UNIVERSIDADE FEDERAL DO AMAZONAS
INSTITUTO DE COMPUTAÇÃO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

INFERÊNCIA DE CONTEXTO PARA DISPOSITIVOS MÓVEIS UTILIZANDO APRENDIZAGEM POR REFORÇO

LEONARDO LIRA GUIMARÃES

Amazonas
Maio de 2015

LEONARDO LIRA GUIMARÃES

INFERÊNCIA DE CONTEXTO PARA DISPOSITIVOS MÓVEIS UTILIZANDO APRENDIZAGEM POR REFORÇO

Dissertação apresentada ao Programa de Pós-Graduação em Informática do Instituto de Ciências Exatas da Universidade Federal do Amazonas como requisito parcial para a obtenção do grau de Mestre em Informática.

Orientador: Horácio Antonio Braga Fernandes de Oliveira

Amazonas

Maio de 2015

Ficha Catalográfica

Ficha catalográfica elaborada automaticamente de acordo com os dados fornecidos pelo(a) autor(a).

Guimaraes, Leonardo Lira
G963i Inferência de contexto para dispositivos móveis utilizando aprendizagem por reforço / Leonardo Lira Guimaraes. 2015
111 f.: il.; 31 cm.

Orientador: Horácio Antonio Braga Fernandes de Oliveira
Dissertação (Mestrado em Informática) - Universidade Federal do Amazonas.

1. Inferência de Contexto. 2. Aprendizagem Por Reforço. 3. Dispositivos Móveis. 4. CoRe-RL. I. Oliveira, Horácio Antonio Braga Fernandes de II. Universidade Federal do Amazonas III. Título

FOLHA DE APROVAÇÃO

"Inferência De Contexto Para Dispositivos Móveis Utilizando Aprendizagem Por Reforço"

LEONARDO LIRA GUIMARÃES

Diessertação de Mestrado defendida e aprovada pela banca examinadora contituída pelos Professores:

Prof. Horácio Antonio B. Fernandes de Oliveira - PRESIDENTE

Prof. Eduardo Freire Nakamura - MEMBRO INTERNO

Prof. Carlos Maurício Seródio Figueiredo - MEMBRO INTERNO

Prof. José Luiz de Souza Pio - MEMBRO EXTERNO

Manaus, 25 de Maio de 2015

Aprendi que para ser melhor é preciso aprender. Dedico essa dissertação à mulher que muito me ensinou – minha mãe (Lídia) – e à mulher com quem muito ainda aprenderei – minha futura esposa (Ingrid). Obrigado pelos reforços positivos e negativos!

Agradecimentos

À minha mãe, Lídia Lira Guimarães, que me proporcionou uma boa educação e sempre foi um exemplo como pessoa, me ensinando, na prática, os princípios que moldaram o meu caráter, os quais levarei para o resto da minha vida. Obrigado por me sustentar tanto financeiramente como emocional e espiritualmente durante o período do mestrado, espero em breve poder retribuir todo o carinho e dedicação que recebi ao longo desses anos.

À minha noiva, Ingrid Sampaio, que sempre me ajudou e acompanhou, me aconselhando e incentivando em momentos difíceis. Demonstrando amor e compreensão, mesmo nos dias mais estressantes. Obrigado por estar sempre junto e por me ouvir, demonstrando cuidado e interesse.

À minha irmã, Laila Rocco, e cunhado, Ricardo Rocco, que também foram muito importantes neste processo de formação e aprendizado, por se preocuparem comigo e com meu desempenho na pós.

Aos demais familiares, aos mais próximos e aos mais distantes, agradeço pelo suporte prestado e por se importarem comigo e com meu futuro. Sou grato pelos exemplos que tive: de profissionais que trabalham no que gostam e sabem como gastar o que ganham.

Aos amigos da igreja, muitos dos quais foram fundamentais tanto diretamente como indiretamente pelo meu desenvolvimento e crescimento como profissional. Obrigado pelas orações e pelos incentivos.

Ao meu orientador, Horácio Fernandes, que me adotou como orientando desde o terceiro período do curso de Engenharia da Computação, me proporcionando experiências acadêmicas que com certeza marcaram a minha vida. Obrigado pelos puxões de orelha, sempre seguidos de palavras de incentivos e motivação, as quais me trouxeram até este ponto.

Aos demais professores, que, através de suas aulas, trabalhos e sermões, me fizeram crescer acadêmica e profissionalmente. Muitos dos quais foram verdadeiros mestres (no sentido mais completo da palavra), tendo em vista seus próprios exemplos como

professores, pesquisadores, cientístas e engenheiros.

Ao Instituto de Computação (ICOMP) da Universidade Federal do Amazonas (UFAM), que é um instituto de referência na universidade e no Brasil. Obrigado por me proporcionarem os melhores recursos (treinamentos, disciplinas, salas, laboratórios e professores) dentre os disponíveis, os quais foram de grande importância para o meu enriquecimento acadêmico-profissional.

À Deus dirijo minha maior gratidão. Por ser presente e atuante em minha vida, pelo cuidado a mim demonstrado em diversos momentos desta jornada. Mesmo sem merecer, Ele proporcionou momentos únicos em minha vida, sempre suprindo minhas necessidades e me dando forças pra continuar e vencer os obstáculos e desafios que surgiram à minha frente. Obrigado Pai pela tua fidelidade e amor, sem Ti eu não seria nada.

"Entrega o teu caminho ao Senhor,
confia nEle e o mais Ele fará"
(Salmo 37.5)

Resumo

Os avanços das tecnologias de comunicação sem fio e de hardware impulsionaram a popularização do uso de dispositivos móveis. Cada vez mais, estes dispositivos ganham novos recursos de hardware (i.e., sensores e outros gadgets) e software (e.g., reconhecimento facial, de voz, gestos) a fim de que a interação humano-computador ocorra de forma mais natural. Esses recursos deram aos dispositivos móveis uma capacidade maior de percepção do ambiente e das condições nas quais os usuários se encontram, possibilitando o desenvolvimento de aplicações cada vez mais proativas e sensíveis ao contexto.

Um sistema sensível ao contexto é capaz de modificar seu comportamento de acordo com os contextos inferidos do ambiente. Entretanto, interpretações errôneas dos dados coletados podem induzir ações inapropriadas e indesejadas nas aplicações. Embora exista uma variedade de técnicas de inferência na literatura (e.g., regras, ontologias, que utilizam aprendizagem supervisionada e não supervisionada), em geral, elas não consideram se as inferências foram de fato adequadas para os contextos do usuário. Além disso, a maioria dessas técnicas utiliza modelos estáticos de inferência (i.e., que não são capazes de se ajustar à mudanças nas condições do ambiente), o que representa uma limitação dessas técnicas quando aplicadas ao domínio das aplicações móveis.

Neste trabalho, é proposta uma nova técnica de inferência de contexto para aplicações móveis – chamada de CoRe-RL – que utiliza aprendizagem por reforço a fim de que sejam produzidas inferências cada vez mais adequadas aos contextos do usuário. Nesta técnica, a aprendizagem ocorre de maneira incremental e conforme o usuário interage com o sistema, permitindo que a inferência seja ajustada por meio de recompensas (reforços positivos) e punições (reforços negativos) associadas aos contextos inferidos. Como os contextos estão continuamente sendo aprendidos, a técnica proposta também permite às aplicações um gerenciamento flexível de contextos, ou seja, é possível que novos contextos (rótulos) sejam cadastrados e aprendidos ao longo do tempo. O funcionamento da técnica é divido em duas etapas – classificação e adap-

tação. O CoRe-RL utiliza o método dos K vizinhos mais próximos (modificado) na classificação. A aprendizagem (adaptação) é baseada em exemplos, mas também faz ajustes sobre os modelos (ranking de características) que ponderam as características mais relevantes de cada contexto, na etapa de classificação.

Com o intuito de testar e avaliar o desempenho da técnica proposta, foi desenvolvido, como estudo de caso deste trabalho, um aplicativo que implementa todas as funcionalidades e recursos do CoRe-RL. Através deste aplicativo, foram realizados experimentos práticos de avaliação da classificação e adaptação, em dois cenários específicos: no primeiro cenário havia um único contexto; e no segundo haviam três. Por meio dos experimentos práticos, observou-se que, de acordo com o limiar de corte usado, é possível obter bons desempenhos na classificação mesmo com uma base pequena e com um ranking pouco ajustado. Além disso, demonstrou-se que o CoRe-RL melhora seu desempenho, convergindo para o desempenho ótimo, de acordo com a ocorrência das interações.

Palavras-chave: Inferência de Contexto, Aprendizagem Por Reforço, Dispositivos Móveis.

Abstract

Advances in wireless communication and computer hardware technologies have boosted the popularity of mobile devices. Increasingly, these devices gain new features of hardware (i.e., sensors and other gadgets) and software (e.g., facial, voice and gestures recognition) so that the human-computer interaction can occur more naturally. These features allowed a greater awareness of the environment and the conditions under which the users are, enabling the development of applications ever more proactive and sensitive.

A context aware system can modify its behavior according to the inferred context of the environment. However, erroneous interpretations of the collected data may induce inappropriate and unwanted actions in applications. Although there is variety of inference techniques in the literature (e.g., rules, ontologies, that uses supervised and unsupervised learning), generally, they do not consider whether the inferences were indeed suitable to the user contexts. Furthermore, most of these techniques uses static inference models (i.e., they are unable to adjust themselves to changes in the environment conditions), which represents a limitation of these techniques when applied to the field of mobile applications.

This work proposes a new context reasoning technique for mobile applications – called CoRe-RL – which uses reinforcement learning in order that the produced inferences could be ever more suitable to the user's contexts. In this technique, learning occurs in an incremental manner and as the user interacts with the system, allowing the inference to be adjusted by the rewards (positive reinforcements) and punishments (negative reinforcements) associated to the inferred contexts. As the contexts are continuously being learned, the proposed technique also allows a flexible context management to the applications, which enables new contexts (labels) to be registered and learned over time. The operation of the technique is divided into two stages – classification and adaptation. The CoRe-RL uses a modified version of the K nearest neighbors in the classification stage. The learning (adaptation) stage is based on examples, but also makes adjustments on the models (features ranking) which weigh the most relevant

features of each context in the classification stage.

In order to validate and evaluate the proposed technique, it was developed, as a case study of this work, an application that implements all of the functionality and capabilities of CoRe-RL. Through this application, practical experiments for evaluating the classification and adaptation were executed in two specific scenarios: there was a single context in the first scenario; and in the second, there were three. Through the practical experiments, it was observed that, in accordance to the cutting threshold used, it is possible to obtain good performances in the classification even with a small base and with a slightly adjusted ranking. Furthermore, it was demonstrated that the CoRe-RL improves its performance, converging to the optimal performance, in accordance to the occurrence of new interactions.

Keywords: Context Reasoning, Reinforcement Learning, Mobile Devices.

Lista de Figuras

Lista de Tabelas

Sumário

Capítulo 1

Introdução

Uma coisa é querer aprender.
Outra é querer garantias de que não vai errar.

Geraldo Eustáquio

Os computadores modernos realizam, com eficiência, tarefas descritas por meio de algoritmos. Os seres humanos, por sua vez, são capazes de realizar as mesmas tarefas de formas diferentes e mais adequadas, pois consideram e são influenciados pelo ambiente em que estão inseridos. Se dermos aos computadores atuais a capacidade de identificar contextos, a partir do ambiente, estes também poderão realizar tarefas de forma mais adequada para determinados contextos (e.g., uma aplicação móvel poderá se comportar de formas diferentes, tendo em vista diferentes ambientes físicos e/ou usuários). Portanto, segundo Dey & Abowd [1999], ao levar em consideração as informações contextuais do ambiente, aumenta-se a riqueza da comunicação na interação humano computador e torna-se possível a criação de serviços computacionais cada vez mais úteis para o usuário.

Nos últimos anos foram intensificadas as pesquisas e discussões na área da computação móvel e ubíqua, devido ao barateamento e grande popularização dos dispositivos móveis. Estes dispositivos, são equipados cada vez mais com um maior número de sensores (e.g., acelerômetros, sensor de luminosidade, sensor de temperatura) e novas interfaces (e.g., reconhecimento facial, reconhecimento de voz, suporte a gestos) possibilitando uma interação mais natural para o usuário. Este cenário proporcionou o surgimento de milhares de aplicações voltadas para dispositivos móveis. Contudo, a maioria destas não utiliza os recursos dos aparelhos para prover serviços mais inteligentes, que considerem o contexto do usuário (i.e., qualquer informação que pode ser usada

para caracterizar a situação de uma determinada entidade [Dey & Abowd, 1999]). A sensibilidade ao contexto traz um novo potencial de funcionalidades e recursos, ao proporcionar a criação de serviços e aplicações mais inteligentes, adaptáveis, proativas e que exigem menos interação do usuário.

A partir da perspectiva de Sistemas Sensíveis ao Contexto, um contexto possui um ciclo de vida que envolve quatro etapas básicas [Perera et al., 2013]: Aquisição de Dados Contextuais, Modelagem de Contexto, Inferência de Contexto e Distribuição. Dentre essas etapas destacamos a Inferência de Contexto, que é a etapa em que informações contextuais de alto nível são produzidas a partir de dados brutos e/ou informações contextuais de baixo nível. Nesta fase, os dados são tratados e processados, com o intuito de extrair informações contextuais úteis para as aplicações. Portanto, caso a técnica de inferência empregada seja inadequada para o domínio de aplicação em que está sendo utilizada, serão produzidas informações contextuais erradas e/ou imprecisas para as aplicações.

O domínio das aplicações para dispositivos móveis tem como foco o usuário. Portanto, deseja-se identificar informações que caracterizem a situação do usuário. Contudo, usuários diferentes possuem características, hábitos, comportamentos e interesses diferentes, o que dificulta a criação de soluções gerais que funcionem bem para todos os usuários. Por este motivo, um sistema sensível ao contexto voltado para aplicações móveis, deve ser ajustável e capaz de se adaptar aos diferentes ambientes e usuários.

1.1 Motivação

Para que um sistema sensível ao contexto se ajuste a um usuário, é necessário que o sistema seja capaz de aprender, com o próprio usuário (i.e., características, hábitos, comportamentos e interesses). Portanto, justifica-se a utilização de aprendizagem de máquina para que o sistema infira contextos com base em um modelo extraído do próprio ambiente.

Em geral, as técnicas de aprendizagem de máquina que são aplicadas nas abordagens de sistemas móveis sensíveis ao contexto são técnicas offline. Ou seja, o sistema treina um classificador a partir de uma base de dados, e somente depois que o classificador atinge certo grau de eficácia (i.e., aprende um modelo específico) é que o sistema será usado na prática para inferir contextos. Contudo, devido às peculiaridades da computação móvel, o ideal é que sejam utilizadas técnicas que aprendem de modo online (i.e. conforme os dados vão sendo coletados).

Existem três categorias principais, nas quais as técnicas de aprendizagem de má-

quina são classificadas: Aprendizagem Supervisionada, Aprendizagem Não Supervisionada e Aprendizagem por Reforço.

Na aprendizagem supervisionada, o sistema aprende a rotular dados a partir de exemplos previamente rotulados. As abordagens que utilizam aprendizagem supervisionada conseguem bons resultados na inferência de contexto em ambientes restritos (controlados) e com usuários específicos. Em geral, estas abordagens requerem um conjunto (base) de dados previamente classificados, para que o sistema aprenda a classificar novos dados, o que nem sempre é possível em sistemas móveis. Embora existam técnicas incrementais de aprendizagem supervisionada, elas não são empregadas na inferência para dispositivos móveis, pois precisam de um "professor" que indique, em caso de erro, o contexto correto que deveria ter sido inferido. A única entidade capaz de dizer o contexto exato de um usuário é o próprio usuário. Portanto, o único "professor" capaz de rotular corretamente os contextos é o próprio usuário. Entretanto, a cada inferência solicitada por qualquer aplicação há a necessidade de se perguntar ao usuário qual o resultado esperado. Esta exigência constante de interações é um problema, uma vez que compromete a proatividade das aplicações.

A aprendizagem não supervisionada realiza o agrupamento e classificação de dados não rotulados a partir das características e padrões de comportamentos dos mesmos. Entretanto, o resultado desses agrupamentos não possuem rótulos, a não ser que cada agrupamento criado seja nomeado por um "professor". Isto dificulta que aplicações interessadas em contextos específicos utilizem os resultados da aprendizagem não supervisionada. Por exigir pouca interação com o usuário, Mayrhofer et al. [2003] considera a utilização de aprendizagem não supervisionada de forma online, entretanto também ressalta a necessidade que o usuário rotule adequadamente os agrupamentos. Esta tarefa nem sempre é trivial, pois, o sistema deverá apresentar os padrões de forma inteligível a qualquer usuário. Além disso, nem todo padrão encontrado representa um contexto significativo para o usuário, ou para as aplicações.

Assim como nas outras técnicas de aprendizagem de máquina, a aprendizagem por reforço utiliza os dados contextuais (estímulos) produzidos pelo ambiente, para produzir uma resposta (ação). Entretanto, o sistema também considera um sinal de reforço (*feedback*) produzido pelo ambiente (usuário e/ou aplicações). Desta forma, uma técnica de inferência de contexto que utilize aprendizagem por reforço, será capaz de considerar se o resultado da inferência foi realmente útil (adequado) para o usuário, aprendendo com esta informação. Portanto, não há a necessidade de que o usuário informe o contexto esperado a cada inferência, basta que as aplicações ou serviços retornem um sinal de reforço positivo, caso a inferência produzida pelo sistema tenha sido adequada, ou negativo caso contrário. O sinal de reforço pode ser obtido basicamente

de duas formas: explícita, através de interfaces que permitam ao usuário informar se o resultado (ou a ação decorrente deste resultado) foi adequado; e implícita, levando em consideração a forma como o usuário reagiu ao resultado (e.g., se houve interação, quantas interações e quanto tempo ele usou o serviço ou informação provido com base na inferência).

Diversas abordagens de sistemas sensíveis ao contexto para dispositivos móveis já foram propostas na literatura. Várias dessas abordagens são frameworks conceituais que definem a arquitetura, organização e funcionamento desses sistemas. Essas abordagens tem como foco prover sensibilidade ao contexto para outras aplicações, tornando secundária, em alguns casos, a escolha da técnica de inferência a ser utilizada. Ainda assim, é possível destacar as principais categorias de técnicas de inferência utilizadas em sistemas móveis sensíveis ao contexto: baseadas em regras [Garlan et al., 2002; Capra et al., 2003; Fahy & Clarke, 2004; Biegel & Cahill, 2004]; baseadas em ontologias [Korpipaa et al., 2003; Devaraju et al., 2007; Guo & Zhang, 2010; Riboni & Bettini, 2010]; que utilizam lógica fuzzy [Korpipaa et al., 2003]; que utilizam aprendizagem supervisionada [Korpipaa et al., 2003; Biegel & Cahill, 2004; Riboni & Bettini, 2010; Park et al., 2011]; que utilizam aprendizagem não supervisionada [Mayrhofer et al., 2003].

Embora já tenham sido propostas abordagens que utilizam aprendizagem de máquina para inferir contexto com base em um modelo de um ambiente real, em geral, nessas abordagens a aprendizagem ocorre de maneira offline. Como essas abordagens não continuam aprendendo com o ambiente, elas não são capazes de se adequar aos diferentes usuários e ambientes físicos (i.e., melhorar suas inferências com o tempo), nem de se adaptar às possíveis mudanças do ambiente físico ou de comportamentos e interesses do usuário. Por esses motivos, é importante que a aprendizagem na inferência de contexto seja também incremental.

Durante o levantamento bibliográfico feito neste trabalho não foram encontradas abordagens destinadas a computação móvel, que utilizassem a aprendizagem por reforço na inferência de contexto. As abordagens mais parecidas são voltadas para área de ambientes inteligentes [Ranganathan & Campbell, 2003; Rizou et al., 2010; Kabir et al., 2015], e consideram que os usuários interagem com um ambiente físico específico. Entretanto, na computação móvel, tanto o ambiente físico, como os comportamentos do usuário podem mudar ao longo do tempo, o que torna o desafio de inferir contextos ainda maior.

1.2 Objetivos

Tendo em vista o problema e as motivações apresentadas na Seção 1.1, o objetivo geral de pesquisa deste trabalho é: Propor e avaliar uma técnica incremental de inferência de contexto para aplicações móveis capaz de aprender com o próprio ambiente, por meio de sinais de reforço (i.e. aprendizagem por reforço), a fim de que sejam produzidas inferências cada vez mais adequadas ao contexto do usuário.

Os objetivos específicos deste trabalho incluem:

- Aplicar a técnica proposta em uma aplicação móvel, a fim de que seja possível avaliar o desempenho da mesma em ambientes reais;
- Demonstrar a viabilidade da utilização da técnica proposta na inferência de contexto, através da avaliação do desempenho da classificação em cenários específicos.
- Demonstrar que a técnica proposta aprende a inferir contextos, através da verificação da ocorrência de melhorias no desempenho da classificação, conforme se aumenta a quantidade de interações em cenários específicos.

1.3 Principais Contribuições

A principal contribuição deste trabalho consiste na avaliação de uma nova técnica de inferência de contexto, demonstrando a viabilidade de uso da mesma em ambientes de computação móvel, e que a técnica é capaz de aprender a inferir contextos de modo incremental por meio de reforços referentes a adequabilidade do resultado da inferência. A avaliação é feita através de experimentos práticos com uma aplicação móvel real, construída como um estudo de caso da abordagem proposta.

A fim de que a técnica proposta possa ser aplicada em sistemas móveis sensíveis ao contexto, também é proposto, neste trabalho, um framework que suporta as peculiaridades da técnica, permite um gerenciamento flexível de contexto e é capaz de prover sensibilidade ao contexto às aplicações. A seguir, são listadas algumas vantagens que justificam a utilização da abordagem proposta, em sistemas móveis sensíveis ao contexto:

- Gerenciamento de contexto Flexível - usuários e aplicações podem criar rótulos para os contextos que deverão ser inferidos. Ou seja, não há uma limitação sobre quais contextos serão aprendidos;

- Ajustável aos diferentes usuários - pode aprender modelos diferentes para um mesmo contexto, tendo em vista usuários ou ambientes diferentes;
- Adequável - melhora os resultados conforme se aumenta o número de interações para um mesmo ambiente;
- Adaptável - é capaz de mudar o seu comportamento, caso o ambiente mude.

