Amund Tveit

Customizing Cyberspace

Methods for User Representation
and Prediction

Dr.ing.-thesis 2004:28

Fakultet for informasjonsteknologi, matematikk og
elektroteknikk
Institutt for datateknikk og informasjonsvitenskap

NTNU

Amund Tveit

Customizing Cyberspace: Methods for User Representation and Prediction

Department of Computer and Information Science
Norwegian University of Science and Technology
N-7491 Trondheim, Norway

Customizing Cyberspace: Methods for User Representation and Prediction
Amund Tveit
amund.tveit@idi.ntnu.no
http://www.idi.ntnu.no/~amundt/

Norwegian University of Science and Technology (NTNU)
http://www.ntnu.no/
N-7491 Trondheim, Norway
Dr.ing.-thesis 2004:28
ISBN 82-471-6260-1 (printed)
ISBN 82-471-6259-8 (electronic)
ISSN: 0809-103X

Department of Computer and Information Science (IDI), NTNU
http://www.idi.ntnu.no/
Sem Sælands vei 7-9, N-7491 Trondheim, Norway
IDI Report 2004:02
ISSN 1503-416X

Abstract

Cyberspace plays an increasingly important role in people's life due to its plentiful offering of services and information, e.g. the Word Wide Web, the Mobile Web and Online Games. However, the usability of cyberspace services is frequently reduced by its lack of customization according to individual needs and preferences.

In this thesis we address the cyberspace customization issue by focusing on methods for user representation and prediction. Examples of cyberspace customization include delegation of user data and tasks to software agents, automatic pre-fetching, or pre-processing of service content based on predictions. The cyberspace service types primarily investigated are Mobile Commerce (e.g. news, finance and games) and Massively Multiplayer Online Games (MMOGs).

First a conceptual software agent architecture for supporting users of mobile commerce services will be presented, including a peer-to-peer based collaborative filtering extension to support product and service recommendations.

In order to examine the scalability of the proposed conceptual software agent architecture a simulator for MMOGs is developed. Due to their size and complexity, MMOGs can provide an estimated "upper bound" for the performance requirements of other cyberspace services using similar agent architectures.

Prediction of cyberspace user behaviour is considered to be a classification problem, and because of the large and continuously changing nature of cyberspace services there is a need for scalable classifiers. This is handled by proposed classifiers that are incrementally trainable, support a large number of classes, and supports efficient decremental untraining of outdated classification knowledge, and are efficiently parallelized in order to scale well.

Finally the incremental classifier is empirically compared with existing classifiers on: 1) general classification data sets, 2) user clickstreams from an actual web usage log, and 3) a synthetic game usage log from the developed MMOG simulator. The proposed incremental classifier is shown to an order of magnitude faster than the other classifiers, significantly more accurate than the naive bayes classifier on the selected data sets, and with insignificantly different accuracy from the other classifiers.

The papers leading to this thesis have combined been cited more than 50 times in book, journal, magazine, conference, workshop, thesis, whitepaper and technical report publications at research events and universities in 20 countries. 2 of the papers have been applied in educational settings for university courses in Canada, Finland, France, Germany, Norway, Sweden and USA.

Contents

List of Figures

List of Tables

Preface

This thesis is submitted to the Norwegian University of Science and Technology in partial fulfillment of the requirements for the degree *Doktor Ingeniør.* This work has been conducted at the Department of Computer and Information Sciences (IDI), Trondheim, Norway.

Acknowledgments

The work presented in this thesis has been financed by the Norwegian University of Science and Technology through the Electronic Commercial Agents (ElComAg) project funded by the Norwegian Research Council.

First, I would like to thank my supervisor Professor Mihhail Matskin for his continuous advice, coaching and support. Together with Professor Arne Sølvberg he has made my work possible by getting funding for and managing the ElComAg project.

I would like to thank all my students, in particular Espen, Håvard, Jørgen, Kristoffer and Øyvind. I would like to thank my colleague and friend Magnus Lie Hetland for the interesting discussions, and our jointly published research in particular. I would like to thank my co-entrepreneur and friend Thomas Brox Røst for providing a fruitful link to the IT industry through our company Agentus AS, and a thank to our investors Alexander Vik (Xcelera Inc.) and Hans Eirik Olav (Juno Finans AS).

I would like to thank the people I've been working with and for in educational settings during the PhD scholarship time: Associate Professor Stig Frode Mjølsnes, Associate Professor Øystein Nytrø, Torbjørn Krøvel, Professor Arne Halaas, Associate Professor Pinar Ozturk, Per Kristian Lehre, and Professor Agnar Aamodt.

I would like to thank the people I've been sharing offices with during the PhD scholarship time: Torgeir, Alf Inge, Alexander, Aleksander, and Jinghai.

I would like to thank all past and present department colleagues, in particular: The Lunch Bunch, Sobah, Reidar, Carl-Fredrik, Elisabeth, Erlend, Pauline, Herindrasana, Terje, Roxana, Lasse, Staal, Herman, Anne, Norun, Berit, Bård, Anders, Arne Dag, Ellen, Eli, Alf, Erik, Birgit, Lisbeth, Kjell, Kjetil, and Kristianne.

I would like to thank Assistant Professor Brian Blake (Georgetown, USA), Senior Researcher Lawrence Cavedon (Stanford, USA), Associate Professor Zakaria Maamar (Zayed, UAE), Assistant Professor Anthony Scime (SUNY Brockport, USA), Managing Editor Helen Zhang (Zhejiang, China) and Research Scientist Steven Willmott (Catalunya, Spain) for allowing me to participate in the international research community by being a reviewer for journals, conferences and workshops. I would like to thank Associate Professor Ruck Thawonmas (Ritsumeikan, Japan) and his research team for finding interest in Zereal.

I would like to thank all my friends and family, in particular Bjørn, Ingvar, Jorun, Jørgen, Jørund, Kari, Kåre, Lisa, Marius, Marte, Nils Christian, and Åse.

My deepest thanks goes to my dearest Wenche for her great love and support, and to our daughter Anneli for just being herself. I would especially like to thank my father Halvard, my mother Inger and my brother Gisle for all the support.

I would also like to thank everyone I should have thanked that I've forgotten to thank already!

Amund Tveit
March 18, 2004

Part I

Context

Chapter 1

Introduction

1.1 Introduction

Cyberspace was first presented in William Gibson's classic book Neuromancer in 1984. We choose to interpret cyberspace as being all systems, services and underlying infrastructure known as the Internet. This view is shared in the dictionary WordNet which considers cyberspace and the Internet to be synonymous (Miller [1995]).

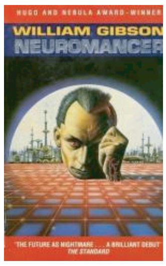

HUGO AND NEBULA AWARD-WINNER
WILLIAM GIBSON
NEUROMANCER
"THE FUTURE AS NIGHTMARE . . . A BRILLIANT DEBUT"
THE STANDARD

Figure 1.1: William Gibson's Neuromancer

This thesis is titled *Customizing Cyberspace* because it describes approaches for *user representation* and *user prediction* that can be used for the customization of cyberspace services to individual users. Such customization is frequently called *personalization*. Personalization of computer-based services is often studied in the fields of marketing and human-computer interaction, but we choose to take a (distributed) artificial intelligence approach with software agents and classification due to scope of the Electronic Commercial Agents (ElComAg) project.

Cyberspace Definitions

Definition 1.1 (Cyberspace, Gibson [1984]). *A consensual hallucination experienced daily by billions of legitimate operators, in every nation, by children being taught mathematical concepts...A graphical representation of data abstracted from the banks of every computer in the human systems. Unthinkable complexity. Lines of light ranged in the non-space of the mind, clusters and constellations of data. Like city lights, receding...*

Definition 1.2 (Cyberspace, Illingworth [1996]). *An informal word first thought to have been used by the novelist William Gibson to refer to the total data on all the computers on all the networks in the world. The word has passed into common use as a way of referring to any large collection of network-accessible computer-based data.*

Topic-Related Definitions

Definition 1.3 (Cyberspace Services). *Commercial or non-commercial (multi-)user application-level services offered through a Cyberspace media such as the wireless or wired Internet. Example services include Massively Multiplayer Online Games, Mobile Commerce, Electronic Commerce, E-mail and the World Wide Web.*

Definition 1.4 (Cyberspace User Representation). *Abstractions, software and (log) data representing actions and profiles for users of cyberspace services.*

Definition 1.5 (Cyberspace User Prediction). *Algorithmic prediction of a user's (near) future action or preference based on the user representation data describing the user itself, and other users in the same or related cyberspace services. Applications include pre-fetching and personalization of digital content, and automatic recommendations of products and services. Examples of methods enabling this are classification and collaborative filtering .*

User representation *and* user prediction are both considered because they are closely related, i.e. you can't perform algorithmic and (partially) automatic user prediction in cyberspace without user representation data.

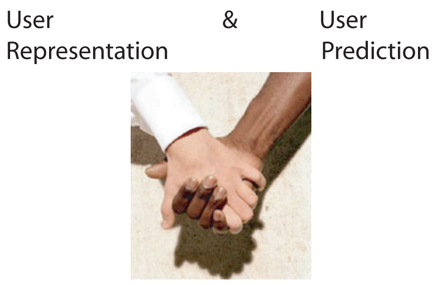


Figure 1.2: Thesis Topic Relations

1.2 The ELCOMAG Project

This thesis is a result of the **Electronic Commercial Agents** (ElComAg) project. The focus of the project was on facilities for trading of knowledge through electronic commerce (e-commerce), including these topics:

1. Conceptual Modeling of Knowledge and Trading Process
2. The design of sale-sites (electronic marketplaces)
3. The design of agents for serving the trade process
4. Information system architecture for knowledge trade

The main objectives of this thesis are related to topic 3 and 4, in particular with respect to agent-based information system architecture for e-commerce. We consider e-commerce to be a part of cyberspace.

The word *serving* (part of topic 3) is a key issue in this thesis since the underlying motivation for all the presented work is to better *serve* the cyberspace users. Users in general prefer cyberspace services that have the following characteristics:

1. Are easy and flexible to use
2. Provide relevant and interesting information
3. Have a decent response time

Characteristic 1 is studied in the field of Human Computer Interaction (HCI), and is considered outside the scope of this thesis. Amazon.com's product recommendation system is an example on how to provide relevant and interesting information (characteristic 2). Response time can be improved with predictive pre-fetching of content (characteristic 3). If these characteristics are in place the cyberspace service might have a prosperous future if it is at the same time timed, priced and marketed in the right way.

Example: Amazon.com's product recommendation

Amazon.com provides an automatic product recommendation service. The algorithms underneath use *user representation* data from customer profiles together with logs of usage and similar purchases by other customers to suggest products that might be of interest to the user (*user prediction*). Details about the recommender algorithms is described by Linden et al. [2003].

Application Areas of Interest

Instead of focusing on all every of the e-commerce field we've selected to primarily investigate two of its novel subfields:

1. Mobile commerce (wireless e-commerce)
2. Massively multiplayer online games (subscription-based)

One of the key challenges in mobile commerce is how to efficiently deal with the resource constraints a mobile client device imposes (e.g. bandwidth, processing power and memory, energy and screen-size). Some of these constraints have in recent years been relaxed, e.g. WiFi networks for increased bandwidth and fuel cells for increased energy and uptime, but to a large extent they are still valid, at least compared to their wired counterparts. Based on the ElComAg project's scope we selected to investigate how software agents can be of assistance to serve users of mobile commerce.

One of the key challenges in massively multiplayer online games (MMOGs) is to deal with the scale and complexity caused by the hundreds of thousands of concurrent players operating in very large virtual worlds. In terms of the richness and rapid frequency of actions performed by the players, massively multiplayer online games are in the near future likely to *dwarf* the current large-scale Internet and E-commerce systems such as Ebay, Amazon.com and Google.

Another key challenge of MMOGs is to ensure that players (i.e. subscribing customers) have a good time and do not get bored. This issue has up till now not been adressed from a data mining viewpoint, possibly because there still are plenty of unsolved issues related to handling the scale and complexity of MMOGs.

Computer games, including MMOGs, have been considered as the killer applications of human-level AI, Laird and Kent [2001]. We share this view, but stress that human-level AI in MMOGs is neither the goal nor the result of this thesis.

1.3 Research Questions

The corresponding main research question that this thesis attempts to answer is:

MRQ: How can we provide scalable and flexible user representation and prediction methods for increased automatic customization of cyberspace services?

This question is important because if answered it can lead to what users consider as better cyberspace services. And if users become more satisfied with the cyberspace services they are likely to use them more hence providing increased revenue for the cyberspace service providers. The reason the services become better is because the user representation and prediction methods can provide more targeted, interesting and relevant information for each individual user and the performance can potentially be improved by the prediction methods

The set of more specific research questions are as follows:

RQ1: How can mobile commerce customers be supported by software agents?

1. How to handle resource constraints imposed by mobile devices?
2. How to represent the customer's profile and interests?
3. Which types of customer services can be handled?
4. Can the approach be generalized to handle electronic commerce?
5. How can customers get product or service recommendations?

RQ2: How to test the proposed solution to **RQ1**, and extending it toward supporting Massively Multiplayer Online Games?

1. Why are Massively Multiplayer Online Games (MMOGs) a relevant test case?
2. How can data for customer personalization be gathered in the MMOG case?
3. How do customer data gathering in MMOGs compare to the existing standard for Web Usage Logging?
4. How can customer data from MMOG services be used for customer personalization?

RQ3: How can classification algorithms be used for customization of cyberspace services, e.g. mobile commerce and massively multiplayer online games?

1. Why is classification appropriate for customization of cyberspace services?
2. How can classifier algorithms scale to handle a large amount of customers and customer personalization types?
3. How can classifier algorithms handle changes in data over time?
4. How to evaluate classification performance of the proposed classifier(s) compared to state-of-the-art classifiers, and what are the results?

Research Question	**User Rep.**	**User Pred.**
MRQ	X	X
RQ1	X	
RQ2	X	X
RQ3		X

Table 1.1: Relation - Research Questions and Thesis Topics

Research Field

Since the Electronic Commercial Agents project goal is about applying agent technology in various e-commerce settings, the research field of study can be considered a part of the **Web Intelligence** (WI) field (Zhong et al. [2003, 2000]). WI is interpreted as the application of Artificial Intelligence (AI) in an Internet context. Software agents belongs to the *Distributed AI* field and classifier algorithms belongs to the *Machine Learning* field, both considered subfields of AI (Moulin and Draa [1996]; Mitchell [1997]). Web Intelligence and its subtopics considered relevant for this thesis will be further described in chapter 3.

1.4 Contributions

The work in this thesis has been a combination of the design of software architecture and algorithms, performing experiments, programming and developing and running computer simulations. This work has been documented in the papers in Part III.

Methods for User Representation

User representation methods proposed are agent-based abstractions of users together with their corresponding profiles (paper A, B and C), usage logs for MMOG services (paper E) and related empirical results (paper D).

Methods for User Prediction

User prediction methods proposed are classifier algorithms (paper F, G and H) and related empirical results on user prediction with user representation data in the form of usage logs from an actual web site and the simulated MMOG setting from paper C (paper I). Note that the classifier algorithms proposed are not limited to user prediction, but also as general incremental linear classifiers.

The main contributions of this thesis are:

C1: A conceptual solution and a supporting platform for implementing and using personal software assistant agents in mobile commerce services. The solution is aimed towards relaxing the restrictions of mobile devices and wireless communications (Paper A).

C2: A conceptual peer-to-peer extension of the platform for supporting scalable and distributed product and service recommendations for mobile commerce customers (Paper B).

C3: A scalable platform for simulating customer user in particular kind of mobile/electronic-commerce service - Massively Multiplayer Online Games (Platform described in paper C and the related empirical performance evaluation in paper D).

C4: Investigation and proposition of requirements for doing customer personalization in Massively Multiplayer Online Game services. This includes the creation of a definition of data mining types for MMOGs, a classification of computer games from a data mining viewpoint, a *comparison* of information gathering for web usage mining and game usage mining, and finally a proposal for a common game log format to enable game usage mining (Paper E).

C5: Investigation and proposition of algorithms that can be used in m/e-commerce personalization, including developing classification algorithms that: scale with a large number of classes (Paper F), utilizes parallelization (Paper H), and handles changes in classification data over time (Paper G).

Contribution	User Rep.	User Pred.
C1	X	
C2	X	X
C3	X	
C4	X	X
C5		X

Table 1.2: Relation - Contributions and Thesis Topics

1.5 Publications

The research leading to this thesis has resulted in the following publications: 1 refereed *journal paper* (Matskin and Tveit [2001]), 1 refereed *book chapter* (Matskin and Tveit [2003]), 5 refereed papers at *international conferences* (Tveit [2002a]; Elster et al. [2003]; Tveit et al. [2003b]; Tveit and Hetland [2003]; Tveit et al. [2003a]), 3 refereed papers at *international workshops and symposia* (Tveit [2001c]; Petersen et al. [2002]; Tveit [2002b]), 1 paper at a *local university conference* (Tveit [2001a]), and 7 *technical reports* (Tveit [2001b, 2002c]; Tveit and Matskin [2002]; Tveit and Engum [2003]; Tveit [2003b,a,c]).

See figure 1.5 for a visual overview of the papers and their relations. Dotted edges represents thematic relation, the other lines mean self-citation (i.e. stronger thematic relation).

1.6 Thesis Outline

This thesis is “paper collection”, i.e. having a contextual description for the papers and the papers themselves. Its two first parts describes the context, main results, evaluations and conclusions. The third part contains the papers, and finally the appendices in the fourth part.

Chapter 2 describes the materials (software and hardware) and methods (quantitative and computer science) applied in the work leading to this thesis

Chapter 3 provides an overview of the Web Intelligence field, and its subtopics related to this thesis. It also goes into some more detail about software agents and classification in web intelligence

Chapter 4 describes the main results (summarized from the papers)

Chapter 5 presents the empirical evaluation of the results

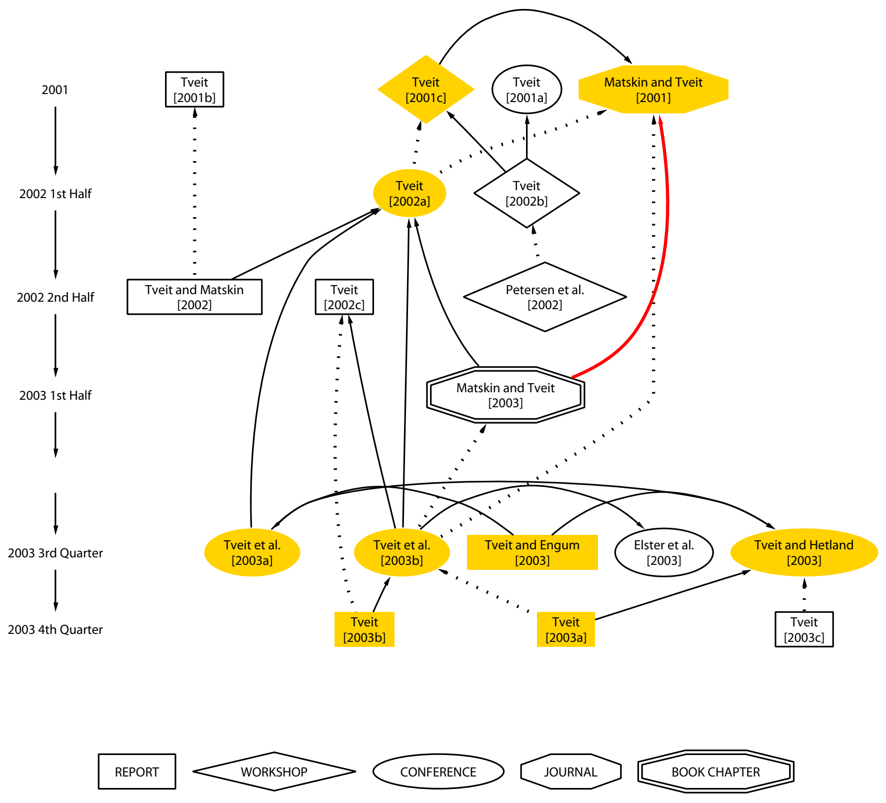


Figure 1.3: The publications and their relations

Chapter 6 summarizes the conclusions and proposes a direction for further research.

The foundation of this thesis is its 6 published papers and 3 technical reports (marked with light grey in figure 1.3).

Paper A describes a software agent architecture for supporting mobile services and users. In order do a proof-of-concept a prototype was developed.

Paper B describes a recommender system extension to the architecture presented in paper A.

Paper C describes a software agent architecture and platform for the simulation of users (players) in massively multiplayer online games. This platform is inspired by the architecture described in paper A.

Paper D describes empirical scalability testing of the simulation platform proposed in paper C; the method applied is factorial experimental design.

Paper E describes information gathering for enabling data mining in massively multiplayer online games together with comparisons to web usage logs.

Paper F describes an algorithm for incremental proximal support vector classification using memoization to speed up classification with multiple classes

Paper G describes an algorithm for incremental and decremental proximal support vector classification

Paper H describes two parallel algorithms for incremental proximal support vector classification using a heap-based tree topology. The first reads data only on leaf-nodes (nodes = computational nodes) in the tree, and the second reads training data on all nodes in the tree

Paper I describes empirical comparisons of the classifier proposed in paper F with existing classifiers such as voted perceptron, c4.5, naive bayes, logistic regression and SVM. Datasets used are data sets from the UCI Machine Learning Repository (Blake and Merz [1998]), an actual web usage log and simulated game usage logs using the simulator described in paper C.

Paper	**User Rep.**	**User Pred.**
A	X	
B	X	X
C	X	
D	X	
E	X	X
F		X
G		X
H		X
I		X

Table 1.3: Relation - Papers and Thesis Topics

Chapter 2

Materials and Methods

This chapter will present the materials and methods applied in the work leading to this thesis. The field of study, computer science, influences the choice of materials and methods

2.1 Materials

Software for *Data Analysis* has been used to investigate data from *Classification* and *Simulation*. *Presentation* software has been used to create graphics for publications and the thesis as well as presentations. *Programming* software has been used to implement ideas presented in this thesis. *Word Processing* has been used to write this thesis, its publications and related reports.

Classification - libSVM, WEKA and IncRidge

Data Analysis - Minitab, Excel, R/RPy and SigmaPlot

Presentation - Visio, Illustrator, PowerPoint and FoilTeX

Programming - Python, C/C++, Java, Perl and PHP

Simulation - Zereal

Word Processing - PDFLatex and Word

Hardware

Desktop - 1GB RAM, 1.4GHz CPU, MS Win XP

Laptop - 1GB RAM, 1GHz CPU, Redhat Linux

Parallel Cluster - 1-2 GB RAM per CPU, Sorceror Linux

2.2 Methods

A quantitative experimental approach together with traditional computer science methods such as prototyping and (parallel) algorithmic design has been used. We applied statistical methods (briefly described below) for quantitative analysis.

Hypothesis testing

In order to test research hypotheses and say something about the significance of empirical results we have applied statistical hypothesis tests. The most frequently used have been the *t-test* together with output from *10-fold cross validation* experiments in order to compare the average value (in our work: classification accuracy and computational performance), Cohen [1995]. In paper I where we compared computational performance and classification accuracy between our proposed classifier with other classifiers on *several* datasets, we used pairwise t-tests on the experimental output from *all* applicable datasets in order to test how the classifiers compared overall, Lange [1999].

Factorial Design

In order to test how parameters of the massively multiplayer online game simulator interacted and controlled the computational performance we used *a full factorial design*, Montgomery [1997]. The result from the factorial design is shown in paper D.

Chapter 3

Web Intelligence

This chapter will present an overview of the Web Intelligence field together with descriptions of its subtopics of particular relevance for this thesis. We first present the Web Intelligence research field and an overview of its subtopics. Secondly we present aspects of the *Software Agents* (Web Agents) subtopic, in particular in a mobile commerce and massively multiplayer online games context. Finally we present aspects of Classification with emphasis on the proximal support vector classifiers.

3.1 Introduction

Web Intelligence (WI) was conceptualized as a scientific research field in year 2000 by Zhong et al. [2000]. It covers research on the exploration and impact of Artificial Intelligence (AI) and advanced Information Technology (IT) applied in Web-related systems, products, services and activities (Zhong et al. [2003]), e.g. e-commerce sites and online games. The WI-AI/IT relation is bijective: AI and IT can be applied for WI, and WI introduces new problems and challenges to AI and IT.

An underlying motivation for establishing the WI field is to develop the *Wisdom Web*. The purpose of Wisdom Web is to enable web-related solutions that are *wise*, i.e. have the capability of using information and knowledge to the best means learned from experience.

Web Intelligence is a research area with nine main topics, see figure 3.1 an overview. Four of these topics - marked with color in the figure - are within the scope of this thesis. Yellow means the topic is *highly related* and light blue means that the topic is *somewhat related* to the topics of this thesis

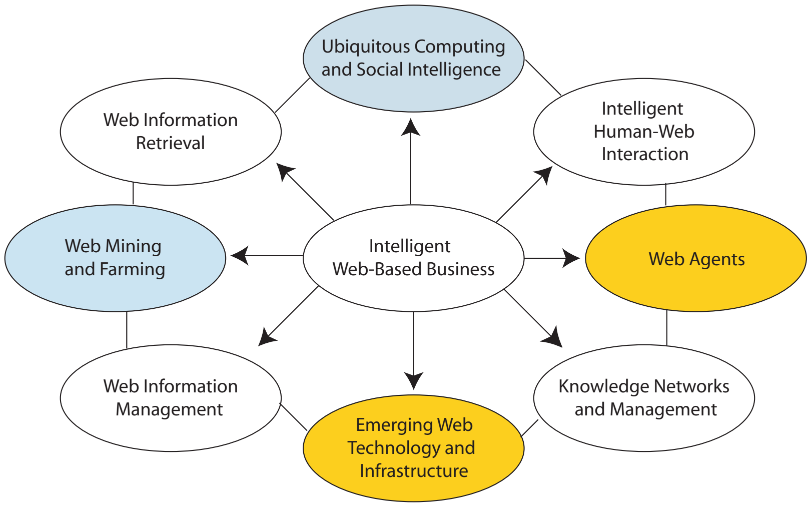

Ubiquitous Computing and Social Intelligence
Web Information Retrieval
Intelligent Human-Web Interaction
Web Mining and Farming
Intelligent Web-Based Business
Web Agents
Web Information Management
Emerging Web Technology and Infrastructure
Knowledge Networks and Management

Figure 3.1: Research Topics of Web Intelligence

Ubiquitous Computing and Social Intelligence covers web intelligence research issues related to ubiquitous web access, e.g. accessing the wireless web using mobile devices. Two examples are:

1. method for adaptive personalization of content for mobile users, Billsus et al. [2002].
2. personalizing a web site for cellular phones by removing redundant links for each user, Kobayashi and Fujioka [2003]

This topic is relevant to the thesis since paper A presents an agent-based platform for the support of *ubiquitous access to mobile commerce services*, and paper B proposes peer-to-peer based product recommendation functionality to the platform.

Web Mining and Farming covers data mining of user profiles and web logs. Two examples are:

1. Detection of correlations between customers and products on e-commerce web sites, Eirinaki and Vazirgiannis [2003]
2. Personalized search engines by altering the ranking of search results based on data mining of click-through data, Kieldaas [2000]

This topic is relevant to the thesis since paper I applies the classifier proposed in paper F for data mining of click-streams from a web usage logs and a synthetic game usage log. Paper E compares web usage mining with data mining in massively multiplayer online games, and proposes game usage logs for the latter approach.

Web Agents covers agent technology, recommender systems and automatic e-mail filtering. Two examples are:

1. Neural Networks for automatic e-mail and anti-spam filtering, Clark et al. [2003]
2. How animations and software agent technology can be used to improve web-based education, Zhong et al. [2003]

This topic is relevant to the thesis since paper A and B utilize agent technology for wireless services, and massively multiplayer online games paper C. Paper D presents empirical performance evaluation of the simulator presented in paper C.

Emerging Web Technology and Infrastructure covers grid computing, peer-to-peer computing, soft computing (e.g. machine learning techniques for classification, association rules and clustering) and web document prefetching. Two examples are:

1. Evolutionary algorithms for real-time prediction of user click-streams, Bonino et al. [2003]
2. Classification of topic-specific web sites using Hidden Markov Tree models, Tian et al. [2003]

This topic is relevant to the thesis since paper F, G and H present approaches for incrementally learning classifiers. The classification algorithm proposed in F is empirically compared to existing classifiers in paper I. Paper B is related to peer-to-peer computing.

In the next sections we first go into detail about software agents in mobile commerce and multiplayer online game settings (i.e. belonging to the web agents topic), and second into classification based on soft computing (i.e. belonging to the emerging web technology and infrastructure topic)

3.2 Software Agents

This section describes the foundation of software agents and then continues with descriptions of software agents in mobile commerce and multiplayer online games. The first part of this section is based on Tveit [2001a].

Agent Terminology

Let us start with explaining what an agent is. An agent is described in Merriam Webster's dictionary as - "*one who is authorized to act for or in the place of another*". When talking about agents in a distributed AI (DAI) context the word *agents* usually refers to entities operating in a software environment, so in order to discern them from other types of agents such as biological agents, they are frequently called *software agents*. A *Web agent* or *Internet Agent* is a software agent operating in the World Wide Web or the Internet. A *Mobile Agent* refers to the agent's movement capabilities: It can move between different environments. An *Intelligent Agent* refers to the (simulated) behavioral and intellectual capabilities of the agent. (Intelligent behavior is the selection of actions based on knowledge). If agents' primary goal is to assist users they are frequently called *Interface Agents* (interfacing the user), Maes [1994]. An excellent overview of software agent types is presented by Nwana [1996].

Whenever the word agent is mentioned in the rest of this section we refer to software agents. In our work we apply interface agents (paper A and B) and mobile agents (paper C).

Examples of Agents

The cyberspace is full of agents of various types, some of them are:

1. Web spiders (collecting data to build indexes to used by a search engine, i.e. Google)
2. Computer viruses and worms (destructive agents)
3. Artificial players or actors in computer games and simulations (e.g. Quake)
4. Trading and negotiation agents (e.g. the auction agent at EBay)
5. The animated paper-clip agent in Microsoft Office (figure 3.2)

Figure 3.2: The Animated Paper-clip Agent

Classification of Agents

A common classification scheme of intelligent agents is the weak and strong notion of agency described by Wooldridge and Jennings [1995]. In the weak notion of agency, agents have their own will (autonomy), they are able to interact with each other (social ability), they respond to stimulus (reactivity), and they take initiative (pro-activity). In the strong notion of agency the weak notions of agency are preserved, in addition agents can move around (mobility), they are truthful (veracity), they do what they're told to do (benevolence), and they will perform in an optimal manner to achieve goals (rationality).

Agents for Web Intelligence

Software Agents or Web Agents have previously been used numerous times in Web Intelligence applications throughout the last decade, e.g. the agent Letizia that supports Web-browsing and the ContactFinder agent that answer bulletin board questions, Mladenic [1999].

Scalable Agent Systems

Agent systems supporting Web Intelligence applications need to be scalable in order to handle thousands of concurrent agents representing users. Underneath two different views on scalability requirements for agent systems are described, the second complements the first by porposing that the dimensions of an agent system should be adaptive.

Multi-agent systems must be *self-building* (automatically find the optimal organization at runtime) and *adaptive* (change this organizational structure according to changes in the environment) in order to be scalable, Turner and Jennings [2001].

Multi-agent systems need to scale in at least 4 dimensions (Omer F. Rana and K. Stout [2000]):

1. Changes in the *number of agents* on *one* platform
2. Changes in the *number of agents* on *a set of* connected platforms
3. Changes in the *size of the data* the agents are operating on
4. Changes in the *diversity of the agents*

The agent-based simulator of massively multiplayer online games presented in paper C supports the 4 dimensions of scalability, but it doesn't support dynamic organizational scalability methods like self-building and adaptivity.

3.2.1 Agents for Mobile Commerce and Services

Agents in mobile commerce and services are frequently of type interface agents that have the purpose of supporting human users of mobile devices. The agent's runtime environment can either be on the mobile device, within the mobile commerce or service system, or at other computer systems such as the user's own PC.

Some examples of agents in mobile commerce and services are:

- mobile agents supporting mobile users that frequently switch between using offline, wireless and wired computers, Gray et al. [1996]
- proxy agents that support the user in browsing of mobile services , Joshi [2000b]
- agents that provide information based on a user's geographical proximity to the mobile services, Finin et al. [2002]; Joshi et al. [2002]
- mobile agents that help users of wireless devices to more easily find and download the software they need, Mena et al. [2002]
- self-organizing and self-configuring agent system supporting mobile devices in an ad hoc wireless environment, Wickramasinghe et al. [2003]
- agent-based web service provisioning for mobile users, Maamar et al. [2004]

In paper A (Matskin and Tveit [2001]) we focus on developing an agent platform supporting mobile users of valued customer membership and subscription services. The platform's core functionality is customization and adaptation of mobile commerce services, and pro-active processing and notification of important events.

The architecture in paper A is further extended in paper B (Tveit [2001c]), the new functionality is a proposed agent-based peer-to-peer approach for product and service recommendation for mobile customers.

3.2.2 Agent-based Simulation of MMOGs

Modeling and simulation of real-world (physical) phenomena is most frequently being done by numerical solutions of partial differential equations (PDE), but in recent years agent-based simulation has shown to be a viable alternative to PDE-based simulation, in particular when simulating individuals in dynamic environments or “messy” systems, Parunak et al. [1998]; Moss [2000].

Agent-based simulation techniques have shown to be useful for social, biological and economical systems, typically using the SWARM system, Hiebeler [1994]. Some examples are agent-based simulation of a suburban sprawl (Rand et al. [2003], agent-based computational economics (Tesfatsion [2002]) and agent-based modeling of biological communities (Hraber and Milne [1996]).

In a Web Intelligence setting multi-agent-based simulation techniques has e.g. been used to model and simulate network security attacks (Gorodetski et al. [2003]), and modeling and simulating of (small world) social networks of web surfers (Haridi [2002]).

Agent-based modeling and simulation for multiplayer games has up till now been primarily focusing on the modeling of intelligent non-personal characters in games with a relatively low number of players and non-personal characters (typically less than 100), e.g. the work on developing intelligent opponents in Quake, Laird [2001]. Other examples include agents that play the Netrek online battle simulation game (Huber and Hadley [1997]), and complexity comparison of a state-machine model with an agent-based BDI model for modeling intelligence in games (Bartish and Thevathayan [2002]).

Probably due to the relative novelty (mid 1990s) of MMOGs there has so far been little published work on (agent-based) simulation of them. A partially related areas with more activity include animation of large crowds for movies (e.g. the MASSIVE system for simulating a large orc army in the Lord of Rings trilogy). One exception is the FreeMMG platform that simulates players of massively multiplayer online games using agents in a hybrid peer-to-peer and client-server multiplayer game simulation model Cecin et al. [2003]. FreeMMG’s purpose is to provide a generic simulation platform that allows cheaper and easier performance and reliability testing of MMOGs.

In paper E (Tveit [2002a]) we propose a data mining approach for MMOGs (*game mining*) by drawing parallels to the established field of web mining. This approach is partially implemented in the MMOG agent-based simulation platform called Zereal presented in paper C (Tveit et al. [2003b]), and empirically tested for computational performance in paper D (Tveit [2003b]) and preliminary testing of player category classification from Zereal logs in paper I (Tveit [2003a]).

3.3 Classification

Classification is an everyday task, some examples are

- Selecting one of several directions (e.g. left, north or up) based on past experience, perception and the navigational goal.
- Recycling your garbage (what material to put in which bin) based on garbage characteristics (e.g. paper, plastic or organic).
- Interpreting symbols with *several fonts and styles* as characters in text (e.g. **bold**, *italic*, Times New Roman or Courier), or in other words *reading*. Search Engines submission pages use such tests to discern between (unwanted) link submissions by software agents and (wanted) submissions by people.

The class or concept is the selected outcome (set of directions, types of bins, or characters in the alphabet for the above examples). Selection of the class is based on apriori knowledge acquired through training or experience.

From a data analysis perspective, classification can be defined as below:

Definition 3.1 (Classification, Han and Kamber [2001]). Classification *is the process of finding a set of* **models** *(or functions) that describe and distinguish data classes or concepts, for the purpose of being able to use the model to predict the class of object whose class is unknown. The derived model is based on the analysis of the set of* **training data** *(i.e., data objects whose class label is unknown)*

The classification model is usually found by using one or several **classification algorithms** (classifiers) that processes the training data. The training data are frequently in the form of **feature vectors**.

