizing the Information Age, Aizu, Japan, pp 271 - 278.

[34] Project Oxygen homepage, Available online: http://oxygen.lcs.mit.edu/, Accessed on 26th October 2010.

[35] Chung, H., Lee, C.J., Selker, T.: 2006, "Lover's cups: drinking interfaces as new communication channels", CHI 2006, Extended abstracts on Human factors in computing systems, Montréal, Canada.

[36] Agarawala, A., Greenberg, S., Ho, G.: 2004, "The Context-Aware Pill Bottle and Medication Monitor", Technical Report 2004-752-17, Dept. of Computer Science, University of Calgary.

[37] Salvador, Z., Bonail, B., Lafuente, A., Larrea, M., Abascal, J., Gardeazabal, L.: 2005, "AmIChair: Ambient Intelligence and Intelligent Wheelchairs", Proc. of the Home Oriented Informatics and Telematics Conference (HOIT'05), 11, pp. 31 - 36.

[38] Yonezawa, T., Clarkson, B., Yasumura, M., Mase, K.: 2001, "Context-Aware Sensor-Doll as a Music Expression Device", CHI 2001, Extended abstracts on Human Factors in computing systems, Seattle, Washington.

[39] Bonanni, L., Arroyo, E., Lee, C.H., Selker, T.: 2005, "Smart Sinks: Real World Opportunities for Context-Aware Interaction", CHI 2005, Extended abstracts on Human Factors in computing systems, Porland, Oregon, pp. 1232-1235.

[40] Linden, G., Smith, B., York, J.: 2003, "Amazon.com Recommendations: Item-to-Item Collaborative Filtering", IEEE Internet Computing, 7(1), pp.76 - 80.

[41] Bellekens, P., Van Kerckhove, G., Kaptein, A.: 2009, "iFanzy: A Ubiquitous Approach Towards a Personalized EPG", Proc. of the EuroITV 2009, Lueven, Belgium, pp. 130 - 131.

[42] Satyanarayanan, M.: 2001, "Pervasive Computing: Vision and Challenges", IEEE Personal Communications, 8(4), pp. 10-17.

[43] Lesser, V., Atighetchi, M., Benyo, B., Horling, B., Raja, A., Vincent, R., Wagner, T., Xuan, P., Zhang, S.X.Q.: 1999, "The Intelligent Home Testbed", Proc. of the Anatomy Control Software Workshop (Autonomous Agent Workshop).

[44] Yoshihama, S., Chou, P., Wong, D.: 2003, "Managing Behaviour of Intelligent Environments", Proc. of the First IEEE International Conference on Pervasive Computing and Communications (PerCom'03), pp. 330.

[45] Mozer, M.C.: 2004, "Lessons from an Adaptive House", In D. Cook and R. Das (Eds.), *Smart environments: Technologies, protocols and applications*, pp. 273 - 294.

[46] Sousa, J.P., Poladian, V., Garlan, D., Schmerl, B., Shaw, M.: 2006, "Task-based Adaptation for Ubiquitous Computing", IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews, Special Issue on Engineering Autonomic Systems, 36(3), pp. 328.

[47] Dey, A.K.: 2009, "Modeling and Intelligibility in Ambient Environments", Journal of Ambient Intelligence and Smart Environments, 1(1), pp. 57 - 62.

[48] Cypher, A., Halbert, D.C., Kurlander, D., Lieberman, H., Maulsby, D., Myers, B.A., Turransky, A.: 1993, "Watch What I Do: Programming by Demonstration", The MIT Press, Cambridge, Massachusetts, London, England.

[49] Chin, J., Callaghan, V., Clarke, G.: 2009, "End-User Customisation of Intelligent Environments", In H. Nakashima, H. Aghajan and J.C. Augusto (Eds.), *Handbook of Ambient Intelligence and Smart Environments*, pp. 371 - 407.

[50] Kobsa, A., Wahlster W.: 1989, "User Models in Dialog Systems", Springer-Verlag, pp. 74-107.

[51] W3C: 2007, "Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 2.0", Available online: http://www.w3.org/TR/2007/WD-CCPP-struct-vocab2-20070430/, Accessed on 26th October 2010.