1.4 Organização da Dissertação

Esta dissertação está dividida em seis capítulos. No capítulo 2 é apresentada uma fundamentação teórica referente aos principais conceitos envolvidos em Computação Sensível ao Contexto, e alguns aspectos que diferenciam os Sistemas Sensíveis ao Contexto (i.e. modelagem de contexto, técnicas de inferência e arquiteturas) entre si. Além disso, também são introduzidos conceitos e as principais técnicas de aprendizagem de máquina aplicadas em Computação Sensível ao Contexto.

No capítulo 3 são apresentados e analisados os trabalhos correlatos. Estes foram divididos em três categorias, tendo em vista as características que os relacionam com este trabalho: 1) Sistemas Sensíveis ao Contexto para Dispositivos Móveis; 2) Aplicações Móveis Sensíveis ao Contexto; 3) (outros) Sistemas Sensíveis ao Contexto que utilizam aprendizagem por reforço.

No capítulo 4 é apresentada uma visão geral do Mob Context, que consiste em uma nova abordagem de Sistema Sensível ao Contexto voltada para o domínio das aplicações móveis, e é detalhado o funcionamento do CoRe-RL, que é a técnica de inferência de contexto (via aprendizagem por reforço) proposta neste trabalho e aplicada no Mob Context.

O capítulo 5 apresenta detalhes da implementação de um aplicativo (Mob Context App) desenvolvido como estudo de caso para avaliação da técnica proposta, além disso, também são detalhados os cenários e a metodologia usada nos experimentos realizados, bem como os resultados obtidos nos mesmos.

Por fim, no capítulo 6 são feitas as conclusões deste trabalho e sugestões para trabalhos futuros.

Capítulo 2

Fundamentação Teórica

O começo de todas as ciências é o espanto de as coisas serem o que são.

Aristóteles

Neste capítulo, serão apresentadas as definições e conceitos da área de Computação Sensível ao Contexto e Aprendizagem de Máquina, levando sempre em consideração o ponto de vista da Computação Móvel, isto é, a capacidade dos usuários moverem fisicamente serviços computacionais, permitindo a realização de tarefas mediadas por computação, independente da localização dos mesmos (e.g., *smartphones*, *tablets*) [Araujo, 2003]. Embora o domínio das aplicações móveis seja o foco deste trabalho, os conceitos e técnicas de aprendizagem de máquina que serão apresentados são aqueles mais usados na inferência de contexto dos diversos domínios da computação sensível ao contexto.

2.1 Computação Sensível ao Contexto

Segundo Bellavista et al. [2012], Sensibilidade ao contexto é a capacidade de se obter e passar à camada de serviço (ou aplicação) qualquer informação relevante que pode caracterizar o ambiente, bem como recursos computacionais, localização física do dispositivo, preferências do usuário e limitações temporais.

Computação Sensível ao Contexto (CSC) tem como objetivo possibilitar uma melhor prestação de serviços através da adaptação proativa do uso, acesso, estrutura e comportamento de informações, serviços, aplicações e recursos físicos no que diz respeito à informação contextual disponível [Soylu et al., 2009]. Chen & Kotz [2000] define

CSC como um paradigma de computação móvel em que as aplicações podem descobrir e tirar proveito de informações contextuais (e.g Localização do usuário, horário, pessoas e dispositivos próximos e atividades do usuário).

Neste trabalho, Computação Sensível ao Contexto é definida como: *Um paradigma de computação onde computadores coletam e processam dados brutos, obtidos a partir de sensores, a fim de extrair informações contextuais do ambiente e utilizá-las na realização de tarefas (e.g., melhorar a interação com usuário, prestar serviços inteligentes, proporcionar proatividade).* A partir da definição apresentada, é possível listar quatro fases principais envolvidas em CSC:

- Sensoriamento - corresponde à aquisição (coleta) de dados brutos a partir de sensores;
- Inferência - é a etapa em que os dados brutos são processados e transformados em informações contextuais, utilizando uma ou mais técnicas de inferência de contexto;
- Decisão - a aplicação deve decidir como irá tirar proveito das informações contextuais inferidas, isto é, escolher a ação mais apropriada para aquele contexto;
- Ação (atuação) - é o resultado da decisão, ou seja, a execução da tarefa escolhida como a mais adequada para o contexto inferido.

A definição apresentada é capaz de englobar a maioria das aplicações, sistemas ou serviços sensíveis ao contexto. Contudo, neste trabalho será apresentado um framework conceitual que é capaz de aprender com as interações do usuário de modo incremental. Desta forma, ao apresentar a abordagem proposta, no capítulo 4, será introduzida uma quinta etapa, chamada de "Adaptação". Na adaptação, o sistema deverá levar em conta se as inferências, ou as ações realizadas, foram adequadas ao contexto do usuário, para que futuramente seja possível produzir inferências mais precisas, ou ações mais adequadas. A adaptação não é requisito fundamental em CSC, por esse motivo esta etapa não é inclusa na lista das quatro fases principais envolvidas em CSC, e nem na definição proposta de CSC.

A fim de esclarecer ainda mais e exemplificar o funcionamento das etapas apresentadas, fundamentais na Computação Sensível ao Contexto, a seguir, será feita uma analogia com o ser humano.

Um elemento básico, presente em qualquer SSC, são os sensores. Estes são responsáveis por observar variáveis específicas do ambiente físico e virtual, no qual o usuário está inserido, permitindo que o sistema capture e extraia informações contextuais deste

ambiente. É possível fazer uma analogia com os orgãos dos sentidos do ser humano (e.g., pele, nariz, boca, olhos e ouvidos), que são responsáveis pela percepção do mundo (i.e., ambiente). Sem a capacidade de perceber o mundo o ser humano não seria capaz sequer de sobreviver, pois não teria informações do ambiente para se adaptar a ele, mesmo que possuisse o melhor corpo e o cérebro mais desenvolvido. De modo similar, um sistema não pode ser sensível ao contexto sem utilizar sensores, pois não é capaz de monitorar, muito menos de identificar contextos.

O ser humano é capaz de agir de forma adequada ao ambiente pois, além de perceber o mundo, também é capaz de interpretar corretamente aquilo que é percebido. De modo análogo, para que um sistema seja sensível ao contexto é necessário que infira corretamente os contextos do ambiente. Portanto, outro elemento de grande importância para os SSC's é a Inferência de Contexto. Entretanto, para que a inferência ocorra, é importante organizar os dados e informações contextuais (modelagem de contexto) e manter alguma representação do ambiente, baseada em conhecimentos prévios ou aprendidos ao longo do tempo pelo sistema. De igual modo, o ser humano representa o ambiente por meio de abstrações, mantém (aprende) o que é relevante na memória e utiliza essas informações para interpretar situações com base em experiências ou conhecimentos aprendidos.

Para adaptar-se a um ambiente, não basta interpretar corretamente o que é percebido do ambiente. É importante também que se faça um bom mapeamento da ação mais adequada para o contexto inferido. Pois, mesmo que a inferência funcione bem, caso o sistema escolha ações inadequadas aos contextos, o maior prejudicado será o usuário, que poderá optar por deixar de usar o sistema. De modo semelhante, o ser humano só sobrevive no ambiente se também for capaz de decidir (escolher) as ações apropriadas para cada situação que vive. Por exemplo, ao sentir frio é apropriado que o ser humano busque calor para manter a temperatura ideal do corpo, caso contrário, devido à incapacidade de se adaptar ao ambiente, dependendo das condições, este poderá não sobreviver. Assim, o terceiro elemento presente nos SSC's é a capacidade de Decisão. Na prática, a maioria das abordagens de SSC atribui esta responsabilidade às aplicações.

O ser humano possui um corpo formado por orgãos e membros com diversas funcionalidades. O homem percebe e interage com o ambiente por meio desse corpo. É possível fazer uma analogia das aplicações de um SSC com os membros do corpo humano responsáveis pela atuação no ambiente (e.g., perna, pé, dedos, mão, braço), pois, as aplicações realizam tarefas e prestam diversos serviços ao usuário (ambiente), e por meio delas ocorrem as interações entre usuário e sistema. Um SSC incapaz de interagir com o ambiente (i.e., sem aplicação) deixa de ser sensível ao contexto para ser apenas

consciente do contexto. Um SSC deve ser capaz de modificar seu comportamento (suas ações) de acordo com os contextos detectados. Assim, a responsabilidade pela tomada de decisão, quanto ao que (e como) será feito, é passada às aplicações, visto que os serviços e tarefas que um SSC pode executar dependem das aplicações.

As subseções seguintes apresentam em detalhes os principais termos usados na definição de computação sensível ao contexto, deste trabalho.

2.1.1 Dados Brutos e Contexto

Dados Brutos são dados coletados a partir de sensores, podendo conter (ou não) informações contextuais (i.e., que caracterizem a situação de uma entidade). Além disso, os dados brutos podem ser imprecisos (i.e., ter um erro associado), conter inconsistências, ruídos e outros defeitos. Em geral, estes problemas são contornados através de um pré-processamento, utilizando técnicas adequadas a cada tipo de dado, a fim de tratar as imperfeições e incertezas presentes nos mesmos.

Contexto é qualquer informação que pode ser usada para caracterizar a situação de uma determinada entidade. Uma Entidade é uma pessoa, um lugar, ou objeto que é considerado relevante para a interação entre um usuário e uma aplicação, incluindo o próprio usuário e a aplicação [Dey & Abowd, 1999]. Esta definição é a mais aceita e referenciada na área, provavelmente por ser genérica suficiente para identificar contextos nos mais diversos cenários. Outra definição, feita por Chen & Kotz [2000], assume que contexto é o conjunto de estados e configurações do ambiente que podem determinar o comportamento de uma aplicação, ou causar a ocorrência de um evento específico (da aplicação) que é relevante para o usuário.

Neste trabalho, as informações contextuais são diferenciadas em dois tipos de acordo com os níveis de representatividade que possuem:

- Informações contextuais de Baixo Nível - são as informações que representam uma característica do ambiente, e são produzidas a partir do processamento dos dados brutos. Por exemplo, um sensor captura uma temperatura de 32 °C, a partir de uma regra simples produz-se a informação contextual de que a temperatura está alta.

- Informações contextuais de Alto Nível - são informações que representam um estado do ambiente (i.e., envolve um conjunto de características), e são resultantes do processamento de informações contextuais de baixo nível. Por exemplo, o sistema infere que o usuário está no contexto “Dormindo”, após observar

as seguintes características:Localização=Casa, Luminosidade=Baixa, Temperatura=Baixa, Velocidade=Nula, Som=Ruído.

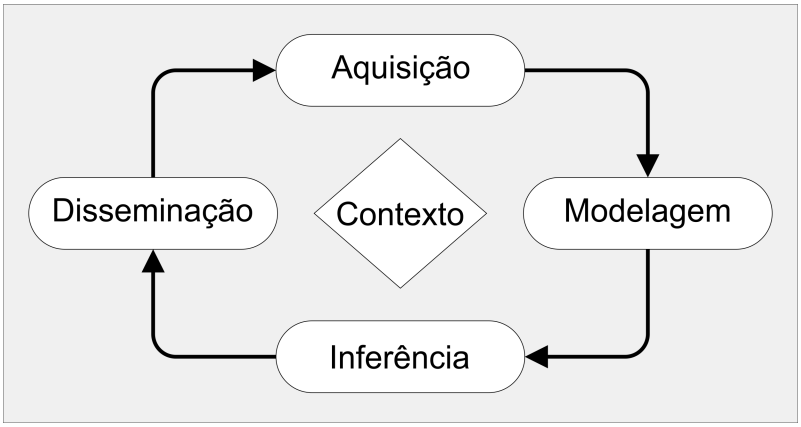


Figura 2.1: Ciclo de Vida do Contexto (Adaptado de Perera et al. [2013]).

Perera et al. [2013] estabelece quatro fases básicas, presentes nas diferentes abordagens propostas da literatura, para o ciclo de vida do contexto em sistemas sensíveis ao contexto. Estas fases, ilustradas na figura 2.1, são apresentadas a seguir:

- Aquisição de Contexto - Os dados e informações contextuais são obtidos a partir de uma ou mais fontes;
- Modelagem de Contexto - As características extraídas do ambiente são representadas e modeladas de forma significativa para o sistema;
- Inferência de Contexto - Os contextos (de baixo nível) modelados são processados a fim de derivar informações contextuais de alto nível;
- Disseminação de Contexto - A informação contextual de alto nível é distribuída para os consumidores (aplicações e serviços) interessados.

2.1.2 Sensores

Na literatura existem diversas definições de sensores, aplicadas a diferentes campos de pesquisa. O termo "sensor" será usado, neste trabalho, para indicar tanto dispositivos físicos, quanto unidades virtuais e lógicas que são capazes de capturar dados brutos do ambiente. Esta definição é bastante ampla, pois engloba as mais diversas fontes de dados do ambiente. Além disso, está de acordo com a classificação dos sensores, quanto a forma de obtenção dos dados, feita por Baldauf et al. [2007] e apresentada a seguir:

- Sensores Físicos - São hardware capazes de capturar praticamente qualquer dado do meio físico (e.g., sensores de luminosidade, movimento, presença, aceleração, toque, temperatura, pressão, câmeras, microfones, GPS). Com o barateamento destes sensores, os dispositivos móveis tornam-se, cada vez mais, capazes de obter informações confiáveis que caracterizem a situação do ambiente, bem como do usuário.

- Sensores Virtuais - São software capazes de capturar e/ou prover informações a partir de aplicações e outros serviços de software. Exemplos de informações contextuais que podem ser obtidas a partir de sensores virtuais: o que o usuário digita, as aplicações que ele mais utiliza, o tempo que ele leva e a frequência com a qual determinada atividade é realizada.

- Sensores Lógicos - Estes sensores fazem uso de diferentes fontes de dados, combinando dados de sensores físicos e virtuais com informações adicionais em bases de dados ou outras fontes, a fim de coletar dados contextuais de complexidade maior. Por exemplo, um sensor lógico pode ser construído para detectar a localização do empregado na empresa, através da análise de logins em computadores desktops, que estão associados a uma localização física em uma base de dados de dispositivos da empresa.

2.1.3 Ambiente

A partir da definição de contexto proposta por Dey & Abowd [1999] e apresentada na seção 2.1.1, o termo "Ambiente" será usado neste trabalho para designar qualquer entidade cuja situação deseja-se caracterizar. Desta forma, reescreve-se o conceito de entidade, para que o termo ambiente seja definido da seguinte forma: *o ambiente pode ser uma pessoa, um lugar (ambiente físico) ou objeto que é considerado relevante para a interação entre usuário e uma aplicação, incluindo o próprio usuário e a aplicação.*

Os ambientes possuem "configurações e estados" [Chen & Kotz, 2000] que caracterizam certos contextos. Entretanto, os sensores disponíveis atualmente são limitados, o que impossibilita a captura de todas as variáveis (configurações e estados) que caracterizam o ambiente. Por este motivo, o termo "ambiente observado" será empregado para denotar a representação computacional do ambiente real, feita por um sistema sensível ao contexto. Assim, o termo poderá ser usado tanto para uma coleção de dados brutos obtida por sensores, como para um conjunto de informações contextuais.

Quanto maior o nível de abstração, menor a quantidade de dados a serem armazenados, e menor será a quantidade de informações disponíveis a respeito do ambiente.

Os dados brutos são como uma foto do ambiente real, eles representam o ambiente com a maior precisão possível, embora não sejam capazes de representar completamente o ambiente. As informações contextuais de baixo nível são conjuntos de características extraídas dessa representação, na analogia da foto, em vez da foto propriamente dita, teríamos uma lista de características extraídas da foto (e.g., "existe 1 objeto na foto", "este objeto é preto", "possui forma retangular", "medindo 40x30cm", "possui um botão no canto inferior" e "uma base em formato circular"). As informações contextuais de alto nível são produzidas com base nas características extraídas da "foto" do ambiente, portanto, neste nível de abstração, não se tem a riqueza de detalhes da foto, nem a lista de características, mas o ambiente é representado apenas por um título (e.g., "Monitor LCD 17"").

Existem sistemas sensíveis ao contexto voltados para os mais diversos domínios. Em alguns destes domínios o ambiente pode ser entendido como estático. Não porque existe um conjunto predefinido de características a serem extraídas do ambiente, mas porque os dados de entrada seguem um padrão que não muda significativamente com o tempo. Por exemplo, no ambiente de uma fábrica de monitores, um sistemas que identifica monitores a partir de fotografias utiliza como entrada fotos de monitores tiradas a partir de ângulos, distâncias e luminosidade parecidas, havendo portanto um controle nas condições do ambiente físico a fim de que não ocorram mudanças significativas de uma foto pra outra, o que afetaria o resultado final.

Os ambientes dos sistemas sensíveis ao contexto voltados para aplicações móveis são naturalmente dinâmicos. Isto decorre do fato de que estão inseridos no mundo real, sem qualquer controle sobre as condições do ambiente. Mesmo que um usuário mantenha uma rotina fixa, as condições do ambiente podem mudar com o tempo. Desta forma, estes sistemas não devem se ater a um modelo estático durante o ciclo de vida do contexto, mas é importante aprender constantemente com o ambiente. Usuários diferentes possuem comportamentos, costumes e hábitos diferentes (e que podem mudar com o tempo), configurando, portanto, ambientes diferentes. Desta forma, os dados coletados, os contextos inferidos, as decisões e ações produzidas serão diferentes.

2.2 Sistemas Sensíveis ao Contexto

Um sistema é sensível ao contexto se utiliza o contexto para prover informações relevantes e/ou serviços para os usuários. Informações são relevantes a depender das atividades do usuário [Abowd et al., 1996]. O termo Sistema Sensível ao Contexto (SSC) pode ser empregado nas mais diversas aplicações da computação ubíqua, englo-

bando áreas como ambientes inteligentes, internet das coisas (Internet of things - IoT) e computação móvel sensível ao contexto.

Nesta seção, serão apresentadas as principais formas de modelagem de contexto, técnicas de inferência de contexto e arquiteturas, empregadas nas diversas abordagens de SSC's disponíveis na literatura. Estes foram os principais aspectos analisados durante a revisão bibliográfica das abordagens de SSC para dispositivos móveis que serão discutidas no capítulo 3 de Trabalhos Correlatos.

2.2.1 Modelagem de Contexto

A Modelagem de Contexto (*Context Modelling*) é uma das fases de grande importância do ciclo de vida dos contextos, pois a depender da forma como são representados os dados, diferentes técnicas de inferência poderão ser aplicadas. Existem diferentes formas de se modelar contexto, que são utilizadas tendo em vista características como: estrutura e complexidade de representação; facilidade de gerenciamento; flexibilidade de uso (em que domínios poderá ser aplicada?); formas de validação dentre outras características.

A seguir serão apresentadas as principais formas de modelagem de contexto usadas em sistemas sensíveis ao contexto identificadas por Chen & Kotz [2000], e incrementadas e discutidas por Strang & Linnhoff-Popien [2004]; Bettini et al. [2010]; Perera et al. [2013]:

- Modelo *Key-Value* - Os contextos são modelados como um par de elementos: a primeira parte é a chave (*key*), referente ao contexto de interesse, e a segunda parte é o valor (*value*) atribuído para aquele contexto. Em geral, é uma estrutura que representa os contextos de forma independente (não relacionados) entre si. Embora seja uma estrutura fácil de gerenciar, a modelagem por *key-value* dificulta a implementação de algoritmos de inferência que consideram as relações entre os contextos. A modelagem utilizando *key-value* não é escalável e nem adequada para o armazenamento de estruturas de dados mais complexas.

- Modelo com esquema de Marcações (*Markup*) - Todos os modelos baseados em marcações utilizam estruturas de dados hierárquicas, consistindo em tags de marcações com atributos e conteúdo. O XML e JSON são exemplos de linguagens de marcação. Estas estruturas são geralmente usadas como formato para a organização de dados intermediários (de complexidade média), ou para representar informações contextuais a serem transferidas por uma rede, ou entre componen-

tes do sistema, pois podem se tornar complexas à medida que a quantidade de níveis de informação envolvidos aumenta.

- Modelos gráficos - São modelos que utilizam esquemas gráficos para modelagem dos contextos. Em geral, modelam contextos com relacionamentos. A UML (Linguagem de Modelagem Unificada) é um exemplo de estrutura que pode ser empregada para modelar contextos graficamente. Diferentes representações de baixo nível podem ser derivadas a partir de modelos gráficos, dentre as quais destacam-se os bancos de dados (e.g., SQL), que podem armazenar e prover acesso rápido à grandes quantidades de dados. Entretanto, podem exigir requisições e configurações cada vez mais complexas a medida que a quantidade de dados e de relacionamentos aumenta.

- Modelos orientados a Objetos - Modelar contexto usando técnicas orientadas a objetos possibilita o uso de todo o poder da orientação a objetos (e.g.,, herança, encapsulamento, reusabilidade, modelagem de relacionamentos). Como a maioria das linguagens de programação de alto nível suportam orientação a objetos, os modelos orientados a objetos podem facilmente ser integrados aos sistemas sensíveis ao contexto.

- Modelos baseados em Lógica - São modelos que utilizam fatos, expressões e regras para representar informações sobre o contexto. Segundo Strang & Linnhoff-Popien [2004], uma lógica define as condições sob as quais uma expressão conclusiva, ou fato, podem ser derivadas a partir de um conjunto de outras expressões ou fatos. Segundo Perera et al. [2013], modelos baseados em lógica podem ser usados para gerar novos contextos de alto nível a partir de contextos de baixo nível, muitas vezes produzindo novos conhecimentos. Além disso, podem modelar eventos e ações e definir condições e restrições. A modelagem por lógica não segue padronizações e é dependente das aplicações, por este motivo, é muito usada de forma suplementar a outras modelagens.

- Modelos baseados em Ontologias - Ontologias representam uma descrição de conceitos e relacionamentos de algum domínio de interesse [Baldauf et al., 2007]. As ontologias são um instrumento promissor para a modelagem de informações contextuais, devido à sua alta, e formal, expressividade e às possibilidades de aplicação de ontologias em técnicas de inferência de contexto. Existem diferentes padronizações e linguagens para criação de ontologias (e.g., RDF - *Resource Description Framework*, OWL - *Web Ontology Language*), bem como ferramentas de desenvolvimento e técnicas de inferência. As ontologias devem ser usadas para

modelar conhecimentos de domínio e estruturar contextos com base nos relacionamentos definidos na ontologia, e não para armazenar os dados coletados.