3.3.1 Typology

Classifiers are either (Rubinstein and Hastie [1997]):

Informative - model the densities of classes and select the class that most likely produce the features. Naive Bayes, Hidden Markov Models and Fisher Discriminant Analysis are examples of informative classifiers.

Discriminative - model the class boundary and membership probability directly. Logistic Regression, C4.5, Artificial Neural Networks, Support Vector Machines and Generalized Additive Models are examples of discriminative classifiers. *The classifiers proposed in paper F, G and H are discriminative.*

3.3.2 Training Methods

Classifiers can be trained using (Duda et al. [2001]):

Batch Training - *all* data from the training data set are presented to the classifier in the training process (this includes historic data in the re-training case).

Stochastic Training - data used in the training process are randomly selected from the training data set, i.e. training data can be considered to be stochastic variables. This training method is used when there are large redundant data sets.

Incremental Training - data are only used once in the training process, i.e. historic data is not used in the training process when the training data grows. If the classifier using incremental training is kept up-to-date at all times it is also called **online training**. This training method is used when the growth and amount of data is so large that storing them is too expensive. *The classifiers proposed in paper F, G and H support incremental and online training.*

Decremental Training - In domains where classification data grows rapidly, **the classes or concepts might change over time**; this is called **concept drift**, Widmer and Kubat [1996]. The corresponding learning task is called **learning drifting concepts**, Schwefel et al. [2003]. In order to efficiently adapt to concept drift the classifier must unlearn the *old concepts*, this is called forgetting or decremental training, Widmer and Kubat [1996]. *The classifier proposed in paper G support decremental training for learning drifting concepts.*

3.3.3 Linear and Nonlinear Classifiers

Classifiers that can discriminate between classes of non-rectangular shapes are called **nonlinear classifiers**, Support Vector Machines and Artificial Neural Networks using nonlinear **kernel** functions are examples of nonlinear classifiers. See figure 3.3.3 for an example of a nonlinear classification problem. The *XOR problem* is a well-known example of a non-rectangular classification problem; it can be solved with a nonlinear classifier but not with a linear classifier. Examples of **linear classifiers** are C4.5, Logistic Regression and Naive Bayes, Goldman and Axtell [1995]. *The classifiers proposed in paper F, G and H are linear.*

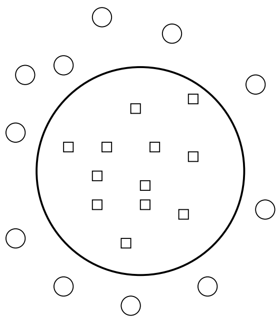


Figure 3.3: A Nonlinear Classification Problem

3.3.4 Sequential and Parallel Classifier Algorithms

Most classifier algorithms are of **sequential** nature , meaning that they can only process one command at a time. With current processor architectures this is only partially true due to multiple pipelines with parallel execution of commands, but the algorithms are usually not designed to take full advantage of parallel execution. The main motivation for designing classifier algorithms of **parallel** nature are to increase the speed of execution and to potentially handle larger classification problems. Examples of parallel classifier algorithms include the SPRINT decision-tree-based classifier by Shafer et al. [1996a] and the Kerneltron hardware Support Vector Machine implementation by Genov and Cauwenberghs [2003].

3.3.5 Important Theorems

The following two theorems provide a sober view on classifier and feature extraction approaches.

No Free Lunch Theorem

According to the *No Free Lunch Theorem* (Duda et al. [2001]), no matter which classifier algorithm is used there exists at least one data distribution where *random guessing is better*. Or in other words: even the “most fancy” nonlinear classifier is not the best for all occasions.

Ugly Duckling Theorem

According to the *Ugly Duckling Theorem* by Duda et al. [2001], there exists no problem-independent way of determining the best set of features. Or in other words: there exist no optimal feature extraction and discretization methods covering *all* problems.

3.3.6 Classifiers for Web Intelligence

Systems and services on the Internet continuously generate *large amounts* of log data, e.g. web servers generate usage logs with click-stream data representing the behavior of online users. If classifiers will be used to handle such data they need to be *scalable* and *incremental*.

The process of predicting behavior based on web usage logs can be considered a classification problem. If the behavior is relatively predictable, it can be used to in-advance personalization of web content based on the user's preferences or pre-fetching of web content to the user's browser cache for improved browsing performance. *Why is this a classification problem?* The next click to an HTML page or multimedia object in a user session can be seen as a *class*, and the previous pages or multimedia objects visited can be seen as a *feature vector*. In order to get high accuracy of next-click classification, it has shown to be beneficial to utilize the inherent graph structure of web sites and have one classifier per web page (paper I).

Requirements for Classifiers in Web Intelligence

Classifiers for Web Intelligence purposes need to have the following properties:

Scalable - Scale well in terms of relatively fast handling large amounts of data by efficiently utilizing computational resources (memory and CPUs). Both the *training* and *classification* process should be scalable.

Incremental - Support incremental (on-line) training since it doesn't have enough time to *re-train* with old data every time new data arrives

Accurate - Provide (in general) high accuracy on the selected problem

Decremental - Can handle drifting concepts since concepts/classes in Web Intelligence applications are not likely to be static over time

Handle non-orthogonal examples - Non-orthogonality is not well handled by some classifiers (i.e. repeated occurences of training data), in a Web Intelligence this is likely to occur and must be properly handled by the classifier algorithm.

Handle Dependency between Features - Some classifiers perform poorly when features are (somewhat) dependent, in Web Intelligence problems where the features are clicks in a click-stream they are likely to be (somewhat) dependent, and this has to be handled by the classifier algorithm.

Naive Bayes is frequently being selected as the default classifier in Web Intelligence; it has been used for e-mail spam detection, text classification, web search and recommender systems. It is relatively scalable (mostly computationally cheap table operations), but it uses a lot of memory in cases where features are continuous or have many value levels. The memory requirements of Naive Bayes can be reduced using discretization techniques, but this may result in reduced classification accuracy. Naive Bayes also assumes that feature values are conditionally independent given the target value (Mitchell [1997]) . *So how can we develop classifiers that meet the above requirements?*

3.3.7 Incremental Proximal Support Vector Classifiers

The theory of *least-squares regression* was first published by Adrien Marie Legendre in 1805, and it was further developed into a statistical tool based on probabilistic theory by Karl Friedrich Gauss in 1809, Christiani and Shawe-Taylor [2000].

Least-squares regression performs poorly when the training vectors are non-orthogonal, i.e. if two distinct vectors X_j, X_k satisfy the relation $X_j = a - bX_k$ when a, b are scalars, Upton and Cook [2002]). *Ridge regression* was introduced to deal with non-orthogonal least-square regression problems, Hoerl and Kennard [1970]. The basic idea of ridge regression is to add a *ridge parameter* to the diagonal of the matrix with training vectors in least-squares regression, if the ridge parameter is nonzero it guarantees the orthogonality of the new matrix, i.e. providing a non-singular regression system that can be solved. Ridge regression is also considered a type of *shrinkage regression.*

Ridge regression is a specialization of *Tikhonov Regularization* using a square loss function (Tikhonov [1963]; Tikhonov and Arsenin [1977]), this also explains the synonym term for ridge regression - *regularized least squares regression* (RLSR). Other specializations of Tikhonov regularization include *support vector machine regression* (SVMR) (using an ϵ-insensitive loss function) and *support vector machine classification* (SVMC) (using a hinge loss function), Vapnik [1999]; Christiani and Shawe-Taylor [2000]; Rifkin [2002].

RLSR, SVMR and SVMC all support kernel mappings in order to handle nonlinear regression (RLSR,SVMR) and classification (SVMC) problems, the purpose of a kernel is to map the nonlinear classification problem into a higher dimensional space where it becomes a linear classification problem. *Regularized* *least squares classification* (RLSC) was first introduced as a type of *neural network* by Poggio and Girosi [1990b], and later extended to handle outliers and negative examples, Girosi et al. [1991]. Poggio and Girosi also showed the equivalence of nonlinear regularization algorithms with multilayer networks, Poggio and Girosi [1990a].

In Bishop [1995], regularization for neural networks is called *weight decay*, and the linear model of weight decay is called *jitter*. Jitter is equivalent to ridge regression.

Fung and Mangasarian introduced the *proximal support vector machine classifier* (PSVMC), Fung and Mangasarian [2001b]. They later introduced algorithms for *multicategory PSVMC* (Fung and Mangasarian [2001a]) and *incremental and decremental PSVMC* (Fung and Mangasarian [2002]). The PSVMC was developed by relaxing the constraints of the ordinary SVMC quadratic optimization problem.

Poggio's PhD student Rifkin showed that the PSVMC is equivalent to a RLSC, and that Fung and Mangasarians two main contributions were 1) very fast ways of computing the linear RLSC and 2) empirical evidence that RLSC have approximately the same classification accuracy as SVMC on benchmark datasets. It was also proved that the SVMC and RLSC have the same generalization bounds, i.e. theoretically supporting the prior empirical evidence, Rifkin [2002]; Rifkin et al. [2003]. Agarwal showed that PSVMC can be transformed into classification using ridge regression, Agarwal [2002] (This is also supported by the above-mentioned relation between ridge regression and regularization).

3.3.8 Our work on incremental PSVM classifiers

The incremental PSVMC proposed by Fung and Mangasarian [2002] showed promising performance and efficient memory utilization; results making it suitable in web intelligence applications, but could it be further improved to

1. efficiently handle incremental classification with multiple categories
2. have more efficient support for decremental learning
3. be efficiently parallelizable in order to handle *very large* classification problems common in cyberspace services (e.g. clickstream prediction on large web sites)

In order to deal with requirement 1 we continued the development of the incremental PSVMC (i.e. RLSC) algorithms proposed by Fung and Mangasarian [2002]. In paper F (Tveit and Hetland [2003]) we proposed memoization in order to add efficient support for incremental multicategory classification with PSVMC.

We tried to handle requirement 2 in paper G (Tveit et al. [2003a]) by proposing an exponential soft-decay method for handling decremental learning for the PSVMC, and showed empirically that it was significantly faster than Fung and Mangasarian's approach.

We handled requirement 3 in paper H (Tveit and Engum [2003]) by parallelizing the incremental PSVMC using two methods based on a heap-based tree topology of processing nodes.

Part II

Synopsis

Chapter 4

Results

This chapter summarizes the results presented in the papers. Each paper is first described briefly together with its relation to the other papers and its relation to the research questions.

Paper	**RQ1**	**RQ2**	**RQ3**
A	X		
B	X		
C		X	
D		X	
E		X	
F			X
G			X
H			X
I	X	X	X

Table 4.1: Relation - Research Questions and Papers

Paper A - Mobile Commerce Agents in WAP-based Services

Paper A presents a conceptual software agent architecture for supporting mobile services and users. The basic idea is to let the software assistant agent that represent a user stay in the wired network in order to reduce the effect of resource constraints on mobile devices. A prototype of the architecture was developed in order to do a proof of concept test. This paper tries to answer research questions

RQ1.1-RQ1.4

Paper B - Peer-to-Peer based Recommendations for Mobile Commerce

Paper B describes a peer-to-peer based recommender system extension to the architecture presented in *paper A*. The basic idea is to provide a distributed version of collaborative filtering where each peer-to-peer query is a vector with votes on products and services. This paper tries to answer research question *RQ1.5*.

Paper C - Scalable Agent-Based Simulation of Players in Massively Multiplayer Online Games

Paper C describes an implemented parallel mobile agent platform for the simulation of users (players) in massively multiplayer online games, This platform is partially based on the architecture described in *paper A and B*, but this platform is geared towards *one* particular service, i.e. massively multiplayer online games. This paper tries to answer research question *RQ2.1*.

Paper D - Empirical Performance Evaluation of the Zereal Massively Multiplayer Online Game Simulator

Paper D describes empirical scalability testing of the simulation platform proposed in *paper C*, the method applied is factorial experimental design from statistics. This paper supports paper C in answering research question *RQ2.1*.

Paper E - Game Usage Mining: Information Gathering for Knowledge Discovery in Massively Multiplayer Online Games

Paper E presents a taxonomy of computer games from a data mining viewpoint, a taxonomy of data mining approaches for massively multiplayer online games inspired by existing the web mining taxonomy, a comparison of information gathering for web usage mining with the proposed *game usage mining* approaches, and finally a proposal for logging of player behavior in massively multiplayer online games. The presented logging approach is partially implemented in *paper C*. This paper tries to answer research questions *RQ2.2-RQ2.4*.

Paper F - Multicategory Incremental Proximal Support Vector Classifiers

Paper F describes an algorithm for incremental proximal support vector classification using memoization to speed up classification with multiple classes. The algorithm is empirically compared to an approach without the use of memoization and empirically shown to be faster. The purpose of this classifier is to provide scalable and memory efficient support prediction of action or clicks on the (mobile) web (*paper A and B*) and massively multiplayer online games (*paper C and E*). This paper tries to answer *RQ3.2*.

Paper G - Incremental and Decremental Proximal Support Vector Classification using Decay Coefficients

Paper G is an extension of the results in *paper F* for doing computationally and memory efficient *decremental* training of the incremental proximal SVM classifier. This paper tries to answer research question *RQ3.3*

Paper H - Parallelization of the Incremental Proximal Support Vector Machine Classifier using a Heap-based Tree Topology

Paper H is a parallelization of the algorithm presented in *paper F*; the purpose is to handle even larger amounts of increasingly growing classification data (in cyberspace services). It tries to go further than paper F in answering research question *RQ3.2*

Paper I - Empirical Comparison of Accuracy and Performance for the MIPSVM Classifier with Existing Classifiers

Paper I presents an empirical comparison of computational performance and classification accuracy of the classifier proposed in *paper F* with existing classifiers. This paper tries to answer research question *RQ3.4*

Paper	**Agents**	**Algorithms**	**Data Mining**	**Simulation**
A	X			
B	(X)	(X)	(X)	
C	X		(X)	X
D	(X)			X
E			X	
F		X	(X)	
G		X	(X)	
H		X	(X)	
I	(X)	(X)	X	(X)

Table 4.2: Papers and their topics

Chapter 5

Evaluation

This chapter evaluates the thesis by considering the answers to the *research questions*, the *contributions* and how these have been addressed in the papers, the *contextual description*, the citations and educational use of papers leading to this thesis, and finally the *lessons learned* from this work.

5.1 Research Questions

The main research question, which was presented in Chapter 1, is:

MRQ: How can we in a scalable computational manner provide methods for user representation and behavior prediction for increased customization of cyberspace services, e.g. mobile commerce and massively multiplayer online game services? (Examples of customization include prediction of interesting cyberspace service content and automatic provision of recommendations of products and services)

The main research question **MRQ** has been answered, in general, by proposing a conceptual software agent architecture for supporting general mobile commerce services, and later peer-to-peer based distributed collaborative filtering for recommendations in such services. *Scalable manner* has been handled by adapting and simulated empirical testing the architecture for a particular type of service - massively multiplayer online games. Prediction behavior has been handled in a scalable manner by empirical testing of the proposed incremental, decremental and parallel classifier algorithms. Finally the empirical results from applying the proposed incremental classifier applied on real web (from a web log) and synthetic (from the developed MMOG simulator) datasets provide a closure.

Our answers to the underlying and more specific research questions are as follows:

RQ1: How can mobile commerce customers be supported by software agents?

1. Resource constraints such as limited bandwidth, memory, processing power and energy common on mobile devices, is handled by letting the software assistant agents represent the *wireless* users from the *wired* network, hence reducing the load on the mobile device used by the wireless user.
2. The customer's profile and interest data are collected, represented and stored in the internals of the software assistant agent. Usage of this data can potentially be controlled by privacy-preserving protocols such as P3P, with services negotiating with the software assistant agent about getting access to the customer data in order to target the catering of service content.
3. Customers can get product or service recommendations by extending the software assistant architecture to support distributed collaborative filtering, sending purchase patterns to "agent friends" in a peer-to-peer manner.
4. The software assistant agent architecture can support various types of mobile services; it was designed with entertainment and games, financial services, news and m-commerce stores in mind.
5. The software assistant architecture can be generalized to handle electronic commerce in addition to mobile commerce since the software assistant agent is available in the *wired* network where electronic commerce takes place. An additional benefit is that the software assistant agent's gathering of customers profile and interest data from mobile commerce services can be used for the electronic commerce services.

RQ2: How to test the proposed solution to **RQ1**, and extending it towards supporting Massively Multiplayer Online Games?

1. A Simulation of Massively Multiplayer Online Game is a relevant test case because MMOGs are probably among the largest and most complex cyberspace services currently available, and can then provide an estimated "upper bound" for the load of other cyberspace services, both in terms of the richness of the services (thousands to millions of tasks to perform and digital locations to visit), and the sheer size (hundreds of thousands of *concurrent* and communicating users in a very large virtual world).
2. Data for Customer Personalization in Massively Multiplayer Online Games can be gathered by logging the actions performed by the user and the avatar controlled by the user. In addition to the time, position

(virtual for the avatar and real for the user), other avatars or non-personal characters involved in the action, and the players preferences, paper E provides the details. But also contextual data such as the *player's input* in form of multimedia and response time may have to be logged; logging of contextual data for customer personalization in MMOGs I claim to be relatively little studied in the international research society.

3. The main differences between usage data gathering in Massively Multiplayer Online Games and the existing Web Usage Logs are that there are no existing standards for MMOG usage logs, better identification of users in MMOGs than on the Web, much more rapid user actions in MMOGs than on the Web (i.e. combat versus browsing), MMOGs usually have more about their users from registration than on the Web, Web Logs have a more limited set of actions than in games (i.e. the HTTP protocol's GET, POST and HEAD)

4. Customers of Massively Multiplayer Online Games usually play because they *enjoy* playing and communicating with other players. If players become *bored* they are likely to unsubscribe, so *automatic boredom detection* followed up by *automatic boredom prevention* are examples of customer personalization in MMOGs. Another issue is *balancing* and fairness: if some item, skill or player type becomes superior to all others (i.e. no rock-scissor-paper like property), the game can be considered unbalanced. In cooperation with students with extensive MMOG playing experience we are working with the detection of balancing problems using visual data mining and dynamic ontologies, Rødseth and Breivik [2003]

RQ3: How can classification algorithms be used for customization of cyberspace services, e.g. for mobile commerce and massively multiplayer online games?

1. Classification can for example be used to predict the next link a user clicks on, e.g. a link to a purchase or content page on the (mobile) web. Classifiers can be trained by the user's own and other users historic path of navigation. These paths are frequently called user click-streams; this has been empirically tested on click-streams extracted from an actual web usage log in paper I. If there exists information about the player types (e.g. killer or explorer) for some of the players in a massively multiplayer online game, this can potentially be used as a training set in order to classify players with unknown player type; this approach has been empirically tested on simulated data in paper I.

2. Scaling up can in general either be done by reducing the problem size (e.g. by using samples of data), increasing the processing resource

(e.g. by buying more powerful hardware), or by improving the efficiency of the underlying algorithms. We have focused on the latter approach by making the classifier incremental (i.e. not having to retrain with *all data* every time new data arrives) and efficiently support a large number of classes (*Cyberspace services continuously generate a large amount of user data and are plentiful with respect to the possible navigational directions that corresponds to classes in the classifier*), making it decremental in order to efficiently "forget" outdated data (*Cyberspace services change over time*), and finally by parallelizing the algorithm in order to make it run efficiently on relatively cheaply available PC-based parallel cluster systems (Cyberspace services are frequently very large).

3. When classification data changes over time it is called concept drift, when this occurs the classifier must *unlearn* obsolete training data; this type of training is called decremental training or simply *forgetting.*
4. Classification performance, both computational performance and classification accuracy, can be empirically compared to other state-of-the-art classifiers using cross-validation (see materials and methods chapter) and several datasets. We selected to compare our C++-based classifier to the Java-based Weka toolkit since there exists many claims that using Java and C++ give similar computational performance Mangione [1998]; Wilson [2003]. The main results (using paired t-tests on results from all datasets) were that our classifier outperformed the WEKA classifiers by one order of magnitude or more for the computational performance. For classification accuracy we found no significant difference from the WEKA-classifiers, with the exception of the naive bayes classifier. The reason that our classifier is more accurate than naive bayes is that click-stream data tends to have relations between attributes, this characteristic disputes the assumptions of independence between attributes that naive bayes has.

5.2 Contributions

A summary of the contributions are given below:

C1: A conceptual solution and a supporting platform for implementing and using personal software assistant agents in mobile commerce services, focusing on subscription and valued customer membership services. The solution is aimed towards relaxing the restrictions of mobile devices and wireless communications (Paper A).

This can be considered as a contribution to the mobile commerce research community (cited by Chang [2002]; Turban et al. [2002]; Vrechopoulos et al.

[2002]). Its contribution to the state-of-the art is that it is very likely to be the first published platform (September 2001) of interface software agents residing between the mobile commerce service and the users where the agents support the users in accessing subscription and valued customer membership services. Our published work resembles the work by Joshi [2000a], but differs since his work focuses on supporting web access (i.e. browsing) from mobile platforms while our work focuses on supporting personalization, reasonable privacy, and flexibility when accessing mobile commerce services from mobile platforms. The YellowStone project (Reticular Systems) also resembles our work but differs since it focuses on using agents to support mobile users in accessing traditional geographically bound services such as restaurants, theaters and sports events; our work is on using agents to support pure virtual services such as stock market information, mobile commerce stores and games.

C2: A conceptual peer-to-peer extension of the platform in C1 for supporting scalable and distributed product and service recommendations for mobile commerce customers (Paper B).

This can be considered as a contribution to the mobile commerce research community (cited by Heinemann et al. [2003a,b]; Munusamy and Leang [2002]; Ding et al. [2002]; Ding and Unnithan [2002]; Nguyen [2002]), the peer-to-peer research community (cited by Lee [2002]), and the recommender system subcommunity of the information retrieval research community (cited by Svensson [2003]; Olsson [2003]; Weiss [2002a,b]). We believe this contributes to the state-of-the-art by very likely being the first published work (July 2001) proposing a peer-to-peer based recommender system. This is not only based on our own litterature study, but also from the publications that later cited our work. A related area is that of trust in peer-to-peer systems, but it differs from our approach by not focusing on recommendations on its own, but rather on how trustworthy the peers of a peer-to-peer network is.

C3: A scalable platform (called Zereal) for simulating customers of a particular kind of m/e-commerce service - Massively Multiplayer Online Games (Platform described in paper C and the related empirical performance evaluation in paper D). This can be considered as a contribution to the computer game research community (cited by Ho et al. [2003]; Thawonmas et al. [2003]; Ho and Thawonmas [2004]), and as a technical contribution to the agent-based simulation research society by providing a scalable platform for running on parallel clusters supporting the message passing interface (MPI).

We believe this contributes to the state-of-the-art by being the only simulation platform for Massively Multiplayer Online Games that is geared towards experiments towards improved player personalization based on game usage mining. The FreeMMG MMOG simulator presented by Cecin et al.

[2003] resembles Zereal but is different in the sense that: 1) FreeMMG is not developed to run on using MPI-based parallel clusters, and 2) FreeMMG is not developed for experimenting with player personalization based game usage mining.

C4: Investigation and proposal of requirements for doing customer personalization in Massively Multiplayer Online Game services. This includes the creation of a proposed new data mining subfield called *Game Mining* covering data mining of MMOGs, a classification of computer games from a data mining viewpoint, a *comparison* of information gathering for web usage mining and game usage mining, and finally a proposal for a common game log format to enable game usage mining (Paper E).

This can be considered as a contribution to the state-of-the-art of the computer game and data mining research societies (cited by Thawonmas [2003]).

Game Mining has also been considered as new research field by Ho and Thawonmas [2004] :

> *"That leads to a new research field called Game Mining, proposed originally by Tveit et al. at NTNU (http://abiody.com/gamemining/) and later but independently by the authors' group, in which data mining techniques are exploited to improve quality of MMOGs in various aspects such as contents, design, story, cost and so forth."*

C5: Investigation and proposition of algorithms that can be used in m/e-commerce personalization, including developing classification algorithms that: Scale with a large number of classes (Paper F), utilizes parallelization (Paper H), and handles changes in classification data over time (Paper G).

This can be considered as a contribution to the data mining and machine learning society (Paper H has been cited by Liu et al. [2003]). This is believed to contribute to the state of the art by providing: 1) computationally efficient support for incremental classification of multiple classes based on memoized algorithmic extensions of the work for incrementally binary classification presented by Fung and Mangasarian [2002] (paper F), 2) more efficient computation and less storage requirements when performing decremental training (handling concept drift) than the approach proposed by Fung and Mangasarian [2002], both an exponential weight-decay reduction of previous classifier knowledge and a hybrid approach supporting both exponential weight-decay reduction together with decrements based on time windows of fixed size (paper G), 3) parallelization with empirical evidence of significant speedup for the incremental classification algorithm presented by Fung and Mangasarian [2002] using a heap-based tree topology of computational nodes (H).

Contributions	RQ1	RQ2	RQ3
C1	X		
C2	X		
C3		X	
C4		X	
C5			X

Table 5.1: Relation - Research Questions and Contributions

5.3 Contextual Description

Due to the multi-disciplinary nature of the work, it was not practically feasible give a full contextual description for this work, but the focus was on giving enough relevant information to support the papers. In particular the related work the proximal support vector classifiers (chapter 3.3.7) was very hard to get an overview of. Parts of the contextual work were taken from the included papers, some partially from other papers by the author, and the rest were written only for the thesis.

5.4 Citations of Our Papers

The number of citations of a scientific publication is frequently being used as an *impact factor* to evaluate research (Garfield [2003]), e.g. between papers in the Science Citation Index. It has also been shown that online available papers are more highly cited (Lawrence [2001]). Note that the number of citations may say little or nothing about the quality of the cited paper, but it can potentially say something about how many people who read or at least looked at the paper.

The publications leading to this thesis have knowingly been cited more than 50 times by researchers at conferences and universities in 20 countries. The most frequently cited papers are: Tveit [2001a] (partially used for chapter 3) with 33 citations, Tveit [2001c] with 15 citations, and Matskin and Tveit [2001]; Tveit et al. [2003b] with 3 citations each.

Educational Use of Our Papers

Some of the papers leading to this thesis have been used in educational settings, e.g. Tveit [2001c] was a part of the:

- Curriculum for *Peer-to-Peer (P2P) Data Management* (2003), a course at University of Leipzig (Germany)
- Recommended readings for *honors projects* (2003) at University of Carleton (Canada).

Tveit [2001a] was a part of the:

- Curriculum for *Systemes Multi-Agent* (2001-2002) at Universite de Savoie (France)
- Related material for the course on *Expert Systems and Heuristic Programming* (2002) at University of Alabama in Huntsville (USA)
- Additional references for the CS 6361 PhD course on *Requirements Engineering* (2002) at the Department of Computer Science, University of Texas at Dallas (USA)
- Related material for *Basic of Knowledge Engineering* (Spring 2002) at Department of Computer Science, Faculty of Technology, University of Vaasaa (Finland)
- Bibliography for the CS 6934 seminar (fall 2002) at the school of computing, University of Utah (USA).
- Recommended reading for course 2G1514 Distributed Artificial Intelligence and Intelligent Agents (spring 2004), Department of Microelectronics and Information Technology (IMIT), Royal Institute of Technology (Sweden).

5.5 Lessons Learned

The work presented in this thesis has been published internationally; this gave valuable feedback through the reviewing process, questions after presentations, and discussions with researchers from several universities and research institutions. The research presented showed to follow (for the author) a surprisingly hard, non-linear and multi-disciplinary path, but with the focus of the Electronic Commercial Agents (ElComAg) project in mind at all times.

Chapter 6

Conclusions and Future Work

This chapter concludes the thesis by summarizing the main results and contributions of this work.

6.1 Summary of Results and Contributions

This thesis proposed user representation and user classification approaches for increased customization (personalization) of cyberspace services. Examples of such customization include delegation of data and tasks to software agents, or automatic pre-fetching or pre-processing of service based on classification-based predictions. The cyberspace service types primarily considered Mobile Commerce (e.g. news, finance and games) and Massively Multiplayer Online Games (MMOGs)

First a conceptual software agent architecture for supporting users of mobile commerce services was presented, then the architecture was extended to support automatic product and service recommendations by proposing a peer-to-peer based distributed collaborative filtering approach.

In order to examine the scalability of the proposed conceptual software agent architecture a simulator for Massively Multiplayer Online Games was developed, MMOGs were selected because they are currently the largest and most complex cyberspace services available, hence providing an estimated "upper bound" for the performance requirements of (most) other cyberspace services using the agent architectures.

Prediction for cyberspace personalization are frequently classification problems (e.g. pre-fetching based on click-stream prediction from usage logs), and due to the services large scale and continuously changing nature classifiers need to handle this, this motivates the parallel, incremental and decremental classifiers proposed.

Finally the incremental classifier was empirically compared with other classifiers (C4.5, Logistic Regression, Voted Perceptron) on general classification data sets, user click-streams from an actual web usage log and a synthetic game usage log from the developed MMOG simulator. The incremental classifier showed to be about 1 order of magnitude (or more) faster than the classifiers compared with, and significantly more accurate than the naive bayes classifier on the selected data sets. We didn't find any significant difference between the classification accuracy of the proposed classifier and the other classifiers on the selected data sets; this may suggest that the proposed classifiers can be useful for user prediction in cyberspace services.

6.2 Directions for Future Work

Opportunities for further work on cyberspace user representation include:

- Implement and empirically test the proposed software assistant agent architecture for several simulated mobile commerce services, or even deploy it for a live service. The peer-to-peer collaborative filtering architecture can potentially be implemented and tested for the recommendation of mobile phone ringing tones and backgrounds.
- Add support for automatic pre-fetching of content based on classification-based prediction trained by simulated or actual mobile user click-streams, and potentially test whether user-positioning data can be used to improve the click-stream predictions.
- Add increased realism to the MMOG simulator by supporting: coalitions, quests and missions, simulation of narration and natural language, improved intelligence support (e.g. BDI and automatic planning), simulation of intra-game e-commerce activities (e.g. trade between players). The increased realism can then be used to test various approaches for logging and data mining of player behavior.

Opportunities for further work on cyberspace user classification include:

- Empiricallly compare with other classification-related methods, e.g. MARS (Friedman [1991]), Sprint (Shafer et al. [1996b], CBR (Aamodt and Plaza [1994]), and GP (Koza [1998]).

- Investigate efficient incremental support for non-linear kernels of the classifiers.

- Add incremental balancing mechanisms that can handle a class-wise unbalanced set of training examples.

- Add parallel support to any new features for the classifier algorithms.

- Investigate whether the linear systems involving symmetric positive definite matrices used in the proposed classifiers can be transformed into a Toeplitz-like (or Hankel-like) system. Introducing Toepliz-like systems can potentially dramatically speed up the final matrix inversion or solution of linear system that usually involves $\Theta(n^3)$ operations (where n is the number of features of the classification examples). Due to the relation between matrix inversion and matrix multiplication (Cormen et al. [1990]) the matrix inversion process can potentially be reduced to $\Theta(n^{2.7})$ operations using Strassen's method (Strassen [1969]) or even $\Theta(n^{2.36})$ operations using Winograd and Coppersmith's method (Coppersmith and Winograd [1990]), these methods have unfortunately shown to be of little practical use. For general complex and real matrices a lower bound for the size of circuit any arithmetic circuit for the product of two matrices of $\Omega(n^2\, log\ n)$ operations (Raz [2003]), so the question is: *can the inverse or solution of the linear system calculated faster than $\Theta(n^2\, log\ n)$ for the (non-general) symmetric and positive definite matrices present in our proposed classifier approaches.* The requirement is probably that one can find a fast transformation from symmetric positive definite matrix systems into a Toeplitz system that can be solved in $\Theta(n\, log\ n)$ operations, Chan and Ng [1996]. It has been hypothesized that *all* matrices are equivalent to Toeplitz matrices (Mackey et al. [1999]), counter examples falsifying that was presented by Amdeberhan and Heinig [2003], but symmetric positive definite matrices have not yet been proven to be different from Toeplitz matrices.

Part III

Papers

Paper A

```
@Article{2001:JDBM:MatskinTveit,
  author   =  {Mihhail Matskin and Amund Tveit},
  title    =  "{Mobile Commerce in WAP-based Services}",
  journal  =  {Journal of Database Management},
  year     =  {2001},
  volume   =  {12},
  number   =  {3},
  pages    =  {27--35},
  month    =  {July--September}
}
```

This paper has been cited by Chang [2002]; Turban et al. [2002]; Vrechopoulos et al. [2002].

MOBILE COMMERCE AGENTS IN WAP-BASED SERVICES

Mihhail Matskin and Amund Tveit

Department of Computer and Information Science

Norwegian University of Science and Technology

N-7491 Trondheim, NORWAY

Phones: +47 73590767, +47 73594480

Fax: +47 73594466

Emails: Mihhail.Matskin@idi.ntnu.no, Amund.Tveit@idi.ntnu.no

ABSTRACT

With the increasing number of e-commerce services for mobile devices, there are challenges in making these services more personalized and to take into account the severely constrained bandwidth and restricted user interface these devices currently provide. In this paper we consider an agent-based platform for support of mobile commerce using wireless (WAP-based) devices. Agents represent mobile device customers in the network by implementing highly personalized customer profiles. The platform allows customization and adaptation of mobile commerce services as well as pro-active processing and notification of important events. Information to the customers is delivered both via WML-decks and SMS messages. Usage of the platform is illustrated by examples of valued customer membership services and subscription services support. Some details of a prototype platform implementation are briefly considered.

KEYWORDS: Mobile services, electronic commerce, intelligent agents, wireless devices, WAP/WML

INTRODUCTION

The increasing number of mobile portable devices in use creates a great opportunity for development of a wide spectrum of mobile e-commerce services. The main advantage of these services is their high availability. Customers with a mobile device can enjoy these e-commerce services regardless of time or location. However, mobile devices, such as cellular phones and PDAs, are constrained by severe restrictions that might complicate practical use of e-commerce services. These restrictions are related to the limitations of wireless data networks when compared to wired networks (less bandwidth, more latency, lower connection stability, less predictability, and less standardized protocols) and to the limitations of mobile handsets when compared to personal computers (small screen size, complicated text input, little memory, slow CPU, and more constrained energy supply).

Additional problems with wide application of mobile e-commerce services are related to higher cost of wireless communications (compared with wired communications), and to the assumption that most users of mobile devices do not have sufficient experience of Internet or PC usage. This puts forward requirements of simplicity and expressiveness to the services.

It is possible that some of the limitations will be relaxed in the future through improved hardware or telecommunication networks technology (Tarasewich & Warkentin, 2000), but at the moment all of them should be taken into consideration when implementing mobile services.

As a basic way of relaxing the above-mentioned problems and limitations we see the following solutions:

- the connection time to the network service should be minimized,
- the precision of delivered information should be high in order to avoid exposing a large amount of useless information to be read on a small screen.

These solutions assume that as much work as possible should be done off-line without the mobile device being directly connected to the network.

Our approach towards reaching this goal is to provide a user of mobile devices with a personal software assistant that represents the customer's profile and interests in e-commerce services. In order to implement such an assistant, we use an agent-based approach and agent technology. The personal assistant operates in the Internet environment, and the users employ WAP-enabled mobile devices to communicate with their assistants to take advantage of e-commerce services.

The rest of the paper is organized as follows. First we describe some details of e-commerce services we would like to implement, as well as basic problems associated with their implementation. Then we briefly consider basic concepts of agent technology and WAP as enabling technologies. Next we propose a solution for mobile e-commerce services utilizing software agents and WAP-based communication. Then we give some additional details of a generic platform and a prototype we have developed for implementation of various mobile e-commerce services. Finally we present our conclusions and an outline of future works.

TWO EXAMPLES OF MOBILE E-COMMERCE SERVICES

As our examples of mobile e-commerce services, we consider valued customer membership service support and subscription-based services.

The purpose of valued customer membership service is to provide members with special offers and with information about available products and services. Usually the service is applied by a shop or a chain of shops to provide membership benefits to their registered customers. Basically the service uses information about registered members to support mailing catalogues or booklets with particular offers. The main problem with such a service is its very low degree of personalization. The same offers and catalogues are usually sent to all members without consideration of their particular interests. This may cause customers to miss out interesting offers as a result of them being hidden amidst a huge amount of non-relevant information. It is also possible that customers simply ignore non-personalized catalogues and offers. In order to achieve better personalization of services, more information about customers’ interests and preferences should be included in the membership database, however this may contradict with privacy requirements (W3C, 2000). Even if customers agree to disclose their preferences to the membership service this can hardly be done in a flexible manner especially with huge centralized databases that assume a standard set of attributes for all customers.