[52] ETSI: 2005, "Human factors (HF); User Profile Management", ETSI Guide, EG 202 325 v1.1.1, Available online: http://webapp.etsi.org/workprogram/Report_WorkItem.asp?WKI_ID=18333, Accessed on 26th October 2010.

[53] Ziebart, B.D., Roth, D., Campbell, R.H., Dey, A.K.: 2005, "Learning Automation Policies for Pervasive Computing Environments", Proc. of the Second International Conference on Autonomic Computing (ICAC'05), pp. 193.

[54] Youngblood, G.M., Holder, L.B., Cook, D.J.: 2005, "Managing Adaptive Versatile Environments", Pervasive and Mobile Computing, 1(4), pp. 373 - 403.

[55] Youngblood, G.M., Heierman, E.O., Holder, L.B., Cook, D.J.: 2005, "Automation Intelligence for the Smart Environment", Proc. of the International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, pp. 1513 - 1514.

[56] Mozer, M.C.: 1998, "The Neural Network House: An Environment that Adapts to its Inhabitants", Proc. of the American Association for Artificial Intelligence Spring Symposium on Intelligent Environments, Menlo Park, California, pp. 110-114.

[57] Si, H., Kawahara, Y., Morikawa, H., Aoyama, T.: 2005, "A Stochastic Approach for Creating Context-Aware Services based on Context Histories in Smart Home", Proc. of Exploiting Context Histories in Smart Environments (ECHISE2005), pp. 3480 - 3495.

[58] Barkhuus L.: 2003, "Is Context-Aware Computing Taking Control Away from the User? Three Levels if Interactivity Examined", Proc. of Fifth Annual Conference on Ubiquitous Computing (UbiComp'03), LNCS 2864, pp. 149 - 156.

[59] Groppe J., Mueller, W.: 2005, "Profile Management Technology for Smart Customizations in Private Home Applications", Proc. of the Sixteenth International Workshop on Database and Expert Systems Applications (DEXA'05), Copenhagen, Denmark, pp. 226 - 230.

[60] Cordier, C., Carrez, F., Van Kranenburg, H., Licciardi, C., Van der Meer, J., Spedalieri, A,. Le Rouzic, J. P., Zoric, J.: 2006, "Addressing the Challenges of Beyond 3G Service Delivery: the SPICE Service Platform", Proc. of Workshop on Applications and Services in Wireless Networks (ASWN '06), Kassel, Germany.

[61] Sutterer, M., Coutand, O., Droegehorn, O., David, K.: 2007, "Managing and Delivering Context-Dependent User Preferences in Ubiquitous Computing Environments", Proc. of the 2007 International Symposium on Applications and the Internet Workshops (SAINTW'07), Article 4, Hiroshima, Japan.

[62] Hagras, H.: 2007, "Embedding Computational Intelligence in Pervasive Spaces", IEEE Pervasive Computing, 6(3), pp. 85-89.

[63] Ball, M., Callaghan, V., Gardner, M., Trossen, D.: 2010, "Achieving Human-Agent Teamwork in eHealth based Pervasive Intelligent Environments," Proc. of 4th International Conference on Pervasive Computing Technologies for Healthcare 2010, pp.1 - 8.

[64] Ball, M., Callaghan, V., Gardner, M., Trossen, D.: 2009, "Exploring Adjustable Autonomy and Addressing User Concerns in Intelligent Environments", Proc. of the 5th International Conference on Intelligent Environments 2009, Spain, IOS Press (2009).

[65] Cole, R.J., Brown, Z.: 2009, "Reconciling Human and Automated Intelligence in the Provision of Occupant Comfort", Intelligent Buildings International, 1(1), pp. 39 - 55.

[66] McCulloch, W.S., Pitts, W.: 1943, "A Logical Calculus of the Ideas Imminent in Nervous Activity", Bulletin of Mathematical Biophysics, 5, pp. 115 - 133.

[67] Rosenblatt, F.: 1958, "The perceptron: a probabilistic model for information storage and organization in the brain", Psychological Review, 65, pp. 386 - 407.

[68] Minsky, M., Papert, S.: 1969, "Perceptrons", Cambridge, MA: MIT Press.

[69] Michalski, R.S.: 1973, "AQVAL/1--Computer Implementation of Variable-Valued Logic System VL1 and Examples of its Application to Pattern Recognition", Proc. the First International Joint Conference on Pattern Recognition, Washington DC, pp. 3 - 17.