2.2.2 Inferência de Contexto

A Inferência de Contexto (do inglês, "*Context Reasoning*") pode ser definida como um método para dedução de informações novas e relevantes para serem usadas por aplicações e usuários, a partir de dados contextuais de várias fontes [Bikakis et al., 2008]. Embora o significado original do termo em inglês seja mais amplo, em geral, a inferência de contexto consiste em mapear informações contextuais de baixo nível em contextos de alto nível.

Existem duas características dos dados brutos que serviram como motivação para a criação das primeiras técnicas de inferência: Imperfeição (i.e., valores desconhecidos, ambíguos, imprecisos ou errôneos) e Incerteza [Perera et al., 2013]. Ambas decorrem da dificuldade de se medir com precisão as variáveis e variações do ambiente físico, devido às limitações dos sensores físicos (e.g., range de valores que se pode medir; condições ambientais, por exemplo, uma faixa de temperatura e humidade, em que o sensor funciona com um erro controlado; estão suscetíveis a ruídos e interferências). Desta forma, o simples tratamento de dados brutos a fim de se obter valores mais confiáveis e corretos, também pode ser entendido como inferência de contexto, tendo em vista que informações novas e relevantes foram deduzidas a partir dos dados coletados.

Nurmi & Floréen [2004] identificam três etapas que podem ocorrer durante a inferência de contexto (no sentido mais amplo - "*Context Reasoning*"): Pré-processamento, quando os dados brutos coletados pelos sensores são processados com o intuito de tratar imperfeições e incertezas, por exemplo, valores que estão faltando podem ser preenchidos, valores discrepantes são desconsiderados e os dados são validados a partir de outras fontes; Fusão de dados, fase em que são combinados os dados de múltiplos sensores, para produzir informações mais precisas, mais completas e mais dependentes (i.e., que não podem ser obtidas a partir de um único sensor); Inferência de contexto (do inglês, "*Context Inference*"), quando informações contextuais de alto nível são produzidas a partir de informações contextuais de baixo nível, provenientes dos passos anteriores. O foco deste trabalho está nesta terceira etapa, ou seja, na Inferência de Contexto propriamente dita.

Existem diversas técnicas (modelos de decisão) que podem ser aplicadas à inferência de contexto para sistemas sensíveis ao contexto, tais como: Árvores de Decisão, Redes Bayesianas, Redes Neurais, K-vizinhos mais próximos, SVM, Clusters, baseadas em Regras, Lógica Fuzzy, baseadas em ontologias, baseadas em lógica probabilística,

dentre outras. Esses métodos podem ser aplicados tanto de forma isolada, como em combinação uns com os outros. Além disso, muitos destes foram criados e são empregados nas áreas de inteligência artificial e aprendizagem de máquina. Embora exista uma grande quantidade de técnicas, nem todas são adequadas ao domínio da computação móvel.

As técnicas de inferência de contexto foram classificadas em seis categorias por Perera et al. [2013], as quais são apresentadas a seguir:

- Aprendizagem Supervisionada - Nesta categoria de técnicas, é necessário antes que se tenha uma coleção de dados de treinamento. Estes dados serão classificados de acordo com os resultados esperados. A partir daí é derivada uma função que pode produzir os mesmos resultados usando os dados de treinamento. No caso de aprendizagem online, é necessário que o usuário rotule os dados sempre que o sistema errar, a fim de treinar o classificador. São exemplos de técnicas que podem ser incluídas nesta categoria: Árvores de decisão, Redes Bayesianas, Redes Neurais Artificiais, Máquinas de Vetores de Suporte (SVM).

- Aprendizagem Não Supervisionada - Esta categoria de técnicas pode encontrar estruturas escondidas (i.e., padrões) em dados não classificados. Neste caso, não há coleção de treinamento, portanto não há erro ou sinal de recompensa para avaliar uma possível solução. Dentre as principais técnicas desta categoria, as que se destacam na inferência de contexto são: a técnica de cluster "*K-Nearest Neighbour*"; e a técnica de redes neurais não supervisionadas "*Kohonen Self-Organizing Map*".

- Regras - Esta é a categoria de técnicas mais simples e direta dentre todas as categorias de técnicas de inferência de contexto apresentadas neste trabalho. Em geral, regras são estruturas que seguem o formato SE-ENTÃO-SENÃO. A utilização de regras proporciona a geração de informações contextuais de alto nível a partir de contextos de baixo nível. A maioria das preferências dos usuários pode ser facilmente codificada em regras. Regras também podem ser usadas na detecção de eventos (que também representam contextos). Ultimamente esse método vem sendo usado intensivamente combinado com técnicas de inferência de contexto baseadas em ontologia.

- Lógica Fuzzy - Permitem inferências aproximadas em vez de inferências rígidas e fixas. Na lógica tradicional, os valores lógicos aceitáveis são 0 (falso) e 1 (verdadeiro), na lógica Fuzzy não há essa rigidez, valores lógicos parciais também

são aceitáveis. A lógica Fuzzy permite que alguns cenários do mundo real sejam representados de forma mais natural, ou seja, é possível ter noções que nem sempre são precisas como: Alto, baixo, escuro, claro, quente, frio, etc. Normalmente esta técnica não é usada sozinha na inferência de contexto, mas sim como complemento para outras técnicas.

- Baseadas em Ontologia - Como o nome sugere, esta categoria de técnicas toma como base o uso das ontologias para a inferência de contexto. São baseadas em descrição lógica, que compreende uma família de representações de formalismos de conhecimentos baseados em lógica. Uma vantagem de se usar inferência baseada em ontologia é que esta se integra bem com a modelagem de contexto que utiliza ontologias, entretanto esta técnica requer a utilização de regras para que seja capaz de identificar valores que estão faltando ou informações ambíguas.

- Lógica Probabilística - Esta categoria de técnicas possibilita que decisões sejam tomadas a partir de probabilidades ligadas aos fatos que estão relacionados ao problema (contexto). Podem ser usadas para combinar dados de sensores provenientes de duas fontes diferentes, além disso, podem ser usadas para identificar as resoluções de conflitos entre contextos. Quase sempre elas são usadas para entender a ocorrência de eventos, pois é possível combinar diferentes evidências para calcular a probabilidade de um evento ocorrer. São exemplos de técnicas que podem ser incluídas nesta categoria: Dempster-Shafer, usado na detecção de eventos, e *Hidden Markov Models*, usado no reconhecimento de atividades em casas inteligentes.

Além destas, também é possível citar uma sétima categoria: a das técnicas que utilizam aprendizagem por reforço. A classificação feita por Perera et al. [2013] não inclui técnicas que utilizam aprendizagem por reforço, provavelmente por não existirem, até então, trabalhos específicos que detalhassem como ocorreria a aprendizagem por reforço na abordagem de SSC proposta. Essas técnicas ainda são pouco utilizadas em SSC's, embora alguns trabalhos comentem, sem entrar em detalhes, sobre a possibilidade de utilizar os reforços para modificar os modelos usados na inferência. As técnicas que utilizam aprendizagem por reforço realizam a inferência de contexto (com base ou não em um modelo) e, após receberem um sinal de reforço, se adaptam para que cada vez mais os reforços recebidos sejam os maiores possíveis.

Enfim, existe uma variedade de técnicas que podem ser aplicadas nas mais diversas aplicações de inferência de contexto, e até mesmo em outros tipos de aplicações. No domínio das aplicações móveis sensíveis ao contexto, uma abordagem de aprendiza-

gem supervisionada talvez não seja interessante por requerer uma coleção de dados de treinamento, ou por exigir que o usuário identifique em qual contexto ele está sempre que o sistema errar. Uma abordagem não supervisionada pode aprender a identificar padrões de comportamentos do usuário, entretanto, pode não ser capaz de informar o que estes padrões representam, exigindo que o usuário rotule tais padrões. Utilizar regras na inferência de contextos de alto nível torna estático o comportamento do sistema, e por não haver aprendizagem, o sistema torna-se menos adaptável e sensível apenas a contextos específicos. A lógica Fuzzy aplicada sozinha em inferência de contexto não permite inferência de contextos de alto nível que utilizam dados de diversas fontes. A utilização de métodos baseados em ontologias é fortemente indicada para cenários menos dinâmicos, nos quais há a possibilidade de se modelar previamente o ambiente (e.g., casas inteligentes), no domínio das aplicações móveis existem contexto que podem ser modelados por ontologias, entretanto também existem contextos que ocorrem de formas diferentes para ambientes físicos e usuários diferentes e que ainda podem mudar com o passar do tempo. De forma similar, os métodos probabilísticos são mais indicados para situações onde as probabilidades são previamente conhecidas ou podem ser facilmente obtidas a partir do ambiente, no caso de aplicações móveis estas probabilidades podem variar de usuário para usuário. Por fim, a utilização de aprendizagem por reforço pode ser particularmente interessante em computação móvel, por permitir um ajuste maior à realidade do usuário, e uma maior adequabilidade às mudanças do ambiente (dinâmico).

No capítulo 3 de Trabalho Correlatos serão apresentadas as abordagens de SSC voltadas para dispositivos móveis, bem como peculiaridades dos métodos usados na inferência. As abordagens analisadas serão agrupadas quanto as técnicas de inferência, permitindo uma noção das categorias de técnicas mais usadas na inferência para dispositivos móveis.

2.2.3 Principais Arquiteturas

Tanto no domínio das aplicações móveis como no de ambientes inteligentes, o objetivo de se ter sensibilidade ao contexto é prover serviços e realizar tarefas de forma proativa (i.e., sem que alguém tenha que solicitar) e adequada ao contexto do usuário. Portanto, nestes domínios, tem-se como foco o usuário. Desta forma, são as ações, comportamentos e atividades que o usuário realiza no ambiente físico ou virtual que irão afetar na forma como os serviços serão prestados em um SSC.

Na seção 2.1, foi apresentada uma definição de Computação Sensível ao Contexto e as etapas envolvidas em CSC. Esses conceitos, bem como, a analogia apresentada,

permitem que se tenha uma visão mais ampla do que a maioria das abordagens de SSC realizam na prática. Assim, embora existam peculiaridades nas diferentes arquiteturas de SSC, a partir da apresentação e discussão dos principais elementos da Computação Sensível ao Contexto (i.e., sensoriamento, inferência, decisão e ação) é possível compreender melhor o funcionamento destes sistemas. Como as abordagens de SSC apresentam soluções distintas para prover sensibilidade ao contexto, é natural que utilizem arquiteturas diferentes, tendo em vista as especificidades dos problemas (e.g., cenários da aplicação, ambientes e domínios de interesse).

Com o passar dos anos diferentes categorias de arquitetura foram sendo propostas e identificadas. Chen & Kotz [2000] apresenta três abordagens diferentes, tendo em vista a forma que as aplicações obtém as informações contextuais: acessando diretamente aos sensores; utilizando um *middleware* com arquitetura centralizada, acessando um servidor de contextos; utilizando um *middleware* com arquitetura distribuída, implementado nos diferentes dispositivos do sistema. Chen & Kotz [2000] defende a separação entre aplicação e sensoriamento por meio de um *middleware* devido à dificuldade no desenvolvimento de aplicações ao utilizar sensores de diferentes marcas, tipos e especificações. Baldauf et al. [2007] diz que a separação entre a detecção e o uso de contexto é necessária para prover e aprimorar a extensibilidade e reusabilidade dos SSC's.

O domínio das aplicações móveis suporta tanto abordagens distribuídas como centralizadas. Entretanto, as abordagens centralizadas são pouco escaláveis (i.e., não suportam um número elevado de dispositivos), ainda assim elas se tornam interessantes quando há necessidade de se poupar recursos dos dispositivos móveis (e.g., memória, bateria, processamento). À medida que as limitações de hardware dos dispositivos móveis diminuem, a utilização de arquiteturas distribuídas torna-se cada vez mais viável e justificável.

Perera et al. [2013] analisou as principais características dos SSC's (identificadas nas abordagens estudadas), classificando-os em seis categorias não exclusivas (i.e., uma abordagem pode ser classificada em mais de uma categoria):

- Arquiteturas Centralizadas - Não permitem a comunicação entre diferentes dispositivos. Abstraem tarefas de baixo nível e permitem que aplicações acessem informações contextuais a partir de um software centralizado. Atuam como uma pilha de camadas (e.g., *middlewares*), permitindo que as aplicações sejam desenvolvidas no topo dela.

- Arquiteturas Distribuídas - São muito usadas no domínio das aplicações móveis. Os dispositivos executam operações localmente, de forma distribuída, e quando

há comunicação entre dispositivos, esta é feita sem a necessidade de um centralizador.

- Baseadas em Componentes - O sistema é composto por componentes principais, fracamente conectados, que interagem entre si.
- Cliente-Servidor - Separa o sensoriamento (obtenção dos dados) do processamento (e.g., inferência de contexto). Clientes coletam dados, Servidores processam e retornam os contextos identificados aos clientes.
- Baseadas em Serviços - A solução é obtida através da utilização de serviços. O sistema é resultado de um conjunto de serviços trabalhando em conjunto.
- Baseadas em Nós - Permite a implantação de "pedaços" de software, com recursos novos ou parecidos, que se comunicam e processam coletivamente dados em redes de sensores.

Estas categorias de arquiteturas não foram identificadas considerando apenas abordagens voltadas para aplicações móveis, mas consideram muitos outros trabalhos de caráter genérico, além de outros domínios de aplicação. Ainda assim, estas categorias foram suficientes para a classificação dos trabalhos estudados nesta dissertação.

2.3 Aprendizagem de Máquina

A Aprendizagem de Máquina é uma sub área da Inteligência Artificial que se preocupa com a questão de como construir programas de computadores que melhorem automaticamente com a experiência. De forma mais precisa, Mitchell [1997] afirma que um programa de computador aprende mediante uma experiência E, com relação a uma classe de tarefas T e uma medida de desempenho P, se o desempenho (medido por P) para as tarefas em T melhora com a experiência E. Portanto, para se ter um problema bem definido de aprendizagem de máquina é necessário identificar essas três características: a classe de tarefas a serem aprendidas; a medida de desempenho a ser melhorada; e a fonte das experiências a partir da qual as tarefas serão aprendidas.

Assim, é possível mapear a Inferência de Contexto em um problema de Aprendizagem de Máquina através da identificação das características levantadas por Mitchell [1997]: a tarefa T consiste em Inferir Contextos; uma medida de desempenho P a ser melhorada pode ser a porcentagem de contextos inferidos corretamente; uma fonte de experiências E pode ser o mundo real ou uma base de treinamento com dados extraídos do mundo real em contextos específicos.

Nas subseções seguintes serão apresentados alguns conceitos e definições da área de aprendizagem de máquina (subseção 2.3.1) que serão úteis para o entendimento da técnica de inferência de contexto proposta neste trabalho (apresentada no capítulo 4), bem como na compreensão dos trabalhos correlatos que utilizam técnicas de aprendizagem de máquina (apresentados no capítulo 3). Não é objetivo deste trabalho apresentar uma revisão detalhada de todas as técnicas de aprendizagem propostas na literatura. Entretanto, será feita uma breve descrição das técnicas empregadas na inferência de contexto para Sistemas Sensíveis ao Contexto (subseção 2.3.2).

2.3.1 Definições Básicas

Mitchell [1997] afirma que os sistemas que aprendem tem como objetivo determinar uma descrição de um dado conceito a partir de conhecimentos prévios (dedução) e de um conjunto de exemplos (deste conceito) fornecidos por um professor (indução).

A indução é a forma de inferência lógica que permite obter conclusões genéricas sobre um conjunto particular de exemplos [Rezende, 2003].Ou seja, na indução observa-se um conjunto de exemplos e a partir desta observação cria-se um conceito que generaliza todos os elementos do conjunto. Entretanto, uma limitação da indução é que podem existir elementos da mesma classe conceitual que não são englobados na generalização.

A dedução difere da indução pois parte de conceitos tidos como verdadeiros, expressos na forma de predicados e sentenças, e a partir das relações causais entre conceitos diferentes consegue-se deduzir novos conceitos. Portanto, se os conceitos usados forem sempre verdadeiros então os conceitos produzidos também serão sempre verdadeiros. Não havendo a possibilidade de que elementos pertencentes a mesma classe conceitual não sejam englobados pelo conceito produzido.

Embora o processo dedutivo dê garantias de que os conceitos produzidos serão corretos, ele também requer que as premissas usadas sejam corretas. Isto restringe a aplicação do processo dedutivo, pois nem sempre é possível ter certeza que os conceitos usados são verdadeiros. Portanto, justifica-se a afirmação, feita por Vallim [2009], de que o foco de pesquisa da aprendizagem de máquina é induzir modelos (e.g., regras, padrões) de forma automática a partir de um conjunto de exemplos.

Em geral, o objetivo dos algoritmos e técnicas indutivas é extrair um bom classificador, tendo em vista as medidas de desempenho escolhidas, a partir de um conjunto de exemplos. Um *classificador* é capaz de predizer a possível classe (i.e., classificar) de um novo exemplo não rotulado. Um *exemplo* (i.e., um caso, registro ou dado) é uma tupla ou vetor de valores de atributos que descreve o objeto de interesse, que neste

trabalho será o ambiente ou contexto do usuário. Um *atributo* representa um aspecto ou uma característica do exemplo. Desta forma, um exemplo pode ser visto como um vetor de características contendo informações contextuais de baixo nível do ambiente. Assim, as técnicas de inferência de contexto se assemelham aos classificadores induzidos via aprendizagem de máquina, visto que também rotulam (classificam) os vetores de características (exemplos) extraídos do ambiente.

Um vetor de características pode pertencer ou não a uma dada classe, além disso, o vetor poderá ser classificado como pertencente ou não àquela classe. Desta forma, podem ocorrer quatro combinações de resultados: V_P verdadeiros positivos - a classificação retornou que o vetor pertence à classe C e estava correto; F_P falsos positivos - a classificação retornou que o vetor pertence à classe C e estava errado; V_N verdadeiros negativos - a classificação retornou que o vetor não pertence à classe C e estava correto; F_N falsos negativos - a classificação retornou que o vetor não pertence à classe C e estava errado. Com base nestas combinações de resultados, são apresentadas, a seguir, as principais medidas de desempenho usadas na avaliação de algoritmos de aprendizagem de máquina.

- Taxa de Erro - avalia a porcentagem de erros do classificador. Representa a proporção de erros com relação ao total de inferências. Exemplo: "o classificador errou em X% das classificações".

$$erro = \frac{F_P + F_N}{V_P + F_P + V_N + F_N} \tag{2.1}$$

- Acurácia Total - avalia a porcentagem de acertos do classificador. Representa a proporção de acertos com relação ao total de inferências. Exemplo: "o classificador acertou em X% das classificações"

$$acur = 1 - err = \frac{V_P + V_N}{V_P + F_P + V_N + F_N} \tag{2.2}$$

- Completude - avalia se todos os exemplos, pertencentes a alguma classe, são classificados (exemplos sem classe não devem ser classificados). Representa o inverso da taxa F_N de falsos negativos. Exemplo: "o classificador conseguiu classificar X% dos exemplos".

$$comp = 1 - \frac{F_N}{V_P + F_P + V_N + F_N} = \frac{V_P + F_P + V_N}{V_P + F_P + V_N + F_N} \tag{2.3}$$

- Consistência - avalia se os exemplos classificados foram classificados corretamente.

Representa o inverso da taxa F_P de falsos positivos. Exemplo: "o classificador não errou na classificação de X% dos exemplos".

$$cons = 1 - \frac{F_P}{V_P + F_P + V_N + F_N} = \frac{V_P + V_N + F_N}{V_P + F_P + V_N + F_N} \tag{2.4}$$

- Valor Preditivo Positivo (Precisão, ou confiabilidade positiva)- avalia a porcentagem de classificações corretas V_P dentre as que retornaram algum resultado positivo ($V_P + F_P$). Exemplo: "o classificador acertou X% de todas as vezes que retornou uma classe para algum exemplo".

$$prec = \frac{V_P}{V_P + F_P} \tag{2.5}$$

- Valor Preditivo Negativo (Confiabilidade negativa)- avalia a porcentagem de acertos (V_N) dentre as classificações que retornaram que o exemplo não pertencia a nenhum contexto ($V_N + F_N$). Exemplo: "o classificador não retornou uma classe corretamente em X% de todas as vezes que não retornou uma classe para algum exemplo".

$$vpn = \frac{V_N}{V_N + F_N} \tag{2.6}$$

Existem três categorias principais de técnicas indutivas de aprendizagem de máquina, que são diferenciadas pelas informações que utilizam pra fazer a indução:

- Aprendizagem Supervisionada - induz modelos a partir de exemplos rotulados. Um professor externo fornece os exemplos de entradas com as respectivas classificações. Após induzir um modelo sobre os exemplos rotulados (exemplos de treinamento), o classificador é avaliado com exemplos não rotulados (exemplos de teste), caso a medida de desempenho esteja em um nível insatisfatório, repete-se o processo de indução com novos exemplos (experiências). Na inferência de contexto, os dados de treinamento seriam vetores de características com seus contextos já identificados.

- Aprendizagem Não Supervisionada - identifica padrões em dados não rotulados. Não existe um professor pra indicar quais os resultados esperados para as entradas. Portanto, neste cenário, as técnicas buscam padrões e agrupam as entradas de acordo com as suas propriedades. Na inferência de contexto, o sistema observa um conjunto de vetores de características coletados do ambiente e faz o agrupamento a partir das propriedades e padrões encontrados, os novos vetores de características vão sendo classificados com base nos agrupamentos já feitos.

Neste caso não há retorno do ambiente ou de um professor indicando se os padrões encontrados são adequados ou não para a classificação.

- Aprendizagem por Reforço - considera reforços na indução de modelos. Neste caso, é como se existisse um professor indicando o quão apropriadas as classificações foram para os vetores de características (as entradas do ambiente). Ou seja, o sistema realiza a inferência de contexto e depois atualiza o modelo usado na inferência a partir dos sinais de reforço oriundos do ambiente.

Além disso, também é possível diferenciar as técnicas de aprendizagem com base no modo de aprendizado. Desta forma, existem abordagens que aprendem de forma incremental e as que aprendem de modo não incremental. As abordagens *não incrementais* (modo *batch*) precisam que todo o conjunto de treinamento esteja presente para o aprendizado, enquanto que as abordagens *incrementais* atualizam a hipótese antiga sempre que novos exemplos são adicionados ao conjunto de treinamento [Russel & Norvig, 1995], ou seja, não precisam reconstruir a hipótese desde o início.