Advertisements and information are usually sent to customers by regular mail. Usage of mobile portable devices for receiving these advertisements is not efficient when the limitations mentioned in the Introduction are taken into account. Nevertheless getting the latest knowledge of good offers for required products could be very valuable when planning shopping, in particular in the case when booklets and catalogues are not readily available.

Our second example of an e-commerce service is support of subscription services for mobile device customers. This is a well-developed service supported by many providers. As an example we take the support of stock market quotes notification. The main problem with this service is similar to the above-mentioned problems with customers’ membership services. Customers wish to be notified about changes in quotes of selected stocks. In order to get such notification, they need to disclose their stock preferences to a service provider. This is not often desirable because the customers may wish to preserve their privacy about stock preferences. It is quite usual that customer's quote notification preferences change over time. In this case personalization is often poor because of the restricted ability of dynamic service customization imposed by the limitations of mobile devices.

ENABLING TECHNOLOGIES

Before we propose our solution to relax the limitations of wireless communications in mobile e-commerce services, we consider the basic features of two recent technologies: software agents and WAP.

Software Agents

In spite of diverse views on agents in different research communities, there is an increasing agreement of what the basic agent characteristics are (Bradshaw, 1997; Huhns et al., 1998, Jennings & Wooldridge, 1998; Nwana, 1996; Wooldridge & Jennings, 1995). We can summarize these characteristics as follows:

- Agents represent an entity (human or another agent) in a computer environment and operates on behalf of its creator,
- Agents employ autonomous behavior and can operate without outside intervention,
- Agents employ pro-active goal-oriented behavior and can take initiative in order to achieve their goals,
- Agents react to changes in the environment: they perceive and affect the environment,
- Agents can communicate with each other and employ negotiation and/or coordination.

The last four properties are usually referred to as weak intelligent agent properties (Wooldridge & Jennings, 1995). Additional properties are usually attached to agents. Among them we consider mobility and learning as very important properties.

At the moment mobile agent platforms are better developed than intelligent agent platforms. However, we would like to underline that considering only mobility as the main agent property is a simplification of the whole agent paradigm, and this might be misleading when potential applications for agents are considered.

The ability of agents to learn is also referred to as an intelligent ability that is very desirable to implement in an agent system. However, we admit that even when agents do not employ learning, they may introduce intelligent abilities into distributed computing and service support by employing autonomy, pro-activity, reactivity and social ability.

When we talk about agents we should remember the following two views to agency:

- Individual view – considers agents which do not cooperate or communicate with other agents,
- Group view (Multi-Agent Systems) – considers the behavior of a collection of autonomous agents communicating with each other in order to solve a given problem.

The first view employs autonomy, pro-activity and reactivity from the above-mentioned agent characteristics, but it doesn't consider communication between agents as a vehicle for common problem solving. The second view underlines social ability and cooperative/competitive behavior between agents.

WAP- Wireless Application Protocol

The basic idea of WAP (Mann, 2000; WAP, 2000) as a standard for wireless communications is to extend the communicative abilities of wireless telephony. In other words cellular phones are no longer considered as just phones, but rather as communication devices capable of running applications and capable of communicating with other devices and applications across a wireless network. The standard takes into account the limitations of wireless data networks compared to wired networks and the limitations of mobile handsets compared to personal computers (see Introduction).

The WAP standard specifies an end-to-end application protocol and a browser-based application environment as two essential elements of wireless communication. The application protocol is a layered communication protocol that is embedded into each WAP-enabled user component. The network side includes a server component that implements the other end of the protocol capable of communicating with any WAP component. The server component often assumes the role of a gateway for routing requests from a user component to an application server. The gateway can be physically located in a telecom network, building a bridge between wireless and computer networks.

A “micro browser” integrated into a handset allows displaying information to the user and accepting user requests. The basis for information representation is WML (Wireless Markup Language), based on a datatype definition (DTD) for XML. The basic unit of WML is a card that specifies a single user interaction screen. Navigation occurs between cards that are grouped into decks. A deck is the top-most element of a WML document, presenting possible alternatives for user interactions.

It is assumed that when the user logs on to a WAP gateway s/he supplies the address of a starting page (WML deck) - a portal - and then performs interactions according to the cards in the deck. It is desirable for the user to be able to find all the information s/he needs linked directly or with as few links as possible from the starting deck. It is also preferable that favorite cards regularly visited by the user can be reached through a shorter chain of links than cards that are only used once in a while.

CONCEPTUAL SOLUTION FOR AGENT BASED MOBILE SERVICES

Our approach towards implementing mobile e-commerce services is based on using personal software assistant agents. Software assistant agents were initially proposed as desktop agents for support of user work with Personal Computers (PC) (Kozierok & Maes, 1993; Lieberman, 1995; Maes, 1994; Van Dyke, 1999). However, we believe that personal assistants for mobile devices, like cellular phones, significantly differ from software assistants for PCs. The main differences are due to restrictions of wireless communications compared to wired communications (see Introduction), limitations in user interaction with mobile devices, short connecting time of customers and services in mobile communication, and a dynamically changing communication context of users of mobile devices (for example, changing geographical location). All this puts forward higher requirements of pro-activity, prediction of user needs, personalization and delivery of precise information for the software assistant of a mobile device customer than for the software assistant of a PC user.

In our solution each mobile device customer has his own software assistant agent which represents his interests in the network. The assistant keeps a customer profile (preferences) and analyzes and generates information to be made available to the customer by taking this profile into account. This solution is presented in Figure 1.

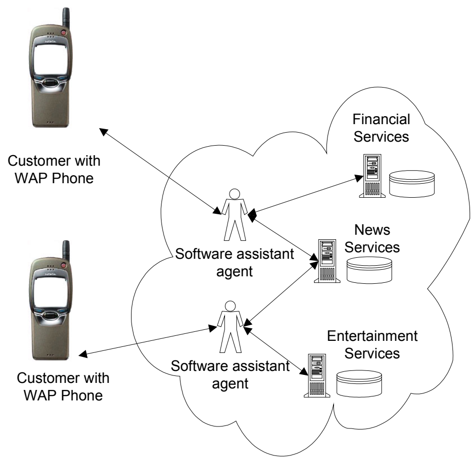


Figure 1 - Mobile Commerce Agent Environment

The assistant is implemented as an autonomous software agent. During periods when it is not necessary to do anything, it may hibernate, but when some event occurs, the agent awakes and performs some actions, such as generating or updating the information to be read by the customer. When the customer connects to the personal portal, s/he will get pro-actively generated and up-to-date information, thus s/he will not need to search for the information explicitly and subsequently wait for the search completion. Autonomy, reactivity, and pro-activity should play a key role in implementing these assistant agents. Pro-active behavior should rely on a customer profile presented in a system. In a more advanced case, agents' learning ability can also be developed, and monitoring of customers' actions may be a source for deciding about a preferred time for the availability of particular information. In some other case, a pro-active reasoning using available sources about a customer (for example, from user profile, from electronic calendar, etc.) can use advance deductive methods. It is assumed that the user may also specify explicit goals for agents and that they may start to act in order to satisfy the goals.

Application of the assistant-based system to the examples of e-commerce services we considered previously could be demonstrated as follows. In the case of customer membership service each registered member can be provided with a personal agent running on a customer's host or on a service provider's host (see Figure 2).

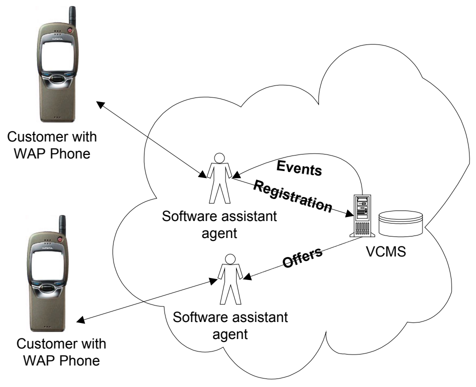


Figure 2 - Agent-based Valued Customer Membership Service (VCMS)

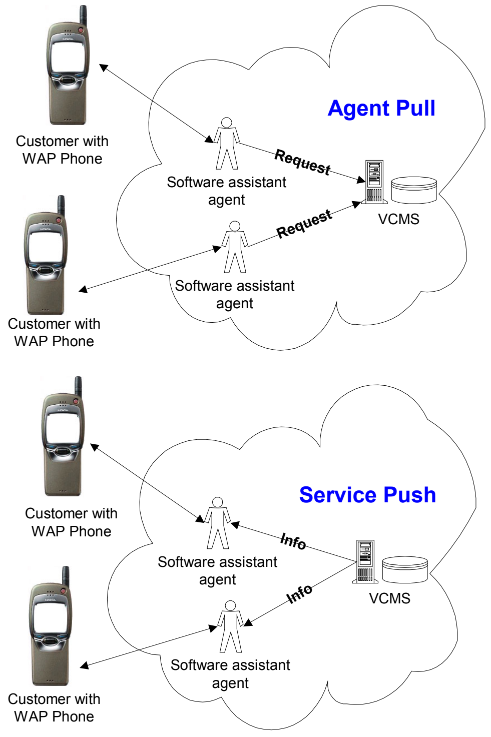


Figure 3 - **Agent-Service communication alternatives**

The agent can have a set of customer preferences presented as his profile. Customer privacy should be supported by corresponding host security, agent encryption, authentication between service host and agents, etc. The offers in the membership services can be implemented in two ways:

- Offers can be posted to members' agents (see Figure 3 bottom),
- Offers can be put into a special message-pool or web-page (see Figure 3 top).

In the first case, the agents perform offer analysis as a reaction to receiving a message from a service provider. In the second case, agents check offers in a common message-pool in some regular (or user defined) time-intervals, and then perform their analysis. The second option seems to be more practical in the case of membership service, because it supports asynchronous access to information and better service scalability.

Analysis of membership offers is done in a distributed and asynchronous fashion and, in particular, it may include filtering offers according to customer preferences.

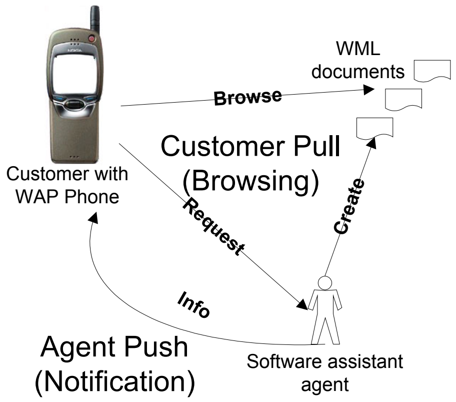


Figure 4 - Agent-Customer communication

Analysis results can be utilized in the following two ways (see Figure 4):

1. Generating data into a page which might be available for browsing by the customer from a mobile device
2. Sending a message to the customer (e-mail, SMS etc.)

Both ways, as well as their combination, might be used for communication with the customer. The first way allows for a more detailed description of the offers, while the second way allows informing the customer in real-time about the offers, as well as providing references to their detailed description.

Functionality of the assistant can be more sophisticated than just filtering incoming data. In particular, it may include pro-active reorganizing of the customer's web/WAP pages. Such reorganizing may take into account time, date, geographical location and/or the customer's calendar book. For example, a customer may have a list of different offers on his web/WAP page which are relevant to his interests. The personal agent, which knows the customer's calendar and preferences, may conclude that just before the customer's vacation, offers of travels (to Mediterranean countries) are of greater interest to the customer than other offers. After such a conclusion has been drawn, the agent can pro-actively find and put such offers on top of the offer list, providing easy offer access and visibility to the customer. The agent's decisions can be based on present customer profile, common knowledge, and the agent's deductive abilities. In another situation geographical location of the customer may actualize some offers related to closely located shops. These offers can be pro-actively put on top of the offer list for the time being when the customer is near this location. In this case the agent pro-actively search for relevant offers from geographically neighboring sources (shops, services, enterprises) and presents the offers to the customer. The architecture for implementing different agent functionality is discussed in the next section.

When the customer precisely specifies his queries for some product, the agent may also actively ask for membership services for required information. In this case the agent initiates a communication (and maybe negotiation) process.

In the case of subscription services and stock market quotes notification the proposed solution will be similar. Customer preferences in this situation may include both names of stocks and conditions when urgent notification is necessary (for example, changing quotes below a predefined threshold or the correlation between changes of quotes for different stocks). The customer's agent can have access to regularly updated stock market quotes in a predefined place. It may filter the quotes information according to the customer's preferences, generate relevant information to the pages read by the customer, and/or notify the customer in the real-time about recognized important events by sending warning messages to the customer's mobile device.

We can summarize general advantages of the proposed conceptual solution as follows:

- The solution provides better service privacy than centralized service provision. By better service privacy we assume that it is not necessary for the customer to disclose private data and preferences to a service provider in order to get a proper service (as it is assumed in the case of centralized service provision). The customer's preferences can be encapsulated into the personal agent who may run in the user's (protected and trusted) computational environment and they can be non-available for non-authorized parties. Of course, we can't completely eliminate the risk of disclosing the private information, however, the proposed solution tries to reduce such a risk.
- The solution may give better flexibility in describing preferences than centralized service provision. There are no requirements to keep a unified definition of attributes presenting customer's preferences.
- The solution provides better scalability than centralized service provision. Analysis and information processing can be asynchronous, decentralized, and distributed.

The above-mentioned properties give better personalization of services and take into account the restrictions of mobile devices when generating output to the customers.

A PLATFORM FOR MOBILE E-COMMERCE SERVICES SUPPORT

Taking into account the conceptual solution presented in the previous section, we propose a general platform for generating systems supporting mobile e-commerce services (see Figure 5).

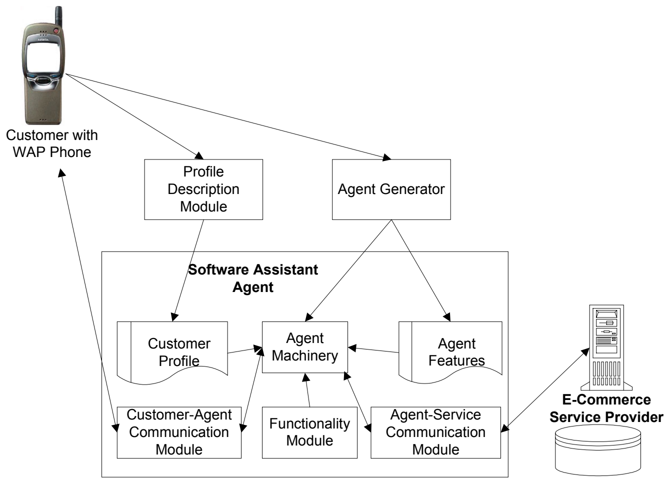


Figure 5 – General architecture of platform

The basic components of the platform are as follows:

- Customer profile description module,
- Customer-Agent communication module,
- Agent-Service communication module,
- Agent functionality module,
- Agent deployment module (agent generator and agent machinery).

Customer Profile Description Module

Each user should be able to present his or her profile by telling the agent what information s/he would like to have and, possibly, where to find it. Since most non-experienced customers probably may find this a hard and time consuming task, it should be possible to create certain default profiles targeted towards large groups of users. This could be, for example, a creation of a profile with the target group being customers interested in sports and fashion. Some other profiles would cover target groups of people interested, for instance, in technology, economy, and politics. To combine the two options (different individual and group profiles) it should be possible to let the user create his profile by combining several profiles together. In a simple case we assume that such combining can be done by manual editing or by performing operations (for instance, intersection, union, or difference) over source profile components. For example, the user could indicate that s/he is interested in sports and technology, but not in fashion, economy, and politics.

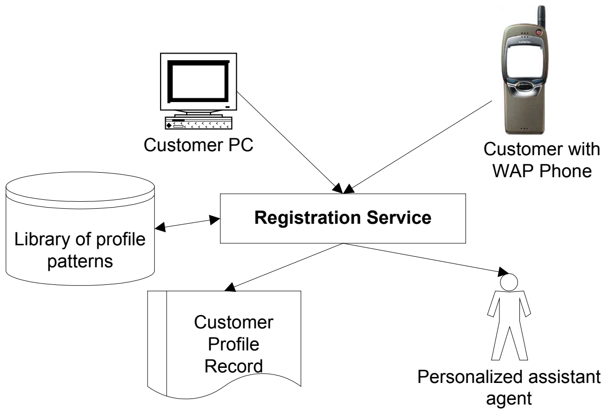


Figure 6 - **Registration service**

In a general case it is possible for the customer to make a registration (profile presentation) for a service both using computer-based and WAP-based handset interfaces as shown in Figure 6.

The result of registration is a generated personal assistant agent that may start running on a customer's PC or on the server of a service provider.

In the current version (see Prototype Description), a customer's profile is presented as a set of patterns together with computational rules used for filtering and analysis of information. However, the profile representation format can be overridden if needed.

During registration the module also provides a customer profile record that can be used by service providers. This record includes customer data which s/he agrees to disclose to the service provider and which can be used, for example, for gathering information about service customers.

Customer-Agent Communication Module

Communication between the customer and the agent can be done in two ways (see Figure 7).

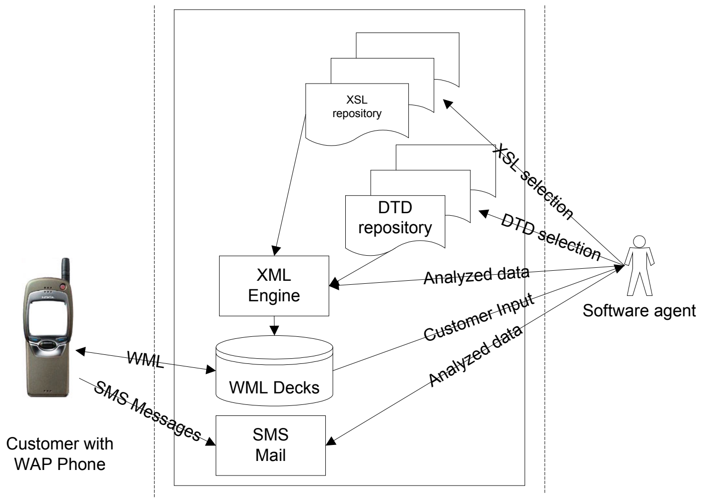


Figure 7 – Customer-Agent communication module

1. In the case of indirect communication, the customer can read pages from the Internet via a WAP-enabled mobile device, and agent prepares such pages containing data to be read by the customer. In this case communication module generates WML-page content accessible by the customer.

2. In the case of direct communication the agent sends SMS (Short Messaging Service) messages to the customer's mobile phone. Contents of the messages may include brief notifications of important events and the URL address of the WML-page(s) where more detailed information can be found.

We would like to note that in case of only short notification messages, cellular phones without WAP could also be used for notification of customers.

Customers can input information for the agent via WML-enabled menus on their phones, and this allows making changes into customer profile in real-time.

Agent-Service Communication Module

Communication between agents and service providers can be organized using pull and/or push techniques (see Figure 3). Both techniques can be utilized depending on situations, and their implementation should be symmetric. This means that both parties can initiate communication and select the communication method.

We believe the pull technique will most often be used for regular service provision and consumption, while the push technique seems to be more suitable for urgent notification. For example, a stock market service provider may regularly put stock quotes into a common message-pool. A similar pool can be used by an agent for requests to a service provider. However, notifications of significant changes in DJIA (Dow Jones Industrial Average) or NASDAQ stock indexes can be pushed directly to agents without letting them wait to read such news from the message-pool after a time interval which is defined for them.

In a more advanced case we believe that service providers can also be represented by software agents, and that these agents may be involved in the communication with customer agents (Matskin, 2000). The communication may include service marketing, customer's needs identification, and price negotiation. This already refers to group view in the agent approach and it may require Agent Communication Language (ACL) based communications (for example, FIPA (FIPA, 2000) or KQML (Finin, 1997)).

Agent Functionality Module

Functionality of the assistant can be different for different applications, and may, for example, include:

- Event handling,
- Filtering information,
- Search for products,
- Comparison of products,
- Finding other agents that carry similar user preferences (Collaborative filtering (Konstan, 1997)),
- Making purchases on behalf of the customer.

In our platform we assume that functionality of the assistant is implemented as an independent software component that can be overridden when needed.

Agent Deployment Module

In order to support the creation of individual agents we propose an architecture of an agent shell. The shell implements general agent functionality and supports easy customization, overriding and extension.

We introduce the agent shell architecture in two steps. First we describe an agent machinery level, and then consider its functional architecture.

Agent shell machinery is based on event-driven execution (see Figure 8), and includes:

- Prioritized Event List handled by embedded event managers,
- Event description list including event type descriptions with references to handlers,
- Event handlers - procedures which are invoked when a corresponding event is dispatched,
- Agent interface, including ACL and low-level communication ports,
- Agent memory that contains models (specifications, classes, types, etc.) and data (raw data, objects, etc.).

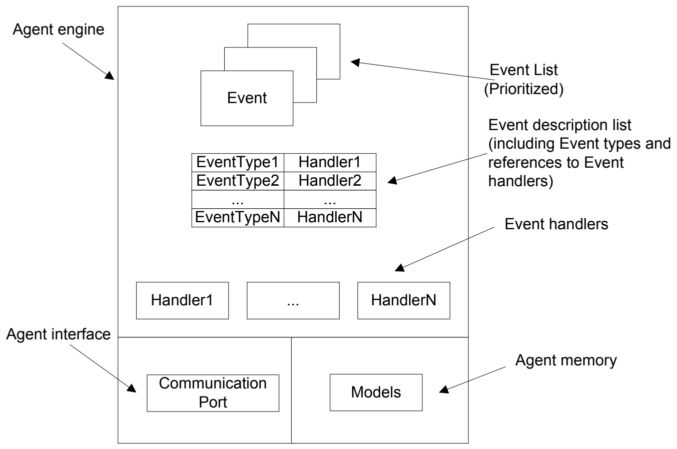


Figure 8 - Agent machinery

Events may arrive to the Event List either

- externally - from the environment, clock, by communicating with other agents, or
- internally - from the Event handlers.

The Event handlers have access to the Event description list and may modify it when necessary. The Agent memory is used by the Event handlers and can be updated by them or due to requests from the outside world.

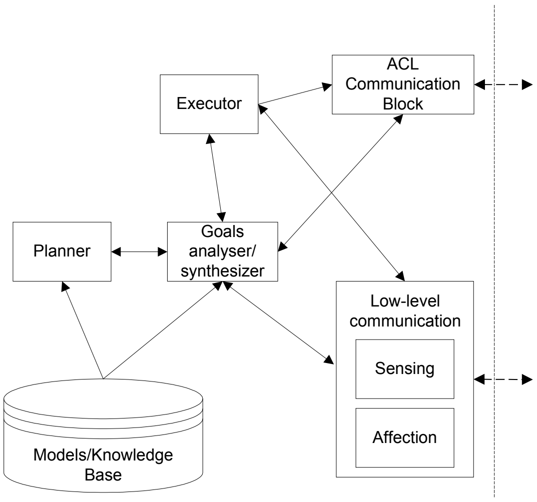


Figure 9 - Agent shell functional structure

An agent creator (a user or an agent) should be provided by tools to describe particular event types, event handlers, communication channels, and agent memory structure and contents. Detailed consideration of such tools is outside the scope of this paper and we refer to (Matskin & Tyugu, 1999; Matskin, 2000) for more details.

The agent shell functionality level is built on top of the agent machinery level and includes the following general blocks (see Figure 9):

- Communication blocks including ACL communication with agents and a low-level (signal-level) communication with the environment.
- Goal Analyzer/Synthesizer (GA) module perceives the environment, synthesizes a goal description, evaluates incoming messages, and selects a plan from a collection of alternative plans.
- Planner module creates a collection of plans based on the goal and Knowledge Base (KB).
- Executor – performs actions according to agent machinery level rules.

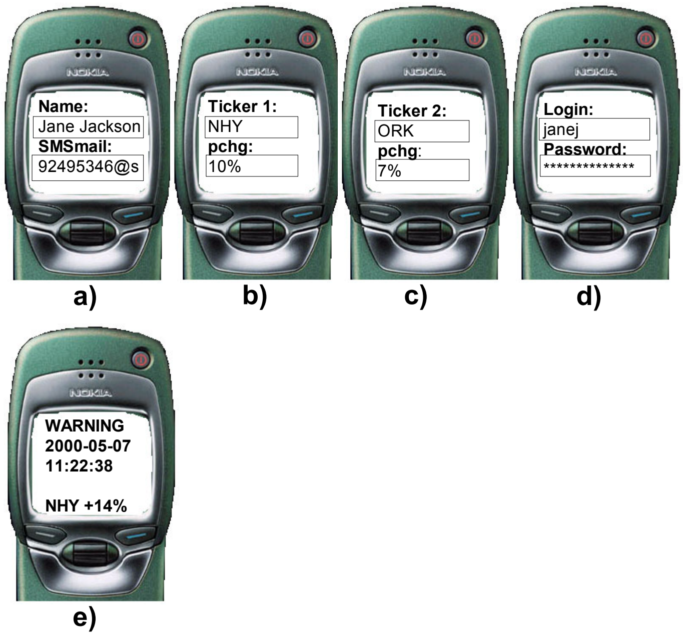
A typical operating scenario for an agent is as follows. When a message arrives or an event occurs it is sensed via Communication blocks and analyzed by the GA module. The GA module tries to find an action from the KB which is a proper reaction to the event. If the action is found, it is put into the Event List together with its handler and the Executor is called. If no action is found, the GA module synthesizes a goal to create a new action sequence (plan) as a reaction to the event, and calls the Planner to produce an optional set of alternative sequences that satisfy the goal. This set of action sequences is returned to the GA module which selects the most suitable action sequence, puts it into the Event List and calls the Executor.
In the current implementation (see Prototype Description) we basically focus on the agent machinery level and leave the higher level functional blocks for future works.

PROTOTYPE DESCRIPTION

In order to test our proposed solution a demo prototype of a WAP-based service for stock analysis of the Norwegian Stock Exchange Market (Oslo Børs) has been developed. In particular, the prototype provides a service for notification of customers about changes in the stock quotas. The notification is done by a personal software agent which sends SMS messages to customer's phone and creates WML-pages. Basic functionality of the prototype includes:

- WML-based customer registration of stock portfolio and corresponding notification conditions,

- SMS-based notification when stock values change above the user-defined threshold (e.g. more than 10% change in stock valuation since last time),
- Scheduled web-crawling of stock data stored in a database to enable further analysis.



a)
Name: Jane Jackson
SMSmail: 92495346@s

b)
Ticker 1: NHY
pchg: 10%

c)
Ticker 2: ORK
pchg: 7%

d)
Login: janej
Password: **************

e)
WARNING
2000-05-07
11:22:38

NHY +14%

Figure 10 - **WAP-based customer interaction**

For testing we used the Nokia 7110 WAP phone and several WAP browsers (Nokia, 2000). Some screen examples of both WML-decks and SMS-messages of the stock market quotes notification service are shown in Figure 10.

A typical way of operation for the prototype is as follows. A user first registers to the service (at http://wap.elcomag.com/) using WAP-enabled phone (see Figure 10.a,b,c,d). In our example, the registration fields are as follows: Customer’s name, SMS mail-address (to allow notification), Stock ticker(s) (e.g. NHY for Norwegian Hydro), the percentage the stock price can change until notification, login-name and the password. Through this registration a personal agent is created. Next time the customer would like to make changes in the presented data s/he needs to enter login name and password (see Figure 10.d). The agent starts to sense and analyze the stock market data and notifies the customer when notification conditions are satisfied (for example, see Figure 10.e, where NHY stock value raises by 14%).

A general structure of the prototype is presented in Figure 11.

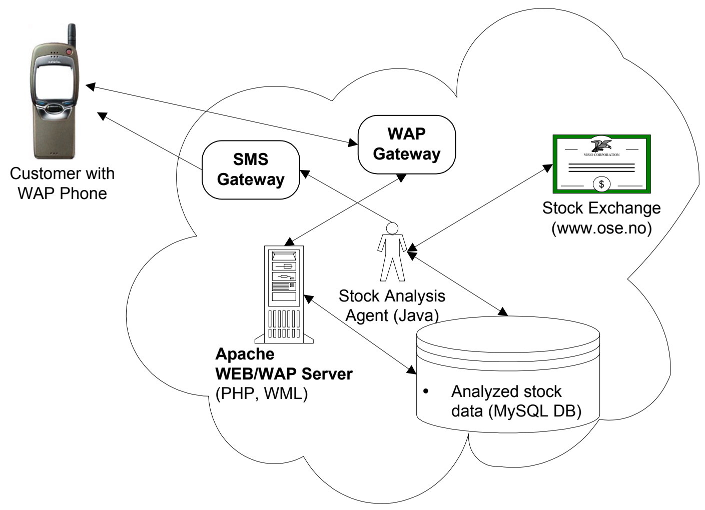

Customer with WAP Phone
SMS Gateway
WAP Gateway
Stock Exchange (www.ose.no)
Stock Analysis Agent (Java)
Apache WEB/WAP Server (PHP, WML)
• Analyzed stock data (MySQL DB)

Figure 11 - Prototype overview

The registration part of the prototype was programmed using PHP scripts to generate WML decks for the WAP-based Graphical User Interface. An Apache HTTP server running on a Windows NT machine hosts the PHP scripts. The agent part of the prototype is programmed in Java. The agent itself periodically downloads an HTML document from the stock exchange (the Norwegian Stock Exchange: http://www.ose.no) site in order to receive up-to-date stock quotes. This document is then parsed and extracted stock quotes are stored in the MySQL database (DuBois, 1999). Then the agent checks if the customer had any stocks that changed more than the registered percentage limit. When these stocks are found, the agent notifies the user about the changes by sending an SMS mail (using a Perl CGI-script on a machine with a mail server). In the prototype system agents run on a trusted Windows NT host of the service provider. However, it is possible for agents to run on a customer's machine connected to the network.

We can summarize experiences obtained from the prototype implementation as follows:

- The prototype demonstrated that the proposed conceptual solution is technically feasible. The prototype contains all basic blocks of the proposed platform with a default functionality which can be overridden.
- General supporting technologies are available. However, some of the WAP browsers and emulators should be improved in order to offer a satisfactory service.

CONCLUSIONS AND FUTURE WORKS

We propose a solution and a supporting platform for implementing and using personal software assistants in mobile e-commerce services. The solution is aimed towards relaxing restrictions of mobile devices and wireless communications. It is based on using software agents as an enabling technology for implementing software assistants for mobile device customers. The basic advantages of the solution include high personalization, reasonable privacy, flexibility, and scalability of agent-enabled mobile e-commerce services. These advantages are based on autonomous, decentralized and distributed processing as well as on the support of different levels of communication.

Future work is planned both for improving functionality and the implementation of the platform and its application to different e-commerce services. In particular, the following work is currently in progress:

- Development of the agent group view in the platform. This includes supporting communication, coordination, and negotiation between customer and service provider agents as well as between different customer agents. This allows cooperative problem solving, collaborative filtering, and information exchange between agents representing different participants in the e-commerce activity. The work is based on our earlier developed Agora architecture and system for support of cooperative work between software agents (Matskin et al., 1998; Matskin, 1999; Matskin et al., 2000).
- Development of a library of agent functional blocks allowing easier creation, personalization, and customization of personal software assistants. This also includes making purchase agreements and payments.
- Consideration of compatibility with existent services and systems using wrappers (Genesereth, 1997).
- Implementation of planning, knowledge base and reflection blocks in agent deployment module.
- Taking geographical location of the customer into account when performing pro-active behavior of query processing (for example, by using GSM positioning or Global Positioning System (Thibodeau, 2000)).
- Application of the platform for implementing a real estate buying assistant agent, advertising service, and product search and comparison services for mobile device customers.

ACKNOWLEDGEMENTS

We are grateful to Arne Sølvberg for fruitful discussions during the early stages of the work. We also would like to thank Thomas Brox Røst and Richard Sanders for their comments and proof reading. This work is partially supported by the Norwegian Research Foundation in the framework of the Distributed Information Technology Systems (DITS) program and the *ElComAg* project.

REFERENCES

Bradshaw, J. M. (Ed.). (1997). *Software Agents*. Menlo Park, CA: AAAI Press/The MIT Press.

DuBois, P. (1999). *MySQL*. Macmillan Technical Publishing.

Finin, T., Labrou, Y. & Mayfield J. (1997). *KQML as an Agent Communication Language*. In: J. M. Bradshaw, (Ed.), Software Agents (pp. 291-316). AAAI Press/The MIT Press: Menlo Park, CA.

FIPA (2000). *Agent Communication Language Specifications*. Available: http://www.fipa.org/repository/aclspecs.html

Genesereth, M. R. (1997). *An Agent –Based Framework for Interoperability*. In: J. M. Bradshaw, (Ed.), Software Agents (pp. 317-345). AAAI Press/The MIT Press: Menlo Park, CA.

Huhns, M. N., Singh, M. P., & Gasser, L. (Eds.). (1998), *Readings in Agents,* Morgan Kaufmann Publishers.

Jennings, N. R., & Wooldridge, M. J. (Eds.). (1998). *Agent Technology: Foundations, Applications and Markets*. Springer Verlag.

Konstan, J. A., Miller, B. N., Maltz, D., Herlocker, J. L., Gordon, L. R., & Riedl, J. (1997). *GroupLens: applying collaborative filtering to Usenet news*. Communications of the ACM, 40(3), 77-87.

Kozierok, R. & Maes, P. (1993). *A Learning Interface Agent for Scheduling Meetings*. Proceedings of the ACM-SIGCHI International Workshop on Intelligent User Interfaces, Florida, 81-93.

Lieberman, H. (1995). *Letizia: An Agent that Assists Web Browsing*. Proceedings of the 14-th International Joint Conference on Artificial Intelligence (IJCAI'95), AAAI Press.

Maes, P. (1994). *Agents that Reduce Work and Information overload*. Communications of the ACM, 37(7), 31-40.

Mann, S. (2000). *Programming Applications with the Wireless Application Protocol: The Complete Developer's Guide*. John Wiley & Sons.

Matskin, M., Divitini, M., & Petersen, S. A.(1998). *An Architecture for Multi-Agent Support in a Distributed Information Technology Application.* Proceedings of the International Workshop on Intelligent Agents in Information and Process Management on the 22nd German Annual Conference on Artificial Intelligence in Bremen (KI-98), 47-58.

Matskin, M. (1999). *Multi-Agent Support for Modeling Co-operative Work*. In: T. Yongchareon, F. A. Aagesen, & V. Wuwongse (Eds.) Intelligence in Networks, (pp. 419-432), Boston/Dordrecht/London: Kluwer Academic Publishers.

Matskin, M., Kirkeluten, O. J., Krossnes, S. B., & Sæle, Ø. (2000*). Agora: A Multi-Agent Platform and its Implementation*. In: H. Arabnia (Ed.) Proceedings of the 2000 International Conference on Artificial Intelligence (IC-AI'2000), Vol. II, (pp. 549-555), CSREA Press.

Matskin, M., & Tyugu, E. (1999). *Shells for Multi-Agent Applications (An Architecture for Agents and Agoras).* Technical report, IDI-nr. 4/99, ISSN-NO: 0802-6394, Norwegian University of Science and Technology (NTNU).

Nokia (2000). *Nokia 7110*. Available: http://www.nokia.com/

Nwana, H. S. (1996). *Software Agents: An Overview*, Knowledge Engineering Review, 11(3), 205-244.

Tarasewich, P., & Warkentin, M. (2000). *Issues in Wireless E-Commerce*. SIGecom Exchanges, Newsletter of the ACM Special Interesting Group on E-Commerce, 19-23

Thibodeau, P. (2000). *Satellites Will Change E-Commerce Landscape*. Computerworld. [Online]. February 21.Available: www.computerworld.com/cwi/story/0,1199,NAV47_STO41428,00.html

Van Dyke, N. W., Lieberman, H., & Maes, P. (1999*). Butterfly: A Conversation-Finding Agent for Internet Relay Chat*. Intelligent User Interfaces 1999: 39-41.

W3C (2000). *The Platform for Privacy Preferences 1.0 (P3P1.0) Specification*. Available: http://www.w3.org/TR/P3P/

WAP (2000). *WAP Forum Specifications* [Online]. Available: http://www.wapforum.org/what/technical.htm

Wooldridge, M. & Jennings, N. (1995). *Intelligent Agents: Theory and Practice.* The Knowledge Engineering Review, 10(2), 115-152.

Paper B

```
@InProceedings{2001:MobCom:Tveit,
  author    = {Amund Tveit},
  title     = "{Peer-to-peer based Recommendations for
                 Mobile Commerce}",
  booktitle = {Proceedings of the First ACM SIGMOBILE
                Mobile Commerce Workshop},
  pages     = {26--29},
  year      = {2001},
  editor    = {Muthy Devarakonda and Anupam Joshi and
                Marisa Viveros},
  address   = {Rome, Italy}
  month     = {July},
  publisher = {ACM Press}
}
```

This paper has been cited by Svensson [2003]; Heinemann et al. [2003a,b]; Olsson [2003]; Adams [2003]; Jang and Lee [2002]; Munusamy and Leang [2002]; Weiss [2002a,b]; Lee [2002]; Ding et al. [2002]; Ding and Unnithan [2002]; Nguyen [2002].