[70] Mitchell, T.M.: 1978, "Version Spaces: An Approach to Concept Learning", Technical Report ADA074462, Department of Computer Science, Stanford University, California.

[71] Quinlan, J.R.: 1986, "Induction of Decision Trees", Machine Learning, 1(1), pp. 81 - 106.

[72] Fisher, D.H.: 1987, "Knowledge Acquisition Via Incremental Conceptual Clustering", Machine Learning, 2(2), pp. 139 - 172.

[73] Rumelhart, D.E., Hinton, G.E., Williams, R.J.: 1986, "Learning Internal Representations by Backpropagating Errors", Nature, Vol. 323, pp. 533 - 536.

[74] Dawkins, R.: 1986, "The Blind Watchmaker", Norton & Company Inc.

[75] Dietterich, T.G.: 1997, "Machine Learning Research: Four Current Directions", AI Magazine, 18(4), pp. 97 - 136.

[76] Mitchell, T.M.: 1997, "Does Machine Learning Really Work?", AI Magazine, 18(3), pp. 11 - 20.

[77] Wolpert, D.H., Macready, W.G.: 1997, "No Free Lunch Theorems for Optimization", IEEE Transactions on Evolutionary Computation, 1(1), pp. 67 - 82.

[78] Mitchell, T.M.: 1997, "Machine Learning", McGraw-Hill.

[79] Carlson, A.J., Cumby, C.M., Rosen, J.L., Roth, D.: 1999, "SNoW user guide", Technical Report 2101, University of Illinois, Urbana, Illinois.

[80] Haykin, S.: 2009, "Neural Networks and Learning Machines", Pearson.

[81] Bishop, C.M.: 2006, "Pattern Recognition and Machine Learning", Springer.

[82] Nurmi, P., Salden, A., Lun Lau, S., Suomela, J., Sutterer, M., Millerat, J., Martin, M., Lagerspetz, E., Poortinga, R.: 2006, "A System for Context Dependent User Modeling", Proc. of OTM Federated Workshops, Montpellier, France, Vol. 4278 of Lecture Notes in Computer Science, pp. 1894 - 1903.

[83] Si, H., Kawahara, Y., Igakura, T., Tonouchi, T., Morikawa, H., Aoyama, T.: 2006, "A Hybrid Context-aware Service Platform Based on Stochastic and Rule-Description Approaches", Proc. 3rd Annual International Conference on Mobile and Ubiquitous Systems: Networks and Services (MOBIQUITOUS 2006), San Jose, California.

[84] Youngblood, M., Cook, D.J., Holder, L.B.: 2005, "Seamlessly Engineering a Smart Environment", Proc. IEEE Conference on Systems, Man and Cybernetics, pp. 548 - 553.

[85] Schaefer, R.: 2004, "Fuzzy Evaluation of User Profiles", Proc. of Workshop on User Profiling at CHI 2004, Vienna, Austria.

[86] Hagras, H., Doctor, F., Callaghan, V., Lopez, A.: 2007, "An Incremental Adaptive Life Long Learning Approach for Type-2 Fuzzy Embedded Agents in Ambient Intelligent Environments", IEEE Transactions on Fuzzy Systems, 15(1), pp. 41 - 55.

[87] Quinlan, J.R.: 1993, "C45: Programs for Machine Learning", Morgan Kaufman.

[88] Quinlan, J.R.: 1987, "Simplifying Decision Trees", International Journal of Man-Machine Studies, 27, pp. 221 - 234.

[89] Mitchell, T.M.: 1977, "Version Spaces: A Candidate Elimination Approach to Rule Learning", Proc. of Fifth International Joint Conference on Artificial Intelligence, Cambridge, USA, pp. 305 - 310.

[90] Cohen, W.W.: 1995, "Fast Effective Rule Induction", Proc. of Twelfth International Conference on Machine Learning, Tahoe City, California, pp. 115 - 123.

[91] Du, Y., Kernchen, R., Moessner, K., Raeck, C., Sawade, O., Tarkoma, S., Arbanowski, S.: 2007, " Context Aware Learning for Intelligent Mobile Multimodal User Interfaces", Proc. of the Eighteenth Annual IEEE International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC’07), Athens, Greece, pp. 1 - 5.