Segundo Rezende [2003], as técnicas de aprendizagem de máquina seguem diferentes paradigmas de aprendizado: *Simbólico*, constroem representações simbólicas (e.g., regras, árvores de decisão) de um conceito através da análise de exemplos e contra-exemplos deste conceito; *Estatístico*, utilizam modelos estatísticos para encontrar uma boa aproximação do conceito induzido (e.g., Redes Bayesianas, Redes Neurais); *Baseado em Exemplos*, classificam exemplos nunca vistos a partir de exemplos similares conhecidos (e.g., Vizinhos mais próximos ou *Nearest Neighbours*, Raciocínio Baseado em Casos); *Conexionista*, envolve unidades altamente interconectadas, que são construções matemáticas simplificadas inspiradas no modelo biológico do sistema nervoso (e.g., Redes Neurais); *Genético*, existe uma população de elementos de classificação que competem para fazer a predição (da classe), onde elementos com as melhores perfomances se proliferam (produzindo variações deles mesmos) enquanto os mais fracos são descartados.

2.3.2 Técnicas Aplicadas em Inferência de Contexto

Esta seção tem como propósito apresentar uma visão geral dos principais algoritmos e técnicas de aprendizagem de máquina empregadas em inferência de contexto para SSC's. Tendo em vista as peculiaridades do domínio das aplicações móveis sensíveis ao contexto, nem todas as técnicas apresentadas nesta seção são adequadas ou viáveis de serem aplicadas na computação móvel. Contudo, é importante ter uma noção do

funcionamento destas técnicas, a fim de que seja possível fazer comparações entre técnicas distintas ou similares.

Árvores de Decisão

São técnicas de aprendizagem supervisionada que constroem uma árvore a partir de um conjunto de dados. A árvore construida pode ser usada para classificar novos exemplos. Segundo Russel & Norvig [1995]; Kotsiantis [2007], São árvores que classificam instâncias ordenando-as com base nos valores das características (atributos). Cada nó em uma árvore de decisão representa uma característica e uma instância a ser classificada, e cada ramo representa um valor que o nó pode assumir. Instâncias são classificadas a partir da raiz e são ordenadas pelos valores das características.

A figura 2.2 apresenta um exemplo de uma árvore de decisão, que pode ser extraída do conjunto de exemplos da tabela 2.1, que classifica as entradas como pertencentes e não pertencentes a classe do contexto Dirigindo. O problema de construção de uma árvore de decisão binária ótima é um problema NP-Completo. Por este motivo os pesquisadores buscam heurísticas para construir árvores de decisão de forma quase ótima.

Tabela 2.1: Exemplo de Base de Treinamento

Características			Contexto
Local	**Velocidade**	**Som**	**Dirigindo?**
CPD	Nula	Ruído	sim
CPD	Baixa	Baixo	sim
CPD	Baixa	Alto	não
CPD	Baixa	Ruído	sim
CPD	Alta	Baixo	não
CPD	Média	Alto	não
Casa	Nula	Ruído	não
Shopping	Baixa	Baixo	não
Outro	Alta	Baixo	não

Redes Bayesianas

São métodos que utilizam um modelo gráfico para relacionamentos probabilísticos sobre um conjunto de variáveis (atributos). A estrutura de uma Rede Bayesiana é um grafo dirigido acíclico, e os nós possuem uma correspondência de um para um com as variáveis (features). Os arcos representam influências casuais entre as variáveis, enquanto a ausência de possíveis arcos representa a independência (condicional) entre

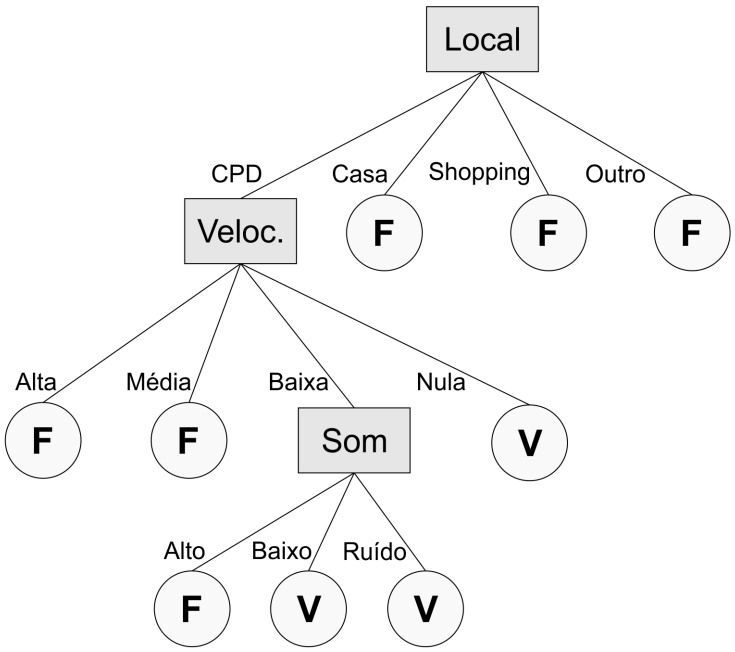


Figura 2.2: Exemplo de Árvore de Decisão.

variáveis [Kotsiantis, 2007]. Redes Bayesianas são comumente usadas na combinação de informações incertas, provenientes de um grande número de fontes, para dedução de informações contextuais de alto nível [Perera et al., 2013].

Em geral, o processo de aprender uma Rede Bayesiana requer duas etapas: 1) Aprender a estrutura do grafo que representará a rede; 2) Determinar os parâmetros da rede (probabilidades condicionais). Segundo Mitchell [1997], uma das principais limitações das Redes Bayesianas é que, dado um problema descrito por N atributos (sem uma estrutura definida), o número de possíveis estruturas (hipóteses) é mais que exponencial em N. Por este motivo, em alguns casos, é mais interessante fazer um balanceamento aproximado da rede, em vez de buscar o balanceamento ótimo de probabilidades.

As Redes Bayesianas ingênuas são Redes Bayesianas onde o grafo (dirigido e acíclico) possui apenas um único pai, representando o nó não observado, e vários filhos, correspondendo aos nós observados. Nestas redes há uma suposição forte de que as variáveis são condicionalmente independentes [Mayrhofer, 2004].

Redes Neurais Artificiais

São técnicas que tentam imitar o sistema biológico dos neurônios através de modelos matemáticos que funcionam de forma análoga ao cérebro. Os elementos aritméticos

básicos de computação correspondem aos neurônios (i.e., as células que realizam o processamento de informações no cérebro), enquanto que a rede neural como um todo corresponde a um conjunto de neurônios interconectados [Russel & Norvig, 1995].

Uma rede neural multicamadas (ilustrada na figura 2.3) consiste em um grande conjunto de unidades (neurônios) conectadas entre si, seguindo um padrão de conexão. As unidades de uma rede neural são, normalmente, classificadas em três tipos, segundo o modelo feed-forward:

- Unidades de entrada - recebem as informações a serem processadas;
- Unidades de saída - onde são encontrados os resultados do processamento;
- Unidades escondidas - localizadas entre as unidades de entrada e de saída, adicionam pesos às entradas a fim de produzir os resultados da saída (esses pesos são ajustados durante a fase de treinamento do classificador);

A rede é treinada sobre um conjunto de pares de dados, para determinar o mapeamento de entrada e saída. Os pesos das ligações entre neurônios são então fixados, e a rede é utilizada para determinar as classificações de um novo conjunto de dados [Kotsiantis, 2007]. São empregadas tipicamente para modelar relacionamentos complexos entre entradas e saídas (aprendizagem supervisionada) ou para encontrar padrões em dados (aprendizagem não supervisionada)[Perera et al., 2013].

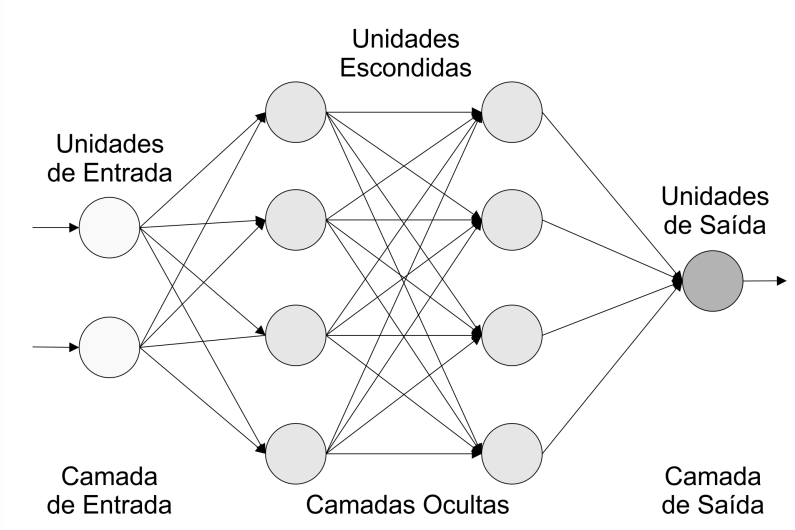


Figura 2.3: Exemplo de Rede Neural. (Adaptado de Russel & Norvig [1995])

Máquinas de Vetores de Suporte (SVM)

A lógica por trás de um SVM gira em torno do conceito de "margem" (ambos os lados de um hiperplano que separam duas classes). Busca-se maximizar a margem, criando a maior distância possível entre o hiperplano separador e as instâncias em ambos os lados [Kotsiantis, 2007]. Os pontos (de dados) da margem, que ficam localizados próximos à outra classe, formam o chamado vetor de suporte [Russel & Norvig, 1995]. Mudando estes pontos, muda-se também o hiperplano e assim, a solução do treinamento. Portanto, o principal objetivo dos SVM's simples é encontrar o hiperplano que melhor divide um conjunto de dados (rotulados) de duas classes diferentes.

As máquinas de vetores de suporte são técnicas de aprendizagem supervisionada que foram desenvolvidas inicialmente para fazer a classificação entre duas classes apenas, mas existem variações que realizam classificação em múltiplas classes através da combinação de vários SVM's simples (de duas classes). Segundo Perera et al. [2013], os SVM's são amplamente usadas em reconhecimento de padrões em computação sensível ao contexto.

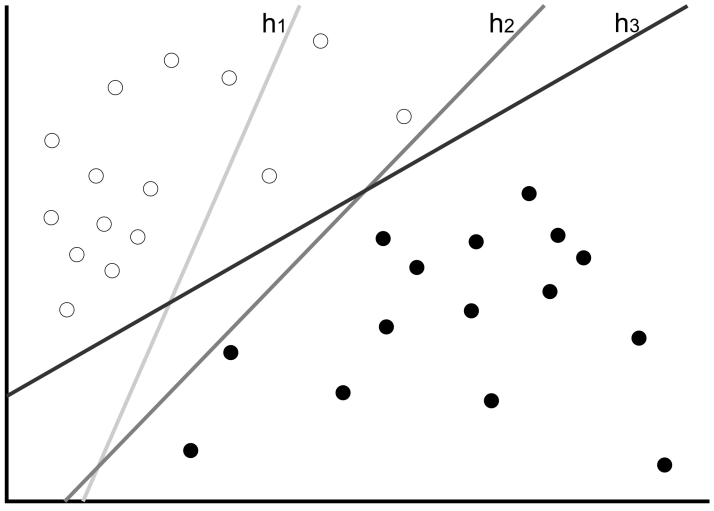


Figura 2.4: Hiperplanos dividindo dois tipos de amostras

A figura 2.4 apresenta três hiperplanos que dividem dois tipos de amostras. O hiperplano da reta h_1 não consegue separar completamente todas as amostras das duas classes (representadas na figura por círculos preenchidos e não preenchidos). Embora o hiperplano da reta h_2 separe completamente as amostras, ele não maximiza a margem, como ocorre no hiperplano da reta h_3. Na figura 2.5 são apresentados os vetores de suporte das amostras da figura 2.4 que podem ser usados para produzir o hiperplano h_3 com margem máxima.

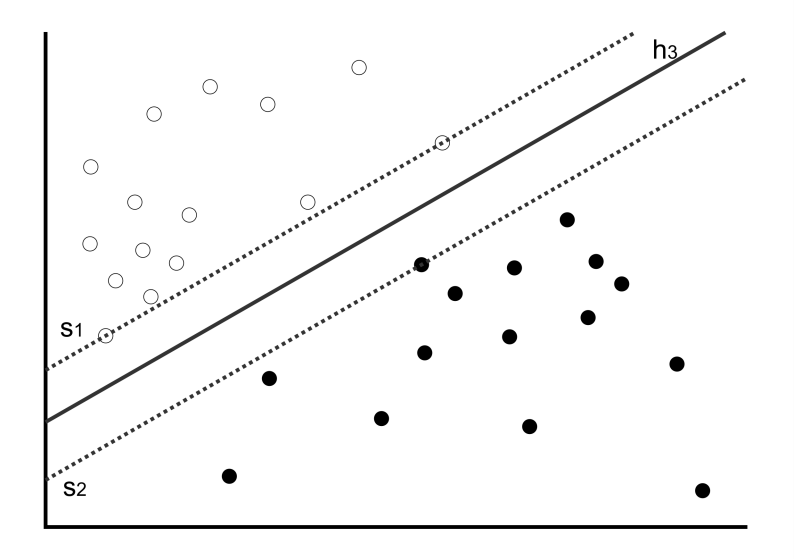


Figura 2.5: Vetores de suporte e hiperplano produzido por um SVM.

Clusters

Consiste em um conjunto de técnicas, cujo objetivo principal é dividir um conjunto de dados em diferentes grupos de objetos similares. Em geral, este conjunto de dados não possui rótulos, portanto, diferentemente dos SVM's, os agrupamentos serão feitos a partir da análise dos atributos dos exemplos de entrada. Ou seja, cada agrupamento é composto por objetos que são similares entre si (possuem as mesmas características) e que são diferentes de objetos de outros grupos.

De acordo com as técnicas aplicadas, o resultado pode ser de diferentes tipos [Rai & Singh, 2010]:

- Clusters exclusivos - um dado pertence apenas a um cluster;
- Clusters sobrepostos - um dado pertence a um ou mais clusters;
- Clusters probabilísticos - um dado pode pertencer a qualquer cluster, com uma determinada probabilidade.

K-means

É uma das técnicas de clusters exclusivos. No *K-means*, os clusters são representados pelos seus respectivos centroides, que são as médias dos pontos presentes no cluster. Segundo Kanungo et al. [2002], o objetivo do *k-means* é minimizar a distância média quadrática entre cada dado e seu respectivo centróide, tendo em vista um conjunto

de n elementos com a atributos e um inteiro k indicando quantos clusters devem ser encontrados.

Esta técnica é usada apenas com atributos numéricos e sofre influência negativa quando ocorrem outliers (i.e., elementos que estão distantes dos demais) [Rai & Singh, 2010]. Além disso, outra limitação do *K-means* é que nem sempre é possível dizer de antemão o valor K de clusters presentes num conjunto de dados.

K-Vizinhos Mais Próximos (K-NN)

É um método de aprendizagem não paramétrica baseado em instâncias, ou seja, permite que a complexidade da hipótese cresça com os dados. Neste método, supõe-se que as propriedades de qualquer ponto X específico da entrada têm probabilidade de serem semelhantes às propriedades dos pontos na vizinhança de X [Russel & Norvig, 1995].

A maior dificuldade é determinar o que é a vizinhança do ponto X. Caso os limites da vizinhança sejam muito próximos a X, a vizinhança não englobará nenhum outro ponto, caso os limites sejam muito afastados, a vizinhança poderá conter todo o conjunto de dados. Por este motivo, no K-NN determina-se um valor K que representa a quantidade de pontos (mais próximos de X) que serão considerados vizinhos. Desta forma, os limites da vizinhança não serão fixos, mas irão variar de acordo com a densidade de dados ao redor de cada ponto. Por outro lado, o valor de K deve ser grande o suficiente para assegurar uma estimativa significativa.

De modo resumido, o funcionamento do K-NN pode ser explicado de acordo com os seguintes passos:

- cálculo da distância entre a instância atual e as demais que já foram rotuladas;
- identificação das K instâncias mais próximas da atual;
- retorno do rótulo (classe) que mais se repete, a partir dos K vizinhos mais próximos;

Originalmente, Cover & Hart [1967] propuseram o método K-NN considerando a distância Euclidiana como forma de calcular a proximidade entre a instância atual e às demais instâncias rotuladas. Entretanto, a distância euclidiana restringe a aplicabilidade do K-NN à dados onde as características são numéricas. Weinberger et al. [2006] afirma que, idealmente, a métrica de distância para a classificação usando o K-NN deve ser adaptada para o problema particular que está sendo resolvido.

Como a técnica proposta neste trabalho utiliza o K-NN na classificação de contextos, é importante apresentar, neste embasamento teórico, o método de cálculo de distância usado neste trabalho - a distância de Gower [J. C. Gower, 1971].

O método proposto originalmente por J. C. Gower [1971] calcula um coeficiente de similaridade entre vetores, cujo valor varia entre 0 e 1. A similaridade indica numericamente o quão parecidas são duas instâncias entre si, e a distância, por outro lado, indica o afastamento entre elas. A Similaridade de Gower $S(x, y)$ para dois vetores x e y tende a 1 quanto mais similares forem x e y, e tende a 0 quanto mais diferentes forem. Desta forma, a distância de Gower $D(x, y)$ para os vetores x e y é dada pela diferença

$$D(x, y) = 1 - S(x, y) \tag{2.7}$$

Assim, a distância de Gower entre dois vetores será nula quando ambos forem iguais, e será 1 quando as diferenças entre eles forem máximas, implicando em uma distância máxima tendo em vista os ranges de valores de cada característica dos vetores. A similaridade de Gower para os vetores x e y é definida pela seguinte equação:

$$S(x, y) = \frac{\sum_{j=1}^{v} w_j \delta_{x,y}^j s_{x,y}^j}{\sum_{j=1}^{v} w_j \delta_{x,y}^j} \tag{2.8}$$

sendo v o número total de características dos vetores, w_j o peso (representando a relevância) da característica j, $\delta_{x,y}^j$ a indicação da possibilidade (0 ou 1) de se fazer a comparação entre x e y, e $s_{x,y}^j$ a similaridade dos valores das características j de x e y.

No caso de variáveis (características) nominais assimétricas podem ocorrer vetores com valores ausentes para determinadas características, por este motivo, $\delta_{x,y}^j = 0$ quando tanto a característica j de x (x_j) como de y (y_j) estiverem ausentes, e $\delta_{x,y}^j = 1$ quando ou x_j ou y_j estiverem presentes.

De acordo com o tipo da característica j, existem duas formas de se calcular o valor de $s_{x,y}^j$:

- se j for nominal:

$$s_{x,y}^j = \begin{cases} 1 & \text{se } x_j = y_j \\ 0 & \text{se } x_j \neq y_j \end{cases} \tag{2.9}$$

- se j não for nominal (i.e., é numérica):

$$s_{x,y}^j = 1 - \frac{|x_j - y_j|}{r_j} \tag{2.10}$$

Como o range de valores para a j-ésima variável r_j é considerado na equação 2.10, então a parte fracionária acaba sendo um cálculo normalizado da distância entre os valores de x_j e y_j. Como o interesse original estava na similaridade dos vetores,

utilizou-se o complemento dessa distância, para representar a similaridade entre as características j de dois vetores. Assim, o valor de $s^j_{x,y}$ na equação 2.10 varia entre 0 (quando a distância relativa de x_j e y_j é máxima) e 1 (quando a distância relativa de x_j e y_j é nula, ou seja, $x_j = y_j$).

Exemplificando, se $x = \{amor, 3, 5, paz, 0\}$ e $y = \{odio, 1.5, 5, paz, 1\}$, considerando que as características 1 e 4 são nominais, e todas as outras características (i.e., 2,3 e 5) são numéricas com range $r_j = 5$, então é possível calcular a distância de gower da seguinte forma:

$$S(x,y) = \frac{\sum_{j=1}^{5} w_j \delta^j_{x,y} s^j_{x,y}}{\sum_{j=1}^{5} w_j \delta^j_{x,y}}$$

$$S(x,y) = \frac{w_1\delta^1_{x,y}d^1_{x,y} + w_2\delta^2_{x,y}d^2_{x,y} + w_3\delta^3_{x,y}d^3_{x,y} + w_4\delta^4_{x,y}d^4_{x,y} + w_5\delta^5_{x,y}d^5_{x,y}}{w_1\delta^1_{x,y} + w_2\delta^2_{x,y} + w_3\delta^3_{x,y} + w_4\delta^4_{x,y} + w_5\delta^5_{x,y}}$$

$$S(x,y) = \frac{w_1 d^1_{x,y} + w_2 d^2_{x,y} + w_3 d^3_{x,y} + w_4 d^4_{x,y} + w_5 d^5_{x,y}}{w_1 + w_2 + w_3 + w_4 + w_5}$$

$$S(x,y) = \frac{w_1(0) + w_2(1 - \frac{|3-1.5|}{5}) + w_3(1 - \frac{|5-5|}{5}) + w_4(1) + w_5(1 - \frac{|0-1|}{5})}{w_1 + w_2 + w_3 + w_4 + w_5}$$

$$S(x,y) = \frac{0 + w_2(0.7) + w_3(1) + w_4(1) + w_5(0.8)}{w_1 + w_2 + w_3 + w_4 + w_5}$$

Caso todas as características tenham peso 1 então,

$$S(x,y) = \frac{0 + (0.7) + (1) + (1) + (0.8)}{5} = \frac{3.5}{5} = 0.7 \quad D(x,y) = 0.3$$

Caso as características 3 e 4 tivessem peso 5 e as demais tivessem peso 1, então,

$$S(x,y) = \frac{0 + (0.7) + (5) + (5) + (0.8)}{13} = \frac{11.5}{13} \approx 0.885 \quad D(x,y) \approx 0.115$$

Caso as características 1 e 2 tivessem peso 5 e as demais tivessem peso 1, então,

$$S(x,y) = \frac{0 + (3.5) + (1) + (1) + (0.8)}{13} = \frac{6.3}{13} \approx 0.485 \quad D(x,y) \approx 0.315$$

Conforme apresentado neste exemplo, de acordo com os pesos atribuídos às características, é possível aplicar um viés (do inglês, *bias*, pode ser traduzido como uma tendência) no cálculo da distância entre vetores. Desta forma, é possível considerar um conhecimento prévio que indique a relevância das características no cálculo de distância.