Peer-to-peer based Recommendations for Mobile Commerce

Amund Tveit[*]
Department of Computer and Information Science
Norwegian University of Science and Technology
N-7491 Trondheim, Norway
amund.tveit@idi.ntnu.no

ABSTRACT

With the increasing number of mobile commerce facilities, there are challenges in providing customers useful recommendations about interesting products and services.
In this paper a Peer-to-Peer (P2P) based collaborative filtering architecture for the support of product and service recommendations for mobile customers is considered. Mobile customers are represented by software assistant agents that act like peers in the processing of recommendations.

1. INTRODUCTION

Mobile commerce, or e-commerce in the wireless web, is providing commercial services that are accessible using mobile devices, typically a mobile phone. The main advantage of such services is their high availability, independent of physical location and time.

However, mobile commerce has its limitations, particularly regarding communication and the resources of mobile devices. Communication have less and more expensive bandwidth, higher latency, lower connection stability, less predictability, and currently less standardized protocols [10]. Mobile devices in contrast to their desktop PC alternatives, are severely constrained due to less processing and memory resources, smaller and less convenient user interfaces, in addition to a limited energy supply.

Problem

An important question related to most types of commercial activities is: *Which products or services should be recommended to a particular customer in order to make him/her satisfied?*

In e-commerce settings the automization of personalized recommendations has been sought solved using methods such as information filtering and collaborative filtering [4]. *Information filtering* recommendations is based on the analysis of a profile describing a (potential) customer's needs and preferences. *Collaborative filtering* is based on using votes or opinions about products and services from *similar* customers in order to give recommendations. Collaborative filtering has been shown as the best approach of these two methods, but it is highly dependent on a large number of customers to give good recommendations and it doesn't scale well in terms of processing.

In this paper an approach for making a scalable recommendation system for mobile commerce using a Peer-to-Peer (P2P) is considered. Peers are represented as software assistant agents interfacing a mobile customer. This approach adds the opportunity of product and service recommendations to the mobile commerce agent architecture presented in [6].

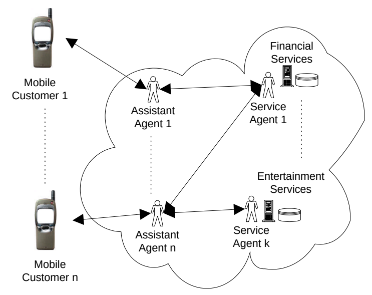


Figure 1: Mobile Commerce Agents

The rest of the paper is organized as follows. Section 2 describes collaborative filtering theory. Section 3 describes the proposed approach for P2P-based recommendations. Section 4 describes related work, and finally the conclusion with future work.

[*]http://www.idi.ntnu.no/~amundt/

2. COLLABORATIVE FILTERING

The most important step for creating recommendations using collaborative filtering for a particular customer u is to find an ordered set of customers $\mathcal{N} = \{N_0, .., N_l\}$; $u \notin \mathcal{N}$, where the ordering is based on similarity of N_i and u, this should be performed for each customer u (i.e. $O(n^2)$ calculations of similarities with n customers in the database) [9].

Similarity between two customers u and N_i can be found by using a proximity measure. The essential data used in the calculation of the similarity is the two customers' vectors with votes on products or services. Votes are typically numerical values that are either entered by the user for products s/he dis/liked, or values added by the e-commerce provider system (e.g. if a customer has browsed or purchased a product or service, it may get a good score automatically). Two common proximity measures are the *Pearson correlation* or the *cosine measure*, shown below for customer a and b. One advantage of the *cosine measure* is that it is suitable for dimension reduction with sparse data sets (i.e. few products or services voted for).

$$corr_{ab} = \frac{\Sigma_i(r_{ai} - \bar{r_a})(r_{bi} - \bar{r_b})}{\sqrt{\Sigma_i(r_{ai} - \bar{r_a})^2 \Sigma_i(r_{bi} - \bar{r_b})^2}} \qquad (1)$$

$$cos(\vec{a}, \vec{b}) = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|_2 * \|\vec{b}\|_2} \qquad (2)$$

When all proximity measures between all customers are calculated, a neighborhood is formed (a reasonable size subset of customers in $\mathcal{N}$ for a particular customer u). If the the vectors in the data set are sparse, a aggregate neighborhood approach can be used (calculates the centroid of the neighborhood before adding new customers). If the vectors in the data set are dense, a center-based approach can be used (picking the l most similar customers from $\mathcal{N}$).

When a set of neighborhood vectors is found, recommendations can be calculated by selecting items in the vectors from the neighborhood that the customer u hasn't voted for, e.g. if a customer u hasn't voted for particular book b, and another (i.e. the most similar) customer u' in the neighborhood has given the book a high rating, the book will be recommended for customer u.

3. PROPOSED APPROACH

The main idea of this approach is to transform the problem of finding good product and service recommendations using collaborative filtering, into a search problem that can take advantage of the scalability and privacy possible in a P2P-based architecture [8] similar to Gnutella or Freenet.

3.1 Query Flow

3.1.1 Gnutella

In Gnutella, a query (i.e. for a MP3 file) from a Peer node in are broadcasted to all its neighbours. The neighbour Peers (independently) check if the query matches one of their hosted files, if so, they return a found message back to the sending node, otherwise they decrease the TTL field (TTL = Time To Live) and then pass on the query to their neighbour peers. If the TTL count reaches 0, the message is not

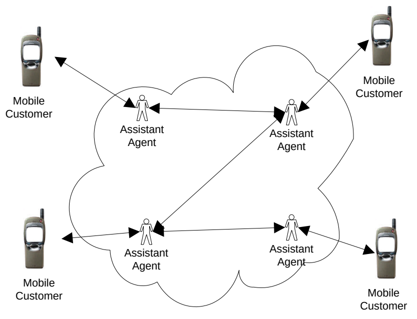


Figure 2: P2P Network of Agents

sent any further. In other words, the query is broadcasted in "all directions" from the quering node, but only the query matches are sent back the network path they arrived from [8]. This flow of queries ensures that a peer doesn't know if a received query was from a neighborhood peer, or another peer that broadcasted the message, this gives a limited degree of privacy for queries.

3.1.2 Proposed Approach

Queries are broadcasted to neighbour Peers as for Gnutella, but a query is not a text string (as the case for Gnutella's file search), but rather a vector with votes on products and recommendations. This voting vector is the same as used in collaborative filtering.

3.2 Routing Algorithms

3.2.1 Gnutella

In Gnutella, there is a very simple routing algorithm that either stops broadcasting from a peer if a match is found, or continues broadcasting messages to other peers if no match was found.

3.2.2 Proposed Approach

When a peer receives a query voting vector: It calculates the proximity (e.g. with Pearson correlation from collaborative filtering) between the voting vector in the message and cached previous messages at the node. If the proximity measure is higher than a threshold τ, the cached voting vector is sent back to the neighbour that sent the query voting vector (and possibly further back to the origin from there). If the proximity measure is lower than τ, the query voting vector is broadcasted further to the receiving peer's neighbours. If there are too many messages received per time unit, they are simply passed on to neighbour peers without calculating the proximity measure against cached items. The size of the cache tries to balance against the traffic, if the traffic is high the cache is decreased in order to be able to calculate proximity measures.

3.3 Dynamic Network Topology

3.3.1 Gnutella

Gnutella have a mainly static peer network topology, i.e. peers can disconnect or connect to the network, but they rarely switch places or routing tables after the initial connection the network.

3.3.2 Proposed Approach

Since each peer is calculating the proximity on incoming messages versus cached messages, the peer is able to monitor the resemblance of traffic from different neighbour peers. So if two neighbour peers send similar types of queries to a peer, it can ask the peers if they wan't each others addresses, and if both reply yes the peers establish a new connection. The reason for doing this is to try to cluster similar peers in terms of number of hops to improve performance and quality of recommendations.

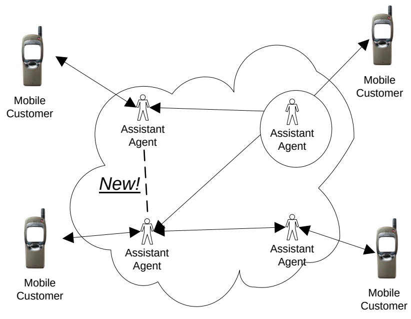


Figure 3: Dynamic Network Topology

3.4 Ranking methods

When a potentially large set of voting vectors has been returned to querying node they have to be ranked in order to get the best recommendations on top of the list before communicating to the mobile user (similar to Internet search engines). The straightforward solution would be to use collaborative filtering (rank based on the calculation proximity measures and then the neighborhood), this may sound like the same as performing traditional collaborative filtering, but the difference is that the voting vectors to be processed is only a relatively small subset of *all* voting vectors for all users of the system, this is because of the filtering done by peers that processed the query.

An alternative approach could be to return only the recommendation and the proximity measure, e.g. if the querying voting vector, represented as with $item : vote$ elements, is $[10 : 0, 22 : 2, 37 : 8]$ and the matching vector is $[10 : 0, 22 : 2, 24 : 9, 37 : 8]$ then only the recommendation $24 : 9$ together with the proximity measure will be returned, this is more efficient both regarding bandwidth usage and a useful preprocessing that makes ranking into a simple sorting job (of proximity measures).

3.5 Query Compression scheme

If voting vectors are sparse, it is more efficient to represent them as hash tables than vectors for internal processing, but to get more efficient compression when sending the vector between peers one can take advantage of the *binary interpolative compression algorithm* [7].

The binary interpolative compression algorithm was developed to efficiently compress inverse files used in document indeces in information retrieval, it utilizes the difference and context information between sorted numeric values for compression. By requiring the data to have certain properties, it can at best compress some items to 0 bytes (depending on position, value and neighbour values in the array). In order to work, the algorithm requires the data to be: *numeric*, *sorted* and *in a known numeric range*. The index positions representing items in the vector (and hash table keys) fulfill these requirements (e.g. ISBN numbers representing books), and can therefore gain from using binary interpolative compression.

3.5.1 Compression example:

Original query vector represented using a hash table (item:vote pairs)

$$\vec{q} = \{24 : 5, 37 : 8, 45 : 0\} \tag{3}$$

for item 24 with rating 5, item 37 with rating 8, and item 45 with rating 0 respectively. The compressed representation is created with

$$IPolComp([24, 37, 45], 3 - 1, 24, 45) : 580 \tag{4}$$

(i.e. compressing the list of items, and appending the votes for items after the separating colon)

3.6 Implementation

The system is planned to be implemented using the java programming language.

3.6.1 Feasible Performance?

Critical issues for high performance in peer-to-peer networks are efficient handling of requests (i.e. processing and routing time), and network bandwidth and topology [8].

Calculating similarities using a proximity measure such as the cosine (2) has linear algorithmic running time with respect to the vector length $|\vec{q}|$. Routing based on comparing an incoming query vector $\vec{q}$ with the set of cached vectors C has a run time of $O(|C||\vec{q}|)$, this can potentially be reduced by ordering the vectors in C based on proximities (whenever there are few messages to handle for the peer) and use binary search to find the closest match(es) for $\vec{q}$ in C. For each similar vector $\vec{v}$ in C $\vec{q}$ is forwarded in $\vec{v}$'s direction (i.e. the neighbour peer which $\vec{v}$ was received from).

Network bandwidth for each peer is not easily changed, but the topology can be changed with a relatively low cost since it involves a simple interaction with neighbour peers about allowing them to be introduced to each other. However, in order to avoid the peer-to-peer topology to become a complete graph, the dynamic topology heuristic should be able to forget, i.e. remove peers from the routing table if they are of little relevance (get low proximity measures) to the majority of requests to the peer.

These approaches are similar to existing and scalable peer-to-peer networks such as gnutella and freenet, and supports the feasibility of the proposed approach.

3.6.2 Trust

Handling trust in the system could possibly be done by introducing trusted third-parties or authority peers/hubs which keeps public keys for the peers in the network. Each query should then be signed by the peer's private key so that it is possible to check that a query came from a valid peer. In order to decrease the opportunity for malicious behavior, e.g. hidden and fraudiant advertising in recommendations, one can add the opportunity for mobile users to add simple feedback on recommendations. If a vector $\vec{v}$ gave a good recommendation to a user, the origin peer p of $\vec{v}$ could gain trust points which could for instance increase the rating of the user in terms of recommendation ordering and degree of replication of p's queries throughout the peer-to-peer network.

4. RELATED WORK

Varshney et al. propose a taxonomy and application framework of mobile commerce [11]. The W^3IQ proxy based recommender system presented by Joshi uses peer-to-peer (proxy-to-proxy) based recommendations of URLs satisfying information requests [5]. Related work on personalization for mobile users include work by Cotter and Smyth [3]. They present PTV, a recommender system for TV-programmes based on both collaborative filtering and content-based recommendation strategies. Personalization for mobile users is supported by the Gulliver system by Austaller et al. [2]. Sarwar et al. gives an overview of various recommendation algorithm approaches for e-commerce [9].

5. CONCLUSIONS

In this paper a P2P-based approach for scalable recommendations for mobile commerce has been presented. The main contribution is the P2P-based recommendation approach described and the efficient compression algorithm of the communication of voting vectors.

This P2P-based approach differs from others, in the sense that there is no free-riding [1]. Free-riding is when peers do not produce and consume equal balanced of data, production is equal to hosting files, and consumption is equal to downloading files (for Napster and Gnutella). In the presented approach consumption equals production, since a query is a voting vector which is also the data object of interest in the P2P service.

Future work include implementing the system for the recommendation of mobile phone ringing tones and logos, and try to meet challenges such as: How to deal with fraudiant behavior? How to maintain consistent and unique identification of products and services in the voting vectors used in queries? How to keep cache consistency? How to deal with privacy and encryption of queries and results? Where are the bottlenecks for true scalability (millions of mobile commerce users and profiles). Is a true P2P-based systems for recommendations or is hybrid approach like Napster or Gnutella with Gnutellahosts needed?

6. ACKNOWLEDGEMENTS

I would like to thank my supervisor Mihhail Matskin, and my Agentus colleague Thomas Brox Røst for motivating discussions. This work is partially supported by the Norwegian Research Council in the framework of the Distributed Information Technology Systems (DITS) program and the ElCoMAg project.

7. REFERENCES

[1] Adar E., and Huberman B. A. Free Riding on Gnutella. *First Monday*, 5(10), October 2000.

[2] Austaller G., Hartl A., Kappel G., Lechleitner C., Muhlhauser M., Reich S., and Rudisch R. Gulliver Beans: Generating Device Optimized and Individualized Content for WAP Applications. In *Proceedings of the 9th International World Wide Web Conference*, 2000.

[3] Cotter P., and Smyth B. WAPing the Web: Content Personalization for WAP-Enabled Devices. In *Proceedings of the International Conference on Adaptive Hypermedia and Adaptive Web-based Systems*, pages 99–108, 2000.

[4] Good N., Schafer J. B., Konstan J., Borchers A., Herlocker B., and Riedl J. Combining Collaborative Filtering with Personal Agents for Better Recommandations. In *Proceedings of the 1999 Conference of the American Association of Artificial Intelligence (AAAI-1999)*, pages 439–446, 1999.

[5] Joshi A. On Proxy Agents, Mobility and Web Access. *Mobile Networks and Applications, Special Issue on Software Architectures for Mobile Applications*, pages 233–241, 2000.

[6] Matskin M., and Tveit A. Mobile Commerce Agents in WAP-based Services. *Journal of Database Management - Special Issue on Mobile Commerce*, pages 27–35, July-September 2001.

[7] Moffat A., and Stuiver L. Exploiting clustering in inverted file compression. In *Proceedings of the 1996 IEEE Data Compression Conference*, pages 82–91, 1996.

[8] Oram A., editor. *Peer-to-Peer: Harnessing the Power of Disruptive Technologies*. O'Reilly and Associates, 2001.

[9] Sarwar B., Karypis G., Konstan J., and Riedl J. Analysis of Recommendation Algorithms for E-Commerce. In *Proceedings of the 1st ACM SIGecom Conference on Electronic Commerce*, 2000.

[10] Tsalgatidou A., and Veijalainen J. Mobile Electronic Commerce: Emerging Issues. In *Proceedings of EC-WEB*, pages 477–486, September 2000.

[11] Varshney U., Vetter R. J., and Kalakota R. Mobile Commerce: A New Frontier. *IEEE Computer*, 33(10):32–38, 2000.

Paper C

```
@InProceedings{2003:SCAI:Tveit,
  author    = {Amund Tveit and {\O}yvind Rein and
               J{\o}rgen Vinne Iversen and Mihhail Matskin},
  title     = "{Scalable Agent-Based Simulation of Players in
                Massively Multiplayer Online Games}",
  booktitle = {Proceedings of the 8th Scandinavian Conference on
               Artificial Intelligence},
  year      = {2003},
  editor    = {Bj{\o}rnar Tessem},
  address   = {Bergen, Norway}
  series    = {Frontiers in Artificial Intelligence and Applications},
  month     = {November},
  publisher = {IOS Press}
}
```

This paper has been cited by Ho et al. [2003]; Thawonmas et al. [2003]; Ho and Thawonmas [2004].

Scalable Agent-Based Simulation of Players in Massively Multiplayer Online Games

Amund Tveit Øyvind Rein Jørgen V. Iversen
Mihhail Matskin

Department of Computer and Information Science,
Norwegian University of Science and Technology
N-7491 Trondheim, Norway

Abstract. We propose a parallel mobile agent platform - Zereal - for scalable and flexible simulation of Massively Multiplayer Online Games (MMOGs). Players and NPCs in Zereal currently have a sense-reason-act behavior, with reasoning based on Markov chains in addition to hierarchical plans specified in XML. Zereal's primary purpose is to be a MMOG simulation tool that enables testing of usage logging approaches for CRM and data mining purposes, and secondly to allow flexible testing of behavioral AI models for players and non-personal characters. Zereal has shown to be close to linearly scalable in terms of number of players, and has been successfully tested with more than 100 thousand players on 20 CPUs on a Linux-based cluster. Zereal is implemented using Python, MPI and C++.

1 Introduction

Over the last couple of years an emerging commercial online games market has come into existence, fuelled by the increasing availability of cheap Internet access. The main share of this market consists of *Massively Multiplayer Online Games* (MMOGs). The characteristic property of MMOGs is that a large number of players, sometimes even in the six or seven figure realm, take part in a persistent virtual world where they communicate, cooperate, fight, build virtual characters, and undertake game-related missions [11, 24].

The worldwide MMOG market is anticipated to grow annually by 70% to USD 2.7 billion in year 2006 [12]. Traditionally the game industry has been highly product-focused, but with the introduction of MMOGs games have become more like services, where players not necessarily pay for the MMOG game client, but instead pay a monthly subscription fee for having access to the MMOG world [8, 15]. In order to gain and keep subscribing players there is a need to focus on *Customer Relationship Management* (CRM) and *Data Mining* methods [1, 21]. The overall motivation for such methods in a MMOG context is to figure out more about the players and their preferences so the MMOG can be continously improved and keep on generating revenue.

1.1 Research Problem and Approach

A fundamental requirement for data mining and CRM methods is the need for *relevant usage data*. For MMOGs there are currently no open standards for usage data logging, the data is being logged in a proprietary manner by the various MMOG vendors. This makes it hard to develop data mining and CRM approaches that can used for several MMOGs. So the primary research problem is: **How to enable a scalable and flexible simulation environment for testing out approaches for player usage logging in MMOGs?**.

Since the environment needs to simulate *autonomous MMOG participants* (i.e. players and non-personal characters), we choose to use autonomous intelligent *agents* as the primary abstraction. Due to *several types and numbers of agents* we select the Multi-Agent System (MAS) approach from Distributed Artificial Intelligence as the underlying architecture [17].

The rest of this paper is organized as follows. Section 2 describes and discusses the Zereal MMOG simulation platform, section 3 describes related work, and finally the conclusion.

2 Zereal Platform

The overall goal is to create a MMOG simulation platform that provides a (coarse) simulation of active players that can be used to test various approaches for player usage logging.

2.1 Overall Architectural Choices

Creating a realistic activity level with thousands of concurrent players and NPCs is considered to be of greater importance than having a very accurate simulation of each player's and NPC's behavior. However, the architecture should also provide a flexible interface for testing out various intelligence mechanisms.

Requiring support for thousands of concurrent players and NPCs encourages the selection of a *distributed or parallel architecture*, this combined with the prior selection of agents as the main abstraction encourages use of *mobile agents* for the representation of players and NPCs. Due to the relative simplicity of implementation and deployment of message-based parallel systems we choose a platform of type *message-based parallel mobile agent architecture*. A second motivation for parallel message-based systems is that they can run on most current supercomputers and cluster systems. Parallel cluster systems are frequently used to serve MMOGs, see figure 1 for a typical industry configuration [4, 26, 7].

2.2 Simulation Platform Requirements

The six fundamental requirements for our MMOG simulation platform are:

1. Virtual World Model
2. Item Model
3. Player Models
4. Non-Personal Character Models
5. Scalability
6. Logging of Player behavior

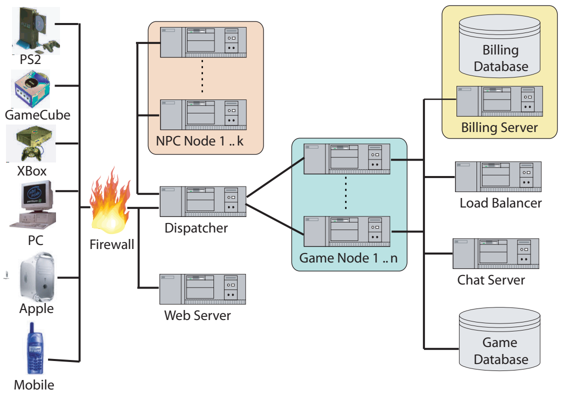


Figure 1: Parallel Cluster Architecture for *an actual* MMOG

2.3 Virtual World Model

Zereal's **MMOG world representation** is a roguelike model, mostly inspired by Nethack (roguelike refers to the game Rogue). Roguelike means that it has simple graphics, typically a 2D environment with a bird's-eye view of the game. Roguelike also usually means that the the game has an enormous variety and amount of features [13], but in our simulation platform we've chosen to provide only some basic features.

The virtual world map is represented as 2D rectangel with borderwalls and doors on the outer limit of the map, and internal walls with doors inside the map to split it up in rooms or smaller outside spaces. Researchers at the Intelligent Computer Entertainment Laboratory at the Ritsumeikan University in Japan has developed a GUI client to Zereal called ZerealViewer, see figure 2 for a screenshot.

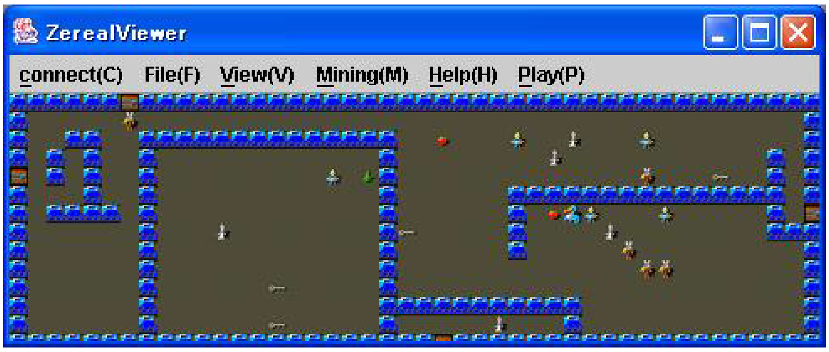


Figure 2: Parts of a typical Zereal Virtual World

Each map is a subworld which resides on its own processor in the cluster and the subworlds borders each other after a squared model and send players between them. This world representation is easy to represent in memory, and extension of the world is only limited by memory and power on the clusternodes. Most landscapes can be ”made believe” in the maps, and the usually resource demanding task of sensing has less complex data to process. Players and items are represented as ASCII characters in visualization and integer ID's in the internal representation. For scalability reasons it is recommended to map the virtual world into a square (or close to a square) of subworlds, each mapping to one CPU, this ensures a relatively equal load on each CPU (in average), except along the borders (where they have fewer neighbours). A future revision of the platform will enable wrapping between the outmost subworlds (like the game Asteroids).

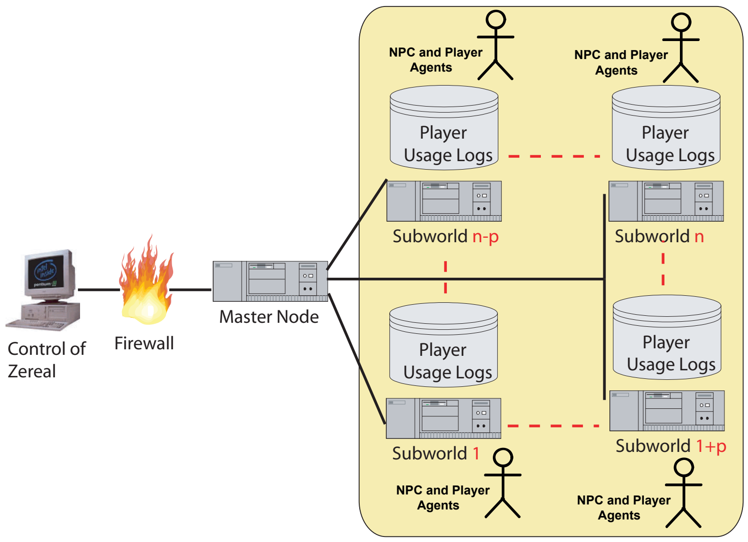


Figure 3: Zereal Macro Architecture

2.4 Item Models

Items in Zereal can be of several types: doors, keys, food, potions and weapons. Items occupy one square on the map, and except for doors, all items can be picked up by players. *Doors* transports the agents from room to room or subworld to subworld. If all neighbouring squares on the other side of a door is occupied, agents will not be able to move through them, but in case of a door on the border, agents will be transmitted to the next world but end up in a queue (limbo) untill some of the squares has been freed. Agents may need keys to open locked doors. *Food* and *potions* increases or (if poisonous) decrease the players hitpoints and strength when consumed.

2.5 Player Models

Zereal's **player models** are inspired by the typical player categories found in *actual MMOGs*:

1. *Achievers* - people who are entertained by building their avatar character with more skills, hit points, strength etc.
2. *Socializers* - people who wants to build and maintain social relations such as friendship and love, they are typically attracted to people with high intra-game status.

3. *Killers* - people who sees killing as the soul purpose of the game, they typically believe that a player's status depends on how many creatures or other players his/her avatar has killed.

4. *Explorers* - people who roams around the gameworld to discover new places, features, items and non-personal characters.

In general, social status in the MMOG is usually the underlying motivation for all player types in MMOGs [10].

Both **players** and **non-personal characters** are represented as mobile agents with a sense-reason-act approach in Zereal, this is because they need to have autonomous, social, reactive, and pro-active capabilities, hence following the weak properties of agents [25]. The selection of agents as the primary abstraction is also motivated by previous representations of intelligent life-like entities in virtual environments [6, 3] Mobility is selected since the players have to move in a world topology that is spread out on several computational nodes, mobile agents have previously been used in various virtual environment architectures [16].

2.6 Non-Personal Character Models

For our platform we chose the *non-personal characters models* to be equal to player models of the type *killer*, but with minor alterations regarding mobility. The alterations are that NPCs are less mobile than the killers, i.e. they mainly stay within the same subworld at all times.

2.7 Zereal Sense-Reason-Act Approach

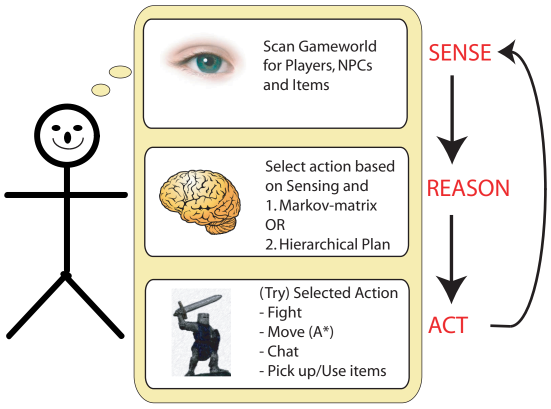


Figure 4: Zereal Agent Architecture

Simulated players and NPCs **senses** using a scan-based vision algorithm that have a configurable vision radius. The vision algorithm can detect walls, items, (other) players and non-personal characters.

There are three types of **reasoning** mechanisms for players and NPCs in Zereal:

1. *Greedy and Agressive* - Both players and NPCs have no social need and will try to pursue the closest player or monster in order to kill it (*Killer* and *Monster* in figure 5).
2. *Action selection using Markov Chain* - The next action is dependent on the previous action and on the possible actions at the current time. E.g. if there are no enemies in sight the 'fight' action in the Markov matrix is pruned and the probabilies for the available actions are normalized. (*MarkovPlayer* in figure 5)
3. *Action selection from hierarchical plan* - The next action is selected from a predefined hierarchical plan (predefined in XML). It also has a long term memory and is able to reason about past experiences, and it is able to use the pathfinding mechanism for determining where to move next. This reasoning mechanism is inspired by the Hap reactive and adaptive architecture used in the Oz project [14]. Plans have so-far been developed for the player types *killer* and *explorer*. (*PlanAgent* in figure 5)

2.8 Scalability of the Zereal Platform

We have used factorial experimental design and identified five (six) factors that have effect on the **scalability** of the simulator. These are: number of players and NPCs, number of CPUs, the vision-radius and size of the world. In addition we have the reasoning factor which is likely to be highly dependent on the reasoning algorithm [22]. By reasoning factor we mean the amount of computational resources spent on AI algorithms for reasoning in the sense-reason-act cycle for each agent.

2.9 Logging of Player Behavior

The main output of Zereal is the log files that are written to disk. The contain player data that can be an important source for data mining in order to find player patterns in MMOGs. Examples of *interesting data mining problems* in MMOGs are:

- How to detect bored or frustrated players?
- How to detect unbalanced features of the game? (E.g. weapons that beat *all* other weapons)
- Which features and data need to be logged to solve the above 2 questions?

The current logging of game events are of 5 types:

1. DEBUG - internal debug messages
2. GAME - game specific events like moves and attacks
3. EVENT - game events regarding receiving and sending agents between subworlds (i.e. CPUs), agent killing and logon/logoff.
4. WARNING - non-critical errors
5. ERROR - severe errors

Logging is done to the local disk of each Subworld's node, and can potentially be joined on the master node for data mining.

Current format of the gamelog file:

```
date||agentID||event||startpos||stoppos||agent type
```

Extracts from a gamelog file:

```
2003-05-22: 12:0:1||6000052||walk||(4,10)||(3,9)||PlanAgent
2003-05-22: 12:0:2||5000018||leaveworld:WEST||(1,2)||(0,3)||PlanAgent
```

Thawonmas et. al used Zereal and its corresponding gamelogs for identification of player types in MMOGs [20].

2.10 Implementation of Zereal

Using Python as the main **implementation** language gives the advantage of high expressiveness and short development time. Regarding performance and memory usage, Python easily competes with Java [18]. Using the stackless implementation of Python gives in adition an extremely fast thread switching, which on most current platforms can reach beyond 1 million switches per second. The heavier tasks like the vision algorithm and A* pathfinding are coded in C/C++ and later glued in to a python module using SWIG. SWIG is a tool for automating the C API coding in order to integrate C/C++ librarys and classes into modules available for languages like Python, Ruby, Perl and Java. PyMPI is used as the interface between the simulator and MPI (Message Passing Interface). The simulator has been tested on parallel clusters with processors based on AMD architecture (at NTNU, Norway) and Intels Xeon architecture (at Ritsumeikan University, Japan).

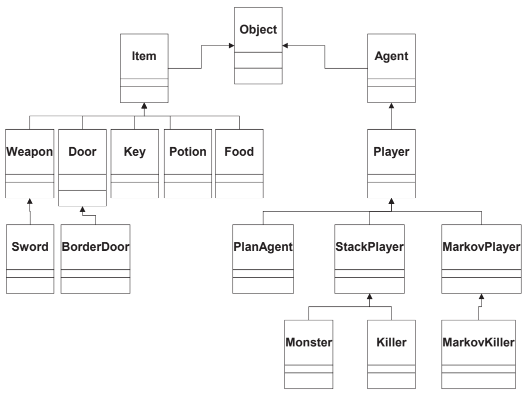


Figure 5: Zereal Class Diagram

3 Related Work

The Creatures software platform simulates agents, represented as virtual pets with human-like behavior, in a graphical environment [9]. Creatures differs from Zereal by focusing on supporting a few simultaneous agents as opposed to Zereal's support for a very large number of simultaneous player or NPC agents.

Scheutz and Rommer proposed an architecture for interactive believable user agents with personality [19]. Their approach differs from Zereal by focusing on natural language support for human-like agents and not on massive scalability in terms of number of agents.

The SimHuman platform resembles Zereal since it simulates human-like agents with planning capabilities [23], but it differs from Zereal since it focuses on a small number of simultaneous agents and a 3D graphical presentation.

Swarm is a toolkit for large scale agent-based simulation, it differs from Zereal by being generic and not geared toward simulation of MMOGs such as Zereal. Another difference is the language support, Swarm supports Objective C, Scheme and Java; Zereal supports Python and C/C++ but is easily extensible to many other languages using the Simplified Wrapper Interface Generator (SWIG).

Python for Massively Multiplayer Virtual Worlds (PMMVW) resembles Zereal in the sense that it provides support for a MMOG and is Python-based, the main difference is that Zereal is focused on simulation of players and PMMVW supports actual humans who play [2].

In addition can various robotic simulations and software for large-scale multiuser animations be considered related, but not equal to the Zereal approach.

4 Empirical Results

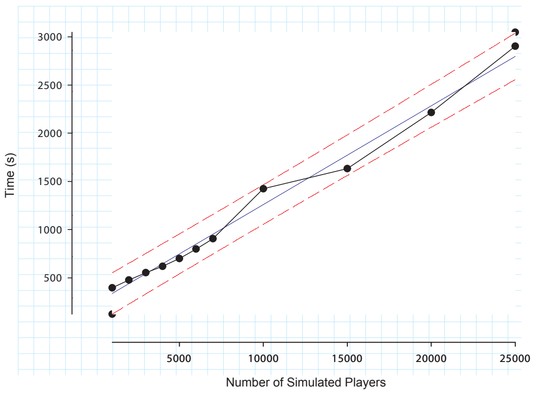


Figure 6: Zereal scalability (with regression line and prediction interval)

In figure 1 the wallclock runtime performance for 5 minutes simulated time in the game (i.e. 300 simulated seconds) as a function of the number of simulated players. The simulation is performed on 5 Athlon 1.6 GHz CPUs on an Linux-cluster. The largest simulation performed so far with Zereal is with 160 thousand agents (simulated players and NPCs) on 20 CPUs. The setup for the experiment was with simulated players and NPCs with Markov chain type reasoning (the largest number of hierarchical planning agents tested so far is 50000). By doubling the number of players and keeping the number of simulated cycles fixed the number of sense-reason-act cycles to be performed doubles, this can explain the close-to-linear scalability.

5 Conclusion and Future Work

We have presented the Zereal Mobile Agent-based Massively Multiplayer Online Game Simulation platform. The implementation has been shown to be close to lineary scalable in terms of number of players simulated. Primary contribution is the platform itself and its scalable implemention that can be used as a testbed for research on MMOGs, in addition to be a MMOG simulation platform Zereal can with minor adaptions provide a generic scalable agent-platform, based on MPI communication it is relatively simple to use on various parallel supercomputers or grid-based environments.

Possible future extensions include: items that can tell the players how they are supposed to be used, player coalition support, guilds, quests, improved intelligence support (e.g. BDI, automatic planning and emotion engine), simulation of natural language based interaction between players and/or NPCs, simulation of e-commerce activities in a MMOG [5] and improved graphical interface support using the VR-environment provided by Silicon Graphics' RAVE.

References

[1] Henrik Andersen. *The CRM Handbook - from group to multi-individual*. PricewaterHouseCoopers, 2000.

[2] Jason Asbahr. Python for Massively Multiplayer Virtual Worlds. O'Reilly Open Source Convention, published Online, July 2001.

[3] Ruth Aylett and Michael Luck. Applying Artificial Intelligence to Virtual Reality: Intelligent Virtual Environments. *Applied Artificial Intelligence*, 14(1):3–32, January 2000.