[92] Agrawal, R., Srikant, R.: 1994, "Fast Algorithms for Mining Association Rules", Proc. of International Conference on Very Large Databases (VLDB'94), Santiago, Chile, pp. 487 - 499.

[93] Giraud-Carrier, C.: 2000, "A Note on the Utility of Incremental Learning", AI Communications, 13(4), pp. 215 - 223.

[94] Schlimmer, J.C., Granger, R.H.: 1986, "Incremental Learning from Noisy Data", Machine Learning, 1(3), pp. 317 - 354.

[95] Michalski, R.S., Mozetic, I., Hong, J., Lavrac, N.: 1986, "The Multi-Purpose Incremental Learning System AQ15 and its Testing Application to Three Medical Domains", Proc. of Fifth National Conference on Artificial Intelligence, Philadelphia, pp. 1041 - 1045.

[96] Sclimmer, J.C., Fisher, D.: 1986, "A Case Study of Incremental Concept Induction", Proc. of Fifth National Conference on Artificial Intelligence, Philadelphia, pp. 496 - 501.

[97] Utgoff, P.E.: 1989, "Incremental Induction of Decision Trees", Machine Learning, 4(2), pp. 161 - 186.

[98] Gallant, S.I.: 1990, "Perceptron-Based Learning Algorithms", IEEE Transactions on Neural Networks, 1(2), pp. 179 - 191.

[99] Littlestone, N.: 1988, "Learning Quickly when Irrelevant Attributes Abound: A New Linear-threshold Algorithm", Machine Learning, 2(4), pp. 285 - 318.

[100] Hertz, J., Krogh A., Palmer, R.G.: 1991, "Introduction to the Theory of Neural Computation", Addison-Wesley.

[101] Carpenter, G.A., Grossberg, S.: 1987, "A Massively Parallel Architecture for a Self-Organising Neural Pattern Recognition Machine", Computer Vision, Graphics and Image Processing, 37(1), pp. 54 - 115.

[102] Carpenter, G.A., Grossberg, S.: 1987, "ART2: Stable Self-Organising of Pattern Recognition Codes for Analog Input Patterns", Applied Optics, 26(23), pp. 4919 - 4930.

[103] Carpenter, G.A., Grossberg, S.: 1990, "ART3: Hierarchical Search using Chemical Transmitters in Self-Organising Pattern Recognition Architectures", Neural Networks, 3, pp. 129 - 152.

[104] Carpenter, G.A., Grossberg, S., Rosen, D.B.: 1991, "Fuzzy ART: Fast Stable Learning and Categorization of Analog Patterns by an Adaptive Resonance System", Neural Networks, 4(6), pp. 759 - 771.

[105] Carpenter G.A., Grossberg, S., Reynolds, J.H.: 1991, "ARTMAP: Supervised Real-time Learning and Classification of Nonstationary Data by a Self-organizing Neural Network", Neural Networks, 4(5), pp. 565 - 588.

[106] Carpenter, G.A., Grossberg, S., Reynolds, J.H.: 1995, "A Fuzzy ARTMAP Nonparametric Probability Estimator for Nonstationary Pattern Recognition Problems", IEEE Transactions on Neural Networks, 6(6), 1330 - 1336.

[107] Vo, M.T.: 1994, "Incremental Learning using the Time Delay Neural Network", Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing, Adelaide, Austrailia, 3, pp. 629 - 632.

[108] Fu, L., Hsu, H., Principe, J.C.: 1996, "Incremental Backpropagation Learning Networks", IEEE Transactions on Neural Networks, 7(3), pp. 757 - 761.

[109] Engelbrecht, A.P., Brits, R.: 2001, "A Clustering Approach to Incremental Learning for Feedforward Neural Networks", Proc. of International Joint Conference on Neural Networks, Washington DC, 3, pp. 2019 - 2024.

[110] Zhang, B.T.: 1994, "An Incremental Learning Algorithm that Optimizes Network Size and Sample Size in One Trial", Proc. of IEEE International Conference on Neural Networks, Orlando, Florida, pp. 215 - 220.

[111] Grippo, L.: 2000, "Convergent On-Line Algorithms for Supervised Learning in Neural Networks", IEEE Transactions on Neural Networks, 11(6), pp. 1284 - 1299.