Mapa Auto-Organizável de Kohonen (KSOM)

Representa uma classe de algoritmos de redes neurais da categoria de aprendizagem não supervisionada. Uma propriedade central do KSOM é que este método forma uma projeção não linear de um conjunto de dados com múltiplas dimensões em uma grade (grid) regular de poucas dimensões (geralmente bidimensional). A rede é composta por nós (neurônios), onde cada nó possui um vetor de pesos da mesma dimensão dos vetores de dados de entrada. Normalmente, estes nós são dispostos sobre uma grade (grid) bidimensional. Desta forma, conforme a rede aprende, o mapa auto-organizável descreverá um mapeamento de um espaço de entrada de muitas dimensões, em um espaço de apenas duas dimensões (conforme as dimensões da rede) [Kohonen, 2001].

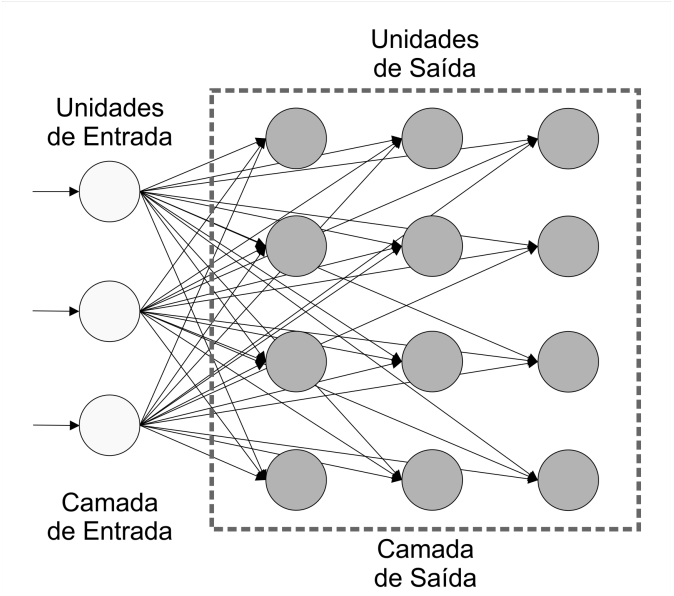


Figura 2.6: Exemplo de criação de um Mapa auto-organizável de Kohonen.

A figura 2.6 ilustra um mapa auto-organizável de Kohonen, onde três unidades de entrada estão conectadas com doze unidades de saída, dispostas em um grid 4x3. No KSOM não existem unidades escondidas, as unidades da camada de entrada estão conectadas diretamente com todas as unidades da camada de saída. Os pesos de cada unidade são aprendidos de forma iterativa.

Q-Learning

É um método para agentes aprenderem como agir de forma ótima em domínios markovianos controlados. Equivale a um método incremental de programação dinâmica que funciona melhorando sucessivamente suas avaliações de qualidade para determinadas

ações em estados particulares [Watkins & Dayan, 1992]. Ou seja, o agente aprende experimentando as consequências de suas ações, sem precisar construir mapas, modelos ou representações do domínio (*Model Free*).

Um agente que utiliza *Q-learning* procede da seguinte forma:

- experimenta uma ação em um estado particular;
- avalia a consequência da ação em termos da recompensa ou punição recebida;
- estima o valor do estado que a ação o levou;

Após experimentar todas as ações em todos os estados repetidamente, o agente aprende as que são melhores de forma geral, a partir da estimativa de recompensa descontada a longo prazo. Segundo Kazakov & Kudenko [2001], a técnica é aplicável apenas a agentes reativos (i.e., aqueles que baseiam suas escolhas apenas no estado atual).

```
Q-learn (g : fator de desconto) {
        Inicializa Q(s,a) = 0 para todos os pares de estado-ação (s,a) existentes;
        s = estado inicial;

        repita pra sempre {
                selecione e execute uma ação a;
                s' = novo estado;
                E = maior valor de Q(s',a') sobre todas as ações a';
                Q(s,a) = R(s') + g*E;
                s = s';
        }
}
```

Figura 2.7: Algoritmo do *Q-learning*. (Adaptado de Kazakov & Kudenko [2001])

A figura 2.7 representa uma adaptação do algoritmo do *Q-learning* (apresentado por Kazakov & Kudenko [2001]). Inicialmente os valores Q associados a todos os possíveis pares de estado (s) ação (a) são inicializados como 0. A partir do estado inicial o algoritmo seleciona e executa uma ação a, observa para qual estado (s') aquela ação o levou, e atualiza o valor de $Q(s, a)$ considerando o reforço recebido no novo estado ($R(s')$) e o valor $Q(s', a')$ descontado de g (sendo a' a ação que possui a maior utilidade no estado s'). Em seguida, repete-se o algoritmo a partir dos passos de seleção e execução de uma ação a, considerando o novo estado s' como estado inicial.

Embora a inferência de contexto não se preocupe tanto com a ação que será escolhida, mas sim em prover os contextos de alto nível a serem considerados na escolha desta ação, ainda assim existem trabalhos que utilizam *Q-learning* a fim de melhorar as

inferências conforme se interage com o ambiente. Por exemplo, Kabir et al. [2015] aplica *Q-learning* em um SSC voltado para uma casa inteligente a fim de que o SSC se adapte a medida que as escolhas do usuário mudam. Contudo, uma das maiores limitações que inviabiliza a utilização de *Q-learning* em computação móvel, é a necessidade de se conhecer, de antemão, o conjunto de estados (contextos) do ambiente e as possíveis ações que o SSC poderá realizar.

2.3.3 Aprendizagem por Reforço

A inteligência artificical gira em torno da definição de agentes inteligentes. Russel & Norvig [1995] definem um agente como algo ou alguém capaz de observar (perceber) um determinado ambiente a partir de sensores e atuar sobre este ambiente por meio de efetores (i.e., mecanismos responsáveis por atuar em resposta a certos estímulos). Um agente racional é aquele capaz de escolher a ação correta ou a mais adequada para qualquer percepção do ambiente, maximizando alguma medida de desempenho.

Um agente racional será inteligente se também for autônomo, ou seja, se puder melhorar seu desempenho (aprender) conforme interage com o ambiente. Desta forma, um agente inteligente não escolhe as ações com base apenas em conhecimento prévio, mas também considera suas próprias experiências na tomada de decisão. Considerando um ambiente em que não há um professor para indicar quais as ações são indicadas para cada estado do ambiente, um agente inteligente deverá aprender a escolher suas ações com base apenas em suas percepções do ambiente. Ou seja, o agente aprende a partir da observação das consequências positivas (recompensas) ou negativas (punições) de suas ações. Neste cenário não é possível utilizar aprendizagem supervisionada, por não haver a figura de um professor. Além disso, uma abordagem de aprendizagem não supervisionada não poderia mapear em ações, os padrões indentificados na entrada, sem considerar se a ação foi boa ou ruim para um determinado estado.

Aprender a atuar de maneiras que produzem recompensas é um sinal de inteligência [Watkins, 1989]. A aprendizagem por reforço aborda a questão de como um agente autônomo que percebe e atua em seu ambiente pode aprender a escolher ações ótimas para atingir suas metas [Mitchell, 1997]. A figura 2.8 ilustra as interações de um agente inteligente que aprende por meio de reforços.

O reforço é uma consequência perceptível decorrente de uma ação. Esta consequência pode ser uma recompensa (i.e., reforço positivo), indicando que a ação foi adequada, ou uma punição (i.e., reforço negativo) caso contrário. Dependendo da aplicação, os reforços poderão ser obtidos em qualquer estado do ambiente (e.g., um robô aprendendo a caminhar sobre uma fita), ou somente em estados terminais (e.g., após

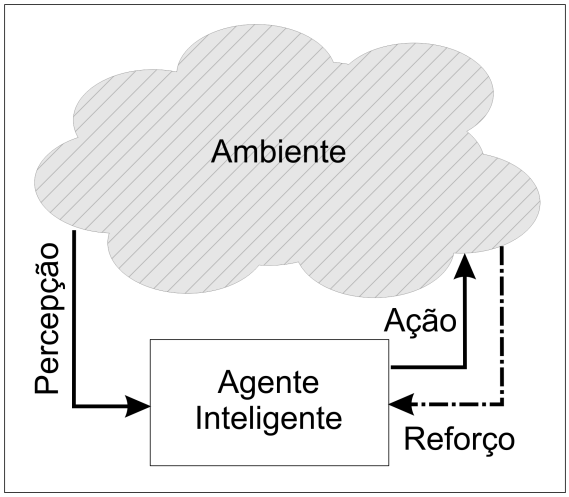


Figura 2.8: Esquema de um Agente Inteligente que Aprende por Reforços.

um checkmate ou empate em uma partida de xadrez). Além disso, o reforço pode ser obtido de forma implícita, o agente observa o ambiente e captura a informação do reforço (e.g., o robô deve ser capaz de perceber se saiu ou não da fita), ou de forma explícita, o ambiente informa ao agente o sinal de reforço (e.g., o jogo de xadrez informa que o agente perdeu ou venceu a partida).

Uma abordagem incremental de Sistema Sensível ao Contexto pode ser modelada como um problema de aprendizagem por reforço considerando que: o SSC é um agente inteligente que percebe o ambiente por meio de sensores; age no ambiente prestando serviços e realizando tarefas de acordo com o contexto detectado; avalia se a ação escolhida foi apropriada (ou não) para o contexto do usuário; e aprende com esta informação.

O objetivo da aprendizagem por reforço é que o agente aprenda a se comportar de maneira ótima em um ambiente (inicialmente) desconhecido. Por outro lado, aprendizagem ótima é a utilização adequada das informações disponíveis na tomada de decisão, considerando tanto os erros anteriores como os possíveis sucessos futuros [Watkins, 1989]. Desta forma, um agente que aprende de modo ótimo não necessariamente se comporta de maneira ótima, mas converge para este comportamento.

Para que o desempenho da inferência de contexto seja ótimo, é necessário que a inferência atribua rótulos corretos para todos os diferentes estados do ambiente. Contudo, a garantia de um desempenho ótimo depende da exploração exaustiva de todas as combinações de rótulos para cada estado do ambiente, o que é inviável do ponto de vista computacional para ambientes com muitos estados. Por outro lado, para que a aprendizagem da inferência seja ótima é necessário considerar os desempenhos de inferências passadas tentando maximizar os ganhos futuros.

O dilema entre exploração e desempenho é um assunto bastante discutido na literatura, tendo em vista a dificuldade e importância de se balancear essas duas formas de tomada de decisão. Este dilema consiste em escolher entre as melhores ações para o estado atual, com base no conhecimento do sistema, e as ações que não são atualmente as melhores, mas que ampliariam a base de conhecimento sobre o ambiente. Sem exploração não há aprendizagem, pois o agente irá tomar sempre as mesmas decisões com base no modelo criado, sendo incapaz de melhorar seu desempenho (i.e., não há inteligência). Por outro lado, decidir sem considerar o desempenho não é racional nem inteligente.

2.4 Considerações Sobre o Capítulo

Neste capítulo, foram apresentados conceitos e fundamentos da Computação Sensível ao Contexto. Após esclarecer a diferença entre contexto e dados brutos, os contextos também foram diferenciados de acordo com aquilo que as informações contextuais representam: contextos de baixo nível representam estados do ambiente, enquanto contextos de baixo nível representam características do ambiente. Estas definições são importantes para o entendimento do que a técnica proposta faz.

Além disso, também foi apresentada uma definição de sensores que vai além da intuição de que sensores apenas capturam informações do ambiente físico. As diferentes categorias em que os sensores podem ser enquadrados ampliam a visão dos desenvolvedores de aplicações sensíveis ao contexto, proporcionando uma variedade de possibilidades a serem exploradas na área de CSC.

Como a abordagem proposta neste trabalho consiste em um Sistema Sensível ao Contexto, os fundamentos apresentados neste capítulo também trataram aspectos específicos destes sistemas, bem como, as principais formas de modelagem de contexto, as principais arquiteturas usadas em SSC e as principais técnicas de inferência de contexto. Ao estudar as formas de modelagem de contexto observou-se que as diferentes modelagens podem ser mais ou menos adequadas à determinadas técnicas de inferência de contexto. Portanto, é recomendável que a escolha da forma de modelar contextos considere o domínio de aplicação e a técnica de inferência que deverá ser utilizada no SCC. Com relação às arquiteturas, em geral, elas representam formas diferentes de organizar e realizar as etapas envolvidas em CSC, o que afeta nos tipos de técnicas de inferência, ou nas formas que elas poderão ser usadas. Por exemplo, ao usar uma arquitetura centralizada ou cliente-servidor em uma aplicação móvel, é possível aplicar técnicas de inferência que exigem muito dos recursos dos dispositivos móveis, enquanto

que em arquiteturas distribuídas estas técnicas podem ser inviáveis, caso não se possa adaptá-las para poupar recursos.

Também foram apresentadas as principais categorias de técnicas de inferência de contexto mais comumente usadas em SSC, sendo que várias delas já foram aplicadas em diferentes abordagens voltadas para computação móvel. Alguns autores consideram que a aprendizagem por reforço pode ser usada como complemento a outras técnicas de inferência, mas não detalham como fazer. Entretanto, mesmo que se utilize aprendizagem por reforço como complemento de outras técnicas, a possibilidade de se aprender a inferir contextos (i.e., reduzir os erros na inferência) conforme ocorrem as interações do usuário, já caracteriza uma nova categoria de técnicas de inferência. E é nessa categoria que a técnica proposta neste trabalho está inclusa.

Por fim, foram apresentados alguns fundamentos da área de Aprendizagem de Máquina, dentre os quais destacamos as definições de algumas medidas de desempenho, as quais foram usadas para avaliar o estudo de caso desenvolvido, neste trabalho, para validar a técnica de inferência proposta neste trabalho. Também foi dada uma visão geral sobre as principais técnicas de aprendizagem de máquina que são aplicadas em inferência de contexto para SSC's, mesmo que algumas delas ainda não tenham sido, ou sejam pouco viáveis de serem aplicadas em computação móvel. Além disso, detalhou-se um pouco mais sobre a aprendizagem por reforço, tendo em vista os conceitos oriundos da área de Inteligência Artificial.

Capítulo 3

Trabalhos Correlatos

Existem muitas hipóteses em ciência que estão erradas. Isso é perfeitamente aceitável, eles são a abertura para achar as que estão certas

Carl Sagan

Os trabalhos correlatos apresentados neste capítulo estão divididos em seções referentes às três categorias seguintes: Sistemas Sensíveis ao Contexto para Dispositivos Móveis; Aplicações Móveis Sensíveis ao Contexto; Sistemas Sensíveis ao Contexto que utilizam aprendizagem por reforço.

3.1 Sistemas Sensíveis ao Contexto para Dispositivos Móveis

Nesta seção, serão apresentadas as principais abordagens de sistemas sensíveis ao contexto destinadas para aplicações móveis. Além do funcionamento, serão destacadas as arquiteturas, formas de modelagem de contexto e técnicas de inferência adotadas. Ao final, será apresentada uma taxonomia e a classificação destas abordagens segundo a taxonomia utilizada.

3.1.1 Projeto *Aura* (2002)

É um sistema orientado a atividades, que funciona sobre uma arquitetura distribuída e baseada na utilização de serviços. Estes serviços são disponibilizados através de ser-

vidores: Servidor Principal e Servidores Intermediários (auxiliavam na comunicação, tendo em vista as limitações de conectividade da época). Os contextos são obtidos através de requisições feitas pelo próprio usuário, e também pela aplicação cliente. Segundo Garlan et al. [2002], esta aplicação possui duas características: Proatividade e Autoajuste (adaptação). Além disso, utiliza internamente os seguintes componentes: Gerenciador de Tarefas; Observador de Contexto; Gerenciador de Ambiente; Fornecedores de Contexto.

Este sistema tem como objetivo minimizar as distrações sobre a atenção humana, proporcionando um ambiente que se adapta ao contexto do usuário e suas necessidades. A modelagem de contexto é baseada em esquemas de marcação (perfis e preferências do usuário) e a inferência utiliza regras criadas a partir das intenções do usuário, armazenadas nos esquemas de marcação. Desta forma, o sistema detecta apenas contextos pré-estabelecidos ou pré-configurados, tomando como base as preferências e intenções do usuário. Assim, para que novos contextos sejam detectados, o sistema deverá ser atualizado exigindo a criação de novas regras, e que o usuário atualize suas preferências.

3.1.2 *Hydrogen* (2003)

A abordagem *Hydrogen* é o nome do *framework* proposto por Hofer et al. [2003], e consiste em uma arquitetura distribuída que utiliza três camadas (ilustradas na figura 3.1): Camada de Aplicação, Camada de Gerenciamento de Contexto e Camada de Adaptadores. A aplicação faz requisições ao servidor de contextos (camada de gerenciamento), o qual utiliza dados provenientes de diversos adaptadores (e.g., adaptador de rede, de localização, de tempo, de usuários) para prover contextos aos usuários. O objetivo principal desta arquitetura era fazer a separação entre sensoriamento e aplicação. Entretanto, o autor não comenta sobre as técnicas de inferência que serão usadas, e considerava que a inferência dependeria dos objetivos e tipos de aplicações, passando a elas a responsabilidade de inferir contexto.

Na abordagem *Hydrogen* os dados brutos coletados pelos adaptadores são armazenados no servidor de contextos, e repassados para as aplicações, as quais devem decidir como inferir contextos de alto nível. Esta abordagem modela contextos através de orientação a objetos (Java), e utiliza esquemas de marcação (XML) durante a transferência de informações entre camadas. Enquanto outros sistemas sensíveis ao contexto centralizam as informações contextuais, a abordagem *Hydrogen* tenta evitar a dependência de uma base central (e.g., Servidor, Pontos de acesso). Embora a aquisição de dados, o gerenciamento de contexto seja feita no próprio dispositivo, ainda assim é possível trocar informações contextuais entre dispositivos (peer-to-peer).

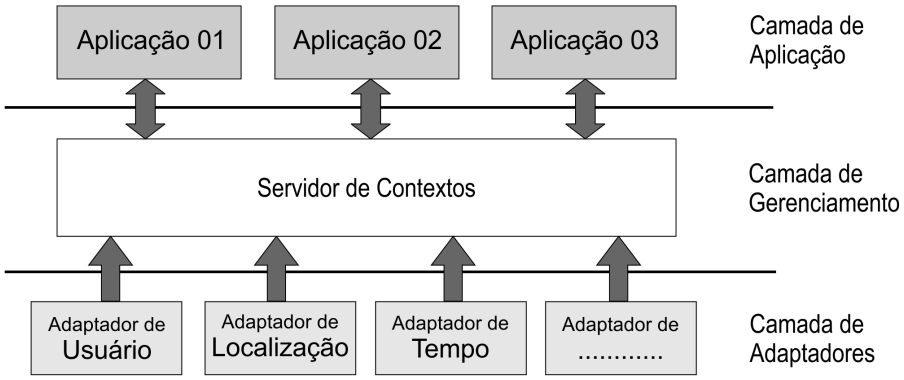


Figura 3.1: Arquitetura da abordagem *Hydrogen*. (Adaptado de Hofer et al. [2003])

A abordagem *Hydrogen* também tem sua importância por ser a primeira a utilizar apenas dispositivos móveis para detectar contextos, permitindo a interação e troca de informações contextuais entre dispositivos sem a necessidade de uma estação base. Desta forma, as aplicações poderiam considerar informações contextuais de outros dispositivos (contexto remoto), com os quais é possível se comunicar, além das informações contextuais do próprio dispositivo (contexto local).

3.1.3 CARISMA (2003)

O termo vem de "*Context-Aware Reflective mIddleware System for Mobile Applications*", ou seja, é um sistema de *middleware* reflexivo e sensível ao contexto para aplicações móveis. O foco do CARISMA está em sistemas móveis extremamente dinâmicos, ou seja, que requerem muita adaptação [Capra et al., 2003]. A modelagem de contexto é feita através de esquemas de marcação (XML), e utiliza regras para realizar a inferência de contexto. Além disso, o *middleware* provê primitivas que descrevem como as mudanças de contextos devem ser tratadas a partir do uso de políticas. Devido a possibilidade de ocorrência de conflitos entre políticas, há a necessidade de se "refletir" sobre os conflitos a fim de resolvê-los, sempre levando em consideração os interesses dos usuários.

Os interesses dos usuários são registrados através de metadados que utilizam esquemas de marcação, estes registros podem ser de duas categorias: passiva e ativa. Na categoria passiva são definidas ações para o *middleware* de acordo com eventos detectados (contextos), usando regras. A categoria ativa permite que se mantenham regras, configurações de contexto e relacionamentos entre serviços usados pela aplicação, dizendo como o sistema deve se comportar sob diferentes condições do ambiente ou do usuário.

Como o CARISMA tem uma preocupação maior com a resolução de conflitos entre regras, concluímos que o *middleware* apresentado é secundário, ou seja, o objetivo não era propor uma nova abordagem de sistemas móveis sensíveis ao contexto. Mesmo assim, este trabalho tem sua importância por tratar uma das principais limitações da utilização de regras na inferência de contexto (i.e., conflitos em ambientes dinâmicos), em tempo de execução, e não de forma estática (durante o projeto e implementação).

3.1.4 *Framework* de Gerenciamento de Contexto (2003)

Korpipaa et al. [2003] propõem um *framework*, de arquitetura distribuída, para gerenciamento de contexto que provê métodos sistemáticos de aquisição, processamento e distribuição de informações contextuais úteis para as aplicações, a partir do ambiente em que o usuário está inserido. Este *framework* possibilita o reconhecimento de contextos semânticos em tempo real, mesmo na presença de incertezas e ruídos, e distribui as informações contextuais de uma forma baseada em eventos. Além disso, também utiliza uma ontologia expansível para definir contextos que os clientes podem usar.