[4] Butterfly.net: Powering Next-Generation Gaming with On-Demand Computing. Published Online by IBM: `ftp://ftp.software.ibm.com/solutions/pdfs/g325-1938-00.pdf`, January 2003.

[5] Nizami Cummins. Integrating E-Commerce and Games. *Journal of Personal and Ubiquitous Computing*, 6(5-6):362–370, December 2002.

[6] Sorabain Wolfheart de Lioncourt and Michael Luck. Motivating intelligent agents for virtual environments. In *Proceedings of the 2nd International Workshop on Intelligent Virtual Agents (IVA 1999)*, 1999.

[7] Anne C. Elster, Otto Anshus, Amund Tveit, and Cyril Banino. Recent trends in cluster computing. In *Proceedings of the International Conference on Parallel Computing (forthcoming)*. Elsevier Science, September 2003.

[8] Markus Friedl. *Online Game Interactivity Theory*. Charles River Media, 1 edition, October 2002.

[9] Stephen Grand, Dave Cliff, and Anil Malhotra. Creatures: Artificial Life Autonomous Software Agents for Home Entertainment. In *Proceedings of the 1st International Conference on Autonomous Agents*. ACM, ACM Press, 1997.

[10] Neal Hallford and Jana Hallford. *Swords & Circuitry: A Designer's Guide to Computer Role-Playing Games*. Prima Publishing, 1 edition, 2001.

[11] Moon Ihlwan. The champs in online games. *Business Week*, (30):27–28, July 2001.

[12] Zona Inc. and Executive Summary Consulting Inc. State of Massive Multiplayer Online Games 2002: A New World in Electronic Gaming. Technical report, Zona Inc., Redwood City, California, USA, October 2002.

[13] Petri Kuittinen. Introduction to Roguelike Games. Published Online: `http://www.hut.fi/˜eye/roguelike/intro.html`, May 2000.

[14] A. Bryan Loyall and Joseph Bates. Hap: A Reactive, Adaptive Architecture for Agents. Technical Report CMU-CS-91-147, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, June 1991.

[15] Ian Maccines, Janusz Moneta, Julio Caraballo, and Dominic Sarni. Business Models for Mobile Content: The Case of M-Games. *The International Journal of Electronic Commerce & Business Media: Special Section on Electronic Commerce in Entertainment and Media*, 12(4):218–227, 2002.

[16] Gonzalo Mndez, Pedro Prez, and Anglica de Antonio. An Overview of hte Use of Mobile Agents in Virtual Environments. In Anglica de Antonio, Ruth Aylett, and Daniell Ballin, editors, *Proceedings of the 3rd International Workshop on Intelligent Virtual Agents (IVA 2001)*, number 2190 in Lecture Notes in Artificial Intelligence, pages 126–136. Springer-Verlag, 2001.

[17] Bernhard Moulin and Brahim Chaib Draa. An Overview of Distributed Artificial Intelligence. In Greg M. P. O'Hare and Nicholas R. Jennings, editors, *Fundamentals of Distributed Artificial Intelligence*, pages 3–56. John Wiley and Sons, 1996.

[18] Lutz Prechelt. An Empirical Comparison of Seven Programming Languages. *IEEE Computer*, 33(10):23–29, October 2000.

[19] Matthias Scheutz and Brigitte Rommer. Autonomous Avatars? From Users to Agents and Back. In Anglica de Antonio, Ruth Aylett, and Daniell Ballin, editors, *Proceedings of the 3rd International Workshop on Intelligent Virtual Agents (IVA 2001)*, number 2190 in Lecture Notes in Artificial Intelligence, pages 61–71. Springer-Verlag, 2001.

[20] Ruck Thawonmas, Ji-Young Ho, and Yoshitaka Matsumoto. Identification of Player Types in Massively Multiplayer Online Games. In *Proceedings the 34th Annual Conference of the International Simulation and Gaming Association (ISAGA)*. ISAGA, To be published by Springer-Verlag, 2003.

[21] Amund Tveit. Game Usage Mining: Information Gathering for Knowledge Discovery in Massively Multiplayer Games. In Hamid R. Arabnia and Youngson Mun, editors, *Proceedings of the International Conference on Internet Computing (IC'2002), session on Web Mining*, volume III, pages 636–642. CSREA Press, June 2002.

[22] Amund Tveit. Scalability Analysis of the Zereal Massively Multiplayer Game Simulator. Technical Report ISSN: 0802-6394, 12/02, IDI, NTNU, December 2002.

[23] Spyros Vosinakis and Themis Panayiotopoulos. SimHuman: A Platform for Real-Time Virtual Agents with Planning Capabilities. In Anglica de Antonio, Ruth Aylett, and Daniell Ballin, editors, *Proceedings of the 3rd International Workshop on Intelligent Virtual Agents (IVA 2001)*, number 2190 in Lecture Notes in Artificial Intelligence, pages 210–224. Springer-Verlag, 2001.

[24] Rusel Demaria & Johnny L. Wilson. *High Score! The Illustrated History of Electronic Games*. McGraw-Hill/Osborne, 1 edition, 2002.

[25] Michael J. Wooldridge and Nicholas R. Jennings. Intelligent Agents: Theory and Practice. *The Knowledge Engineering Review*, 2(10):115–152, 1995.

[26] Terazona White Paper. Available from Zona Inc., May 2003.

Paper D

```
@TechReport{2003:TR:TveitA,
  author      = {Amund Tveit},
  title       = "{Empirical Performance Evaluation of the
                  Zereal Massively Multiplayer Online Game
                  Simulator}",
  institution = {Divison of Intelligent Systems,
                 Department of Computer and Information Science,
                 Norwegian University of Science and Technology},
  year        = {2003},
  month       = {November}
}
```

Empirical Performance Evaluation of the Zereal Massively Multiplayer Online Game Simulator

Amund Tveit

Division of Intelligent Systems
Department of Computer and Information Science,
Norwegian University of Science and Technology,
N-7491 Trondheim, Norway
`amundt@idi.ntnu.no`

Abstract. This paper presents a factorial experimental design for testing which factors controlling the scalability of the Zereal Massively Multiplayer Online Game simulator. The analysis show that the factors explain approximately 97% of the model (measured in square of errors), which is surprisingly high since the simulation is stochastic.

1 Introduction

In this paper we will investigate the scalability of the Zereal Massively Multiplayer Online Game Simulator, and how much of the response variable that can be explained by the factors (e.g. simulator parameters).

The selected experimental method is factorial design since it enables the measurement of both single impact *and* interaction effects of the parameters used to control Zereal. Selected outcome measurement is wallclock time for a fixed number of simulation steps. More details about the statistical analysis and data can be found in appendix C

1.1 Purpose of Zereal

The main purpose of the Zereal simulator is to provide a scalable (research) testbench for testing models of players, monster intelligence, and various data analysis and data mining approaches to massively multiplayer online games [2]. The alternative way of performing such research would require tight cooperation with major massively multiplayer game vendors like Microsoft, Sony, Electronic Arts, which requires extensive negotiation rounds since (real) player data is one of their greatest assets.

1.2 Goal of analysis

Get a detailed overview of how (and how much) various parameters (i.e. factors) affect the runtime of the simulation.

2 Choice of Factors and Levels

Five factors believed to have effect on the outcome of the experiment are chosen

1. number of agents of type MarkovKillers (per CPU)
2. number of agents of type PlanAgents (per CPU)
3. number of agents of type Monsters (per CPU)
4. number of CPUs
5. vision Radius of agents (for all types)

Table 1. Factor levels

Level	MarkovKillers	PlanAgents	Monsters	CPUs	Radius
1	500	500	50	10	10
2	1000	1000	200	17	20

The number of CPUs and monsters are likely to be very significant for the runtime since they are controlling the amount of movement that MarkovKillers and to lesser extent PlanAgents do. Movement between CPUs leads to (timewise costly) network traffic. Since monsters can't move, they are not likely to increase network traffic, and with computationally cheap Markov-based action-selection they are not likely to increase the CPU load much either. The effect of sight radius is uncertain, but it can potentially lead to more network traffic and more CPU load.

Interaction between factors is likely to occur, in particular between CPUs and monsters, since increasing the number of CPUs and monsters together will give more network traffic due to more doors between subworlds at the CPUs. Note that the number of agents are (initally) *per* CPU (i.e. not total number of agents. This can potentially reduce the effect of the CPUs since the information is somehow present in the factors describing the number of agents.

The selected levels presented in appendix C has been determined in discussions with the implementors (MSc students) of the simulator.

Since this is a computer simulation the selected factors can be completely controlled and set to the wanted levels. Examples of factors that can't be controlled is the load on the computer cluster, but the batch job scheduler minimizes the risk of high load from other applications or systems simultaneous with our experiments.

Choice of Response Variable The selected response variable is the (wall clock) time for running simulations of 100 cycles (i.e. simulating 1 second per cycle) with the variations of levels shown in 1.

Other response variables of interest could be the load of CPUs of the computer systems or type of actions performed by the agents. These are however more computationally expensive to measure than the wallclock time.

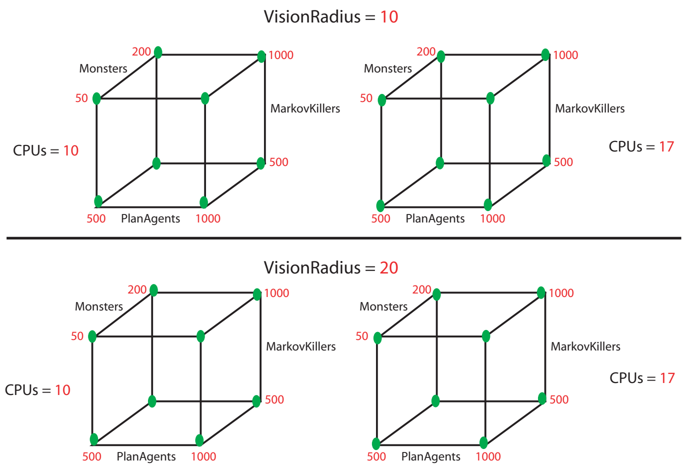


Fig. 1. 2^5 Factorial Combinations

Wall clock time response is measured using the unix shell method *time*. The accuracy of *time* is in milliseconds.

Choice of Experimental Design Selected experimental design is a factorial design with two levels [1]. The design is shown in figure 1.

The experiment is complete factorial (full resolution). There is no need to create this design with blocks.

Experimentation The order of runs, one repeat of 2^5 combinations = 32 runs, are fully randomized using a uniform distribution. The purpose of the randomization is to lower the potential effect on load caused by non-simulator processes on the cluster.

3 Analysis

3.1 Variance Model

The wall clock time T_{ijklm} is assumed to be a function of the parameters in the following way:

- MarkovKillers: α_i
- PlanAgents: β_j

– Monsters: γ_k
– CPUs: δ_l
– Radius: ϵ_m

$$\begin{aligned} T_{ijklm} &= \mu + \alpha_i + \beta_j + \gamma_k + \delta_l + \epsilon_m \\ &(\alpha\beta)_{ij} + (\alpha\gamma)_{ik} + (\alpha\delta)_{il} + (\alpha\epsilon)_{im} \\ &(\beta\gamma)_{jk} + (\beta\delta)_{jl} + (\beta\epsilon)_{jm} + (\gamma\delta)_{kl} + \\ &(\gamma\epsilon)_{lm} + (\delta\epsilon)_{lm} + \zeta_{ijklm} \end{aligned}$$

The error ζ_{ijklm} come from stochastic elements of the experiment (e.g. cpu load) in addition to higher order interaction effects.

Calculations was performed using Minitab's General Linear Model (output in section 6.2). By selecting a confidence level of 5%[1], the following factors have significant effect (checking the P-value):

1. MarkovKillers
2. PlanAgents
3. Monsters
4. CPUs
5. Radius
6. MarkovKillers*Radius
7. PlanAgents*Radius
8. Monsters*Radius

The main factors were likely to have effect, but that 2-factor interactions involving the vision radius are not directly obvious.

The significant main and interaction factors together explain approximately 97.6% of the model based on calculations with Sum of Squares, the exact expression is (1.0 - 206214/211197)*100. The rest of approximately 2.4% is due to true variance from stochastic elements in the experiments.

A plausible explanation of the high main effect of vision radius (fig 1) is that the amount of vision processing increases proportional to the square of the vision radius (processing area = $\pi \cdot (visionradius)^2$). Since adding more agents *and* increasing the vision radius rapidly increases the load of an agents vision algorithm (i.e. sees a larger area with higher density of other agents), the significant 2-factor interactions involving agents and vision radius (fig 1) makes sense.

The main effects satisfy the initial assumptions, but the 2-factor interactions involving vision radius were quite surprising at first.

There is no need to perform more experiments in order to deal with alias structures since the experiment is a based on full factorial design (i.e. disabling alias structures).

[1] not the same α as in the model above!

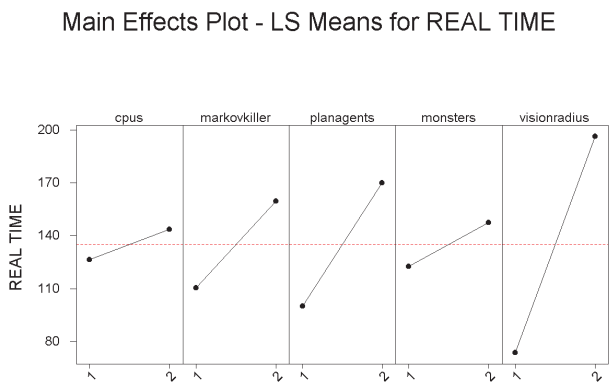


Fig. 2. Main Effects Plot

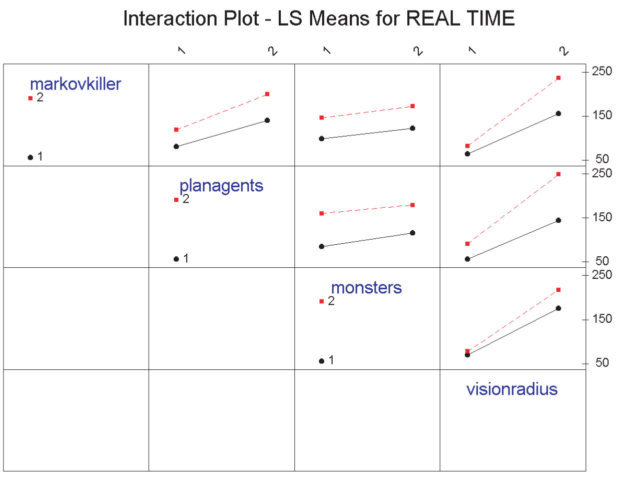


Fig. 3. Interaction Plots

3.2 Scalability of Zereal

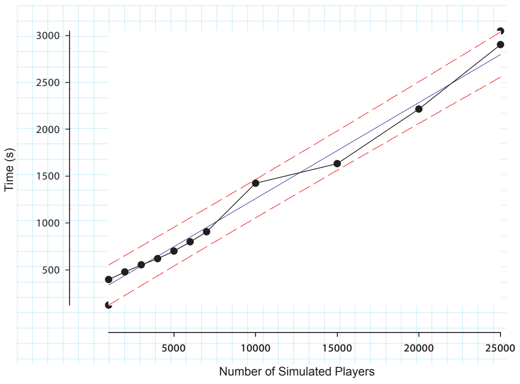


Fig. 4. Zereal scalability (with regression line and prediction interval)

In figure 1 the wallclock runtime performance for 5 minutes simulated time in the game (i.e. 300 simulated seconds) as a function of the number of simulated players. The simulation is performed on 5 Athlon 1.6 GHz CPUs on an Linux-cluster. The largest simulation performed so far with Zereal is with 160 thousand agents (simulated players and NPCs) on 20 CPUs. The setup for the experiment was with simulated players and NPCs with Markov chain type reasoning (the largest number of hierarchical planning agents tested so far is 50000). By doubling the number of players and keeping the number of simulated cycles fixed the number of sense-reason-act cycles to be performed doubles, this can explain the close-to-linear scalability.

4 Conclusion and Recommendations

It has been determined that there is significant interaction between factors in the experiments (MarkovKillers*Radius, PlanAgents*Radius and Monsters*Radius)

The main factors and including all interactions between them explain approximately 97% of the response variable, this is a surprisingly high value since the simulator has highly stochastic behavior of killers and monsters. However, since each run is over a period of 100 cycles, and for each cycle the killers (and monsters) perform one action (or inaction), one gets an “averaging” effect on the measured wallclock time.

In future experiments it would be useful to track the movements of agents between CPUs, and measure how the agents are balanced between CPUs, another factor that also would be interesting to add is the load of the cluster operating system (explaining some of ζ_{ijklm}), and a third interesting factor is topology of the simulated game world and how it is mapped onto the CPUs.

4.1 Scalability of the Zereal Platform

We have used factorial experimental design and identified eight factors that have effect on the **scalability** of the simulator. These significant factors are:

1. MarkovKillers
2. PlanAgents
3. Monsters
4. CPUs
5. Radius
6. MarkovKillers*Radius
7. PlanAgents*Radius
8. Monsters*Radius

References

1. Douglas C. Montgomery. *Design and Analysis of Experiments*, chapter 10, pages 461–466. John Wiley & Sons, Inc., 4th edition, 1997.
2. Amund Tveit, Øyvind Rein, Jørgen Vinne Iversen, and Mihhail Matskin. Scalable Agent-based Simulation of Players in Massively Multiplayer Online Games. In *Proceedings of the 8th Scandinavian Conference on Artificial Intelligence*, Frontiers in Artificial Intelligence and Applications. IOS Press, 2003.

Paper E

```
@InProceedings{2002:IC:Tveit,
  author    = {Amund Tveit and Gisle B. Tveit},
  title     = "{Game Usage Mining: Information Gathering for
                Knowledge Discovery in
                Massively Multiplayer Online Games}",
  booktitle = {Proceedings of the International Conference on
               Internet Computing},
  pages     = {24--27},
  year      = {2002},
  editor    = {Hamid R. Arabnia and Youngsong Mun},
  address   = {Las Vegas, USA}
  volume    = {3},
  month     = {June},
  publisher = {CSREA Press}
}
```

This paper has been cited by Thawonmas [2003].

Game Usage Mining: Information Gathering for Knowledge Discovery in Massive Multiplayer Games

Amund Tveit[*]
Department of Computer and Information Science
Norwegian University of Science and Technology

Gisle B. Tveit
Department of Thermal Energy
Norwegian University of Science and Technology

Abstract

Multiplayer games provide rich sources of information for data mining. The primary purpose of data mining in games is to find patterns of behavior, structure or content in order to improve the overall gameplay, hence keeping players longer and increasing the revenue of the game service. In this paper we define the term *game usage mining* and describe it from *web usage mining* perspective, with the main emphasis on data gathering. A classification of game types from a data mining perspective is also suggested. Based on the discussion we propose content for a *common game log format* suitable for game usage mining.

Keywords: Massive Multiplayer Games, Web Usage Mining, Information Gathering

1 Introduction

Multiplayer games, both on the wireless and the traditional Internet, are excellent sources of information for data mining. We can mine usage patterns of both human and virtual players (avatars), or discover patterns in the games content and structure.

[*]amund.tveit@idi.ntnu.no, IDI/NTNU, N-7491 Trondheim, Norway

In this paper we compare the process of information gathering for behavior mining in massive multiplayer games (game usage mining) to the similar process known from web usage mining.

1.1 Motivation

The main motivation for performing data mining in computer games is to discover patterns, e.g. rules or statistics, that can possibly be used to improve the game. Then paying players become more satisfied and stay longer, which again increases the revenue of the game service. This is of particular importance for wireless games, since keeping a player is equivalent of making more money. Wireless Internet revenue models are either proportional to *money per time unit* spent by the player (e.g. WAP over GSM), or increasingly more common, *money per byte* served to the player (e.g. UMTS, I-Mode and WAP over GPRS).

Areas and connections between areas (e.g. doors or tunnels), are frequent building structures in games. From a web usage mining perspective they can be seen as relatively analog to web pages and links (i.e. URLs), respectively. Player behavior can also be detected, including actions and speech act utterances. This makes it possible generate game logs that resemble web logs in structure, and hence can gain from applying web usage mining methods with only minor adaptions.

1.2 Terminology

In order to have a clear vocabulary to describe data mining in multiplayer games (from now on called *game mining*) we define three main types: 1) *game content mining* – discovery of patterns in multimedia or textual content in games (e.g. room layout), 2) *game structure mining* – discovery of structural patterns in form of paths and connections binding the game world together (e.g. hallways between rooms), and 3) *game usage mining* – discovery of human and avatar behavior patterns. The described types of game mining are inspired by well-known types of web mining – web content mining, web structure mining and web usage mining [2].

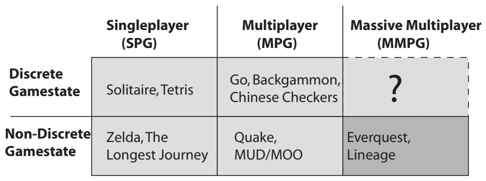
1.3 Research Problem

What are the similarities and differences of information gathering for web usage mining and game usage mining?

The rest of this paper is organized as follows. Section 2 describes a game classification schema. Section 3 describes the Game Usage Mining concept. Section 4 compares information gathering in a web and game context. Section 5 describes the proposed game log approach, and finally the conclusion with future work.

2 Game Classification

From a information gathering viewpoint, games are proposed classified according to figure 1, with examples for each class.



	Singleplayer (SPG)	Multiplayer (MPG)	Massive Multiplayer (MMPG)
Discrete Gamestate	Solitaire, Tetris	Go, Backgammon, Chinese Checkers	?
Non-Discrete Gamestate	Zelda, The Longest Journey	Quake, MUD/MOO	Everquest, Lineage

Figure 1: Computer Game Classification

In the *discrete game-state class* we consider games that have a discrete search space and a turn-based gameplay, e.g. in chinese checkers it is not possible to do fractional (non-integer) moves of marbles, and players have to wait until their turn. In Quake [1] the search space (from a practical view) is non-discrete, and players don't have to wait until their turn, hence Quake belongs to the *non-discrete game-state class.*

In the *singleplayer game class*, the games only have one simultaneous (human) player, as in case of Solitaire where the human plays, and the computer only shuffles the cards.

The difference between the *multiplayer* and the *massive multiplayer* classes is not absolute, but the prior covers games with $2 - 100$ simultaneous players (or same order of magnitude), and the latter covers games with the number of players being $\gg 100$. An example of a (non-discrete game-state) massive multiplayer game is NCSoft's Lineage with approximately 110,000 simultaneous players [4]. We were not able to find examples of discrete gamestate massive multiplayer games.

Another classification schema for computer games is based on genres (e.g. action games, role-playing games, adventure games, sports game etc.) [6]. Genres are suited to classify what the game is about, but not so well suited to classify the amount and what type of data that can be gathered from the game, hence being less useful in a data mining context.

Games can also be classified according to their network architecture (e.g. single node, peer-to-peer, client/server or server-network) [8], which is useful for describing where to collect the data, but doesn't say anything about what type of data to collect.

3 Game Usage Mining Concept

In figure 2 a mobile massive multiplayer game setting is shown. Players interact with the Massive Multiplayer Game Service (MMGS). The MMGS sends events about user actions to the Game Usage Miner (GUM), if the GUM discovers patterns of interest (e.g. player logoff probability) they get passed on to Recommender Service (RS). When the RS receives a pattern it will give a recommendation back to the MMGS about how to alter the player experience. The purpose of this is to improve the gaming experience for players. Methods used in the RS can e.g. be collaborative filtering or case-based reasoning. Another purpose of the GUM is to provide realtime metrics of the game, typically macro numbers such as mood of the community (e.g. number of players likely to log off in the near future).

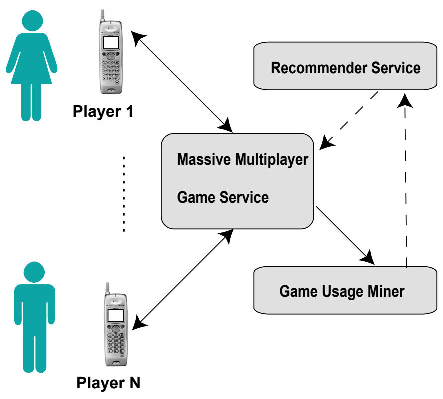


Figure 2: Game Usage Mining Concept

4 Web vs Game Usage Mining

In this section we compare information gathering in non-discrete game-state massive multiplayer games with information gathering from web browsing. The purpose of the information gathering is to enable game usage mining and web usage mining, respectively.

For web usage mining, the primary information source is the highly standardized web log (i.e. common or extended log format) created by the web server. Unfortunately, there are currently no standardized information sources that supports game usage mining in massive multiplayer games.

4.1 User and Player Session

User session time $t_{session}$ is an important concept in web usage mining, it is the estimate of user's active period, i.e. before browsing to another web site or stop surfing.

In game usage mining we propose an approach for estimating player sessions. However, if the player pays per time unit to play, it is usually very simple to determine $t_{session}$, since the user probably will explicitly log off instead of idling, or a least be logged out automatically after a particular idle time of the client (in order to avoid high bills). If the client tells the game server that the logout was caused by timeout for a period t_{idle}, the accurate session time $t_{session}$ can easily be determined by:

$$t_{session} = t_{stop} - t_{start} - t_{idle} \tag{1}$$

Where t_{start} and t_{stop} are timestamps for the start and stop of player session, respectively.

4.2 Log rate

Logging of usage data in web usage mining is purely event driven. When the client (browser) requests a web page or an object it causes a log entry line ($LogEntry_t$) to be written to the web server log. The time resolution of the log is usually limited to whole seconds, so if the web server receives several requests in a second, they are written to the web log in an arbitrary order.

Game playing generate many more events than web browsing, since events are based on real-time actions in the game (e.g driving a car, talking to other avatars, flying on a dragon's back etc.). To avoid overload of the game service's logging mechanism, it has to collect events in memory for a period $t_{collect}$, then possibly prune events of less importance and write the important events during the period $t_{collect}$ to the log. If $t_{collect}$ is relatively short (e.g. 1s or less, depending on game type), we propose that game log entries (from possibly several players) collected in that period can have an arbitrary order as the case for web logs. The reason for this is to avoid costly synchronization in parallel implementations.

4.3 Player Information Attributes

This section describes information attributes of the player – the human customer of the gaming service.

Player identification

The user's IP address is the simplest identification mechanism in web usage mining. However, since many users can have the same IP address (e.g. if surfing through a firewall or proxy gateway), the IP address may not provide a unique user identification.

User identification can be improved by using cookies (small information files residing at the user's browser which can be written to and read by the web server), but cookies have gotten a lot of critics due to privacy issues. This has resulted in that many users disable the cookie support in their browsers.

Player identification ($player_{id}$) is in games is usually much simpler, since the player is becoming uniquely identified when entering the game (e.g. by username or by mobile phone number) and stays identified the whole playing session.

Player Position

For wireless games, the position $player_{pos}$ of the (human) player may be possible to detect (e.g. using GPS positions obtained from the player's wireless device). With the $player_{pos}$, interesting attributes such as $player_{speed}$ and $player_{direction}$ can be estimated (during a session), this can possibly add useful knowledge in a game usage mining context.

Player Characteristics

On the web it is possible to get user system characteristics such as browser version and operating system from the extended web log format.

Games on the other hand, can possibly obtain more interesting data about users, information might include the player's real-life name, address, e-mail, connecting IP address etc (based on manual input by the player when registering).

4.4 Avatar Information Attributes

This section describes information attributes of the avatar – the virtual character that the player controls in the game.

Avatar Actions

A web browsing user generates actions in form of HTTP requests, the most frequent request-type is "GET" used to retrieve pages or objects, but also "POST" and "HEAD" occur relatively frequently.

Action types ($avatar_{action}$) in games are usually much more plentiful and represented as verbs, examples include "jump", "fire", "walk" and "say" etc.

Parameters of Avatar Actions

Parameters of an action in web browsing are usually an encoded path requesting a particular HTML file, application/script or multimedia object. This path may contain its own parameters, e.g. if the path requests a script that sets some variables.

Examples of action parameters ($avatar_{actionparameters}$) in games include: natural language or slang phrases (corresponding to action "say"), power or distance (corresponding to action "fire").

Avatar Path

On the web users can browse between web pages, objects and scripts (discrete movement) by clicking on URLs. However, if items are cached (e.g. by a reverse or standard proxy server, a content provisioning service or the browser cache), browsing will not be detected and logged by the web server. To improve the quality of web logs. java-applets that send messages from the browser to the web server can be applied, other attempts to improve path quality include various heuristic methods [3, 9].

In games it is easier to determine the complete path, but the path is more complex to represent than on the web since it's not necessarily discrete. The player can move the avatar in any direction and not only select between a relatively small set of directions. Possible approaches to represent paths include storing the global game coordinates of the avatar's position ($avatar_{pos}$) at every $t_{collect}$ period, or game area relative coordinates ($avatar_{rpos,area}$), generalized direction ($avatar_{dir}$) and speed ($avatar_{speed}$) together with position data.

Avatar Characteristics

Avatar characteristics may include gender, type of avatar (e.g. elf, troll), occupation (e.g. wizard, warrior) etc. This information can be used in creating profiles of the players.

5 Common Game Log Content

This section summarizes the presented usage attributes found in games, and proposes content for a *common game log.*

To reduce duplicate information, the game log is proposed divided into two files: 1) *player.log* - which contains information about the user that doesn't need to be repeated for each action the player does. This file is updated only for every new session, it's maximum update frequency is $\frac{1}{t_{session}}$, and 2) *playeractions.log* - which contains information logs of the players action, it's maximum update frequency is $\frac{1}{t_{collect}}$.

Attributes of player.log

$player_{id}$, $session_{id}$, $timestamp$, $player_{characteristics}$, $avatar_{characteristics}$

Attributes of playeractions.log

$player_{id}$, $session_{id}$, $player_{pos}$, $timestamp$, $avatar_{pos}$, $avatar_{action}$, $avatar_{actionparameter}$

5.1 Log file format

One problem with usage log files in massive multiplayer games with thousands of players is that they're likely to grow (in size) very rapidly, this has to be considered when storing the data.

An intuitive way of representing the data would be to define XML Schemas or DTDs for player.log and playeractions.log. Unfortunately XML is quite verbose, so an easily compressed, though open and standardized, file format is probably preferable over XML. However, the best approach is probably to allow several types of game log encoding, e.g. both XML and a compressed format. The requirements for a multi-format solution should be simple translation mechanims between the formats.

5.2 Game Usage Mining Process

Figure 3 shows the details the Game Usage Mining part of figure 2.

Information gathering of game log files described earlier is shown in figure 3, part A. In order the make the system scale up to a large number of players, we need to create higher-level aggregated information, e.g. a sequence of rapid avatar position changes will be interpreted as running. This aggregation process is shown in figure 3, part B.

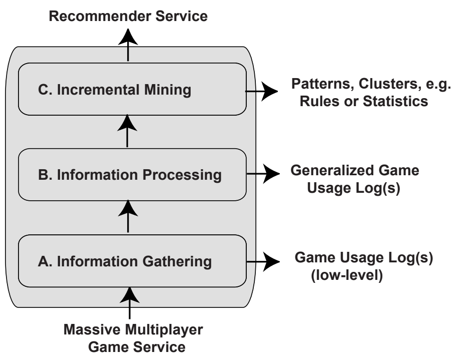

Recommender Service
C. Incremental Mining
Patterns, Clusters, e.g. Rules or Statistics
B. Information Processing
Generalized Game Usage Log(s)
A. Information Gathering
Game Usage Log(s) (low-level)
Massive Multiplayer Game Service

Figure 3: Game Usage Mining Process

An hierarchical approach of dividing high-level actions (e.g. performing a large task) into several sub-tasks with corresponding actions have shown useful in the creation of intelligent Quake monsters [5]. This supports our aggregation approach of actions.

We suggest that the behavior of non-personal characters (e.g. monsters) are also logged and processed in the same way as the players. This in order to be able to recreate the sessions and the experience of the player. These data must be combined with global information about the game (storyline, major events, user interface) in the mining process (part C). Results of the mining process are patterns (e.g. rules or statistics) that can either be used as input to a recommender service or as metrics to the game service operator(s).

Acknowledgements

We would like to thank Magnus Lie Hetland for fruitful discussions about computer games in general. We would also like to thank Professor Mihhail Matskin. This work is supported by the Norwegian Research Council in the framework of the Distributed Information Technology Systems (DITS) program and the (ElComAg) project.

6 Conclusion

The contribution of this paper has been threefold, first we defined types of data mining in computer, second we provided a classification of computer games from a data mining viewpoint, and third we compared information gathering in web usage mining and game usage mining as well as proposing a common game log format to enable game usage mining.

Future work include determining the actual representation of game log files (e.g. which attributes and which concrete efficient representation), determine what type of usage mining is most useful in massive multiplayer computer games (e.g clustering, classification and incremental sequence mining [10, 7]), and how existing web usage mining architectures and systems can be adapted to a support scalable game usage mining setting.

References

[1] Ahmed Abdelkhalek, Angelos Bilas, and Andreas Moshovos. Behavior and performance of interactive multi-player game servers. In *Proceedings of IEEE International Symposium on Performance Analysis of Systems and Software*. IEEE, November 2001.

[2] Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. Web mining: Information and pattern discovery on the world wide web. In *Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'97)*. IEEE, November 1997.

[3] Robert Cooley, Bamshad Mobasher, and Jaideep Srivastava. Data preparation for mining world wide web browsing patterns. *Knowledge and Information Systems*, 1(1):5–32, February 1999.

[4] Moon Ihlwan. The champs in online games. *Business Week*, pages 27–28, July 23, 2001.

[5] John Laird. It knows what you're going to do: Adding anticipation to a quakebot. In *Proceedings of the 5th International Conference on Autonomous Agents*. ACM, ACM Press, 2001.

[6] John Laird and Michael Van Kent. Human-Level AI's Killer Application: Interactive Computer Games. *AI Magazine*, 22(2):15–26, 2001.

[7] Florent Masseglia, Pascal Poncelet, and Maguelone Teisseire. Web usage mining: How to efficiently manage new transactions and new clients. In *Proceedings of the 4th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD'00)*, September 2000.

[8] Jouni Smed, Timo Kaukoranta, and Harri Hakonen. Aspects of networking in multiplayer computer games. In Loo Wai Sing, Wan Hak Man, and Won Wai, editors, *Proceedings of International Conference on Application and Development of Computer Games in the 21st Century*, pages 74–81, November 2001.

[9] Jaideep Srivastava, Robert Cooley, Mukund Deshpande, and Pang-Ning Tan. Web usage mining: Discovery and applications of usage patterns from web data. *SIGKDD Explorations*, 1(1):12–23, January 2000.

[10] Ron Sun. Introduction to sequence learning. In Ron Sun and C. Lee Giles, editors, *Sequence Learning - Paradigms, Algorithms, and Applications*, volume 1828 of *Lecture Notes in Computer Science*, pages 1–10. Springer-Verlag, 2001.

Paper F

```
@InProceedings{2003:KES:Tveit,
  author    = {Amund Tveit and Magnus Lie Hetland},
  title     = "{Multicategory Incremental Proximal Support
               Vector Classifiers}",
  booktitle = {Proceedings of the 7th International Conference
              on Knowledge-Based Information \& Engineering
              Systems (KES'2003)},
  pages     = {386--392},
  year      = {2003},
  series    = {Lecture Notes in Artificial Intelligence (LNAI)},
  number    = {2773 (Part I)},

  publisher = {Springer-Verlag}
}
```

Multicategory Incremental Proximal Support Vector Classifiers

Amund Tveit and Magnus Lie Hetland

Department of Computer and Information Science,
Norwegian University of Science and Technology,
N-7491 Trondheim, Norway
{amundt,mlh}@idi.ntnu.no

Abstract. Support Vector Machines (SVMs) are an efficient data mining approach for classification, clustering and time series analysis. In recent years, a tremendous growth in the amount of data gathered has changed the focus of SVM classifier algorithms from providing accurate results to enabling incremental (and decremental) learning with new data (or unlearning old data) without the need for computationally costly retraining with the old data. In this paper we propose an efficient algorithm for multicategory classification with the incremental proximal SVM introduced by Fung and Mangasarian.