[112] Polikar, R., Udpa, L., Udpa, S.S., Honavar, V.: 2001, "Learn++: An Incremental Learning Algorithm for Supervised Neural Networks", IEEE Transactions on Systems, Man and Cybernetics, 31(4), pp. 497 - 508.

[113] Segal, R.B., Kephart, J.O.: 2000, "Incremental Learning in SwiftFile", Proc. of Seventeenth International Conference on Machine Learning, Stanford University, California, pp. 863 - 870.

[114] Carbonell, J.G., Michalski, R.S., Mitchell, T.M.: 1983, "Machine Learning: A Historical and Methodological Analysis", AI Magazine, 4(3), pp. 69 - 79.

[115] DAIDALOS, Available online: http://www.ist-DAIDALOS.org/, Accessed on 26th October 2010.

[116] Farshchian, B., Zoric, J., Mehrmann, L., Cawsey, A., Williams, M.H., Robertson, P., Hauser, C., 2004, "Developing Pervasive Services for Future Telecommunication Networks", Proc. WWW/Internet 2004, pp. 977 - 982.

[117] Williams, M.H., Taylor, N., Roussaki, I., Robertson, P., Farshchian, B., Doolin, K., 2006, "Developing a Pervasive System for a Mobile Environment", Proc. eChallenges - Exploiting the Knowledge Economy, pp. 1695 - 1702.

[118] McBurney, S., Papadopoulou, E., Taylor, N., Williams, M.H., 2009, "Comparing Two Different Architectures for Pervasive Systems from the Viewpoint of Personalisation", Proc. eChallenges, IOS Press, (ISBN: 978-1-905824-13-7).

[119] McBurney, S., Papadopoulou, E., Taylor, N., Williams, M.H., 2009, "User Preference Management in a Pervasive System should be a Trusted Function", Proc. The IASTED International Conference on Systems (ICONS 09).

[120] Williams, M.H., Roussaki, I., Strimpakou, M., Yang, Y., MacKinnon, L., Dewar, R., Milyaev, N., Pils, C., Anagnostou, M., 2005, "Context-Awareness and Personalisation in the DAIDALOS Pervasive Environment", International Conference on Pervasive Systems (ICPS 05), pp. 98-107.

[121] Williams, M.H., Yang, Y., Taylor, N., McBurney, S., Papadopoulou, E., Mahon, F., Crotty, M., 2006, "Personalised Dynamic Composition of Services and Resources in a Wireless Pervasive Computing Environment", Proc. First International Symposium on Wireless Pervasive Computing, pp. 377 - 382.

[122] Papadopoulou, E., Williams, M.H., Taylor, N., McBurney, S.: 2006, "Redirecting Communication in a Pervasive System", Proc. IEEE Sponsored eChallenges, IOS Press, pp. 1688 - 1694.

[123] Yang, Y., Williams, M.H., Taylor N., McBurney, S., Papadopoulou, E.: 2006, "Handling Personalized Redirection in a Wireless Pervasive Computing System with Different Approaches to Identity", Proc. First International Symposium on Wireless Pervasive Computing, 6 pp.

[124] Angermann, M., McBurney, S., Kuhmuench, C., Mahon, F., Mitic, J., Robertson, P., Whitmore, J., 2006, "Integrating and Demonstrating Pervasiveness in a Scenario Driven Approach", Proc. IEEE sponsored eChallenges, 8 pp.

[125] Papadopoulou, E., McBurney, S., Taylor, N., Williams, M.H., Abu-Shaaban, Y.: 2009, "User Preferences to Support Privacy Policy Handling in Pervasive/Ubiquitous Systems", International Journal on Advances in Security, 2(1), pp. 62 - 71.

[126] Papadopoulou, E., McBurney, S., Taylor, N., Williams, M.H.: 2008, "Linking Privacy and User Preferences in the Identity Management for a Pervasive System", Proc. IEEE/WIC/ACM International Conference on Web Intelligence (WI-08), pp. 192 - 195.

[127] Papadopoulou, E., McBurney, S., Taylor, N., Williams, M.H.: 2008, "Using Personalisation to Support Privacy in Ubiquitous Systems", Poster Supplement for the 10th International Conference on Ubiquitous Computing (Ubicomp 08), South Korea.