A arquitetura interna do *framework* possui quatro entidades funcionais principais: Gerenciador de Contexto - funciona como um servidor para as diversas aplicações que poderão estar interessadas em informações contextuais, interage com os demais componentes da arquitetura, disponibiliza informações contextuais em um Quadro Negro (Blackboard, as informações ficam visíveis para todas as aplicações e serviços), ou pode gerar eventos quando requisitado por aplicações específicas; Servidor de Recursos - se conecta com qualquer fonte de dados contextuais, e publica as informações contextuais no Quadro Negro do gerenciador de contextos; Serviço de Reconhecimento de Contexto - permite que desenvolvedores ou processos adicionem ou removam, de forma online, serviços de reconhecimento de contexto; Aplicações - Podem visualizar as informações contextuais no Quadro Negro e subscrever por um evento (contexto) específico.

O objetivo central deste *framework* é gerenciar contexto, por este motivo, é proposto um *framework* genérico, onde é possível utilizar qualquer técnica durante a inferência de contexto de forma transparente para a aplicação. Durante a implementação do *framework* utilizou-se inferência baseada em ontologia, lógica fuzzy e um classificador Bayesiano. Os experimentos práticos mostraram que o classificador Bayesiano conseguiu bons resultados de inferência (poucos falsos positivos e poucos falsos negativos), entretanto o autor comenta que a maior parte dos resultados foi obtida a partir de cenários restritos (parecidos com os cenários de treinamento), ou seja, em ambientes controlados, sem considerar outros contextos que pudessem afetar os resultados, o que nem sempre acontece no mundo real.

3.1.5 Reconhecimento e Predição de Contexto (2003)

Mayrhofer et al. [2003] apresenta uma arquitetura para o reconhecimento e predição de contexto, que aprende a partir dos comportamentos do usuário. Os dados coletados pelos sensores são classificados, a fim de detectar padrões comuns nos valores de entrada. Estes padrões são interpretados como estados de uma máquina de estados abstrata que age como um identificador de contextos. Quando um usuário avança de um contexto para outro, as leituras dos sensores irão mudar, e um novo estado ficará ativo, refletindo a mudança de contexto. Desta forma, o autor afirma que a partir da trajetória percorrida na máquina de estados é possível fazer previsões de outras trajetórias prováveis no futuro.

Esta abordagem utiliza aprendizagem não supervisionada para inferir e prever contextos, e é voltada para dispositivos móveis executando sem a intervenção de usuários, por um longo período de tempo. A arquitetura geral utilizada não é detalhada no trabalho, mas sim a arquitetura para realizar a inferência e predição de contexto, a qual consiste em quatro partes principais: Extração de características (*feature extraction*), classificação, rotulagem (*labeling*) e predição. Os sensores disponíveis produzem dados que são transformados em um vetor de entrada para o passo de classificação (utiliza clusters, aprendizagem não supervisionada); Em seguida, a rotulagem associa os contextos reconhecidos (padrões identificados) à nomes significativos, especificados pelo usuário; A predição realiza previsões de futuros estados (contextos) do sistema, com base nos dados aprendidos a partir dos comportamentos do usuário.

Embora haja certa flexibilidade do sistema em relação a inserção de novos contextos, a rotulagem é feita pelo usuário após a etapa de classificação, desta forma, o usuário deverá entender o que aquele padrão de comportamento representa, para então mapeá-lo em um contexto, o que nem sempre é uma tarefa fácil, podendo ocasionar em rotulagens erradas. Além disso, mesmo que o usuário seja capaz de atribuir rótulos aos padrões encontrados, as aplicações não terão como identificar o que os rótulos representam.

3.1.6 CASS (2004)

O CASS ("*Context-Awareness Sub-Structure*"), em português, Subestrutura de Sensibilidade ao Contexto, é um *middleware* cliente-servidor destinado a suportar aplicações sensíveis ao contexto em dispositívos móveis [Fahy & Clarke, 2004]. Este sistema trata as limitações de hardware dos dispositivos móveis (e.g., memória, processador, bateria) através do paradigma cliente-servidor, ao tirar do cliente a responsabilidade de proces-

sar e executar tarefas que podem exigir muitos recursos. Além disso, o CASS também utiliza dados coletados por nós sensores (estáticos ou móveis) espalhados no ambiente.

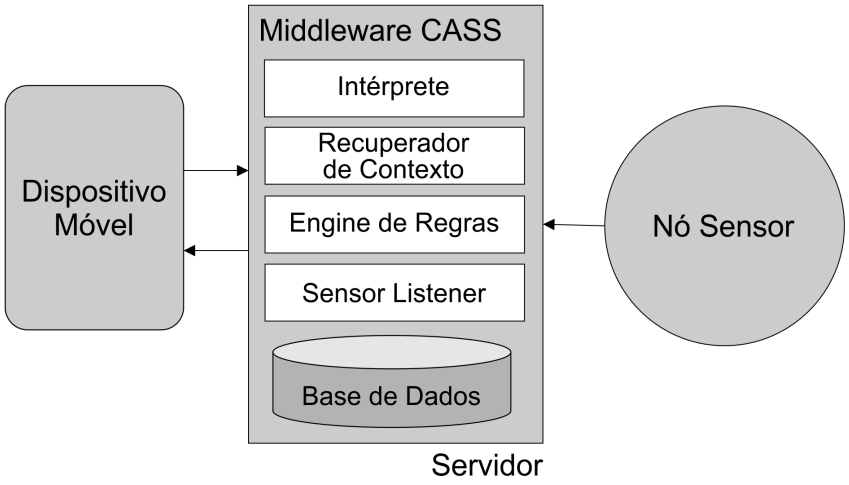


Figura 3.2: Arquitetura cliente-servidor usada na abordagem CASS. (Adaptado de Fahy & Clarke [2004])

A figura 3.2 apresenta a arquitetura cliente servidor do CASS. Nesta arquitetura, além dos clientes (dispositivos móveis) e servidores (acessíveis por um *middleware*), ainda são utilizados nós sensores que proveem dados do ambiente para o servidor. No servidor os dados coletados são armazenados, e através de técnicas de inferência de contexto baseadas em regras, abstrações contextuais de alto nível são produzidas, armazenadas, e posteriormente reportadas aos clientes interessados. O CASS armazena tanto as regras quanto as informações de contexto em bases de dados utilizando SQL, entretanto, a modelagem de contexto é muito parecida com o modelo *key-value*. A arquitetura interna do *middleware*, localizado no servidor, é formada por: Intérprete (utiliza uma base de conhecimento), Recuperador de Contexto (Interage com o cliente, informando contextos), Engine de Regras (inferência de contexto) e Sensor Listener (responsável por registrar atualizações de dados contextuais provenientes dos sensores).

O CASS permite que os desenvolvedores superem as limitações de memória e processador inerentes aos dispositivos de mão, enquanto dá suporte a grandes números sensores de baixo nível e outras fontes de contexto. O fato de que a aplicação cliente não realiza inferência de contexto é um dos pontos fortes do CASS, no sentido de que polpa recursos computacionais dos dispositivos, e permite que dispositivos mais limitados também sejam sensíveis ao contexto. Entretanto, isto exige que a aplicação cliente esteja sempre conectada com o servidor, o que nem sempre é, ou será, possível.

3.1.7 STEAM (2004)

STEAM, nome dado ao *framework* proposto por Biegel & Cahill [2004]. Na verdade, é a sigla de *Scalable Timed Events And Mobility*. É um serviço de *middleware* (com arquitetura distribuída) consciente de localização baseado em eventos, projetado especificamente para ambientes de redes ad-hoc sem fio.

Utiliza como base um modelo de objeto sensível, que encapsula três entidades principais: Captura Sensorial (*Sensory Capture*), Hierarquia de Contexto (*Context Hierarchy*), Engine de Inferência (*Inference Engine*). Neste modelo sensores reportam eventos, os quais são capturados pelo objeto sensível. Este, por sua vez, aplica uma hierarquia de contexto e infere contextos com base em regras, produzindo contextos para atuadores (aplicações). Quando os eventos são capturados, o objeto sensível aplica um esquema probabilístico de fusão de dados, baseado em Redes Bayesianas. A modelagem de contexto utiliza hierarquias, implementadas com esquemas de marcação (XML).

Além disso, foi desenvolvida uma ferramenta visual para facilitar o desenvolvimento de aplicações, a qual reduz a necessidade de escrever código. A "programação" é feita arrastando e soltando componentes e configurando alguns parâmetros, são gerados códigos XML e CLIPS (usado na criação de regras para a inferência de contextos de alto nível) automaticamente.

3.1.8 CaSP (2007)

A sigla significa plataforma de serviços sensíveis ao contexto ("*Context-aware Service Platform*"). É um *framework* que utiliza conjuntos de interfaces e protocolos de comunicação para coletar (e identificar) contextos a partir de sensores [Devaraju et al., 2007]. O *framework* coleta, no lado do cliente, dados brutos a partir de sensores físicos, via programas de abstração de sensores escritos pelos desenvolvedores dos mesmos, traduz esses dados para o formato reconhecido pela plataforma CaSP. Em seguida, o cliente entrega os dados formatados para o servidor, via conexão TCP, afim de que o mesmo modele e infira contextos.

O CASP é um *middleware* que funciona sob a arquitetura Cliente-Servidor. Os contextos são modelados através de esquemas de marcações (XML) e ainda por ontologias. O servidor utiliza técnicas baseadas em ontologia para inferir contextos, a partir dos dados coletados pela aplicação cliente. É interessante comentar sobre a API desenvolvida, que auxilia no envio (*report*) dos dados a partir dos programas de implementação dos sensores, sem exigir que desenvolvedores de aplicações tenham que lidar com tarefas de programação e mecanismos de comunicação com a plataforma. Desta

forma, o CaSP encoraja a criação e desenvolvimento de novas aplicações sensíveis ao contexto.

3.1.9 *Feel@Home* (2010)

Feel@Home é um *framework* de gerenciamento de contexto que suporta interações entre diferentes domínios (e.g., casa inteligente, escritório inteligente, dispositivos móveis) [Guo & Zhang, 2010]. Suporta dois tipos de interações: intradomínio e entre domínios. De acordo com os domínios, determinadas entidades podem ter papéis diferentes do ponto de vista do modelo Produtor-Consumidor (e.g., um usuário de dispositivo móvel que é um consumidor neste domínio, passa a ser um produtor, informando contextos ao domínio de um escritório inteligente).

O gerenciamento de contexto é feito de acordo com domínios. Os gerenciadores de contexto de domínio consideram como produtor qualquer fonte de dados contextuais, e consideram consumidor qualquer entidade interessada em informações contextuais. Em sua arquitetura existem três componentes: Servidor Global de Administração (GAS); Gerenciadores de Contexto de Domínio; Requisições de Usuários. As requisições são recebidas pelo GAS (Servidor), que decide qual domínio é relevante, e precisa ser acionado para responder à requisição. A modelagem de contexto é feita através do uso de ontologias, e os dados são armazenados em bancos de dados, por isso a inferência de contexto utiliza técnicas baseadas em ontologias.

O *Feel@Home* é um *framework* de caráter genérico, cujo foco é dar suporte ao gerenciamento de contexto tanto em domínios específicos, como em múltiplos domínios, mas também é um sistema sensível ao contexto voltado para dispositivos móveis.

3.1.10 COSAR (2010)

COSAR é um sistema de reconhecimento de atividades com arquitetura centralizada para dispositivos móveis [Riboni & Bettini, 2010]. Utiliza dados provenientes tanto de sensores embarcados no dispositivo, como de sensores "vestíveis" (*wearable*, ou seja, sensores de outros dispositivos carregados/vestidos pelo usuário) e de sensores espalhados no ambiente. O COSAR é uma abordagem que utiliza ontologias (modelagem) e inferência baseada em ontologia combinada com técnicas de aprendizagem estatística supervisionada, para fazer o reconhecimento de atividades dos usuários.

A arquitetura do COSAR contém: um conjunto de Sensores externos ao dispositivo móvel (i.e., sensores no ambiente, e sensores "vestíveis"); o próprio Dispositivo Móvel do Usuário, que faz a fusão dos dados dos sensores externos com os dos sensores

embarcados, a fim de criar um vetor de características (*features vector*), que será usado para inferir a atividade do usuário; A camada de infraestrutura (fora do dispositivo móvel), que realiza o reconhecimento de padrões (modelos estatísticos), e inferência ontológica, possuindo ainda um Servidor de Localização e Provedor de Rede (para a comunicação com o dispositivo móvel e com a camada de serviços).

Assim como no trabalho de Korpipaa et al. [2003], foram obtidos bons resultados para um conjunto específico de contextos (atividades), testados em um ambiente estático e controlado, sem mudanças ou interferências de outros contextos que poderiam afetar a eficácia do classificador. A solução é interessante, por utilizar um algoritmo híbrido para inferência de contexto (neste caso, reconhecimento de atividades), onde os processamentos pesados são feitos fora do dispositivo móvel.

3.1.11 *Context Viewer* (2011)

É um sistema para compartilhamento de contexto, que utiliza Redes Bayesianas (aprendizagem supervisionada) para inferir contextos de alto nível em dispositivos móveis. Devido aos recursos limitados dos smartphones, foi adotado o modelo Cliente-Servidor como paradigma de arquitetura. O sistema possui os seguintes módulos: Coletor de Log (cliente), Pré-Processador (servidor), Reconhecedor de Contexto (servidor), Gerenciador de Rede Social (servidor) e Gerenciador de Serviços (cliente). O coletor de log coleta logs do dispositivo móvel (contextos de baixo-nível, capturados por sensores). Esses logs são enviados para o servidor, onde o pré-processador produz logs mais significativos. O reconhecedor de contexto detecta e armazena os contextos de alto nível referentes às atividades, emoções e relacionamentos do usuário. O gerenciador de rede social constrói redes sociais móveis, com base em contextos de alto nível, incluindo os relacionamentos do usuário, e por fim o gerenciador de serviços é responsável por apresentar as informações contextuais ao usuário, e requisitar ao servidor as requisições do usuário.

Os modelos de Redes Bayesianas desenvolvidas no *Context Viewer* são para usuários genéricos, e podem não funcionar bem para todos os usuários [Park et al., 2011]. Mais uma vez observa-se que, ao utilizar aprendizagem supervisionada, deve-se assumir que existe um modelo determinado e fixo para o ambiente. O que ocorre com muita frequência em ambientes de casas, escritórios e empresas inteligentes, mas com pouca frequência no cenário de um usuário de dispositivo móvel.

3.1.12 Classificação e Taxonomia

Após a revisão biliográfica, as diferentes abordagens de Sistemas Móveis Sensíveis ao Contexto foram classificadas considerando os seguintes aspectos:

- Arquitetura: (Co) Baseadas em Componentes; (D) Distribuídas; (S) Baseadas em Serviços; (N) Baseadas em Nós; (Ce) Centralizadas; (CS) Cliente-Servidor.
- Modelagem de Contexto: (KV) Modelo *Key-Value*; (M) Esquemas de Marcação; (G) Modelagem Gráfica ou Relacional; (OO) Orientada a Objetos; (L) Lógica; (On) Ontologias;
- Inferência de Contexto: (AS) Aprendizagem Supervisionada; (AN) Aprendizagem Não Supervisionada; (AR) Aprendizagem por Reforço; (R) Regras; (F) Lógica Fuzzy; (On) Técnicas baseadas em Ontologias; (P) Lógica Probabilística.

A tabela 3.1 apresenta a classificação das abordagens estudadas de acordo com os aspectos taxonômicos adotados:

Tabela 3.1: Classificação dos Sistemas Móveis Sensíveis ao Contexto estudados.

Abordagem	**Arquitetura**	**Modelagem**	**Inferência**
Aura	(S)	*(M)	(R)
Hydrogen	(D)	(OO), (M)	–
CARISMA	(D)	(M)	(R)
Korpippa (2003)	(D)	(On)	(On), (F), (AS)
Mayrhoffer (2003)	(D)	–	(AN)
CASS	(CS), (N)	*(G), *(KV)	(R)
STEAM	(D)	(M)	(AS),(R)
CaSP	(CS)	(M), (On)	(On)
Feel@Home	(D)	(On), (G)	(On)
COSAR	(C)	(On)	(On), (AS)
Context Viewer	(CS)	–	(AS)

As abordagens que não citam algum dos aspectos taxonômicos utilizados estão com o respectivo item marcado com um traço (–). Embora algumas abordagens não comentem explicitamente, em alguns casos é possível deduzir a forma de modelagem utilizada com base nas afirmações feitas pelos autores. Na tabela 3.1, os campos que foram deduzidos estão marcados com um asterísco (*).

3.2 Aplicações Móveis Sensíveis ao Contexto

Nesta seção, serão apresentados alguns dos principais aplicativos sensíveis ao contexto da atualidade. Esses aplicativos são descritos e analisados de um ponto de vista de usuário e com base nas informações disponibilizadas pelos desenvolvedores em seus respectivos sites.

3.2.1 IFTTT

IFTTT é um aplicativo móvel (para Android e iOS) que permite a criação de conexões poderosas a partir de uma simples afirmação: Se isto então aquilo (*If This Then That*) [IFTTT, 2015]. Recentemente, o IFTTT foi dividido em dois aplicativos que se diferenciam quanto aos gatilhos usados para iniciar tarefas: *Do Recipes* (fórmulas "Faça") - a tarefa é realizada por requisição explícita do usuário, através de um clique de botão por exemplo; e *If Recipes* (fórmulas "se") - a tarefa é realizada se um evento específico for detectado. As fórmulas "se" são as que tornam esta aplicação sensível ao contexto.

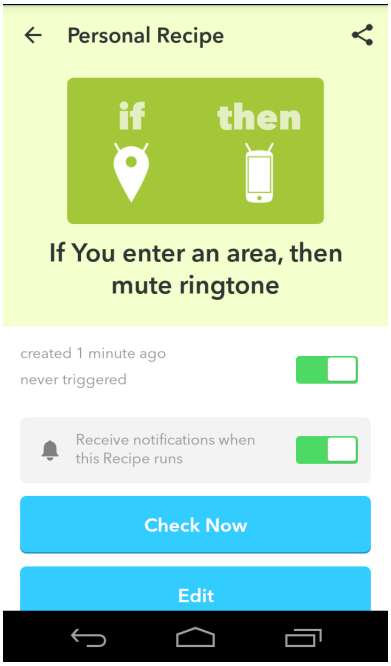


Figura 3.3: Tela de Definição dos Gatilhos no IFTTT.

A figura 3.3 apresenta uma tela com um exemplo de gatilho definido no IFTTT. Mesmo no caso de gatilhos e tarefas que não envolvem serviços de aplicações da nuvem, o IFTTT não funciona (i.e., é incapaz de executar uma tarefa definida em uma regra) se não estiver conectado à internet. Portanto, supõe-se que a arquitetura utilizada não é distribuída.

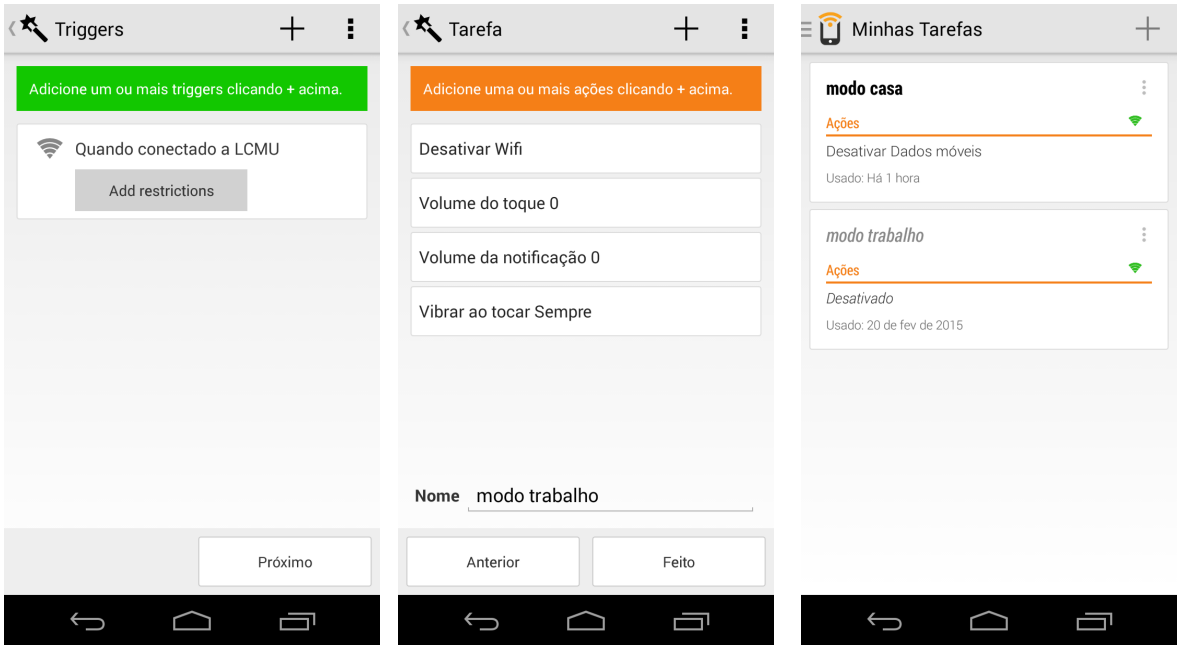
O IFTTT é uma aplicação sensível ao contexto pois o seu comportamento é determinado pela ocorrência de eventos (contextos pontuais). O aplicativo utiliza "canais" que podem ser de dois tipos: Gatilhos - informam de forma exata os contextos e as ocorrências de eventos (i.e., disparam gatilhos); Ações - prestam serviços ao usuário (i.e., executam tarefas). O formato das fórmulas "se" não permite que o aplicativo utilize mais de uma informação contextual para prover um serviço, o mapeamento é de uma condição para uma ação. Desta forma, o IFTTT não realiza inferência de contexto, mas cria conexões entre eventos e ações (ambos fornecidos por outros sistemas e aplicações). Ainda assim, o mapeamento de eventos em ações realizado pelo aplicativo, poderia ser melhor explorado se usasse um sistema sensível ao contexto como um canal de gatilho para a realização de tarefas. Desta forma, contextos de alto nível fornecidos pelo sistema seriam mapeados em ações pelos usuários. Uma limitação do IFTTT é que a aplicação requer acesso à internet para funcionar.

3.2.2 *Trigger*

O *Trigger* é um aplicativo para dispositivos móveis (especificamente para a plataforma Android), sensível ao contexto do usuário, que utiliza gatilhos para modificar o comportamento do aparelho. Segundo Egomotion [2014], na versão gratuita, o *Trigger* utiliza as redes sem fio (wi-fi), tags NFC (*Near Field Communication* - "Comunicação por Campo de Proximidade") e o Bluetooth do dispositivo como gatilhos, além disso, na versão paga é possível ainda utilizar o nível de bateria, localização e gatilhos temporais.