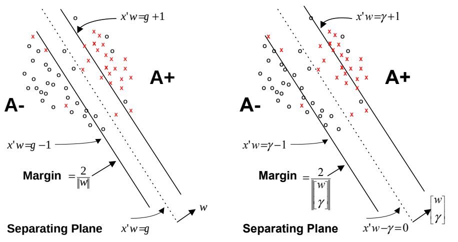
1 Introduction

Support Vector Machines (SVMs) are an efficient data mining approach for classification, clustering and time series analysis [1–3]. In recent years, a tremendous growth in the amount of data gathered (for example, in e-commerce and intrusion detection systems) has changed the focus of SVM classifier algorithms from providing accurate results to enabling incremental (and decremental) learning with new data (or unlearning old data) without the need for computationally costly retraining with the old data. Fung and Mangasarian [4] introduced the Incremental and Decremental Linear Proximal Support Vector Machine (PSVM) for binary classification and showed that it could be trained extremely efficiently, with one billion examples (500 increments of two million examples) in two hours and twenty-six minutes on relatively low-end hardware (400 MHz Pentium II).

In this paper we propose an efficient algorithm based on memoization, in order to support Multicategory Classification for the Incremental PSVM.

2 Background Theory

The standard binary SVM classification problem with soft margin (allowing some errors) is shown visually in Fig. 1(a) and as a constrained quadratic programming problem in (1). Intuitively, the problem is to maximize the margin between the solid planes and at the same time permit as few errors as possible, errors being positive class points on the negative side (of the solid line) or vice versa.


x'w=g +1
A+
A-
x'w=g −1
Margin $= \frac{2}{\|w\|}$
w
Separating Plane x'w=g
x'w=γ+1
A+
A-
x'w=γ−1
Margin $= \frac{2}{\left\|\begin{bmatrix} w \\ \gamma \end{bmatrix}\right\|}$
$\begin{bmatrix} w \\ \gamma \end{bmatrix}$
Separating Plane x'w−γ=0

(a) Standard SVM classifier

(b) Proximal SVM classifier

Fig. 1. SVM and PSVM

$$\begin{aligned} \min_{(w,\gamma,y)\in\mathbb{R}^{n+1+m}} \quad & \{ve'y + \tfrac{1}{2}w'w\} \\ \text{s.t. } & D(Aw - e\gamma) + y \geq e \\ & y \geq 0 \end{aligned} \tag{1}$$

$$A \in \mathbb{R}^{m\times n},\ D \in \{-1,+1\}^{m\times 1},\ e = 1^{m\times 1}$$

Fung and Mangasarian [5] replaced the inequality constraint in (1) with an equality constraint. This changed the binary classification problem, because the points in Fig. 1(b) are no longer bounded by the planes, but are clustered around them. By solving the equation for y and inserting the result into the expression to be minimized, one gets the following unconstrained optimization problem:

$$\min_{(w,\gamma)\in\mathbb{R}^{n+1+m}} f(w,\gamma) = \tfrac{\nu}{2}\|D(Aw - e\gamma) - e\|^2 + \tfrac{1}{2}(w'w + \gamma^2) \tag{2}$$

Setting $\nabla f = \left(\frac{\partial f}{\partial w}, \frac{\partial f}{\partial \gamma}\right) = \mathbf{0}$ one gets:

$$\underbrace{\begin{pmatrix} w \\ \gamma \end{pmatrix}}_{\mathcal{X}} = \begin{pmatrix} A'A + \frac{I}{\nu} & -A'e \\ -e'A & \frac{1}{\nu} + m \end{pmatrix}^{-1} \begin{pmatrix} A'De \\ -e'De \end{pmatrix} = \underbrace{\left(\frac{I}{\nu} + E'E\right)^{-1}}_{\mathcal{A}^{-1}} \underbrace{E'De}_{\mathcal{B}} \tag{3}$$

$$E = [A - e],\ E \in \mathbb{R}^{m\times(n+1)}$$

Fung and Mangasarian [6] later showed that (3) can be rewritten to handle increments (E^i, d^i) and decrements (E^d, d^d), as shown in (4). This decremental approach is based on time windows.

$$\begin{aligned}\mathcal{X} &= \begin{pmatrix} w \\ \gamma \end{pmatrix} \\ &= \left(\frac{I}{\nu} + E'E + (E^i)'E^i - (E^d)'E^d\right)^{-1} \left(E'd + (E^i)'d^i - (E^d)'d^d\right) \ , \end{aligned} \tag{4}$$

where

$$d = De \ .$$

3 Incremental Proximal SVM for Multiple Classes

In the multicategorical classification case, the (incremental) class label vector d^i consists of m_i numeric labels in the range $\{0, \ldots, c-1\}$, where c is the number of classes, as shown in (5).

$$\mathcal{X} = \begin{pmatrix} w_0 \ldots w_{c-1} \\ \gamma_0 \ \cdots \ \gamma_{c-1} \end{pmatrix} = \mathcal{A}^{-1}\mathcal{B} \tag{5}$$

3.1 The Naive approach

In order to apply the proximal SVM classifier in a "one-against-the-rest" manner, the class labels must be transformed into vectors with $+1$ for the positive class and -1 for the rest of the classes, that is, $\Theta(cm_i)$ operations in total, and later $\Theta(cm_in)$ for calculating $(E^i)'d$ for each class. The latter (column) vectors are collected in a matrix $\mathcal{B} \in \mathbb{R}^{(n+1)c}$. Because the training features represented by E^i are the same for all the classes, it is enough to calculate $\mathcal{A} \in \mathbb{R}^{(n+1)^2}$ once, giving $\Theta(m_i(n+1)^2 + (n+1)^2)$ operations for calculating $(E^i)'E^i$ and adding it to $\frac{I}{\nu} + E'E$. The specifics are shown in shown in Algorithm 1.

Theorem 1. *The running time complexity of Algorithm 1 is $\Theta(cm_{inc}n)$.*

Proof. The conditional statement in lines 3–7 takes $\Theta(1)$ time and is performed m_{inc} times (inner loop, lines 2–8) per iteration of *classId* (outer loop, line 1–10). Calculation of the matrix-vector product $\mathcal{B}[classId,]$ in line 9 takes $\Theta((n+1)m_{inc})$ per iteration of *classId*. This gives a total running time of

$$\Theta(c \cdot (m_{inc} + m_{inc}(n+1))) = \Theta(cm_{inc}n) \ .$$

□

Algorithm 1 *calcℬ_Naive*(E_{inc}, d_{inc})

```
Require: $E_{inc} \in \mathbb{R}^{m_{inc}x(n+1)}$, $d_{inc} \in \{0,\ldots,c-1\}^{m_{inc}}$ and $n, m_{inc} \in \mathbb{N}$
Ensure: $\mathcal{B} \in \mathbb{R}^{(n+1)xc}$
 1: for all $classId$ in $\{0,\ldots,c-1\}$ do
 2:   for all $idx$ in $\{0,\ldots,m_{inc}-1\}$ do
 3:     if $d_{inc}[idx] = classId$ then
 4:       $d_{classId}[idx,] = +1$
 5:     else
 6:       $d_{classId}[idx,] = -1$
 7:     end if
 8:   end for
 9:   $\mathcal{B}[classId,] = E'_{inc} d_{classId}$
10: end for
11: return $\mathcal{B}$
```

3.2 The Memoization Approach

The $d_{classId}$ vectors, c in all, (in line 3 of Algorithm 1) are likely to be unbalanced, that is, have many more -1 values than $+1$ values. However, if there are more than two classes present in the increment d_i, the vectors will at least share one index position where the value is -1. With several classes present in the increment d_i, the matrix-vector products $(E^i)' d_{classId}$ actually perform duplicate calculations each time there exists two or more $d_{classId}$ vectors that have -1 values in the same position.

The basic idea for the memoization approach (Algorithm 3) is to only calculate the $+1$ positions for each vector $d_{classId}$ by first creating a vector $F = -[E^i_{\cdot j}]$ (a vector with the negated sum of E's columns, equivalent to multiplying $E^{i'}$ with a vector filled with -1) and then to calculate the $d_{classId}$ vectors using F and only switching the -1 to a $+1$ by adding the row vector of E twice if the row in $d_{classId}$ is equal to $+1$. In order to do this efficiently, an index of d_i for each class ID has to be created (Algorithm 2).

Algorithm 2 *buildClassMap*(d_{inc})

```
Require: $d_{inc} \in \{0,\ldots,c-1\}^{m_{inc}}$ and $m_{inc} \in \mathbb{N}$
1: $classMap$ = array of length $c$ containing empty lists
2: for all $idx = 0$ to $m_{inc} - 1$ do
3:   append $idx$ to classMap[$d_{inc}[idx,]$]
4: end for
5: return $classMap$
```

Theorem 2. *The running time complexity of Algorithm 2 is* $\Theta(m_{inc})$.

Proof. Appending idx to a the tail of a linked list takes $\Theta(1)$ time, lookup of $classMap[d_{inc}[idx,]]$ in the directly addressable arrays $classMap$ and d_{inc} also takes $\Theta(1)$ time, giving a total for line 3 of $\Theta(1)$ time per iteration of *idx*. *idx* is iterated m_{inc} times, giving a total of $\Theta(m_{inc})$ time.

□

Algorithm 3 calc$\mathcal{B}$_Memo($E_{inc}, d_{inc}, E_{inc}{}' E_{inc}$)

```
Require: $E_{inc} \in \mathbb{R}^{m_{inc} \times (n+1)}$, $d_{inc} \in \{0, \ldots, c-1\}^{m_{inc}}$ and $n, m_{inc} \in \mathbb{N}$
Ensure: $\mathcal{B} \in \mathbb{R}^{(n+1)xc}$, $\mathcal{F} \in \mathbb{R}^{(n+1)}$
1: $classMap = buildClassMap(d_{inc})$
2: for all $classId$ in $\{0, \ldots, c-1\}$ do
3:     $\mathcal{B}[classId,] = E_{inc}{}' E_{inc}[n]$
4:     for all $idx$ in $classMap[classId,]$ do
5:         $\mathcal{B}[idx, classId,] = \mathcal{B}[idx, classId,] + 2 \cdot \sum_{j=0}^{n} E_{inc}[idx, j]$
6:     end for
7: end for
8: return $\mathcal{B}$
```

Theorem 3. *The running time complexity of Algorithm 3 is $\Theta(n(c + m_{inc})$.*

Proof. Calculation of *classMap* (line 1) takes $\Theta(m_{inc})$ time (from Theorem 2). Line 3 takes $\Theta(n+1)$ time per iteration of *classId*, giving a total of $\Theta(c(n+1))$. Because *classMap* provides a complete indexing ($|\bigcup_{u=0}^{c-1} classMap[u]| = m_{inc}$) of the class labels in d_{inc}, and because there are no repeated occurrences of *idx* for different *classId*s ($\bigcap_{u=0}^{c-1} classMap[u] = \emptyset$), line 5 will run a total of m_{inc} times.

This gives a total running time of

$$\begin{aligned}\Theta(&m_{inc} + (n+1)m_{inc} + c(n+1) + m_{inc}) \\ &= \Theta(n(c + m_{inc})) \ .\end{aligned}$$

□

Corollary 1. *Algorithms 1 and 3 calculate the same $\mathcal{B}$ if provided with the same input.*

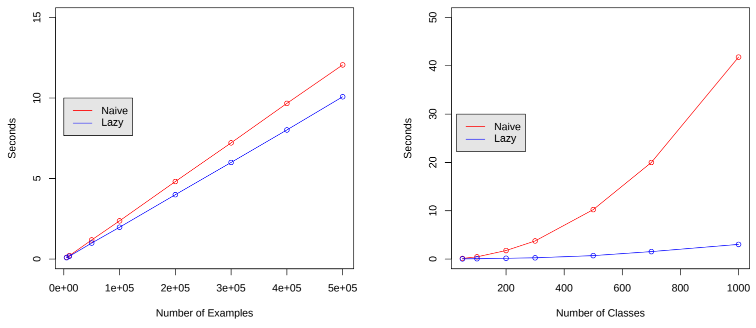
4 Empirical Results

In order to test and compare the computational performance of the incremental multicategory proximal SVMs with the naive and lazy algorithms, we have used three main types of data:

1. Forest cover type, 580012 training examples, 7 classes and 54 features (from UCI KDD Archive [7])
2. Synthetic datasets with a large number of classes (up to 1000 classes) and 30 features

3. Synthetic dataset with a large number of examples (10 million), 10 features and 10 classes

The results for the first two data sets are shown in Fig. 4; the average time from tenfold cross-validation is used. For the third data set, the average classifier training times were 18.62 s and 30.64 s with the lazy and naive algorithm, respectively (training time for 9 million examples, testing on 1 million).



(a) Runtime vs examples (cover type) (b) Runtime vs classes (synthetic)

Fig. 2. Computational performance: training time

The incremental multicategory proximal SVM was been implemented in C++ using the CLapack and ATLAS libraries. The tests were run on an Athlon 1.53 GHz PC with 1 GB RAM running Red Hat Linux 2.4.18.

Acknowledgements

We would like to thank Professor Mihhail Matskin and Professor Arne Halaas. This work is supported by the Norwegian Research Council in the framework of the Distributed Information Technology Systems (DITS) program, and the ElComAg project.

5 Conclusion and Future Work

We have introduced the multiclass incremental proximal SVM and shown a computational improvement for training the multiclass incremental proximal SVM, which works particularly well for classification problems with a large number of classes. Another contribution is the implementation of the system (available on request).

Future work includes applying the algorithm to demanding incremental classification problems, for example, web page prediction based on analysis of click streams or automatic text categorization. Algorithmic improvements that need to be done include (1) develop balancing mechanisms (in order to give hints for pivot elements to the applied linear system solver for reduction of numeric errors), (2) add support for decay coefficients for efficient decremental unlearning, (3) investigate the appropriateness of parallelized incremental proximal SVMs, (4) strengthen implementation with support for tuning set, kernels as well as one-against-one classifiers.

References

1. Burbidge, R., Buxton, B.F.: An introduction to support vector machines for data mining. In Sheppee, M., ed.: Keynote Papers, Young OR12, University of Nottingham, Operational Research Society, Operational Research Society (2001) 3–15
2. Huang, J., Shao, X., Wechsler, H.: Face pose discrimination using support vector machines (svm). In: Proceedings of 14th Int'l Conf. on Pattern Recognition (ICPR'98), IEEE (1998) 154–156
3. Muller, K.R., Smola, A.J., Ratsch, G., Scholkopf, B., Kohlmorgen, J., Vapnik, V.: Predicting time series with support vector machines. In: ICANN. (1997) 999–1004
4. Fung, G., Mangasarian, O.L.: Incremental support vector machine classification. In Grossman, R., Mannila, H., Motwani, R., eds.: Proceedings of the Second SIAM International Conference on Data Mining, SIAM (2002) 247–260
5. Fung, G., Mangasarian, O.L.: Multicategory Proximal Support Vector Classifiers. Submitted to Machine Learning Journal (2001)
6. Schwefel, H.P., Wegener, I., Weinert, K., eds.: 8. Natural Computing. In: Advances in Computational Intelligence: Theory and Practice. Springer-Verlag (2002)
7. Hettich, S., Bay, S.D.: The UCI KDD archive. `http://kdd.ics.uci.edu` (1999)

Paper G

```
@InProceedings{2003:DAWAK:Tveit,
  author    = {Amund Tveit and Magnus Lie Hetland and
               H{\aa}vard Engum},
  title     = "{Incremental and Decremental Proximal
                Support Vector Classification using
                Decay Coefficients}",
  booktitle = {Proceedings of the 5th International Conference
               on Data Warehousing and Knowledge Discovery
               (DAWAK'2003)},
  year      = {2003},
  series    = {Lecture Notes in Computer Science (LNCS)},
  number    = {2737},
  pages     = {422--429},
  editor    = {Yahiko Kambayashi and Mukesh Mohania and
               Wolfram Woss},
  address   = {Prague, Czech Republic}
  month     = {September},
  publisher = {Springer-Verlag}
}
```

Incremental and Decremental Proximal Support Vector Classification using Decay Coefficients

Amund Tveit, Magnus Lie Hetland and Håvard Engum

Department of Computer and Information Science,
Norwegian University of Science and Technology,
N-7491 Trondheim, Norway
{amundt,mlh,havare}@idi.ntnu.no

Abstract. This paper presents an efficient approach for supporting decremental learning for incremental proximal support vector machines (SVM). The presented decremental algorithm based on decay coefficients is compared with an existing window-based decremental algorithm, and is shown to perform at a similar level in accuracy, but providing significantly better computational performance.

1 Introduction

Support Vector Machines (SVMs) is an exceptionally efficient data mining approach for classification, clustering and time series analysis [5, 12, 4]. This is primarily due to SVMs highly accurate results that are competitive with other data mining approaches, e.g. artificial neural networks (ANNs) and evolutionary algorithms (EAs). In recent years tremendous growth in the amount of data gathered (e.g. user clickstreams on the web, in e-commerce and in intrusion detection systems), has changed the focus of SVM classifier algorithms to not only provide accurate results, but to also enable online learning, i.e. incremental and decremental learning, in order to handle concept drift of classes [2, 13].

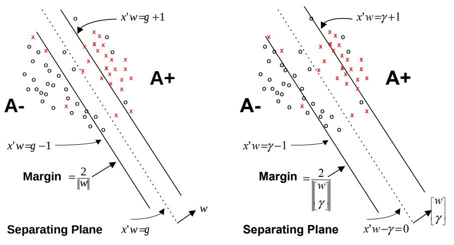
Fung and Mangasarian introduced the Incremental and Decremental Linear Proximal Support Vector Machine (PSVM) for binary classification [10], and showed that it was able to be trained extremely fast, i.e. with 1 billion examples (500 increments of 2 million) in 2 hours and 26 minutes on relatively low-end hardware (400 MHz Pentium II). This has later been extended to support efficient support of incremental multicategorical classification [16]. Proximal SVMs has also been shown to perform at a similar level of accuracy as regular SVMs and at the same time being significantly faster [9].

In this paper we propose a computationally efficient algorithm that enables decremental support for Incremental PSVMs using a weight decay coefficient. The suggested approach is compared the current time-window based approach proposed by Fung and Mangasarian [10].

2 Background Theory

The basic idea of Support Vector Machine classification is to find an optimal maximal margin separating hyperplane between two classes. Support Vector Machines uses an implicit nonlinear mapping from input-space to a higher dimensional feature-space using kernel-functions, in order to find a hyperplane of problems which are not linear separable in input-space [7, 18]. Classifying multiple classes is commonly performed by combining several binary SVM classifiers in a tournament manner, either one-against-all or one-against-one, the latter approach requiring substantial more computational effort [11].

The standard binary SVM classification problem with soft margin (allowing some errors) is shown visually in Fig. 1(a). Intuitively, the problem is to maximize the margin between the solid planes and at the same time permit as few errors as possible, errors being positive class points on the negative side (of the solid line) or vice versa.



(a) Standard SVM classifier

(b) Proximal SVM classifier

Fig. 1. SVM and PSVM

The standard SVM problem can be stated as a quadratic optimization problem with constraints, as shown in (1).

$$\begin{aligned} \min_{(w,\gamma,y)\in\mathbb{R}^{n+1+m}} \quad & \{ve'y + \tfrac{1}{2}w'w\} \\ \text{s.t. } D(Aw - e\gamma) + y &\geq e \\ y &\geq 0 \end{aligned} \tag{1}$$

$$A \in \mathbb{R}^{m\times n},\ D \in \{-1,+1\}^{m\times 1},\ e = 1^{m\times 1}$$

Fung and Mangasarian [8] replaced the inequality constraint in (1) with an equality constraint. This changed the binary classification problem, because the points in Fig. 1(b) are no longer bounded by the planes, but are clustered around them. By solving the equation for y and inserting the result into the expression to be minimized, one gets the following unconstrained optimization problem:

$$\min_{(w,\gamma)\in\mathbb{R}^{n+1+m}} f(w,\gamma) = \frac{\nu}{2}\|D(Aw - e\gamma) - e\|^2 + \frac{1}{2}(w'w + \gamma^2) \tag{2}$$

Setting $\nabla f = \left(\frac{\partial f}{\partial w}, \frac{\partial f}{\partial \gamma}\right) = \mathbf{0}$ one gets:

$$\underbrace{\begin{pmatrix} w \\ \gamma \end{pmatrix}}_{\mathcal{X}} = \begin{pmatrix} A'A + \frac{I}{\nu} & -A'e \\ -e'A & \frac{1}{\nu} + m \end{pmatrix}^{-1} \begin{pmatrix} A'De \\ -e'De \end{pmatrix} = \underbrace{\left(\frac{I}{\nu} + E'E\right)^{-1}}_{\mathcal{A}^{-1}} \underbrace{E'De}_{\mathcal{B}} \tag{3}$$

$$E = [A - e],\ E \in \mathbb{R}^{m\times(n+1)}$$

Agarwal has showed that the Proximal SVM is directly transferable to a ridge regression expression [1]. Fung and Mangasarian [10] later showed that (3) can be rewritten to handle increments (E^i, d^i) and decrements (E^d, d^d), as shown in (4). This decremental approach is based on time windows.

$$\begin{aligned} \mathcal{X} &= \begin{pmatrix} w \\ \gamma \end{pmatrix} \\ &= \left(\frac{I}{\nu} + E'E + (E^i)'E^i - (E^d)'E^d\right)^{-1} \left(E'd + (E^i)'d^i - (E^d)'d^d\right)\ , \end{aligned} \tag{4}$$

where $d = De$

.

3 PSVM Decremental Learning using Weight Decay Coefficient

The basic idea is to reduce the effect of the existing (old) accumulated training knowledge $E'E$ with an exponential weight decay coefficient α.

$$\begin{pmatrix} w \\ \gamma \end{pmatrix} = \left(\tfrac{I}{\nu} + \alpha \cdot E^{'}E + E^{i^{'}}E^{i}\right)^{-1} \left(\alpha \cdot E^{'}d + E^{i^{'}}d^{i}\right)\ ;\ \alpha \in \langle 0, 1] \tag{5}$$

As opposed to the decremental approach in expression (4), the presented weight decay approach *does not require storage of increments* $(E^{i^{'}}E^{i}, E^{i^{'}}d^{i})$ later to be retrieved as decrements $(E^{d^{'}}E^{d}, E^{d^{'}}d^{d})$.

A hybrid approach is shown in expression (6), where one has both a soft decremental effect using the weight decay coefficient α as well as a hard decremental effect using a fixed window of size W increments.

$$\begin{pmatrix} w \\ \gamma \end{pmatrix} = \left(\frac{I}{\nu} + \alpha \cdot E^{'}E + E^{i^{'}}E^{i} - \alpha^{W} \cdot E^{d^{'}}E^{d}\right)^{-1} \cdot \left(\alpha \cdot E^{'}D + E^{i^{'}}D^{i} - \alpha^{W} \cdot E^{d^{'}}D^{d}\right) \; ; \alpha \in \langle 0, 1] \tag{6}$$

4 Related Work

Syed et al. presented an approach for handling concept drift with SVM [2]. Their approach trains on data, and keeps only the support vectors representing the data before (exact) training with new data *and* the previous support vectors. Klinkenberg and Joachims presented a window adjustment based SVM method for *detecting* and handling concept drift [13]. Cauwenberghs and Poggio proposed an incremental and decremental SVM method based on a different approximation than used by us [6].

5 Empirical results

In order to test and compare our suggested decremental PSVM learning approach with the existing window-based approach we created synthetic binary classification data sets with simulated concept drift. This was created by sampling feature values from a multivariate normal distribution where the covariance matrix $\Omega = I$ (identity matrix) and the mean vector μ was sliding linearly from only $+1$ values to -1 values for the positive class case, and vice versa for the negative class [14], as shown in algorithm 1.

Algorithm 1 *simConceptDrift($nFeat, nSteps, nExPerStep, start$)*

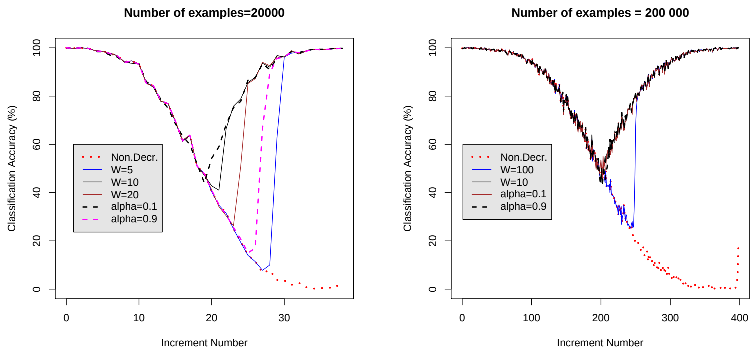
```
Require: $nFeat, nSteps, nExPerStep \in \mathbb{N}$ and $start \in \mathbb{R}$
Ensure: Linear stochastic drift in $nSteps$ from $start$ to $-start$
1: $center = [start, \ldots, start]$ {vector of length $nFeat$}
2: $origcenter = center$
3: for all $step$ in $\{0, \ldots, nSteps - 1\}$ do
4:     for all $synthExampleCount$ in $\{0, \ldots, nExPerStep - 1\}$ do
5:         sample example from multivar.gauss.dist with $\mu = center$ and $\sigma^2$'s $= 1$
6:     end for
7:     $center = origcenter \cdot (1 - 2 \cdot \frac{step+1}{nStep-1})$ {concept drift}
8: end for
```

5.1 Classification Accuracy

For the small concept drift test (20000 examples with 10 features and 40 increments of 500 examples, figure 2(a)), the weight decay of $\alpha = 0.1$ performs slightly better in terms of unlearning than a window size of $W = 5$, and a weight decay of $\alpha = 0.9$ performs between unlearning with $W = 10$ and $W = 20$, and the unlearning performance varies quite a bit with α.

For the medium concept drift test (200000 examples with 10 features and 400 increments of 500 examples, figure 2(b)), the value of α matters less, this due to more increments shown and faster exponential effect of the weight decay coefficient than in the small concept drift test.

As seen in both figure 2(a) and 2(b), there is "dip" in classification performance around their respective center points (increment number $\approx$ 20 and 200). This is caused by concept drift, i.e. the features of the positive and negative class are indiscernible.



(a) Short timespan (b) Medium timespan

Fig. 2. Classification Accuracy under Concept Drift

5.2 Computational Performance

As shown in figure 5.2 the computational performance (measured in wallclock time) of the weight decay based approach is almost twice as fast as the window-based approach except for large windows (e.g. $W = 1000$). The performance difference seems to decrease with increasing increment size, this is supported by the P-values from T-test comparisons. 21 out of 27 T-tests (tables 1-3) showed significant difference in favor of the weight decay based approach over the window based approach. Performed T-tests were based on timing of ten repeated runs of each presented configuration of α, w and increment size.

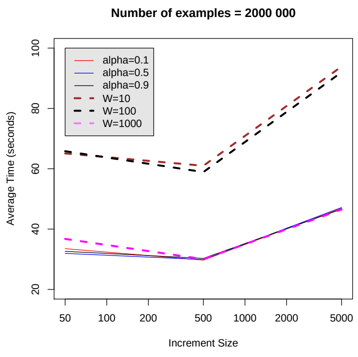


Fig. 3. Computational Performance (Long timespan)

	w=10	w=100	w=1000
α=0.1	0.00	0.00	0.01
α=0.5	0.00	0.00	0.00
α=0.9	0.00	0.00	0.00

Table 1. P-values for increment size 50 (Comp. Perf.)

5.3 Implementation and Test environment

The incremental and decremental proximal SVM has been implemented in C++ using the CLapack and ATLAS libraries [3, 19]. Support for Python and Java interfaces to the library is currently under development using the “Simplified Wrapper and Interface Generator”[15]. A Linux cluster (Athlon 1.4-1.66 GHz nodes, Sorceror Linux) has served as the test environment.

Acknowledgements

We would like to thank Professor Mihhail Matskin and Professor Arne Halaas. This work is supported by the Norwegian Research Council.

6 Conclusion and Future Work

We have introduced a weigth decay based decremental approach for proximal SVMs and shown that it can replace the current window-based approach. The

	w=10	w=100	w=1000
α=0.1	0.00	0.00	0.25
α=0.5	0.00	0.00	0.39
α=0.9	0.00	0.00	0.67

Table 2. P-values for increment size 500 (Comp. Perf.)

	w=10	w=100	w=1000
α=0.1	0.00	0.00	0.67
α=0.5	0.00	0.00	0.79
α=0.9	0.00	0.00	0.57

Table 3. P-values for increment size 5000 (Comp. Perf.)

weight decay based approach is significantly faster than the window-based approach (due to less IO-requirements) for small-to-medium increment and window sizes, this is supported by simulation and p-values from T-Test.

Future work includes applying the approach on demanding incremental classification and prediction problems. e.g. game usage mining [17]. Algorithmic improvements that needs to be done include 1) develop incremental multiclass balancing mechanisms, 2) investigate the approriateness of parallellized incremental proximal SVMs, 3) strengthen implementation with support for tuning set and kernels.

References

1. Deepak K. Agarwal. Shrinkage Estimator Generalizations of Proximal Support Vector Machines. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 173–182. ACM Press, 2002.
2. Nadeem Ahmed, Huan Liu, and Kah Kay Sung. Handling Concept Drifts in Incremental Learning with Support Vector Machines. In *Proceedings of the fifth International Conference on Knowledge Discovery and Data Mining*, pages 317–321. ACM Press, 1999.
3. E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.
4. Asa Ben-Hur, David Horn, Hava T. Siegelmann, and Vladimir Vapnik. Support Vector Clustering. *Journal of Machine Learning Research*, 2:125–137, 2001.
5. Robert Burbidge and Bernhard F. Buxton. An introduction to support vector machines for data mining. In M. Sheppee, editor, *Keynote Papers, Young OR12*, pages 3–15, University of Nottingham, March 2001. Operational Research Society, Operational Research Society.

6. Gert Cauwenberghs and Tomaso Poggio. Incremental and Decremental Support Vector Machine Learning. In *Advances in Neural Information Processing Systems (NIPS'2000)*, volume 13, pages 409–415. MIT Press, 2001.
7. Nello Christiani and John Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*, chapter 6, pages 93–111. Cambridge University Press, 1st edition, 2000.
8. Glenn Fung and Olvi L. Mangasarian. Multicategory Proximal Support Vector Classifiers. *Submitted to Machine Learning Journal*, 2001.
9. Glenn Fung and Olvi L. Mangasarian. Proximal support vector machine classifiers. In *Proceedings of the 7th ACM Conference on Knowledge Discovery and Data Mining*, pages 77–86. ACM, 2001.
10. Glenn Fung and Olvi L. Mangasarian. Incremental Support Vector Machine Classification. In R. Grossman, H. Mannila, and R. Motwani, editors, *Proceedings of the Second SIAM International Conference on Data Mining*, pages 247–260. SIAM, April 2002.
11. Chih-Wei Hsu and Chih-Jen Lin. A Comparison of Methods for Multi-class Support Vector Machines. *IEEE Transactions on Neural Networks*, 13(2):415–425, 2002.
12. Jeffrey Huang, Xuhui Shao, and Harry Wechsler. Face pose discrimination using support vector machines (svm). In *Proceedings of 14th Int'l Conf. on Pattern Recognition (ICPR'98)*, pages 154–156. IEEE, 1998.
13. Ralf Klinkenberg and Thorsten Joachims. Detecting Concept Drift with Support Vector Machines. In Pat Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*. Morgan Kaufmann, 2000.
14. Kenneth Lange. *Numerical Analysis for Statisticians*, chapter 7.3, pages 80–81. Springer-Verlag, 1999.
15. Simplified wrapper and interface generator. Online, http://www.swig.org/, March 2003.
16. Amund Tveit and Magnus Lie Hetland. Multicategory Incremental Proximal Support Vector Classifiers. In *Proceedings of the 7th International Conference on Knowledge-Based Information & Engineering Systems (forthcoming)*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 2003.
17. Amund Tveit and Gisle B. Tveit. Game Usage Mining: Information Gathering for Knowledge Discovery in Massive Multiplayer Games. In *Proceedings of the International Conference on Internet Computing (IC'2002), session on Web Mining.* CSREA Press, June 2002.
18. Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*, chapter 5, pages 138–146. Springer-Verlag, 2nd edition, 1999.
19. Richard C. Whaley, Antoine Petitet, and Jack J. Dongarra. Automated Empirical Optimization of Software and the ATLAS Project". *Parallel Computing*, 27(1-2):3–25, 2001.

Paper H

```
@TechReport{2003:TR:TveitEngum,
  author      = {Amund Tveit and H{\a}vard Engum},
  title       = "{Parallelization of the Incremental
                  Proximal Support Vector Machine Classifier
                  using a Heap-based Tree Topology}",
  institution = {Divison of Intelligent Systems,
                 Department of Computer and Information Science,
                 Norwegian University of Science and Technology},
  year        = {2003},
  month       = {August}
}
```

Note: This paper was accepted to ECML/PKDD 2003 Workshop on Parallel and Distributed Machine Learning in Croatia, but it was withdrawn due to travel funding constraints.

This paper has been cited by Liu et al. [2003]; Gibbs [2003]

Parallelization of the Incremental Proximal Support Vector Machine Classifier using a Heap-based Tree Topology

Amund Tveit and Håvard Engum

Department of Computer and Information Science,
Norwegian University of Science and Technology,
N-7491 Trondheim, Norway
{amundt,havare}@idi.ntnu.no

Abstract. Support Vector Machines (SVMs) are an efficient data mining approach for classification, clustering and time series analysis. In recent years, a tremendous growth in the amount of data gathered has changed the focus of SVM classifier algorithms from providing accurate results to enabling incremental (and decremental) learning with new data (or unlearning old data) without the need for computationally costly retraining with the old data. In this paper we propose two efficient parallelized algorithms based on heaps of processing nodes for classification with the incremental proximal SVM introduced by Fung and Mangasarian.

1 Introduction

Support Vector Machines (SVMs) is an exceptionally efficient data mining approach for classification, clustering and time series analysis [1–3]. This is primarily due to SVMs highly accurate results that are competitive with other data mining approaches, e.g. artificial neural networks (ANNs) and evolutionary algorithms (EAs). In recent years tremendous growth in the amount of data gathered (e.g. user clickstreams on the web, in e-commerce and in intrusion detection systems), has changed the focus of SVM classifier algorithms to not only provide accurate results, but to also enable online learning, i.e. incremental and decremental learning, in order to handle concept drift of classes [4, 5].

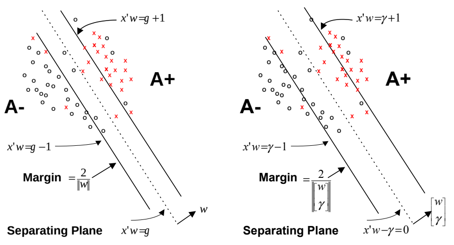
Fung and Mangasarian introduced the Incremental and Decremental Linear Proximal Support Vector Machine (PSVM) for binary classification [6], and showed that it was able to be trained extremely fast, i.e. with 1 billion examples (500 increments of 2 million) in 2 hours and 26 minutes on relatively low-end hardware (400 MHz Pentium II). This has later been extended to support efficient support of incremental multicategorical classification [7] and soft-decay decremental learning [8]. Proximal SVMs has also been shown to perform at a similar level of accuracy as regular SVMs and at the same time being significantly faster [9].

In this paper we propose and compare two parallelization approaches of the Incremental SVM classifier. The algorithms presented are based on heaps of CPUs represented as tree topologies.

2 Background Theory

The basic idea of Support Vector Machine classification is to find an optimal maximal margin separating hyperplane between two classes. Support Vector Machines uses an implicit nonlinear mapping from input-space to a higher dimensional feature-space using kernel-functions, in order to find a hyperplane of problems which are not linear separable in input-space [10, 11]. Classifying multiple classes is commonly performed by combining several binary SVM classifiers in a tournament manner, either one-against-all or one-against-one, the latter approach requiring substantial more computational effort [12].

The standard binary SVM classification problem with soft margin (allowing some errors) is shown visually in Fig. 1(a). Intuitively, the problem is to maximize the margin between the solid planes and at the same time permit as few errors as possible, errors being positive class points on the negative side (of the solid line) or vice versa.