[128] Papadopoulou, E., McBurney, S., Taylor, N., Williams, M.H.: 2008, "Using User Preferences to Enhance Privacy in Pervasive Systems", Proc. Third International Conference on Systems (ICONS 08), Mexico, IEEE Computer Society, pp. 271 - 276.

[129] Papadopoulou, E., McBurney, S., Taylor, N., Williams, M.H., Lo Bello, G.: 2008, "Adapting Stereotypes to Handle Dynamic User Profiles in a Pervasive System", Proc. Fourth International Conference on Advances in Computer Science and Technology (ACST 08), pp. 7 - 12.

[130] McBurney, S., Williams, M.H., Taylor, N., Papadopoulou, E.: 2007, "Managing User Preferences for Personalisation in a Pervasive Service Environment", Proc. Third Advanced International Conference on Telecommunications (AICT), pp. 32 - 37.

[131] Papadopoulou, E., McBurney, S., Taylor, N., Williams, M.H.: 2008, "A Dynamic Approach to Dealing with User Preferences in a Pervasive System", Proc. International Conference on Intelligent Pervasive Computing (IPC-08), pp. 409 - 416.

[132] McBurney, S., Papadopoulou, E., Taylor, N., Williams, M.H.: 2008, "Adapting Pervasive Environments Through Machine Learning and Dynamic Personalization", Proc. International Conference on Intelligent Pervasive Computing (IPC-08), pp. 395 - 402.

[133] McBurney, S., Papadopoulou, E., Taylor, N., Williams, M.H.: 2009, "Implicit Adaptation of User Preferences in Pervasive Systems", Proc. Fourth International Conference on Systems (ICONS 09), pp. 56 - 62.

[134] MIT Media Lab, The Reality Mining Dataset, Available online: http://reality.media.mit.edu/dataset.php, Accessed on 26th October 2010.

[135] Gallacher, S., Papadopoulou, E., Taylor, N., Williams, M.H., Abu-Shabaan, Y.: "Dynamic Context-Aware Personalisation in a Pervasive Environment", Submitted to the Journal of Pervasive and Mobile Computing, 6th August 2009.

[136] Webb G.I., Pazzani, M.J., Billus, D.: 2001, "Machine Learning for User Modeling", User Modeling and User-Adapted Interaction, Vol. 11, pp. 19 - 29.

[137] Baddeley, A.: 1997, "Human Memory: Theory and Practice", Psychology Press.

[138] Baddeley, A.: 1999, "Essentials of Human Memory", Psychology Press.

[139] Lyapunov, A.M.: 1992, "General Problem on Stability of Motion", Taylor and Francis, London.

[140] PERSIST project wiki, Available online: http://www.ict-persist.eu, Accessed on 22nd October 2010.

[141] Taylor, N.K.: 2008, "Personal eSpace and Personal Smart Spaces", Proc. First PerAda Workshop on Pervasive Adaptation (SASO '08), pp. 156 - 161.

[142] Crotty, M., Taylor, N., Williams, H., Frank, K., Roussaki, I., Roddy, M.: 2009, "A Pervasive Environment Based on Personal Self-Improving Smart Spaces", Ambient Intelligence 2008, Springer-Verlag, Heidelberg, pp. 58 - 62.

[143] SOCIETIES project wiki, Available online: http://www.ict-societies.eu, Accessed on 25th October 2010.

[144] Personal Smart Space open source: http://sourceforge.net/projects/psmartspace, Accessed on 25th October 2010.

[145] UCI Machine Learning Repository: http://archive.ics.uci.edu/ml/, Accessed on 26th October 2010.

[146] Clark, P., Niblett, T.: 1989, "The CN2 Induction Algorithm", Machine Learning, 3(4), pp. 261 - 283.

[147] Clark, P., Niblett, T.: 1987, "Induction in Noisy Domains", In I Bratko & N. Lavrac (Eds.), *Progress in Machine Learning*, pp. 11 - 30. Sigma Press.

[148] Kononenko, I., Simec, E.: 1995, "Induction of Decision Trees using RELIEFF", In G. Della Riccia, R. Kruse & R. Viertl (Eds.), *Mathematical and Statistical Methods in Artificial Intelligence*, CISM Courses and Lectures No. 363. Springer Verlag.