A figura 3.4 apresenta as telas de definição dos gatilhos (figura 3.4a), escolha das tarefas (figura 3.4b) e gerenciamento dos contextos (figura 3.4c). Na definição dos gatilhos o usuário determina as condições para que um determinado gatilho seja disparado, enquanto que, na definição das tarefas, o usuário escolhe quais ações deverão ser realizadas quando um determinado gatilho for disparado. A tela de gerência de contextos permite que o usuário visualize, edite, ative e desative os gatilhos e tarefas de acordo com as suas próprias necessidades.

O aplicativo funciona de forma distribuída, sem a necessidade de comunicação com um servidor. Além disso, a inferência de contexto é baseada em regras criadas pelos próprios usuários, o que permite que o usuário faça, de forma apropriada, o mapeamento de contextos em ações. Por permitir que o próprio usuário defina regras, contextos e ações, é possível afirmar que o *Trigger* é uma aplicação que permite um gerenciamento de contexto de forma flexível. Entretanto, não é ajustável ao usuário, pois infere contextos utilizando apenas as regras criadas pelo usuário, o que impossibilita a aprendizagem de modelos diferentes (melhoria das regras) para os contextos criados.



(a) Definição dos Gatilhos (b) Definição das Tarefas (c) Gerência de Contextos

Figura 3.4: Exemplos de Telas do *Trigger*

O *Trigger* não melhora suas inferências conforme o número de interações cresce, nem modifica seu comportamento caso o usuário mude de comportamento, ou seja, se o usuário não modificar as regras criadas o *Trigger* irá manter sempre os mesmos comportamentos para as mesmas situações. Portanto o aplicativo por si só é incapaz de se ajustar, adequar ou adaptar ao usuário, esses três aspectos dependem do interesse do usuário modificar as regras criadas.

3.2.3 *Tasker*

É um aplicativo pago, destinado a dispositivos móveis da plataforma Android, voltado para a automatização de tarefas. Segundo Tasker [2015], o *Tasker* é um aplicativo que executa tarefas (conjunto de ações) baseadas em contextos (aplicações, horário, data, localização, eventos e gestos) de acordo com perfis pré definidos pelo usuário.

O *Tasker* é um aplicativo muito mais robusto que o *Trigger*, pois além de utilizar gatilhos e mapeá-los em ações, também pode considerar *loops*, variáveis e condições na realização de tarefas, além de suportar plugins e ferramentas de terceiros [CraftyAppsEU, 2015]. O aplicativo permite a criação de perfis onde são criados contextos específicos para o perfil selecionado, possibilitando um maior grau de ajuste do dispositivo para o usuário. Entretanto, de modo semelhante ao *Trigger*, a inferência é baseada em regras e a ajustabilidade, adequabilidade e adaptabilidade do *Tasker*

também dependem do usuário.

Dentre os aplicativos analisados, o *Tasker* é um dos que mais se aproxima de um Sistema Sensível ao Contexto genérico, pois permite a interação com outras aplicações, através de um provedor de conteúdos (*ContentProvider*), e a criação de aplicações simples através de um plugin gratuito (*Tasker App Factory*). Embora a robustez do *Tasker* seja seu ponto forte, o aplicativo peca por ser pouco intuitivo para usuários novos e não familiarizados com os conceitos envolvidos no uso da aplicação.

3.3 Sistemas Sensíveis ao Contexto que utilizam Aprendizagem por Reforço

Nesta seção serão apresentados três SSC's que consideram a aprendizagem por reforço na inferência de contexto. Estes trabalhos não foram incluídos na seção 3.1, por não serem voltados para as aplicações móveis. Contudo, estão relacionados com este trabalho por aplicarem aprendizagem por reforço a fim de inferir contextos ou melhorar a inferência.

3.3.1 Um *middleware* para Agentes Sensíveis ao Contexto (2003)

Ranganathan & Campbell [2003] propõe um Middleware que facilita o desenvolvimento de agentes sensíveis ao contexto em ambientes de computação ubíqua. Este *middleware* permite que os agentes obtenham facilmente as informações contextuais, raciocinem sobre elas usando diferentes lógicas (e.g., lógica de primeira ordem, lógica temporal, regras e aprendizagem de máquina) e então se adaptem às mudanças de contextos (aprendizagem por reforço). O *middleware* utiliza ontologias na modelagem de contexto a fim de garantir que os diferentes agentes tenham o mesmo entendimento semântico sobre as diferentes informações contextuais.

O *middleware* utiliza uma arquitetura distribuída (voltada principalmente para ambientes inteligentes) e utiliza CORBA para possibilitar que agentes distribuídos se encontrem e se comuniquem uns com os outros. Além disso, agentes diferentes podem usar técnicas diferentes de inferência e considerar os resultados inferidos por outros agentes. Na abordagem proposta, existem seis tipos diferentes de agentes (ilustrados na figura 3.5): Provedores de Contexto (capturam informações contextuais do ambiente), Sintetizadores de Contexto (deduzem contextos de alto nível e fornecem essas informações a outros agentes), Consumidores de Contexto (são as alicações sensíveis

ao contexto que utilizam as informações tanto dos Provedores como dos Sintetizadores), Serviço de Busca de Provedor de Contexto (anunciam os serviços dos Provedores de Contexto, permitindo que os Consumidores encontrem provedores apropriados), Serviço de Histórico de Contexto (permite que outros agentes acessem contextos passados), Servidor de Ontologia (mantém ontologias que descrevem os diferentes tipos de informações contextuais.

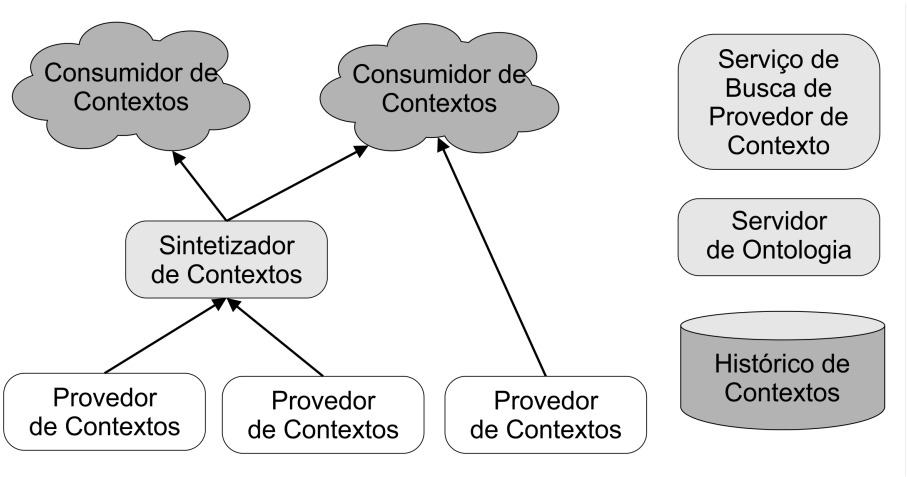


Figura 3.5: Arquitetura do *middleware* para Agentes Sensíveis ao Contexto. (Adaptado de Ranganathan & Campbell [2003])

Apesar do foco não estar voltado para aplicações móveis, não seria complicado usar este *middleware* com dispositivos móveis, bastaria modelar os dispositivos como agentes inteligentes. Apesar de afirmar que o *middleware* implementado suporta e utiliza aprendizagem por reforço, o autor não deixa claro o funcionamento da técnica implementada. Mas apresenta a aprendizagem por reforço como um complemento à inferência feita pelas diferentes técnicas que poderão ser usadas. Isto é justificável, pois o *middleware* proposto é de caráter genérico, e dependendo da técnica de aprendizagem usada na inferência os reforços serão usados de formas diferentes.

3.3.2 Um Sistema para Inferência de Contexto Distribuída (2010)

Rizou et al. [2010] apresenta um novo sistema que possibilita a inferência de contexto distribuída de uma forma genérica e independente do algoritmo de inferência utilizado. A abordagem se baseia na ideia de dividir o processo de raciocínio (inferência e decisão) em uma rede de correlacionadores de raciocínio, utilizando um modelo centrado em situações que possui padrões predefinidos (*templates*) de situações que são armazenados

como um conhecimento prévio do sistema. Os *templates* de situações são construídos a partir de contextos observáveis e unidade de processamento chamadas de Operadores. O sistema suporta diferentes técnicas de inferência por meio de diferentes operadores. Em geral cada template de situaçao forma um grafo de operadores que executam de forma cooperativa a inferência de contexto.

O sistema opera em duas fases distintas: Fase de inicialização - o sistema cria um plano lógico que descreve uma tarefa de inferência para um operador, em seguida o plano lógico é transformado em um plano físico, onde os operadores livres são colocados em componentes físicos de acordo com algum critério (e.g., minimizar o tráfego, ou *delay*); Fase de execução - a tarefa de inferência é executada de uma forma distribuída na rede física e ao mesmo tempo o sistema se adapta o plano físico de acordo com as condições da rede. A arquitetura é baseada em três camadas principais: Camada de Modelo do Mundo, Camada de Inferência de Contexto e Camada de Aplicação. Dentro da camada de Inferência existe um componente chamado Adaptação ao Feedback (retorno) que é responsável pela reconfiguração de operadores durante a fase de execução, com base nos reforços do usuário.

A inferência utilizada na implementação do sistema foi baseada em Redes Bayesianas distribuídas, mas o autor afirma que o sistema suporta outras técnicas, portanto, a adaptação ao feedback será diferente caso a técnica utilizada seja outra. Da forma como foi implementado, os reforços do usuário fazem com que o sistema recalcule os pesos da Rede Bayesiana, utilizando um algoritmo proposto por Zweigle et al. [2009]. A modelagem de contexto foi feita de forma orientada a objetos.

Apesar de ser possível a utilização do sistema para inferência de contextos no domínio da Computação Móvel, este sistema assume que dispositivos diferentes estão interessados nos contextos de um mesmo ambiente. O que torna a abordagem incompatível com a realidade das aplicações móveis que deverão se ajustar a diferentes usuários (ambientes).

3.3.3 Uma Aplicação de Casa Inteligente que usa *Q-learning* (2015)

Kabir et al. [2015] propõe uma aplicação sensível ao contexto capaz de prover serviços de acordo com escolhas predefinidas do usuário. A aplicação utiliza a distância de Mahalanobis em um classificador baseado nos K vizinhos mais próximos para realizar a inferência dos serviços predefinidos. Além disso, a aplicação combina características de aprendizagem supervisionada com aprendizagem por reforço, permitindo ainda que a aplicação se adapte quando a escolha do usuário mudar (utilizando *Q-learning*).

A arquitetura da aplicação é centralizada e considera três camadas: sensores, *middleware* e aplicação. A camada de sensores obtém as informações do ambiente, enquanto o *middleware* é responsável por armazenar, gerenciar e analisar as informações contextuais com base em uma ontologia. A aplicação em si é composta por alguns componentes principais: Módulo de Recebimento de Contexto - recebe informações contextuais de baixo nível de um provedor de contexto presente no *middleware*, e classifica estas informações em contextos de disparo de serviço, contextos que modificam serviços e reforços; Engine de Inferência - aplica o K-NN para determinar o serviço que será prestado dentre os serviços aprendidos; Módulo provedor de Serviços - responsável por prover serviços conforme solicitado pela engine de inferência; Módulo de Adaptação - considera os reforços explícitos e implícitos (tempo de utilização do serviço) do usuário balanceando a matriz Q.

Quando um par Estado-Ação $Q(a, s)$ atinge certo limiar, então a ação correspondente (serviço) e a informação contextual é adicionada à base de treinamento usada na inferência. Na próxima vez, a Engine de Inferência irá usar também está informação para escolher a ação mais adequada para o contexto.

A abordagem proposta por Kabir et al. [2015] é voltada para um ambiente restrito e estático, ainda assim, o *Q-learning* só é aplicável porque o autor considera um número fixo de contextos e limita a quantidade de estados possíveis do sistema. Caso fosse possível adicionar ou remover contextos (serviços), o sistema precisaria refazer a matriz Q. Além disso, se houvesse um número muito alto de combinações de estados, a matriz iria requerer cada vez mais espaço para ser usada. Portanto, embora a aplicação utilize *Q-learning* como técnica de aprendizagem por reforço, ela acaba sendo pouco escalável.

Na computação móvel é possível aplicar a técnica usada nesta aplicação de casa inteligente, contudo os recursos computacionais dos dispositivos móveis são mais escassos que os dos computadores modernos, o que exigiria um maior cuidado com o crescimento da matriz Q. Tornando esta abordagem pouco interessante para ser usada em um Sistema Sensível ao Contexto mais genérico capaz de prover informações contextuais a diferentes aplicações.

3.4 Considerações Sobre o Capítulo

Neste capítulo, foram apresentadas diferentes abordagens de Sistemas Móveis Sensíveis ao contexto. O funcionamento das abordagens estudadas foi analisado segundo a taxonomia proposta na seção 3.1.12 (i.e., segundo a arquitetura, forma de modelagem e técnica de inferência usada). Além disso, observa-se que a maioria das abordagens

atribui pouca importância para a dinamicidade dos ambientes do domínio das aplicações móveis. Em geral, as abordagens utilizam, de modo estático, um conjunto de regras, uma taxonomia ou um modelo aprendido na inferência de contextos, sem considerar que as condições do ambiente podem ser diferentes entre usuários, ou que podem mudar ao longo do tempo. Os sistemas que não podem se adaptar, dependendo das mudanças ocorridas no ambiente, poderão provocar ações inapropriadas nas aplicações, implicando na possível inutilização desses sistemas.

Além das abordagens de SSC voltadas para o domínio de aplicações móveis, também foram apresentados três aplicativos sensíveis ao contexto disponíveis para uso nos smartphones atuais. Em todos eles, a técnica de inferência utilizada é baseada em regras, as quais devem ser definidas pelos usuários. Desta forma, é possível realizar exatamente o que o usuário especificou para as condições (gatilhos) estabelecidas. Entretanto, se as condições mudam, é necessário que o usuário faça modificações de forma manual nas regras afetadas.

Outras abordagens de SSC's, que utilizam aprendizagem por reforço, também foram estudadas neste capítulo, pois nenhuma daquelas voltadas para dispositivos móveis aplica alguma técnica baseada em aprendizagem por reforço. Entretanto, embora essas abordagens sejam incrementais e aprendam a inferir contextos por meio de reforços, elas foram desenvolvidas tendo em vista as peculiaridades de ambientes inteligentes, deixando de lado aspectos importantes na computação móvel (e.g., limitações de hardware e software, suporte a diferentes usuários e/ou diferentes aplicações).

Por fim, com base na análise dos trabalhos correlatos, conclui-se que ainda não se propôs, em computação móvel, uma abordagem sensível ao contexto que utilize, na prática, aprendizagem por reforço. Portanto, uma das inovações deste trabalho consiste na aplicação de aprendizagem por reforço em uma abordagem de sistema móvel sensível ao contexto, além da técnica de inferência propriamente dita.

Capítulo 4

Inferência de Contexto via Aprendizagem por Reforço

O que prevemos raramente ocorre; o que menos esperamos geralmente acontece

Benjamin Disraeli

A principal contribuição deste trabalho, apresentada neste capítulo, consiste na proposição e avaliação de uma técnica de inferência de contexto para dispositivos móveis que aprende mediante o recebimento de reforços, a qual foi chamada de CoRe-RL. Esta técnica realiza duas tarefas principais:

- Classificação - Rotula o vetor de características atual com base nos demais vetores aprendidos (rotulados) e em um ranking de características.

- Adaptação - Modifica o modelo do ambiente de acordo com os reforços recebidos.

Para que haja uma melhor compreensão da técnica (detalhada na Seção 4.2), é importante que antes seja apresentada uma visão geral da arquitetura (Seção 4.1.1) e funcionamento (Seção 4.1.2) do Sistema Sensível ao Contexto no qual ela estará inserida. O nome Mob Context é referente à abordagem de SSC proposta neste capítulo, que utiliza o CoRe-RL na inferência de contextos. Esta abordagem representa uma contribuição secundária deste trabalho.

4.1 Mob Context

O Mob Context é uma abordagem distribuída de sistema sensível ao contexto com foco na disseminação de informações contextuais de alto nível para aplicações em dispositivos móveis, com suporte à aprendizagem por reforço. O objetivo da abordagem é prover contextos de alto nível para as aplicações (passando a estas a responsabilidade da tomada de decisão quanto ao que fazer com as informações recebidas). Além disso, a abordagem idealizada utiliza aprendizagem online baseada em exemplos e reforços informados pelas aplicações.

O gerenciamento flexível de contexto permite que as aplicações criem rótulos, cadastrem, removam ou renomeiem vetores de características rotulados. Portanto, é possível cadastrar um conjunto de exemplos rotulados e utilizá-los como ponto de partida para a inferência, possibilitando um salto na curva de aprendizagem (sem exigir interações com o usuário). No caso básico, em que não há exemplos de vetores rotulados, as aplicações deverão cadastrar pelo menos um vetor com os valores das principais características esperadas, neste caso as características principais terão um peso inicial maior que as demais características. Além disso, novos rótulos podem ser cadastrados também de forma online, ou seja, as aplicações podem cadastrar, remover e renomear os rótulos de acordo com a demanda do usuário.

Outra característica do Mob Context é que ele é ajustável ao usuário. Por ser uma abordagem distribuída que aprende de forma incremental, mesmo que as aplicações utilizem vários vetores de treinamento de caráter genérico (que abranjam uma grande quantidade de usuários), o sistema irá utilizar os reforços do usuário para se ajustar à realidade dele. Esse ajuste pode ocorrer tanto aprendendo novos vetores que estão mais próximos da realidade do usuário, como esquecendo exemplos que produzem muitos reforços negativos. Conforme o sistema for se ajustando ao usuário, ele irá melhorar seu desempenho na inferência, o que indica que o sistema é adequável ao usuário. Além disso, a mudança de comportamento do usuário fará com que o sistema tenha que se adaptar à nova realidade, mais uma vez levando em conta a aprendizagem de novos vetores de características e esquecimento daqueles que fazem o sistema errar.

4.1.1 Arquitetura

A partir das descrições dos trabalhos relacionados, apresentados no capítulo 3, é possível notar que já foram propostas diferentes abordagens de sistemas móveis sensíveis ao contexto. Cada abordagem apresentada, considera uma visão particular do problema da sensibilidade ao contexto, e assim, justificam a utilização das mesmas. Na seção

3.1, foram apresentadas as abordagens de SSC voltadas para a computação móvel, propostas na literatura. Entretanto, nenhuma delas considera em sua arquitetura, ou em seu funcionamento, a utilização de recompensas e punições para avaliar ou aprender a inferir contextos de modo incremental. Além disso, uma limitação de várias abordagens é a pouca flexibilidade no gerenciamento de contexto, pois não suportam a adição e remoção de contextos de modo online. Estes fatores motivaram a proposição do Mob Context.

A abordagem proposta neste trabalho tenta ser genérica ao ponto de não se limitar a uma aplicação ou a uma técnica de inferência, sem se preocupar com a resolução de todos os problemas de computação sensível ao contexto. Desta forma, tendo em vista que o foco do trabalho está na proposição de uma nova técnica de inferência que aprende por meio de reforços, a arquitetura elaborada considera os componentes principais para a realização da inferência de contexto conforme as necessidades da técnica proposta, sem se preocupar tanto com outras questões secundárias (e.g., comunicação entre dispositivos, segurança, predição de contextos futuros, utilização de mais de uma técnica de inferência).

A figura 4.1 ilustra as camadas e os principais componentes da arquitetura interna do Mob Context. Além disso, também são apresentadas as atribuições de cada camada (tendo em vista as fases envolvidas na computação sensível ao contexto) e os diferentes tipos de interações entre componentes e camadas.

A arquitetura do Mob Context está dividida em três camadas principais similares às camadas da abordagem Hydrogen [Hofer et al., 2003]: Coleta (camada de adaptadores, na abordagem hydrogen), responsável pela fase de sensoriamento; Gerenciamento de Contexto, gerencia o acesso às informações contextuais e é responsável pela fase de inferência e adaptação; Aplicação, presta serviços e realiza tarefas de acordo com as informações contextuais disponibilizadas, responsável pela fase de Decisão e Ação e por informar os reforços à camada de gerenciamento.

A camada de Coleta de Contexto (bloco superior da figura 4.1) realiza a captura de dados brutos (representados por pontos), através do uso de sensores. Nesta camada, é realizado um pré-processamento dos dados coletados, em seguida, ocorre a inferência de contextos de baixo nível e, por fim, as informações contextuais são registradas na base de dados da camada de Gerenciamento de Contexto (bloco intermediário da figura 4.1).