(a) Standard SVM classifier

(b) Proximal SVM classifier

Fig. 1. SVM and PSVM

The standard SVM problem can be stated as a quadratic optimization problem with constraints, as shown in (1).

$$\begin{aligned} \min_{(w,\gamma,y)\in\mathbb{R}^{n+1+m}} \quad & \{ve'y + \tfrac{1}{2}w'w\} \\ \text{s.t. } & D(Aw - e\gamma) + y \geq e \\ & y \geq 0 \end{aligned} \tag{1}$$

$$A \in \mathbb{R}^{m\times n},\ D \in \{-1,+1\}^{m\times 1},\ e = 1^{m\times 1}$$

Fung and Mangasarian [13] replaced the inequality constraint in (1) with an equality constraint. This changed the binary classification problem, because the points in Fig. 1(b) are no longer bounded by the planes, but are clustered around them. By solving the equation for y and inserting the result into the expression to be minimized, one gets the following unconstrained optimization problem:

$$\min_{(w,\gamma)\in\mathbb{R}^{n+1+m}} f(w,\gamma) = \frac{\nu}{2}\|D(Aw - e\gamma) - e\|^2 + \frac{1}{2}(w'w + \gamma^2) \tag{2}$$

Setting $\nabla f = \left(\frac{\partial f}{\partial w}, \frac{\partial f}{\partial \gamma}\right) = \mathbf{0}$ one gets:

$$\underbrace{\begin{pmatrix} w \\ \gamma \end{pmatrix}}_{\mathcal{X}} = \begin{pmatrix} A'A + \frac{I}{\nu} & -A'e \\ -e'A & \frac{1}{\nu} + m \end{pmatrix}^{-1} \begin{pmatrix} A'De \\ -e'De \end{pmatrix} = \underbrace{\left(\frac{I}{\nu} + E'E\right)^{-1}}_{\mathcal{A}^{-1}} \underbrace{E'De}_{\mathcal{B}} \tag{3}$$

$$E = [A - e],\ E \in \mathbb{R}^{m\times(n+1)}$$

Agarwal has showed that the Proximal SVM is directly transferable to a ridge regression expression [14]. Fung and Mangasarian [6] later showed that (3) can be rewritten to handle increments (E^i, d^i) and decrements (E^d, d^d), as shown in (4). This decremental approach is based on time windows.

$$\begin{aligned}\mathcal{X} &= \begin{pmatrix} w \\ \gamma \end{pmatrix} \\ &= \left(\frac{I}{\nu} + E'E + (E^i)'E^i - (E^d)'E^d\right)^{-1} \left(E'd + (E^i)'d^i - (E^d)'d^d\right)\ ,\end{aligned} \tag{4}$$

where $d = De$

.

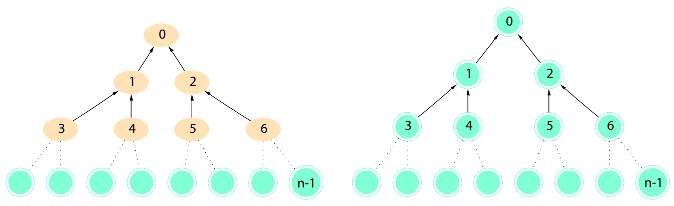
3 Parallelization of the Incremental Proximal SVM

The incremental capabilities of Proximal SVM allows us to efficiently calculate increments $((E^i)'E^i,\ (E^i)'d^i)$ in parallel.

The increments have the dimension numfeatures^2, compared to numfeatures$\cdot$numexamples for E (feature dimension will also be limited due to the curse of dimensionality), in practice is numfeatures^2 $<<$ numfeatures $\cdot$ numexamples. The fact that these increments are relatively small makes them network-wise cheap to send to a parent node for accumulation into $(E'E,\ E'De)$, before finally calculating $\mathcal{X}$ on the top node.

Most algorithms, both sequential and parallel, that involves a tree-based datastructure, perform computationally most efficiently when the tree is being balanced. Another wanted property is the capability of efficient calculation of adresses to child and parent nodes for each nodes in the tree. Heap-numbering of nodes solves both these issues.

In the first approach, as presented in figure 2(a) and algorithm 1, only the *leaf nodes* read increments of examples $((E^i)'E^i, (E^i)'d^i)$, in the second approach shown in figure 2(b) and algorithm 2 *all nodes* read examples.



(a) Read on leaf-nodes only (b) Read on all nodes

Fig. 2. Parallel heap-based tree topologies for Incremental PSVM

3.1 Description of trainParallelReadLeaf Algorithm

The basic flow for algorithm 1 is:

1. data is split (evenly distributed) between *the computational leaf nodes*
2. let the computational leaf nodes read training data from their respective storages.
3. calculate increments $((E^i)'E^i, (E^i)'d^i)$ on the leaf nodes before sending to parent nodes
4. parent nodes wait until they receive increment data from child node(s) (using the MPI receive method in the implementation)
5. parent nodes adds increments and send them to their respective parent nodes, repeated upward the tree
6. the top node finally gets the full incremental training data and is ready for classification
7. the top node (and intermediate nodes) are continously updated from increment data read at the leaf nodes

Algorithm 1 *trainParallelReadLeaf(thisNode, nNodes, incrementSize, nExamples)*

```
1: $topNode \Leftarrow 0$
2: $parentNode \Leftarrow ((thisNode + 1)/2) - 1$
3: $childNode1 \Leftarrow (thisNode + 1) * 2$
4: $childNode2 \Leftarrow (thisNode + 1) * 2 - 1$
5: $nIncrements \Leftarrow nExamples/incrementSize$
6: for $i = 1$ to $nIncrements$ do
7:    if $childNode1 > nNodes - 1$ then
8:       read $incrementSize$ training examples
9:       calculate increment $((E^i)'E^i, (E^i)'d^i)$
10:   end if
11:   if $childNode1 \leq nNodes - 1$ then
12:      (blocking) receive and accumulate increments $((E^i)'E^i, (E^i)'d^i)$ from
         childNode1
13:   end if
14:   if $childNode2 \leq nNodes - 1$ then
15:      (blocking) receive and accumulate increments $((E^i)'E^i, (E^i)'d^i)$ from
         childNode2
16:   end if
17:   if $thisNode \neq topNode$ then
18:      send (accumulated) increment $((E^i)'E^i, (E^i)'d^i)$ to $parentNode$
19:   else
20:      at the topnode, calculate $\mathcal{X}$ from total accumulated $E'E$, $E'd$
21:   end if
22: end for
```

3.2 Description of trainParallelReadAll Algorithm

The basic flow for algorithm 2 is:

1. incoming data is split (evenly distributed) between *all the computational nodes*
2. all computational nodes read training data from their respective storages.
3. leaf nodes then calculates and sends increments to the parent nodes
4. non-leaf nodes calculates increments and wait for increments from their respective child nodes
5. non-leaf nodes adds increments and send them to their respective parent nodes, repeated upward the tree
6. the top node finally gets the full incremental training data (including its own) and is ready for classification
7. the nodes are continously updated from increment data read at all nodes and accumulated upward the tree

4 Empirical Results

Since the two algorithms proposed are exact parallalizations of the incremental proximal SVM, we have chosen to measure speedup (instead of accuracy) on

Algorithm 2 *trainParallelReadAll*($thisNode, nNodes, incrementSize, nExamples$)

```
1: $topNode \Leftarrow 0$
2: $parentNode \Leftarrow ((thisNode + 1)/2) - 1$
3: $childNode1 \Leftarrow (thisNode + 1) * 2$
4: $childNode2 \Leftarrow (thisNode + 1) * 2 - 1$
5: $nIncrements \Leftarrow nExamples/incrementSize$
6: for $i = 1$ to $nIncrements$ do
7:   read $incrementSize$ training examples
8:   calculate increment E'E,E'De
9:   if $childNode1 \leq nNodes - 1$ then
10:    (blocking) receive and accumulate increments $((E^i)'E^i$, $(E^i)'d^i)$ from childNode1
11:  end if
12:  if $childNode2 \leq nNodes - 1$ then
13:    (blocking) receive and accumulate increments $((E^i)'E^i$, $(E^i)'d^i)$ from childNode2
14:  end if
15:  if $thisNode \neq topNode$ then
16:    send (accumulated) increment $((E^i)'E^i$, $(E^i)'d^i)$ to $parentNode$
17:  else
18:    calculate $\mathcal{X}$ from total accumulated $E'E$, $E'd$
19:  end if
20: end for
```

various configurations (number of processing nodes and number of examples in increments). The example dataset is *Forest cover type*, 580012 training examples, 7 classes and 54 features (from UCI KDD Archive [15]). The results are based on average speedups from 10 runs of each configuration.

The parallel incremental proximal SVM has been implemented in C++ using the CLapack and ATLAS libraries for linear system solvers and MPI for communication between CPU nodes [16, 17]. The tests were run on a cluster with Athlon 1.46 GHz / 1 GB RAM nodes running Source Mage GNU/Linux (Linux kernel version 2.4.20).

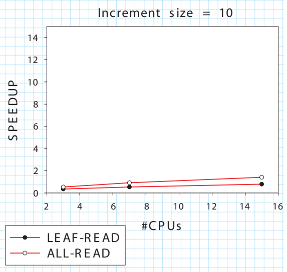
Acknowledgements

We would like to thank Professor Mihhail Matskin. This work is supported by the Norwegian Research Council.

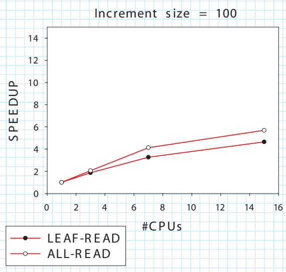
5 Discussion

In figure 3 and 4 the computational speedup of algorithm 1 and 2 compared to the sequential algorithm running on one processing node.

For a small increment size - 10 training examples per increment - the parallel version performs poorly, actually slower than the sequential version, the latter

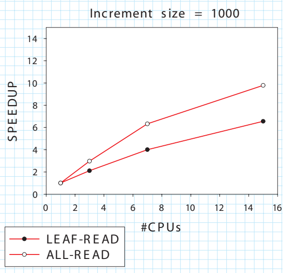


(a) increment size = 10

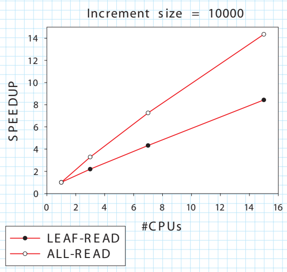


(b) increment size = 100

Fig. 3.



(a) increment size = 1000



(b) increment size = 10000

Fig. 4.

having the reference speedup value of 1. This is mainly caused by network IO (very high frequency of sending and receiving) and by disk IO (reading small increments, less than a disk block, is as time consuming as reading a full block).

Increasing the increment size to 100 improves the performance, and still there is still only little difference between algorithm 1 and 2, this is caused by that the computation is mainly network IO bound.

For an increment size of 1000 and 10000 the problem is becoming mainly disk IO bound, so letting *all* nodes read from disk in algorithm 2 significantly improves the performance compared to reading only with leaf-nodes in algorithm 1. With an increment size of 10000 algorithm 2 is close to being linearly scalable.

6 Conclusion and Future Work

We have introduced two algorithmic approaches for parallelizing the incremental proximal SVM and empirically shown that 1) they both have increasing speedup properties with increasing increment size, this is due to heavier processing per node and less network IO between nodes, and 2) reading on all nodes in the tree performs better than only reading at the leaf nodes.

Future work includes applying the approach on incremental classification and prediction problems. e.g. game usage mining [18]. Algorithmic improvements that needs to be done include 1) develop support for parallelized *decremental* PSVM, 2) add kernel support, 3) add incremental balancing mechanisms to improve accuracy for cases with many, potentially unbalanced, classes.

References

1. Burbidge, R., Buxton, B.F.: An introduction to support vector machines for data mining. In Sheppee, M., ed.: Keynote Papers, Young OR12, University of Nottingham, Operational Research Society, Operational Research Society (2001) 3–15
2. Huang, J., Shao, X., Wechsler, H.: Face pose discrimination using support vector machines (svm). In: Proceedings of 14th Int'l Conf. on Pattern Recognition (ICPR'98), IEEE (1998) 154–156
3. Ben-Hur, A., Horn, D., Siegelmann, H.T., Vapnik, V.: Support Vector Clustering. Journal of Machine Learning Research **2** (2001) 125–137
4. Ahmed, N., Liu, H., Sung, K.K.: Handling Concept Drifts in Incremental Learning with Support Vector Machines. In: Proceedings of the fifth International Conference on Knowledge Discovery and Data Mining, ACM Press (1999) 317–321
5. Klinkenberg, R., Joachims, T.: Detecting Concept Drift with Support Vector Machines. In Langley, P., ed.: Proceedings of the Seventeenth International Conference on Machine Learning (ICML), Morgan Kaufmann (2000)
6. Fung, G., Mangasarian, O.L.: Incremental Support Vector Machine Classification. In Grossman, R., Mannila, H., Motwani, R., eds.: Proceedings of the Second SIAM International Conference on Data Mining, SIAM (2002) 247–260
7. Tveit, A., Hetland, M.L.: Multicategory Incremental Proximal Support Vector Classifiers. In: Proceedings of the 7th International Conference on Knowledge-Based Information & Engineering Systems (KES'03, forthcoming). Lecture Notes in Artificial Intelligence, Springer-Verlag (2003)

8. Tveit, A., Hetland, M.L., Engum, H.: Incremental and Decremental Proximal Support Vector Classification using Decay Coefficients. In: Proceedings of the 5th International Conference on Data Warehousing and Knowledge Discovery (DaWaK'03, forthcoming). Lecture Notes in Artificial Intelligence, Springer-Verlag (2003)
9. Fung, G., Mangasarian, O.L.: Proximal support vector machine classifiers. In: Proceedings of the 7th ACM Conference on Knowledge Discovery and Data Mining, ACM (2001) 77–86
10. Christiani, N., Shawe-Taylor, J.: 6. In: An Introduction to Support Vector Machines and other kernel-based learning methods. 1st edn. Cambridge University Press (2000) 93–111
11. Vapnik, V.N.: 5. In: The Nature of Statistical Learning Theory. 2nd edn. Springer-Verlag (1999) 138–146
12. Hsu, C.W., Lin, C.J.: A Comparison of Methods for Multi-class Support Vector Machines. IEEE Transactions on Neural Networks **13** (2002) 415–425
13. Fung, G., Mangasarian, O.L.: Multicategory Proximal Support Vector Classifiers. Submitted to Machine Learning Journal (2001)
14. Agarwal, D.K.: Shrinkage Estimator Generalizations of Proximal Support Vector Machines. In: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM Press (2002) 173–182
15. Hettich, S., Bay, S.D.: The UCI KDD archive. `http://kdd.ics.uci.edu` (1999)
16. Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., Sorensen, D.: LAPACK Users' Guide. Third edn. Society for Industrial and Applied Mathematics, Philadelphia, PA (1999)
17. Whaley, R.C., Petitet, A., Dongarra, J.J.: Automated Empirical Optimization of Software and the ATLAS Project". Parallel Computing **27** (2001) 3–25
18. Tveit, A., Tveit, G.B.: Game Usage Mining: Information Gathering for Knowledge Discovery in Massive Multiplayer Games. In: Proceedings of the International Conference on Internet Computing (IC'2002), session on Web Mining, CSREA Press (2002)

Paper I

```
@TechReport{2003:TR:TveitB,
  author      = {Amund Tveit},
  title       = "{Empirical Comparison of Accuracy and Performance for
                  the MIPSVM Classifier with
                  Existing Classifiers}",
  institution = {Divison of Intelligent Systems,
                 Department of Computer and Information Science,
                 Norwegian University of Science and Technology},
  year        = {2003},
  month       = {November}
}
```

Empirical Comparison of Accuracy and Performance for the MIPSVM classifier with Existing Classifiers

Amund Tveit

Division of Intelligent Systems
Department of Computer and Information Science,
Norwegian University of Science and Technology,
N-7491 Trondheim, Norway
amundt@idi.ntnu.no

Abstract. This paper presents an empirical comparison of the Multicategory Incremental Proximal Support Vector Machine Classifier (MIPSVM) against the C4.5, Naive Bayes, Voted Perceptron, SMO, SVM and Logistic Regression classifiers on several datasets. The datasets are from the UCI Machine Learning repository, a Web Usage Log, and game usage logs from the Zereal Massively Multiplayer Online Game Simulator.

MIPSVM is found to be close to 1 order of magnitude (or more) faster than the other classifiers in all experiments. Based on pairwise T-test comparisons of accuracy between MIPSVM and the other classifiers, MIPSVM was found to be significantly more accurate than the Naive Bayes classifier, and not having significantly different accuracy than the other classifiers on the tested data sets.

1 Introduction

The objective of this paper is to test both computational performance and classification accuracy of Multicategory Incremental Proximal Support Vector Machine classifier (MIPSVM) [8] against other classifiers on several datasets.

All experiments are performed using the average accuracy from 10-fold cross-validation method, i.e. training on 9/10s of the data and test on the remaining 1/10 for all 1/10s in the dataset (i.e. 10 tests per dataset).For classifation accuracy comparison, graphs of average accuracy percentage from 10-fold cross validation are used, similar for time comparison

Computational performance is measured in average cross-validation wallclock time shown on a logarithmic scale relative to MIPSVM. MIPSVM has a value of 1 (hence not visible!), so if another classifier algorithm has a value of 10 it means it is one order of magnitude (ten times) slower than MIPSVM.

Finally we analyze the results using pairwise T-tests to get a general indication of MIPSVMs accuracy relative to the other classifiers compared with.

2 Classification Datasets Overview

Classification Experiments have been performed on 3 types of data:

1. Three datasets from the UCI machine learning repository [1], experiments on these datasets were performed by former MSc student Håvard Engum (co-supervised by the author) [4].
2. Zereal game simulation output has been used as datasets for the section on "Game Player Classification".
3. Web usage logs from www.jfipa.org has been used as datasets for the section on "Web Intelligence".

2.1 Classification Tools Overview

The classifiers that is compared to MIPSVM implementation in IncRidge are classifiers in the Weka toolkit [10] and the C-SVM algorithm in the LIBSVM toolkit. These toolkits are described briefly below. (Other classifier tools were considered, but not selected since they didn't support 10-fold cross-validation testing).

WEKA WEKA is an akrynom for "Waikato Environment for Knowledge Analysis" and is a software tool that consists of a set of machine earning algorithms including classifiers. Weka is implemented in Java and has been succesfully run on all major computer platforms citewitten:mining. The Weka classifiers used are Naive Bayes, C4.5, Logistic Regression, Voted Perceptron and SMO. Naive Bayes is usually the *default* classifier in many domains, it is simple and gives in general good results. Logistic Regression is considered to be the standard classification method in the domain of medical research [7]. C4.5 has been shown to perform well compared to other classifiers, in fact outperforming Linear Discriminant Analysis and Logistical Analysis for the classification of high performance mutual funds [5]. Sequential Minimal Optimization (SMO) is an approximate and fast method to train Support Vector Machine Classifiers [6].

LIBSVM LIBSVM is an integrated software for support vector classification, regression and distribution estimation. It support multicategory classification and different SVM formulations [2].

In order to get optimal results with LIBSVM, there are a few steps that can be done in advance to enhance LIBSVM's performance both in accuracy and computational efficiency [3].

The first step is to scale the features to the range $[-1, 1]$ or $[0, 1]$. This is because you do not want attributes in greater numeric ranges dominate those in smaller numeric ranges. The other advantage of scaling is that you avoid numerical difficulties of large attribute values when calculating the values of kernel functions. LIBSVM features a tool, $svm - scale$m that scales the data.

The second step is to find optimal values for C and γ, where C is the penalty parameter from the original SVM-formulation and γ is the constant in the Radial Basis Function Kernel. LIBSVM features a tool programmed in Python, easy.py, that finds good values for C and γ These two steps described above is time-consuming. Especially step two takes from five to ten minutes on a somewhat fast computer (AMD Athlon 1.66GHz), depending on the data set. In the experiments, LIBSVM was tested both with the two preparing steps above and without any preparing steps.

2.2 Experiments on UCI datasets

This section presents results from classification experiments performed on three different datasets from the UCI Machine Learning Repository:

1. **Waveform Generator database** - Generated by a C-program. Each class in the dataset is generated from a combination of " of 2 "base" waves with added noise. The 5000 examples have 40 numeric attributes and 3 classes.
2. **Image Segmentation Database** - Data drawn randomly from a database of seven outdoor images. The images were hand segmented to create a classification for every pixel. Each instance has 3x3 region and 19 attributes. There are 7 classes in the dataset named brickface, sky, foliage, cement, window, path and grass. Number of examples are 2310.
3. **Letter Recognition Database** - Data with 20,000 examples of black-and-white rectangular pixels displays of the capital letters in the English alphabet, hence the number of classes are 26, one for each letter. Each example has 16 numerical features.

It was also planned to use the Forest Covertype data in the experiment, but it was too memory intensive for Weka to handle.

Classification Accuracy - UCI Datasets As we can see from the results in figure 1, MIPSVM performs comparably well when it comes to classification accuracy for the Waveform and Image Segment datasets. For the Letter Recognition dataset it performs considerably worse than the other classifiers. This is likely to be caused by that MIPSVM doesn't have any balancing mechanisms one-against-the-rest classifiers may gain from having when there are many classes. Another reason could be that the letter dataset might be not be linearly separable.

Computational Performance - UCI Datasets The computational efficiency is measured in seconds of running time. The experiment was run on a AMD Dual Athlon MP 2100+ (1.66GHz) with 2GB ram. Each configuration is run 10 times, and the average running time is used as a result. The figure shows the *runtime* relative to MIPSVM.

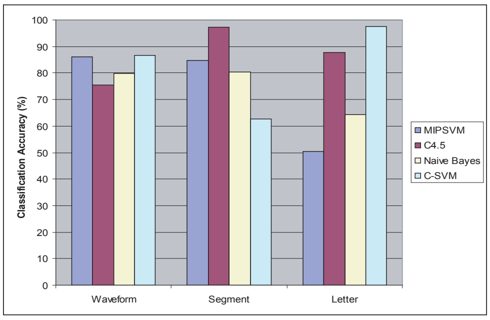


Fig. 1. Accuracy - UCI Datasets

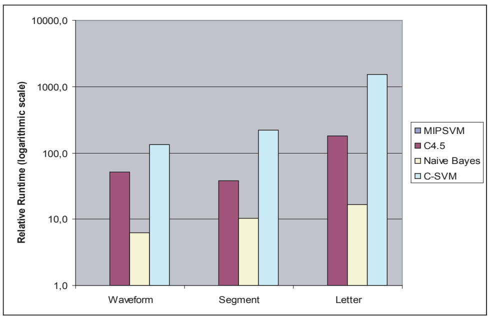


Fig. 2. Performance - UCI Datasets

As shown in figure 2 MIPSVM (IncRidge) *outperforms* the other classifiers with respect to computational efficiency.

2.3 Experiments on Zereal Datasets

This section presents results from classification experiments performed on three different datasets output from the Zereal Massively Multiplayer Online Game Simulator:

1. **zerClassRnd** - classify agents into the 4 classes PlanAgent, MarkovKiller, Killer or the non-personal character category Monster. It has 160 examples.
2. **PlayerMonster** - classify agents into the 2 classes Player or Monster. It has 160 examples
3. **zerealPlayers** - classify agents into the 3 classes PlanAgent, MarkovKiller or Killer. It has 120 examples.

All datasets have 7 numeric features representing frequencies of occurences per agent (i.e. player or monster):

pickupfood - food item frequency (for increased health)
pickupkey - key frequency (for unlocking doors)
pickuppotion - potion frequency (for increased health)
pickupsword - sword frequency (for more powerful combat)
attack - attack frequency (aggressitivity)
leaveworld - frequency for number of times left a subworld
walk - walk frequency (movability)

Classification Accuracy - Zereal Datasets MIPSVM is the best classifier for the *Player Types* dataset, shared 2nd with C4.5 for the *Player vs NPCs* dataset, and 4th for the *Players and NPCs* dataset (figure 3)

Computational Performance - Zereal Datasets Comparison of performance on Zereal datasets were only performed on the *Players and NPCs* dataset since the other datasets were too small to take measurable time with the MIPSVM classifier.

As shown in figure 4 MIPSVM is the fastest classifier, being approximately 2.5 orders of magnitude faster than the SMO classifier and approximately 1.5 orders of magnitude faster than the rest of the classifiers.

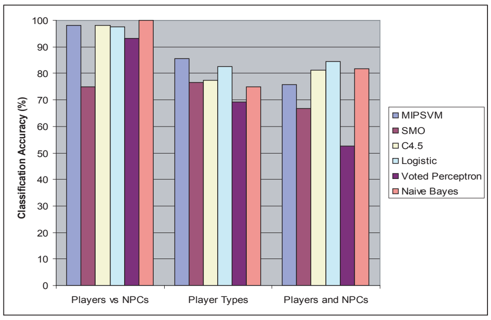


Fig. 3. Accuracy - Player and NPC Types

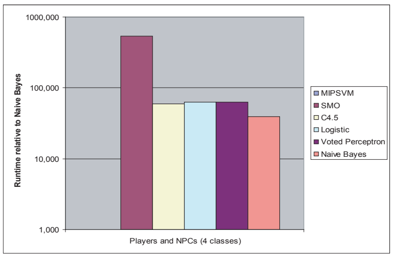


Fig. 4. Performance - Player and NPC Types

2.4 Experiments on Web Usage Logs Datasets

This section presents results from classification experiments performed on four different datasets based on a web usage log from extracted from www.jfipa.org. This web usage log is then preprocessed in order to create 4 types of data for various classification problems:

1. **Intrapage clickstream prediction** - The 8 datasets - P0 to P7 - are for prediction of the next selected page based on clickstreams. There is one classifier per current page, i.e. the previous pages in a user session and the current page is used to predict the next page. Each dataset has 4 numeric attributes representing the pages in the clickstream, with the class being the 5th and last page going to be predicted.
   P0 - 1728 examples and 26 classes
   P1 - 184 examples and 10 classes
   P2 - 166 examples and 3 classes
   P3 - 147 examples and 24 classes
   P4 - 179 examples and 27 classes
   P5 - 270 examples and 24 classes
   P6 - 101 examples and 22 classes
   P7 - 191 examples and 19 classes
2. **Intrasite clickstream prediction** - One dataset for prediction of the next selected page based on clickstreams. This dataset uses one classifier, i.e. does not use the graph structure of links and pages as the *Intrapage clickstream prediction* datasets, but have the same attribute structure. It has 4957 examples and 191 classes.
3. **Search Engine Result Clickstreams** - Two datasets used to answer the questions: 1) *Do people arriving from search engine results differ from those who don't?* and 2) *Do people arriving from results from the Google search engine differ from those from other search engines?* Both datasets have 4957 examples and 2 classes, but different classes for each dataset. These datasets also has the same attribute structure as the previous ones.
4. **Spider and Person Clickstreams** - One dataset used to answer the question: *Are web crawler (agent) clickstreams different from those of users?* This dataset has 9914 examples and 2 classes. This dataset has the same attribute structure as the previous ones.

Classification Accuracy - Web Datasets As observed from figure 5 MIPSVM is among the 2 best classifiers for 4 of the 8 datasets (P0,P1,P2 and P4) and only the worst for one dataset (P6) for the prediction of intrapage clickstreams.

Due to OutOfMemory exceptions Logistic Regression and Voted Perceptron failed to be applied to the Intrasite prediction problem. MIPSVM was a lot less accurate than C4.5, but more accurate than Naive Bayes (figure 6).

For the search engine result clickstream datasets all classifiers except Naive Bayes have very similar performance, but with Logistic Regression and MIPSVM

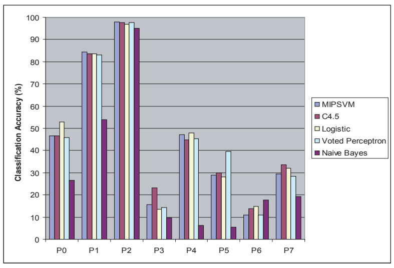


Fig. 5. Accuracy - intrapage clickstreams

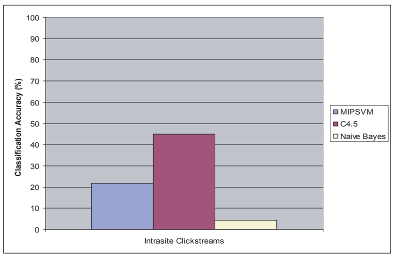


Fig. 6. Accuracy - intrasite clickstreams

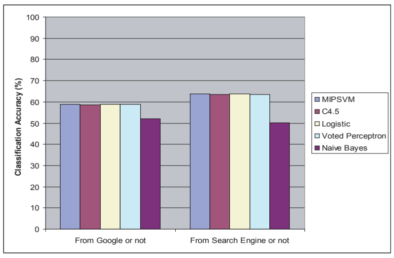


Fig. 7. Accuracy - referring URL type

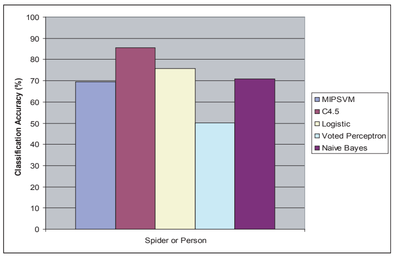


Fig. 8. Accuracy - person vs agent clickstreams

being slightly more accurate than the other classifiers on the *From Search Engine or Not* dataset (figure 7).

C4.5 is the most accurate classifier on the *Spider or Person* dataset with Naive Bayes being the third and slightly more accurate than MIPSVM (figure 8).

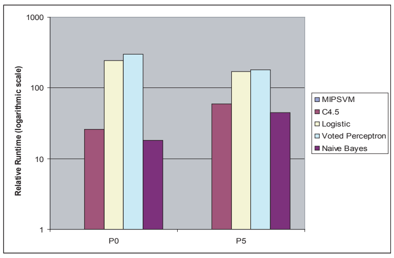


Fig. 9. Performance - intrapage clickstreams

Computational Performance - Web Datasets Comparison of performance for *Intrapage clickstream prediction* were only done on the P0 and P5 datasets since the other datasets were too small to take measurable time with the MIPSVM classifier. MIPSVM is between 1 and 2 orders of magnitude faster than C4.5 and Naive Bayes, and between 2 and 3 orders of magnitudes faster than Logistic Regression and the Voted Perceptron algorithms (figure 9).

MIPSVM is about 1 order of magnitude faster than Naive Bayes and 1.5 orders of magnitude faster than C4.5 on the *Intrasite clickstream prediction* dataset (figure 10)

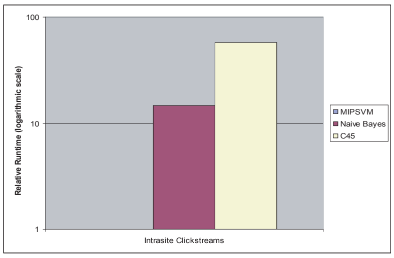


Fig. 10. Performance - intrasite clickstreams

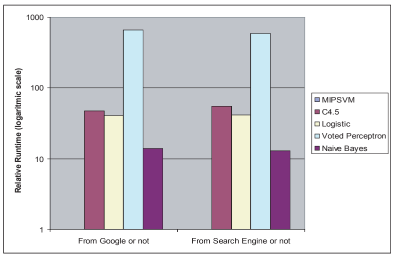


Fig. 11. Performance - referring URL type

On the *Search Engine Result Clickstreams* datasets MIPSVM is between 1 and 1.5 orders of magnitude faster than C4.5, Naive Bayes and Logistic Regression. Compared to Voted Perceptron it is about 2.5 orders of magnitude faster (figure 11).

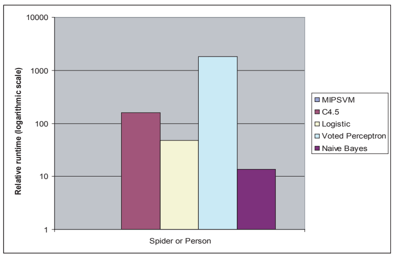


Fig. 12. Performance - person vs agent clickstreams

On the *Spider and Person Clickstreams* dataset MIPSVM is approximately 1 order of magnitude faster than Naive Bayes, 2 orders of magnitude faster than C4.5, 1.5 orders of magnitude faster than Logistic Regression and 3 orders of magnitudes faster than Voted Perceptron (figure 12.

3 Conclusion

Finally MIPSVM was been compared to other classifiers on three main types of classification data:

1. Classification datasets from the UCI Machine Learning Repository
2. Classicication datasets from the Zereal Massively Multiplayer Online Game Simulator
3. Classification datasets from a Web Usage Log extracted from www.jfipa.org

3.1 Computational Performance

MIPSVM is faster than the other classifiers in all experiments. This can be caused by several reasons: the algorithms, implementation overhead, programming language and optimizations in compiler and virtual machines, external libraries used (the ATLAS linear algebra library is used in the qimplementation of MIPSVM [9]).

3.2 Classification Accuracy

In order to make some more general conclusions on MIPSVM's accuracy compared to the other classifiers, pairwise T-tests have been used
These tests concluded that on the 18 datasets
the default configurations of MIPSVM, Naive Bayes and C4.5 were used (appendix C). MIPSVM is *significantly more accurate* than Naive Bayes (5% confidence level, p-value = 0.013), and *not significantly different* from C4.5 (even though C4.5 is more accurate). On the 14 datasets the default configurations of MIPSVM, Logistic Regression and Voted Perceptron were used (appendix C) pairwise T-tests showed that MIPSVM and Logistic Regression have *not significantly different accuracy* (even though Logistic Regression is more accurate), and that MIPSVM and Voted Perceptron have *not significantly different accuracy* (even though MIPSVM is more accurate).

A modestly bold conclusion and recommendation is that MIPSVM is a suitable alternative to Naive Bayes as the *default classifier* when first attacking a classification problem.

3.3 Further Work

Opportunities for further work include:

- Add support for a variable number of features in examples
- Develop support for parallelized *decremental* PSVM
- Add kernel support
- Add incremental balancing mechanisms to improve accuracy for cases with many, potentially unbalanced classes.
- Investigate the performance effect of altering matrix multiplication algorithms
- Investigate whether the symmetric matrix system ($\mathcal{A}$) can be transformed into a Toeplitz or Hankel matrix system for more efficient computation

References

1. C. L. Blake and C. J. Merz. UCI Repository of Machine Learning Databases. Online, `http://www.ics.uci.edu/~mlearn/MLRepository.html`, 1998.
2. Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: A Library for Support Vector Machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.
3. Chih-Chung Chang Chih-Wei Hsu and Chih-Jen Lin. *A practical guide to SVM classification.* Department of Computer Science and Information Technology, National Taiwan University. Paper available at `http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf`.
4. Håvard Engum. IncRidge: A Software Tool for Scalable Incremental and Decremental Classifier Algorithms based on Support Vector Approximations. Technical report, IDI, NTNU, July 2003.

5. Robert C. Norris. Classifying High Performance Mutual Funds: A Comparison C4.5, LDA and Logit. In *Proceedings of the INFORMS Conference on Information Systems and Technology, Session on AI in Business and Industry: Applications and Theory*, May 1996.
6. J. Platt. Sequential Minimal Optimization. Technical Report MSR-TR-98-14, Microsoft Research, 1998.
7. Sloan Rush. Logistic Regression: The Standard Method of Analysis in Medical Research. Technical Report Mathematics #S3, Trinity University, 2001.
8. Amund Tveit and Magnus Lie Hetland. Multicategory Incremental Proximal Support Vector Classifiers. In *Proceedings of the 7th International Conference on Knowledge-Based Information & Engineering Systems (KES'2003)*, number 2773 in Lecture Notes in Artificial Intelligence (LNAI), pages 386–392. Springer-Verlag, 2003.
9. Richard C. Whaley, Antoine Petitet, and Jack J. Dongarra. Automated Empirical Optimization of Software and the ATLAS Project". *Parallel Computing*, 27(1-2):3–25, 2001.
10. Ian H. Witten and Eibe Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1st edition, October 1999.

Part IV

Appendices

Appendix A

Software Tools

This appendix will present software tools developed as a part of this thesis. The languages used were: Java, C++, Python, XML, Perl and SWIG.

A.1 Zereal

Zereal is a simulator of players and non-personal characters in Massively Multiplayer Online Games Tveit et al. [2003b]. It is implemented using Stackless Python as the main language, pyMPI for messaging support on parallel computers, C++ for performance critical algorithms (e.g. path finding and agent vision), and SWIG for the integration of C++ code with Stackless Python. The architecture and technological choices was proposed by Amund Tveit. The initial version of Zereal was implemented by Håvard Engum, Øyvind Rein and Jørgen Vinne Iversen (as project students co-supervised by Amund Tveit) (Engum et al. [2002]), and the final version was developed by the latter two (as MSc students co-supervised by Amund Tveit) (Iversen and Rein [2003]).

Zereal was developed using approximately:

- 4.800 lines of (Stackless) Python code
- 550 lines of C++ code
- 300 lines of Java code
- 288 lines of XML
- 8 lines of SWIG code

A.2 Incridge and pIncridge

Incridge is a classification tool based on approximations to support vector machines, Tveit and Hetland [2003]; Tveit et al. [2003a]. It is implemented using C++ as the main language, and MPI for messaging support on parallel computers (pIncridge only), Tveit and Engum [2003] . The algorithmic design of Incridge and pIncridge was primarily done by Amund Tveit, including the first implementation (of MIPSVM). Later implementations (of IDPSVM and PIPSVM) were done by Håvard Engum (MSc student co-supervised by Amund Tveit) (Engum [2003].