[149] Ratanamahatana, C., Gunopulos, D.: 2002, "Scaling up the Naive Bayesian Classifier: Using Decision Trees for Feature Selection", In Proc. of Workshop on Data Cleaning and Preprocessing (DCAP 2002), at IEEE International Conference on Data Mining (ICDM 2002), Maebashi, Japan.

[150] Clark, P., Boswell, R.: 1991, "Rule Induction with CN2: Some Recent Improvements", In Y. Kodratoff (Ed.), *Machine Learning EWSL-91*, pp. 151 - 163. Springer Verlag.

[151] Syed, N. A., Liu, H., Sung, K. K.: 1999, "Handling Concept Drift in Incremental Learning with Support Vector Machines", In Proc. Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '99), pp. 317 - 321.

PUBLICATIONS

The following papers have been published relating to the work described in the thesis:

Journal Papers

[1] Papadopoulou, E., Gallacher, S., Taylor, N., Williams, H.: "A Personal Smart Space Approach to Realising Ambient Ecologies", Submitted to the Pervasive and Mobile Computing Journal, 15th May 2010.

[2] Gallacher, S., Papadopoulou, E., Taylor, N., Williams, M.H., Abu-Shaaban, Y.: "Dynamic Context-Aware Personalisation in a Pervasive Environment", Submitted to the Journal of Pervasive and Mobile Computing, 6th August 2009.

[3] Papadopoulou, E., McBurney, S., Taylor, N., Williams, M.H., Abu-Shaaban, Y.: 2009, "User Preferences to Support Privacy Policy Handling in Pervasive/Ubiquitous Systems", International Journal on Advances in Security, 2(1), pp. 62 - 71.

Conference Papers

[4] Gallacher, S., Papadopoulou, E., Taylor, N. K., Blackmun, F. R., Williams, M. H., Roussaki, I., Kalatzis, N., Liampotis, N., Zhang, D.: 2011, "Personalisation in a System Combining Pervasiveness and Social Networking", to appear in Workshop on Social Interactive Media Networking and Applications (SIMNA 2011), at IEEE ICCCN 2011, Maui, Hawaii.

[5] Gallacher, S., Papadopoulou, E., Taylor, N.K., Williams, M.H, Blackmun, F.: 2010, "Linking Between Personal Smart Spaces", 2nd PerAada Workshop on User Centric Pervasive Adaptive Systems (UCPA '10), at the 7th International ICST Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (Mobiquitous 2010), Sydney.

[6] Gallacher, S., Papadopoulou, E., Taylor, N., Williams, H.: 2010, "Putting the 'Personal' into Personal Smart Spaces", Proc. of Pervasive Personalisation Workshop, Pervasive 2010, pp. 10 - 17.

[7] Papadopoulou, E., Abu-Shaaban, Y., Gallacher, S., Taylor, N., Williams, M.H.: 2010, "Two Approaches to Handling Proactivity in Pervasive Systems", Proc. International Conference on Information Systems, Technology and Management (ICISTM '10), pp. 64 - 75.

[8] McBurney, S., Papadopoulou, E., Taylor, N., Williams, H., Abu-Shaaban, Y.: 2009, "Comparing Two Different Architectures for Pervasive Systems from the Viewpoint of Personalisation", Proc. eChallenges (e2009), IOS Press, (ISBN: 978-1-905824-13-7).

[9] McBurney, S., Taylor, N., Williams, H., Papadopoulou, E.: 2009, "Giving the User Explicit Control over Implicit Personalisation". Proc. Persist Workshop on Intelligent Pervasive Environments (AISB 09), ISBN 1902956834, pp. 16 - 19.

[10] Abu-Shaaban, Y., McBurney, S., Taylor, N., Williams, M.H., Kalatzis, N., Rousakki, I.: 2009, "User Intent to Support Pro-activity in a Pervasive System", Proc. Persist Workshop on Intelligent Pervasive Environments (AISB 09), ISBN 1902956834, pp. 3 - 8.

[11] Frank, K., Robertson, P., McBurney, S., Kalatzis, N., Roussaki, I., Marengo, M.: 2009, "A Hybrid Preference Learning and Context Refinement Architecture", Proc. Persist Workshop on Intelligent Pervasive Environments (AISB 09), ISBN 1902956834, pp. 9 - 15.

[12] McBurney, S., Papadopoulou, E., Taylor, N., Williams, H.: 2009, "Implicit Adaptation of User Preferences in Pervasive Systems", Proc. Fourth International Conference on Systems (ICONS 09), pp. 56 - 62.

[13] Papadopoulou, E., McBurney, S., Williams, H.: 2009, "A Model for Personalised Communications Control in Pervasive Systems", Proc. The IASTED International Conference on Advances in Computer Science and Engineering (ACSE 09).

[14] McBurney, S., Papadopoulou, E., Taylor, N., Williams, H.: 2009, "User Preference Management in a Pervasive System should be a Trusted Function", Proc. The IASTED International Conference on Advances in Computer Science and Engineering (ACSE 09).

[15] McBurney, S., Papadopoulou, E., Taylor, N., Williams, H.: 2008, "Adapting Pervasive Environments through Machine Learning and Dynamic Personalisation", Proc. International Conference on Intelligent Pervasive Computing (IPC-08), pp. 395 - 402.

[16] Papadopoulou, E., McBurney, S., Taylor, N., Williams, H.: 2008, "A Dynamic Approach to Dealing with User Preferences", Proc. International Conference on Intelligent Pervasive Computing (IPC-08), pp. 409 - 416.

[17] Papadopoulou, E., McBurney, S., Taylor, N., Williams, M.H.: 2008, "Linking Privacy and User Preferences in the Identity Management for a Pervasive System", Proc. IEEE/WIC/ACM International Conference on Web Intelligence (WI-08), pp. 192 - 195.

[18] Papadopoulou, E., McBurney, S., Taylor, N., Williams, M.H.: 2008, "Using Personalisation to Support Privacy in Ubiquitous Systems", Poster supplement for the 10th International Conference on Ubiquitous Computing (Ubicomp 08), South Korea, September 2008.

[19] Papadopoulou, E., McBurney, S., Taylor, N., Williams, M.H., Dolinar, K., Neubauer, M.: 2008, "Using User Preferences to Enhance Privacy in Pervasive Systems", Proc. Third International Conference on Systems (ICONS 08), pp. 271 - 276.

[20] Papadopoulou E., McBurney, S., Taylor, N., Williams, M.H., Lo Bello, G.: 2008, "Adapting Stereotypes to Handle Dynamic User Profiles in a Pervasive System", Proc. Fourth International Conference on Advances in Computer Science and Technology (ACST 08), pp. 7 - 12.

[21] Williams, M.H., Papadopoulou, E., Taylor, N., McBurney, S., Dolinar, K.: 2007, "Conflict between Privacy and Personalisation in a Pervasive Service Environment", Proc. Advances in Computer Science and Technology, pp 172 - 181.

[22] McBurney, S., Williams, M.H., Taylor, N., Papadopoulou, E.: 2007, "Managing User Preferences for Personalization in a Pervasive Service Environment", Proc. The Third Advanced International Conference on Telecommunications (AICT), pp. 32 - 37.

[23] Angermann, M., McBurney, S., Kuhmuench, C., Mahon, F., Mitic, J., Robertson, P., Whitmore, J.: 2006, "Integrating and Demonstrating Pervasiveness in a Scenario Driven Approach", Proc. IEEE sponsored eChallenges, 8 pp.

[24] Papadopoulou, E., Williams, M.H., Taylor, N., McBurney, S.: 2006, "Redirecting Communication in a Pervasive System", Proc. IEEE Sponsored eChallenges, IOS Press, pp. 1688 - 1694.

[25] Yang, Y., Williams, M.H., Taylor, N., McBurney, S., Papadopoulou, E.: 2006, "Handling Personalized Redirection in a Wireless Pervasive Computing System with Different Approaches to Identity", Proc. First International Symposium on Wireless Pervasive Computing, 6 pp.

[26] Williams, M.H., Yang, Y., Taylor, N., McBurney, S., Papadopoulou, E., Mahon, F., Crotty, M.: 2006, "Personalized Dynamic Composition of Services and Resources in a Wireless Pervasive Computing Environment", Proc. First International Symposium on Wireless Pervasive Computing, pp. 377 - 382.