Incridge and pIncridge was developed using approximately:

- 6.600 lines of C++ code
- 2.100 lines of Python code

A.3 Other software developed

This section describes other software developed that is (partially) related to this thesis.

A.3.1 Browsim

Browsim is a simulator of user patterns in news content web sites, it was developed by Amund Tveit using approximately 2.200 lines of Python code. Browsim was used to simulate browsing patterns of mobile users in order to test predictive retrieval, Tveit and Matskin [2002].

A.3.2 jfipa

jfipa is a set of software tools that supports relatively efficient parsing XML-messages of type FIPA Agent Interaction Protocols, it was developed by Amund Tveit using approximately 17.000 lines of Java code. jfipa was empirically found to outperform established Java-based XML-parsers such as Crimson and Xerces on parsing of FIPA messages, Tveit [2002b].

Appendix B

Statistical Analysis

This appendix will present statistical analysis of MMOG simulation and classification.

B.1 Factorial Design for Zereal Performance

This section shows the Minitab analysis output based on the factorial experimental design and analysis of Zereal's performance.

B.1.1 Factorial Design Experimental Data

REAL TIME is the measured response variable.

```
cpus markovkillers planagents monsters radius REAL TIME
1    2             2          2        1      105.332
1    1             1          2        1      43.813
1    2             2          2        2      309.983
2    2             2          2        1      107.871
1    1             2          2        2      208.884
2    1             2          1        2      190.308
1    2             1          1        2      148.754
1    2             1          2        2      187.185
1    1             2          2        1      74.236
1    1             2          1        2      172.335
2    2             1          2        1      70.522
1    2             2          1        2      261.521
```

1	1	1	1	1	35.726
1	1	1	2	2	106.313
1	1	1	1	2	85.47
2	1	1	2	2	175.159
2	1	1	1	1	38.723
2	2	1	1	1	61.683
1	2	2	1	1	91.271
2	2	1	2	2	201.142
1	1	2	1	1	67.877
2	2	2	2	2	330.603
2	2	2	1	1	97.097
2	2	2	1	2	295.952
2	1	1	2	1	71.674
2	1	2	1	1	108.537
2	1	2	2	2	221.136
2	2	1	1	2	158.914
1	2	1	1	1	57.781
1	2	1	2	1	68.737
2	1	2	2	1	79.078
2	1	1	1	2	90.288

B.1.2 Analysis of Variance (ANOVA)

```
General Factorial Design

Factors:   5     Factor Levels: 2; 2; 2; 2; 2
Runs:     32     Replicates:    1

General Linear Model: REAL TIME versus cpus; markovkillers; ...


Factor     Type Levels Values
cpus      fixed      2 1 2
markovki  fixed      2 1 2
planagen  fixed      2 1 2
monsters  fixed      2 1 2
visionra  fixed      2 1 2

Analysis of Variance for REAL TIM, using Adjusted SS for Tests
```

```
Source              DF Seq SS  Adj SS  Adj MS  F      P
cpus                 1 2337    2337    2337    11.49  0.004
markovki             1 19247   19247   19247   94.61  0.000
planagen             1 39210   39210   39210   192.75 0.000
monsters             1 4986    4986    4986    24.51  0.000
visionra             1 120539  120539  120539  592.54 0.000
cpus*markovki        1 237     237     237     1.16   0.297
cpus*planagen        1 1       1       1       0.00   0.953
cpus*monsters        1 32      32      32      0.16   0.697
cpus*visionra        1 268     268     268     1.32   0.268
markovki*planagen    1 900     900     900     4.42   0.052
markovki*monsters    1 9       9       9       0.05   0.832
markovki*visionra    1 7923    7923    7923    38.95  0.000
planagen*monsters    1 282     282     282     1.39   0.256
planagen*visionra    1 9621    9621    9621    47.29  0.000
monsters*visionra    1 2351    2351    2351    11.56  0.004
Error               16 3255    3255    203
Total               31 211197


Unusual Observations for REAL TIM

Obs  REAL TIM      Fit      SE Fit  Residual   St Resid
 16   175.159  147.683      10.085    27.476      2.72R
 26   108.537   84.444      10.085    24.093      2.39R

R denotes an observation with a large standardized residual.

Macro is running ... please wait

Main Effects Plot for REAL TIME

Interaction Plot for REAL TIME
```

B.2 Classification Accuracy - UCI Datasets

	Letter	Segment	Waveform
Weka NBayes	64.23	80.30	79.96
Weka J48	87.76	97.14	75.52
LIBSVM C-SVM	97.45	62.60	86.66
MIPSVM	50.42	84.72	85.98

B.3 Computational Performance - UCI Datasets

TODO: add text here..

	Letter	Segment	Waveform
MIPSVM	1.378	0.1700	1.113
Weka NBayes	22.70	1.742	6.939
Weka J48	249.85	6.490	57.21
LIBSVM C-SVM	2081	37.74	148.5

B.4 Data for Computational Performance - Zereal and Web data

Dataset/Classifier	MIPSVM	SMO	C4.5	LogReg	VotPer	Naiv.Bayes
Players and NPCs	0,01	5,37	0,59	0,63	0,63	0,39
Players vs NPCs	0,00	1,33	0,43	0,59	0,40	0,37
Player Types	0,00	2,85	0,53	0,63	0,51	0,37
Spider or person	0,151	NA	23,927	7,218	271,909	2,057
Google vs others	0,075	NA	3,565	3,085	49,42	1,04
SE vs others	0,078	NA	4,289	3,251	46,102	1,005
Intrasite cs.	0,402	NA	23,063	NA	NA	5,907
Intrapage P0	0,044	NA	1,137	10,67	13,243	0,793
Intrapage P5	0,01	NA	0,593	1,7	1,813	0,449

B.5 Paired T-Tests for Classification Accuracy

This section shows the Minitab analysis output based on paired T-tests of classification accuracy of the proposed MIPSVM classifier implemented in the Incridge tool with the C4.5, Logistic Regression, Naive Bayes and Voted Perceptron classifiers implemented in the Weka tool. Paired T-test was chosen since it allows a "global summary" over all the classification experiments performed (uses the difference between classifier results per row)

B.5.1 Classification Data C4.5 and Naive Bayes

Dataset/Classifier	MIPSVM	C4.5	Naive Bayes
P0	46.5116	46.5856	26.678
P1	84.4444	83.6957	53.804
P2	98	97.5904	95.181
P3	15.7143	23.1293	9.524
P4	47.0588	44.6927	6.145
P5	28.8889	30	5.556
P6	11	13.8614	17.822
P7	29.4737	33.5079	19.372
Intrasite Clickstr.	21.9172	45.0676	4.257
From Google or not	58.8485	58.6645	52.088
From SE or not	63.6768	63.5667	50.212
Spider or Person	69.5661	85.6264	70.92
Waveform	85.98	75.52	79.96
Segment	84.72	97.14	80.3
Letter	50.42	87.76	64.23
Players vs NPCs	98.125	98.125	100
Player Types	85.625	77.5	75
Players and NPCs	75.625	81.25	81.875

B.5.2 MIPSVM and C4.5 Comparison

The P-Value based on the T-test is 0.088, this is not significant according to 5% significance level, hence the experiments performed does not show significant difference between MIPSVM and C4.5.

```
Welcome to Minitab, press F1 for help.
n
Paired T-Test and CI: MIPSVM; C4.5
Paired T for MIPSVM - C4.5

                  N      Mean     StDev   SE Mean
MIPSVM           18     58.64     28.60      6.74
C4.5             18     63.52     27.39      6.46
Difference       18     -4.87     11.40      2.69

95% CI for mean difference: (-10.54; 0.80)
T-Test of mean difference = 0 (vs not = 0):

T-Value = -1.81  P-Value = 0.088
```

B.5.3 MIPSVM and Naive Bayes Comparison

The P-Value based on the T-test is 0.013, this is significant according to a 5% significance level, hence the experiments showed that MIPSVM is more accurate than Naive Bayes on the experiment data sets used.

```
Paired T-Test and CI: MIPSVM; Naive Bayes

Paired T for MIPSVM - Naive Bayes

                  N      Mean     StDev   SE Mean
MIPSVM           18     58.64     28.60      6.74
Naive Bayes      18     49.61     33.26      7.84
Difference       18      9.04     13.76      3.24

95% CI for mean difference: (2.20; 15.88)
T-Test of mean difference = 0 (vs not = 0):

T-Value = 2.79  P-Value = 0.013
```

B.5.4 Classification Data Log. Regr. and Vot. Perc.

```
Dataset            MIPSVM  LogReg. Vot. Perc.
P0                 46.5116 52.8241 45.8333
P1                 84.4444 83.6957 83.1522
```

```
P2                  98      96.988  97.5904
P3                  15.7143 13.6054 14.2857
P4                  47.0588 48.0447 45.2514
P5                  28.8889 28.1481 39.6296
P6                  11      14.8515 10.8911
P7                  29.4737 31.9372 28.2723
From Google or not  58.8485 58.9066 58.8662
From SE or not      63.6768 63.6474 63.5263
Spider or Person    69.5661 75.8523 50.2118
Players vs NPCs     98.125  97.5    93.125
Player Types        85.625  82.5    69.1667
Players and NPCs    75.625  84.375  52.5
```

B.5.5 MIPSVM and Logistic Regression Comparison

The P-Value based on the T-test is 0.151, this is not significant according to a 5% significance level, hence the experiments performed does not show significant difference between MIPSVM and Logistic Regression. (Even though Logistic Regression seems to be slightly more accurate than MIPSVM)

```
Paired T-Test and CI: MIPSVM; Logistic

Paired T for MIPSVM - Logistic

                 N      Mean     StDev   SE Mean
MIPSVM          14     58.04     29.17      7.80
Logistic        14     59.49     28.94      7.73
Difference      14    -1.451     3.563     0.952

95% CI for mean difference: (-3.509; 0.606)
T-Test of mean difference = 0 (vs not = 0):

T-Value = -1.52  P-Value = 0.151
```

B.5.6 MIPSVM and Voted Perceptron Comparison

The P-Value based on the T-test is 0.099, this is not significant according to a 5% significance level, hence the experiments performed does not show significant difference between MIPSVM and Voted Perceptron. (Even though MIPSVM seems to be slightly more accurate than the Voted Perceptron)

```
Paired T-Test and CI: MIPSVM; Voted Perceptron

Paired T for MIPSVM - Voted Perceptron

                 N      Mean     Stdev   SE Mean
MIPSVM          14     58.04     29.17      7.80
Voted Percep    14     53.74     26.44      7.07
Difference      14      4.30      9.07      2.43

95% CI for mean difference: (-0.94; 9.54)
T-Test of mean difference = 0 (vs not = 0):

T-Value = 1.77  P-Value = 0.099
```

Bibliography

A. Aamodt and E. Plaza. Case-based reasoning: Foundational Issues, methodological variations, and systems approaches. *AI Communications*, 7(1):39–54, March 1994.

R. Adams. Wireless Peer-to-Peer Internet, 2003.

D. K. Agarwal. Shrinkage Estimator Generalizations of Proximal Support Vector Machines. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 173–182. ACM Press, 2002.

T. Amdeberhan and G. Heinig. On matrices that are not similar to a Toeplitz matrix and a family of polynomials. Submitted to Linear Algebra & its Applications, March 2003.

A. Bartish and C. Thevathayan. BDI Agents for Game Development. In *Proceedings of the 1st International Conference on Autonomous Agents and Multiagent Systems*, pages 668–669. ACM, ACM Press, 2002.

D. Billsus, C. Brunk, C. Evans, B. Gladish, and M. J. Pazzani. Adaptive interfaces for ubiquitous web access. *Communications of the ACM*, 45(5):34–38, 2002.

C. M. Bishop. *Neural Networks for Pattern Recognition*, chapter 9.2, pages 338–340. Oxford University Press, 1995.

C. L. Blake and C. J. Merz. UCI Repository of Machine Learning Databases. Online, http://www.ics.uci.edu/$\sim$mlearn/MLRepository.htm, 1998.

D. Bonino, F. Corno, and G. Squillero. A Real-Time Evoluationary Algorithm for Web Prediction. In J. Liu, C. Liu, M. Klusch, N. Zhong, and N. Cercone, editors, *Proceedings of IEEE/WIC International Conference on Web Intelligence (WI 2003)*, pages 139–145. IEEE, October 2003.

F. Cecin, J. Barbosa, and C. Geyer. FreeMG: An Hybrid Peer-to-Peer, Client-Server, and Distributed Massively Multiplayer Game Simulation Model. In *Proceedings of the 2nd Brazilian Workshop on Games and Digital Entertainment (WJogos'03)*, November 2003.

R. H. Chan and M. K. Ng. Conjugate gradient methods for Toeplitz systems. *SIAM Review*, 38(3):427–482, 1996.

G. Chang. Limitations of Mobile Commerce. Report, Auckland University of Technology, New Zealand, 2002.

N. Christiani and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods*, chapter 6, pages 93–111. Cambridge University Press, 1st edition, 2000.

J. Clark, I. Koprinska, and J. Poon. A Neural Network Based Approach to Automated E-Mail Classification. In J. Liu, C. Liu, M. Klusch, N. Zhong, and N. Cercone, editors, *Proceedings of IEEE/WIC International Conference on Web Intelligence (WI 2003)*, pages 702–705. IEEE CS Press, October 2003.

P. R. Cohen. *Empirical Methods for Artificial Intelligence*, chapter 6.10 - Cross-Validation: An Efficient Training and Testing Procedure, pages 216–218. MIT Press, 1995.

D. Coppersmith and S. Winograd. Matrix Multiplication via Arithmetic Progressions. *Journal of Symbolic Computation*, 9(3):251–280, March 1990.

T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*, chapter 31.5, pages 762–765. MIT Press, 1st edition, 1990.

M. S. Ding and C. R. Unnithan. Mobile Commerce (mCommerce) Security: An appraisal of current issues and trends. Technical Report SWP 2002/34, School of Information Systems, Faculty of Business and Law, Deakin University, Australia, 2002.

M. S. Ding, C. R. Unnithan, and B. Fraunholz. MCommerce - A Vision in Time: A Preliminary Investigation into Past and Future of Mobile Payments. Technical Report SWP 2002/51, School of Information Systems, Faculty of Business and Law, Deakin University, Australia, 2002.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification.* Wiley-Interscience, 2nd edition, 2001.

M. Eirinaki and M. Vazirgiannis. Web Mining for Web Personalization. *ACM Transactions on Internet Technology*, 3(1):1–27, February 2003.

A. C. Elster, O. Anshus, A. Tveit, and C. Banino. Recent trends in cluster computing. In *Proceedings of the International Conference on Parallel Computing (forthcoming)*. Elsevier Science, September 2003.

H. Engum. IncRidge: A Software Tool for Scalable Incremental and Decremental Classifier Algorithms based on Support Vector Approximations. Technical report, IDI, NTNU, July 2003.

H. Engum, J. V. Iversen, and Ø. Rein. Zereal: A Semi-realistic Simulator of Massively Multiplayer Games. Technical report, IDI, NTNU, December 2002.

T. Finin, A. Joshi, L. Kagal, O. Ratsimor, S. Avancha, V. Korolov, H. Chen, F. Perich, and R. S. Cost. Intelligent Agents for Mobile and Embedded Devices. *International Journal of Cooperative Information Systems*, 11(3-4):205–230, 2002.

J. H. Friedman. Multivariate adaptive regression splines (MARS). *The Annals of Statistics*, 19(1):1–141, 1991.

G. Fung and O. L. Mangasarian. Multicategory Proximal Support Vector Classifiers. *Submitted to Machine Learning Journal*, 2001a.

G. Fung and O. L. Mangasarian. Proximal support vector machine classifiers. In *Proceedings of the 7th ACM Conference on Knowledge Discovery and Data Mining*, pages 77–86. ACM, 2001b.

G. Fung and O. L. Mangasarian. Incremental Support Vector Machine Classification. In R. Grossman, H. Mannila, and R. Motwani, editors, *Proceedings of the Second SIAM International Conference on Data Mining*, pages 247–260. SIAM, April 2002.

E. Garfield. The meaning of the Impact Factor. *International Journal of Clinical and Health Psychology*, 3(2):363–369, 2003.

R. Genov and G. Cauwenberghs. Kerneltron Support Vector 'Machine' in Silicon. *IEEE Transactions on Neural Networks*, 14(5):1426–1434, September 2003.

S. Gibbs. Data Parallelism and the Support Vector Machine. Online Proceedings of the Information Processing Systems (IPS) Laboratory Research Forum, Department of Electrical Engineering, Ohio State University, Colombus, Ohio, USA, 2003.

W. Gibson. *Neuromancer*. Ace Books, 1984.

F. Girosi, T. Poggio, and B. Caprile. Extensions of a theory of networks for approximation and learning. In R. P. Lippmann, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems*, volume 3, pages 750–756. Morgan Kaufmann Publishers, Inc., 1991.

J. Goldman and M. Axtell. On Using Logic Synthesis for Supervised Classification Learning. In *Proceedings of the 7th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'95)*, November 1995.

V. Gorodetski, I. Kotenko, and O. Karsaev. Multi-agent technologies for computer network security: Attack simulation, intrusion detection and intrusion detection learning. *International Journal of Computer Systems, Science & Engineering*, 18(4):191–200, July 2003.

R. S. Gray, D. Kotz, S. Nog, D. Rus, and G. Cybenko. Mobile agents for mobile computing. Technical Report TR96-285, Dartmouth College, NH, USA, 1996.

J. Han and M. Kamber. *Data Mining: Concepts and Techniques.* Academic Press and Morgan Kaufmann Publishers, 2001.

S. Haridi. Parallel Agent Based Simulation on PC Cluster. IBM Workshop on Information Cities and Collaborative Portals: The Gateways to Next Generation Commerce and City Life Over The Internet. Published Online, http://www.research.ibm.com/compsci/hci/workshop/SeifHaridi-simulation.pdf, October 2002.

A. Heinemann, J. Kangasharju, F. Lyardet, and M. Muhlhauser. Ad Hoc Collaboration and Information Services using Information Clouds. In *Proceedings to the 3rd Workshop on Applications and Services in Wireless Networks (ASWN 2003)*, 2003a.

A. Heinemann, J. Kangasharju, F. Lyardet, and M. Muhlhauser. iClouds - Peer-to-Peer Information Sharing in Mobile Environments. Technical Report TK-01/03, Telecooperation Group, Department of Computer Science, Darmstadt University of Technology, Darmstadt, Germany, 2003b.

D. Hiebeler. The Swarm simulation system and individual-based modeling, 1994.

J.-Y. Ho, Y. Matsumoto, and R. Thawonmas. MMOG Player Identification: A Step Toward CRM of MMOGs. In *Proceedings of the 6th Pacific Rim Conference on Multi-Agents (PRIMA-2003)*, Seoul, Korea, November 2003.

J.-Y. Ho and R. Thawonmas. Episode Detection with Vector Space model in Agent Behavior Sequences of MMOGs. In *Proceedings the Future Business Technology Conference (FUBUTEC'2004)*, March 2004.

A. E. Hoerl and R. W. Kennard. Ridge Regression: Biased Estimation for Nonortogonal Problems. *Technometrics*, 12(1):55–67, 1970.

P. T. Hraber and B. T. Milne. Community assembly in a model ecosystem. Technical Report 96-12-094, Santa Fe Institute, December 1996.

M. J. Huber and T. Hadley. Multiple roles, multiple teams, dynamic environment¿ Autonomous Netrek Agents. In *Proceedings of the 1st International Conference on Autonomous Agents*, pages 332–339. ACM, ACM Press, 1997.

V. Illingworth, editor. *Oxford Dictionary of Computing.* Oxford University Press, 4th edition, 1996.

J. V. Iversen and Ø. Rein. Agent-based Simulation of Massively Multiplayer Online Games. Technical report, IDI, NTNU, June 2003.

S. Jang and E. Lee. An Intelligent Mobile Commerce System with Dynamic Contents Builder and Mobile Products Browser. In *Proceedings of the 3rd International Conference on Intelligent Data Engineering and Automated Learning (IDEAL 2002)*, volume Lecture Notes in Computer Science, pages 179–185. Springer-Verlag, August 2002.

A. Joshi. On Proxy Agents, Mobility and Web Access. *Mobile Networks and Applications, Special Issue on Software Architectures for Mobile Applications*, pages 233–241, 2000a.

A. Joshi. On proxy agents, mobility, and web access. *Mobile Networks and Applications*, 5(4):233–241, 2000b.

A. Joshi, T. Finin, and Y. Yesha. Agents, Mobility and M-Services: Creating the Next Generation Applications and Infrastructure on Mobile Ad-Hoc Networks. volume 2538 of *Lecture Notes in Computer Science*, pages 106–118. Springer-Verlag, 2002.

A. Kieldaas. Personalized Search Engines. Technical report, IDI, NTNU, January 2000.

A. Kobayashi and H. Fujioka. Personalizing a Web Site for Cellular Phones. In J. Liu, C. Liu, M. Klusch, N. Zhong, and N. Cercone, editors, *Proceedings of IEEE/WIC International Conference on Web Intelligence (WI 2003)*, pages 432–435. IEEE, October 2003.

J. R. Koza. Genetic programming. In J. G. Williams and A. Kent, editors, *Encyclopedia of Computer Science and Technology*, volume 39, pages 29–43. Marcel-Dekker, 1998.

J. Laird. It Knows What You're going To Do: Adding Anticipation to a Quakebot. In *Proceedings of the 5th International Conference on Autonomous Agents.* ACM, ACM Press, 2001.

J. Laird and M. V. Kent. Human-Level AI's Killer Application: Interactive Computer Games. *AI Magazine*, 22(2):15–26, 2001.

K. Lange. *Numerical Analysis for Statisticians*, chapter 7.3, pages 80–81. Springer-Verlag, 1999.

S. Lawrence. Online or Invisible? *Nature*, 411(6837):521, 2001.

K. J. Lee. Peer-to-Peer Electronic Commerce and Intelligent Systems. Presentation at Department of Industrial Engineering, Korea Advanced Institute of Science and Technology, Published Online: http://space.postech.ac.kr/vod/s011204/ppt1204/ppt1204.pdf, 2002.

G. Linden, B. Smith, and J. York. Amazon.com Recommendations: Item-to-Item Collaborative Filtering. *IEEE Internet Computing*, 7(1):76–80, January/February 2003.

K. Liu, J. Ryan, and H. Kargupta. Distributed data mining bibliography. Published Online: http://www.csee.umbc.edu/~hillol/DDMBIB/, August 2003.

Z. Maamar, Q. Z. Sheng, and B. Benatallah. On Composite Web Services Provisioning in an Environment of Fixed and Mobile Computing Resources. *Information Technology and Management*, 5(3), 2004.

D. S. Mackey, N. Mackey, and S. Petrovic. Is Every Matrix Similar to a Toeplitz Matrix. *Linear Algebra & its Applications*, 297:87–105, 1999.

P. Maes. Agents that Reduce Work and Information Overload. *Communications of the ACM*, 37(7):31–40, 1994.

C. Mangione. Performance Tests show Java as fast as C++. *Java World*, February 1998.

M. Matskin and A. Tveit. Mobile Commerce Agents in WAP-based Services. *Journal of Database Management - Special Issue on Mobile Commerce*, pages 27–35, July-September 2001.

M. Matskin and A. Tveit. Software Agents for Mobile Commerce Services Support. In K. Siau, editor, *Advanced Topics in Database Research*, volume 2, chapter 11, pages 246–266. Idea Group Inc, 2003.

E. Mena, J. A. Royo, A. Illarramendi, and A. Goni. An Agent-based Approach for Helping Users of Hand-Held Devices to Browse Software Catalogs. In M. Klusch, S. Ossowski, and O. Shehory, editors, *Proceedings of the 6th International Workshop on Cooperative Information Agents (CIA)*, volume 2446 of *Lecture Notes in Computer Science*, pages 51–65. Springer-Verlag, 2002.

G. A. Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, November 1995.

T. Mitchell. *Machine Learning.* Computer Science Series. McGraw-Hill, 1st edition, 1997.

D. Mladenic. Text-Learning and Related Intelligent Agents: A Survey. *IEEE Intelligent Systems*, 14(4):44–54, 1999.

D. C. Montgomery. *Design and Analysis of Experiments*, chapter 10, pages 461–466. John Wiley & Sons, Inc., 4th edition, 1997.

S. Moss. Messy Systems - The Target for Multi Agent Based Simulation. In S. Moss and P. Davidsson, editors, *Proceedings of the Second International Workshop on Multi-Agent-Based Simulation*, number 1979 in Lecture Notes in Artificial Intelligence, pages 1–14. Springer-Verlag, 2000.

B. Moulin and B. C. Draa. An Overview of Distributed Artificial Intelligence. In G. M. P. O'Hare and N. R. Jennings, editors, *Fundamentals of Distributed Artificial Intelligence*, pages 3–56. John Wiley and Sons, 1996.

M. Munusamy and H. P. Leang. Characteristics of Mobile Devices and an Integrated M-Commerce Infrastructure for M-Commerce Deployment. In *Proceedings of the second International Workshop on Internet Computing and E-Commerce (ICECE 2002)*, 2002.

E. Nguyen. Mobile Auction Enhances the Online Auction Experience. Master's thesis, Department of Information Systems and Computing, Brunel University, London, Great Britain, 2002.

H. S. Nwana. Software Agents: An Overview. *The Knowledge Engineering Review*, 11(3):205–244, 1996.

T. Olsson. *Bootstrapping and Decentralizing Recommender Systems.* PhD thesis, Licentiate Thesis 2003-006, Department of Information Technology, Uppsala University and SICS, 2003.

Omer F. Rana and K. Stout. What is Scalability in Multi-Agent Systems? In *Proc. of the fourth international conference on Autonomous agents*, pages 56–63, 2000.

H. V. D. Parunak, R. Savit, and R. L. Riolo. Agent-based Modeling vs. Equation-Based Modeling: A Case Study and Users' Guide. In N. G. Jaime, S. Sichman, and R. Conte, editors, *Proceedings of Multi-agent Systems and Agent-based Simulation (MABS'98)*, number 1534 in Lecture Notes in Artificial Intelligence, pages 10–25. Springer-Verlag, 1998.

S. A. Petersen, J. Rao, and A. Tveit. Challenges in Agent-based Support for Virtual Enterprises. In *Proceedings of the 1st International Workshop on Challenges in Open Agent Systems (at AAMAS'2002)*, Bologna, Italy, July 2002.

T. Poggio and Girosi. Regularization algorithms for learning that are equivalent to multilayer networks. *Science*, (247):978–982, 1990a.

T. Poggio and F. Girosi. Networks for Approximation and Learning. In *Proceedings of the IEEE*, volume 78, pages 1481–1497, 1990b.

W. Rand, D. G. Brown, S. E. Page, R. Riolo, M. Zellner, and L. E. Fernandez. An Agent-based Model of Suburban Sprawl. In *Proceedings of the 7th Annual Swarm Researchers Meeting*, 2003.

R. Raz. On the Complexity of Matrix Product. *SIAM Journal on Computing*, 32(5):1356–1369, 2003.

Reticular Systems. Using Intelligent Agents for Wireless E-Commerce Applications: The Yellowstone Project. Online, http://www.reticular.com/Library/yellowstone.pdf, September 1999.

R. Rifkin, G. Yeo, and T. Poggio. Chapter 7: Regularized Least-Squares Classification. In Suykens, Horvath, Basu, Micchelli, and Vandewalle, editors, *Advances in Learning Theory: Methods, Model and Applications*, volume 190 of *NATO Science Series III*. IOS Press, 2003.

R. M. Rifkin. *Everything Old Is New Again: A Fresh Look at Historical Approaches in Machine Learning*. PhD thesis, Massachusetts Institute of Technology (MIT), September 2002.

K. M. Rødseth and E. Breivik. Boredom and Balancing in Massively Multiplayer Online Gaems, December 2003.

Y. D. Rubinstein and T. Hastie. Discriminative vs informative learning. In D. Heckerman, H. Mannila, and D. Pregibon, editors, *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, pages 49–53. AAAI Press, 1997.

H.-P. Schwefel, I. Wegener, and K. Weinert, editors. *Advances in Computational Intelligence: Theory and Practice*. Natural Computing. Springer-Verlag, 2003.

J. C. Shafer, R. Agrawal, and M. Mehta. SPRINT: A Scalable Parallel Classifier for Data Mining. In T. M. Vijayraman, A. P. Buchmann, C. Mohan, and N. L. Sarda, editors, *Proceedings of the 22nd International Conference on Very Large Databases (VLDB)*, pages 544–555. Morgan Kaufmann, 1996a.

J. C. Shafer, R. Agrawal, and M. Mehta. SPRINT: A Scalable Parallel Classifier for Data Mining. In T. M. Vijayaraman, A. P. Buchmann, C. Mohan, and N. L. Sarda, editors, *Proceedings of the 22nd Int. Conf. Very Large Databases, VLDB*, pages 544–555. Morgan Kaufmann, 1996b.

V. Strassen. Gaussian Elimination is Not Optimal. *Numerische Mathematik*, (13):354–356, 1969.

M. Svensson. Explainin and Combining Recommender Algorithms for Decentralized Architectures (ENCORE). Technical report, SICS, Sweden, 2003.

L. Tesfatsion. Agent-based computational economics: Growing economies from the bottom up. *Artificial Life*, 8(1):55–82, 2002.

R. Thawonmas. Ambition of game mining (in japanese. *IPSJ Symposium Series*, 2003(1), 2003.

R. Thawonmas, J.-Y. Ho, and Y. Matsumoto. Identification of Player Types in Massively Multiplayer Online Games. In *Proceedings the 34th Annual Conference* *of the International Simulation and Gaming Association (ISAGA).* ISAGA, To be published by Springer-Verlag, August 2003.

Y. H. Tian, T. J. Huang, W. Gao, J. Cheng, and P. B. Kang. Two-Phase Web Site Classification Based on Hidden Markov Tree Models. In J. Liu, C. Liu, M. Klusch, N. Zhong, and N. Cercone, editors, *Proceedings of IEEE/WIC International Conference on Web Intelligence (WI 2003)*, pages 227–234. IEEE, October 2003.

A. N. Tikhonov. Solution of incorrectly formulated problems and the regularization method. *Soviet Mathematics Dokladi*, 4:1035–1038, 1963.

A. N. Tikhonov and V. Y. Arsenin, editors. *Solution of ill-posed problems.* John Wiley & Sons, 1977.

E. Turban, D. King, J. Lee, M. Warkentin, and M. Chung. *Electronic Commerce 2002: A Managerial Perspective.* Prentice Hall, 2nd edition, Jan 2002.

P. J. Turner and N. R. Jennings. Improving the Scalability of Multi-Agent Systems. In T. Wagner and O. F. Rana, editors, *Proceedings of the Workshop on Infrastructure for Agents, Multi-Agent Systems, and Scalable Multi-Agent Systems*, volume 1887 of *Lecture Notes in Computer Science*, pages 246–262. Springer, 2001.

A. Tveit. A Survey of Agent-Oriented Software Engineering. Online, http://abiody.com/jfipa/publications/AgentOrientedSoftwareEngineering/, May 2001a.

A. Tveit. Acceleration of Web Browsing using Predictive Retrieval. Technical report, Department of Computer and Information Science, NTNU, Sem Sælands vei 7-9, NO-7491 Trondheim, Norway, May 2001b.

A. Tveit. Peer-to-peer based recommendations for mobile commerce. In M. Devarakonda, A. Joshi, and M. Viveros, editors, *Proceedings of the First International Mobile Commerce Workshop*, pages 26–29. ACM Press, July 2001c.

A. Tveit. Game Usage Mining: Information Gathering for Knowledge Discovery in Massively Multiplayer Games. In H. R. Arabnia and Y. Mun, editors, *Proceedings of the International Conference on Internet Computing (IC'2002), session on Web Mining*, volume III, pages 636–642. CSREA Press, June 2002a.

A. Tveit. jfipa: an Architecture for Agent-based Grid Computing. In M. Schroeder and O. F. Rana, editors, *Proceedings of AISB'02 Convention, Symposium on AI and Grid Computing*, London, United Kingdom, April 2002b. Artificial Intelligence and Simulated Behavior (AISB).

A. Tveit. Scalability Analysis of the Zereal Massively Multiplayer Game Simulator. Technical Report ISSN: 0802-6394, 12/02, IDI, NTNU, December 2002c.

A. Tveit. Empirical Comparison of Accuracy and Performance for the MIPSVM Classifier with Existing Classifiers. Technical report, Divison of Intelligent Systems, Department of Computer and Information Science, Norwegian University of Science and Technology, November 2003a.

A. Tveit. Empirical Performance Evaluation of the Zereal Massively Multiplayer Online Game Simulator. Technical report, Divison of Intelligent Systems, Department of Computer and Information Science, Norwegian University of Science and Technology, November 2003b.

A. Tveit. On the Complexity of Matrix Inversion. Technical report, Divison of Intelligent Systems, Department of Computer and Information Science, Norwegian University of Science and Technology, November 2003c.

A. Tveit and H. Engum. Parallelization of the Incremental Proximal Support Vector Machine Classifier using a Heap-based Tree Topology. Technical report, IDI, NTNU, August 2003.

A. Tveit and M. L. Hetland. Multicategory Incremental Proximal Support Vector Classifiers. In *Proceedings of the 7th International Conference on Knowledge-Based Information & Engineering Systems (KES'2003)*, number 2773 in Lecture Notes in Artificial Intelligence (LNAI), pages 386–392. Springer-Verlag, 2003.

A. Tveit, M. L. Hetland, and H. Engum. Incremental and Decremental Proximal Support Vector Classification using Decay Coefficients. In *Proceedings of the 5th International Conference on Data Warehousing and Knowledge Discovery (DAWAK'2003)*, number 2737 in Lecture Notes in Computer Science (LNCS), pages 422–429. Springer-Verlag, 2003a.

A. Tveit and M. Matskin. Acceleration of Mobile Commerce using Predictive Retrieval. Technical report, IDI, NTNU, October 2002.

A. Tveit, Ø. Rein, J. V. Iversen, and M. Matskin. Scalable Agent-based Simulation of Players in Massively Multiplayer Online Games. In *Proceedings of the 8th Scandinavian Conference on Artificial Intelligence*, Frontiers in Artificial Intelligence and Applications. IOS Press, 2003b.

G. Upton and I. Cook, editors. *Dictionary of Statistics.* Oxford University Press, 1st edition, 2002.

V. N. Vapnik. *The Nature of Statistical Learning Theory*, chapter 5, pages 138–146. Springer-Verlag, 2nd edition, 1999.

A. P. Vrechopoulos, I. D. Constantiou, and I. Sideris. Strategic Marketing Planning for Mobile Commerce Diffusion and Consumer Adoption. In *Proceedings of the 1st International Conference on Mobile Business (M-Business 2002)*, July 2002.

M. Weiss. Exploiting Communities of Interest to Find Information on the Web. Discussion White Paper, Carleton University, Ottawa, Canada, Published Online: http://www.scs.carleton.ca/~weiss/conav/papers/conav.pdf, Feb 2002a.

M. Weiss. Searching for Information on the Web Using Accumulated Knowledge of Others. In *Proceedings of the International Sunbelt Social Network Conference XXII, New Orleans, USA*, Feb 2002b.

L. K. Wickramasinghe, S. W. Loke, A. Zaslavsky, and D. Alakahoon. A-GATE: A System of Relay and Translation Gateways for Communication among Heterogeneous Agents in Ad Hoc Wireless Environments. In J.-B. Stefani, I. M. Demeure, and D. Hagimont, editors, *Proceedings of the 4th IFIP Conference on Distributed Applications and Interoperable Systems (DAIS 2003)*, volume 2893 of *Lecture Notes in Computer Science*, pages 212–223. Springer-Verlag, 2003.

G. Widmer and M. Kubat. Learning in the Presence of Concept Drift and Hidden Contexts. *Machine Learning*, 23(1):69–101, 1996.

M. Wilson. C Sharp Performance: Comparison with C, C++, D, and Java. *Windows Developer Network*, pages 2–23, Fall 2003.

M. Wooldridge and N. Jennings. Intelligent Agents: Theory and Practice. *The Knowledge Engineering Review*, 10(2):115–152, 1995.

N. Zhong, J. Liu, and Y. Yao, editors. *Web Intelligence*. Springer-Verlag, Heidelberg, Germany, 1st edition, 2003.

N. Zhong, J. Liu, Y. Y. Yao, and S. Ohsuga. Web Intelligence (WI). In *Proceedings of the 24th IEEE Computer Society International Computer Software and Applications Conference (COMPSAC 2000)*, pages 469–470. IEEE CS Press, 2000.

Index