O Gerenciamento de Contexto é feito pelo Gerente de Contexto, o qual é responsável por: atender às requisições de inferência e gerenciamento (e.g., cadastro, remoção) de contexto; disseminar as informações contextuais de alto nível; receber os sinais de reforço da Aplicação e repassá-los para que a engine de inferência se adapte. Além

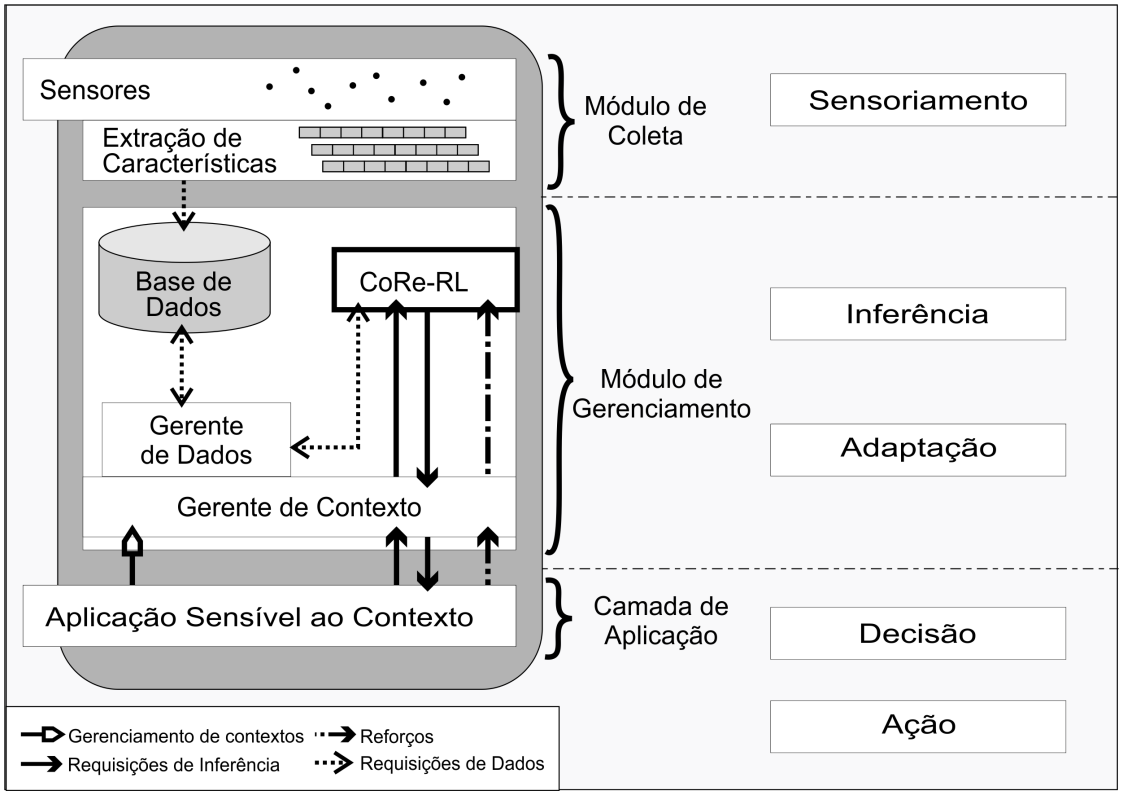


Figura 4.1: Arquitetura interna do Mob Context.

do Gerente de Contexto, a camada de gerenciamento possui uma base de dados, um gerente de dados e um módulo de inferência de contexto.

A Aplicação (bloco inferior da figura 4.1) solicita à camada de gerenciamento o cadastro dos contextos de alto nível que devem ser aprendidos, solicita as informações contextuais, mapeia os contextos em ações (tarefas e serviços) e retorna ao gerenciamento os sinais de reforço (positivos ou negativos).

4.1.2 Funcionamento

Camada de Coleta

O ambiente produz uma infinidade de dados brutos a todo instante. A camada de coleta tem como responsabilidade catalogar e acessar os sensores físicos e virtuais disponíveis nos dispositivos, com o propósito de capturar os dados brutos produzidos pelo ambiente. Como esses sensores monitoram variáveis específicas, isoladamente eles não são capazes de capturar todas as informações que caracterizam o ambiente. Por este motivo, em cada domínio de aplicação, utiliza-se uma combinação de sensores específicos de acordo com as variáveis a serem monitoradas. No Mob Context considera-se que existe um número fixo de sensores físicos e virtuais disponíveis no dispositivo e, consequentemente,

de dados brutos que podem ser coletados, não havendo necessidade de se adicionar ou remover sensores.

A técnica de inferência proposta neste trabalho utiliza vetores de características para inferir contextos. Estes vetores são criados, utilizando regras e lógica fuzzy, a partir da extração de características dos dados brutos coletados. Após a captura, a Camada de Coleta faz o pré-processamento e aplica regras sobre os dados brutos, afim de extrair as características que serão analisadas durante a inferência de contexto. O objetivo da extração de características é obter informações relevantes que possibilitem a caracterização de diferentes contextos no ambiente. A identificação das características que serão relevantes na inferência de contexto depende dos contextos que serão analisados, por isso, a técnica de inferência proposta considera na inferência todas as características passadas e aprende um modelo afim de ponderar as características de maior relevância. Assim, a Camada de Coleta, deverá se preocupar apenas em capturar e informar o maior número possível de características, contanto que essa quantidade não mude com o tempo.

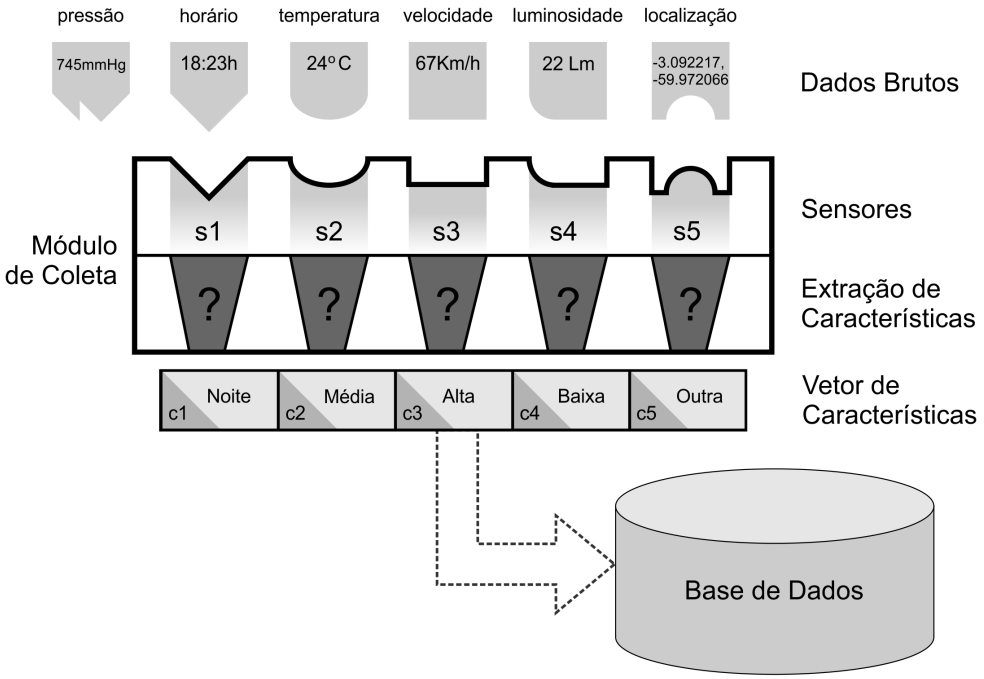


Figura 4.2: Exemplo de Funcionamento da Camada de Coleta.

A figura 4.2 apresenta o funcionamento da Camada de Coleta na abordagem Mob Context. Os dados brutos, representados na figura 4.2 por diferentes formas geométricas, são coletados por sensores específicos, representados por concavidades com formatos equivalentes aos seus respectivos tipos de dados brutos. Por exemplo, o

sensor s2, de concavidade arredondada, monitora a temperatura do ambiente, cujo dado bruto também possui formato arredondado. Mesmo com uma grande quantidade de sensores, ainda é possível que algumas variáveis não sejam monitoradas, por exemplo, a pressão atmosférica de 745 mmHg não é capturada por nenhum sensor na figura 4.2. Por outro lado, quanto mais sensores estiverem disponíveis nos dispositivos, maior será a representatividade dos dados coletados, tornando possível a inferência de um número maior de contextos, contudo, a complexidade das inferências também será maior.

Depois que os sensores coletam os dados brutos do ambiente, é feita a extração de características através da aplicação de regras, as quais estão representadas por interrogações na figura 4.2. Estas regras produzem informações contextuais de baixo nível que caracterizam o ambiente. Por exemplo, após a aplicação de uma regra sobre o valor de $67km/h$ da velocidade, foi produzido o valor "Alta" para a característica "Velocidade" (indicada por c3 na figura 4.2). Por fim, os valores das características (e.g., "Noite", "Média", "Alta", "Baixa", "Outra") são atribuídos às respectivas características (e.g., c1, c2, c3, c4, c5) do vetor de características, o qual é adicionado à base de dados que será acessada pelo Gerente de Dados.

Os vetores de características criados podem ter dados com diferentes propriedades. Ou seja, algumas características não podem ser tratadas da mesma forma que outras tendo em vista os tipos de dados associados. Por exemplo, características nominais (i.e.,representadas por nomes) podem ser mapeadas em números inteiros através da associação dos nomes que ocorrem em números específicos, contudo, elas não possuem propriedades ordinais e não podem ser comparadas considerando a ordinalidade dos números inteiros. Por outro lado, as características numéricas geralmente possuem propriedades ordinais (i.e., podem ser ordenadas), e também podem ser mapeadas em nomes, agrupando os valores em intervalos, contudo ao tratá-las como variáveis nominais desconsidera-se a ordinalidade dessas características, o que também pode ser um problema.

Alguns exemplos de características nominais que podem ser extraídas na computação móvel: Estado da bateria (e.g., "Descarregando", "Carregando na USB", "Carregando na Tomada", "Carregado e Conectado na USB", "Carregado e Conectado na Tomada"), Aplicativo sendo usado em primeiro plano (e.g., "Facebook", "Instagram", "Banco do Brasil", "Gmail", etc.), Orientação do Aparelho (e.g., "Em pé", "Deitado", "De Lado"), Proximidade (e.g., "Próximo", "Distante") entre outros. Alguns exemplos de características numéricas que possuem propriedades ordinais e também podem ser extraídas na computação móvel: Temperatura, Luminosidade, Magnetismo, Velocidade, Intensidade de Aceleração, Turno e Intensidade do Som, entre outros. Durante a coleta os valores destes tipos de características devem ser agrupados em intervalos apro-

priados e irão retornar rótulos similares (e.g., “Nulo”, “Muito Baixo”, “Baixo”, “Médio”, “Alto”, “Muito Alto”).

A técnica proposta neste trabalho utiliza o cálculo da distância entre vetores, durante a classificação. O método utilizado, para calcular a distância, considera vetores com as mesmas características, as quais podem possuir propriedades diferentes entre si. Desta forma, é possível preservar a ordinalidade das características numéricas e ainda considerar dados nominais nos vetores de características. Por isso, é importante que a Camada de Coleta salve os vetores de características na base de dados, mas que também haja uma representação de quais características possuem propriedadades ordinais e quais são estritamente nominais, a fim de que haja a diferenciação desses tipos de características durante a inferência de contexto.

Camada de Gerenciamento

A Camada de Gerenciamento é uma interface entre as aplicações e os contextos extraídos do ambiente. Portanto, o principal objetivo desta camada é abstrair questões referentes à coleta, tratamento, extração de características e inferência de contextos com base nos dados brutos coletados do ambiente. Desta forma, os desenvolvedores de aplicações móveis sensíveis ao contexto podem se preocupar mais quanto ao que fazer (i.e., decisão) e como fazer (i.e., ação), em função das informações contextuais fornecidas.

A Camada de Gerenciamento possui quatro componentes principais: Base de Dados, Gerente de Contexto, Gerente de Dados e o Módulo de Inferência, o qual chamamos de CoRe-RL.

A Base de Dados (ilustrada na figura 4.3) é responsável pelo armazenamento e manutenção das seguintes informações: Uma base de vetores de características coletados; Uma representação simples dos tipos de cada característica usada nos vetores de características; Uma base de vetores de características aprendidos; Uma base para os rankings de características de cada contexto cadastrado.

Os vetores de características coletados são armazenados com uma marcação de tempo (*timestamp*) em uma base de vetores coletados, a qual possui certa volatilidade (i.e., as informações ficam armazenadas por um tempo limitado) a fim de que a memória dos dispositivos não seja comprometida. Desta forma, a estrutura destes vetores compreende apenas em um *timestamp* e uma lista de valores associados às características extraídas dos dados brutos.

A representação dos tipos das características usadas consiste em uma lista (i.e., um vetor) indicando quais características são nominais e quais possuem propriedades

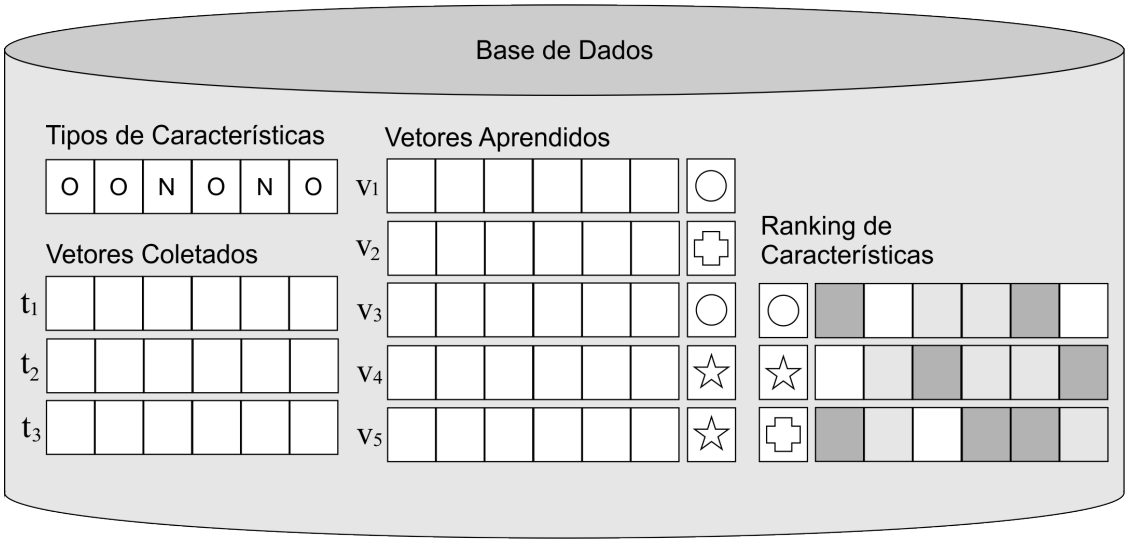


Figura 4.3: Exemplo de Base de Dados.

ordinais (na figura 4.3 esta indicação é feita com as letras 'O', para características ordinais, e 'N', para características nominais). Portanto, esta representação é, na verdade, uma padronização dos tipos de características presentes em todos os vetores utilizados no Mob Context.

A base de vetores aprendidos irá guardar os exemplos cadastrados por uma aplicação, ou aprendidos após a classificação de um vetor coletado. Desta forma, esta base irá crescer conforme o sistema for aprendendo a rotular os vetores coletados (reforços positivos). Mas alguns vetores também poderão ser esquecidos (i.e., removidos da base) após o recebimento de uma série de reforços negativos. Assim, os elementos armazenados nesta base possuirão um identificador único (i.e., um timestamp com a data de inserção na base), um rótulo (i.e., uma string indicando a qual contexto o vetor pertence), um nível de importância (i.e., indica se o vetor deve permanecer na base ou não) e a lista de valores associados às características do vetor.

Para cada novo contexto (rótulo) cadastrado, deve-se criar uma nova instância na base de rankings de características, a qual corresponde ao modelo que será aprendido sobre a relevância das características analisadas. Desta forma, os elementos desta tabela terão a seguinte estrutura: um rótulo único, referente a um contexto específico, e uma lista de números reais, onde cada item está associado a relevância de uma característica específica.

O Gerente de Dados é responsável pelo acesso à base de dados. Este componente contém a implementação de todos os métodos de acesso à Base de Dados (i.e., adição de novas instâncias, consulta, remoção e atualização das instâncias salvas).

O Gerente de Contextos é quem disponibiliza as informações contextuais de alto

nível para a Camada de Aplicação, respondendo a dois tipos principais de requisições: requisições de gerenciamento e requisições de inferência. As requisições de gerenciamento são aquelas relacionadas à consulta, adição e remoção dos contextos utilizados na inferência, enquanto as requisições de inferência são as solicitações de contextos e retorno dos sinais de reforço para os contextos inferidos.

O módulo de inferência faz a classificação dos vetores coletados, repassa os contextos inferidos para o Gerente de Contexto e utiliza os sinais de reforço, caso hajam, para melhorar o modelo utilizado na classificação (i.e., ranking de características) e modificar a base de exemplos aprendidos, aprendendo ou esquecendo vetores de características. A técnica CoRe-RL será explicada com mais detalhes na seção 4.2

Camada de Aplicação

Em um SSC, são as aplicações que interagem diretamente com o usuário, por meio de suas funcionalidades, prestação de serviços e execução de tarefas. Portanto, a camada de aplicação representa um nível de abstração onde as diferentes aplicações estarão implementadas. Embora a abordagem proposta não se preocupe em implementar as aplicações da camada de aplicação, o Mob Context não tem sentido algum se não houverem aplicações implementadas nesta camada. Além disso, será por meio das aplicações que o Mob Context será capaz de inferir e aprender como inferir contextos.

Assim, algumas das principais atribuições da Camada de Aplicação são: criação e remoção de rótulos para os contextos de interesse; solicitação de informações contextuais; decisão e ação de acordo com as informações contextuais obtidas; informação de reforços para os contextos inferidos.

Uma peculiaridade que diferencia o Mob Context das demais abordagens de SSC estudadas, é o gerenciamento flexível de contextos. Como o Mob Context aprende de forma incremental, é permitido que as aplicações consultem, cadastrem e removam contextos da base, por intermédio do Gerente de Contextos, de acordo com as suas necessidades.

Tendo em vista que a inferência de contextos é baseada na comparação de exemplos, durante a criação de um rótulo é necessário especificar algumas características que os vetores de características daquele rótulo provavelmente terão, ou cadastrar um conjunto de vetores de características associandos ao rótulo cadastrado (adiantando um pouco o processo de aprendizagem). Além disso, um contexto só poderá ser removido da base caso a aplicação decida por removê-lo. Portanto, mesmo que o último vetor de um rótulo específico receba uma série de reforços negativos, ele não poderá ser removido da base sem que a aplicação solicite a remoção do contexto. Desta forma, no pior

caso, a base de vetores aprendidos sempre terá pelo menos um vetor de características para cada contexto cadastrado.

O Mob Context funciona como um middleware distribuído que provê informações contextuais de alto nível às aplicações. As informações contextuais são fornecidas mediante as solicitações das próprias aplicações. Nestas solicitações, as aplicações podem especificar um limiar de corte, exigindo certa confiabilidade do resultado retornado pela camada de gerenciamento. Um limiar de corte alto (pela similaridade) faz com que a inferência retorne um vetor, da base de vetores aprendidos, que possua similaridade alta com relação ao vetor do estado atual do ambiente, caso nenhum vetor se adeque a esta condição, o gerenciamento não retornará nenhum contexto. Apesar de parecer ruim não retornar um contexto para uma solicitação de inferência, essa possibilidade é interessante quando o desempenho é mais importante que a aprendizagem, ou quando não for desejável correr o risco de errar e provocar ações inadequadas ao contexto do usuário.

Outra atribuição da Camada de Aplicação, fundamental para a inferência do Mob Context, consiste em informar os reforços observados a partir da interação com o usuário. Os reforços são recompensas e punições para as inferências realizadas, e podem ser obtidos explícitamente, solicitando que o usuário indique se gostou ou não do serviço (ação) realizado, ou implícitamente, avaliando como o usuário interagiu com a aplicação após a prestação do serviço (i.e., realização da ação). As recompensas são reforços positivos, e no Mob Context são representadas por inteiros maiores que zero, enquanto as punições são reforços negativos, representados por inteiros menores que zero.

Na aprendizagem por reforço, um agente inteligente é capaz de mapear estímulos (i.e., sinais de entrada, dados brutos, contextos) em respostas (i.e., comportamentos, ações). Desta forma, um SSC que aprende por reforço deve ser capaz de aprender a como se comportar, e não apenas a inferir contextos. Embora o Mob Context não realize esse mapeamento de forma direta, ele é feito de forma indireta, pois, ao informar contextos para a aplicação, esta poderá executar ações específicas de acordo com o contexto inferido. Uma vez que as aplicações realizem sempre as mesmas ações para os contextos inferidos (i.e., um rótulo produz sempre a mesma ação), então o sistema estará aprendendo a mapear contextos em ações, isto é, estímulos em respostas. Desta forma, a inferência de um determinado contexto produziria um comportamento específico, e o reforço para este comportamento também indicaria se a inferência foi adequada ou não para o contexto do usuário. Entretanto, se uma aplicação é capaz de produzir diferentes ações para um mesmo rótulo inferido, então, mesmo que a inferência tenha acertado, em alguns casos as ações poderão ser adequadas e em outros não.

Isto implicaria em reforços corretos para as ações executadas, porém inadequados para as inferências realizadas, o que atrapalharia na aprendizagem dos contextos.

4.2 CoRe-RL

Nesta seção será apresentada a técnica CoRe-RL de inferência de contexto que aprende por reforços, utilizada no Mob Context. O CoRe-RL é uma técnica baseada em exemplos cujo funcionamento depende de duas etapas principais – Classificação e Adaptação. A Classificação retorna um rótulo inferido para o vetor atual, enquanto a adaptação utiliza os reforços fornecidos pela aplicação para modificar o modelo do ambiente.

4.2.1 Classificação

Na etapa de classificação, o CoRe-RL aplica a técnica dos K vizinhos mais próximos para rotular novos vetores tendo em vista a base de exemplos aprendidos (ou fornecidos pela aplicação). Conforme apresentado na seção 2.3.2, o K-NN classifica um vetor de características pelo rótulo da classe com maior número de vizinhos dentre os K vizinhos mais próximos. O K-NN identifica os vizinhos mais próximos calculando as distâncias euclidianas entre o vetor atual e os demais vetores rotulados. Entretanto, a distância euclidiana não é aplicável aos diferentes tipos de variáveis (e.g.,, não suporta variáveis nominais).

Na literatura, existe um grande número de métodos de cálculo de distância entre vetores de características (e.g.,, distância euclidiana, distância de Hamming, distância de Mahalanobis, coeficiente de correspondência simples, distância de Gower). Esses métodos possuem diversas particularidades que os qualificam para diferentes cenários de aplicação. Neste trabalho, optou-se pela utilização do método de Gower [J. C. Gower, 1971] (explicado na seção 2.3.2, na subseção referente ao K-NN), pois é robusto ao ponto de suportar características de diferentes tipos (e.g.,, características nominais, ordinais, de intervalos e de taxas) e ainda considerar a atribuição de pesos no cálculo da distância. Desta forma, o CoRe-RL não fica restrito a um tipo específico de dado e, ainda pode atribuir pesos às características de acordo com o modelo aprendido.

A classificação feita pelo CoRe-RL é baseada em exemplos (vetores aprendidos e rotulados), mas também considera um modelo aprendido mediante o recebimento de reforços. Este modelo consiste no ranqueamento das características conforme a relevância para a inferência de um contexto específico. Desta forma, para cada contexto (rótulo) cadastrado haverá um modelo (ranking de características) associado, o qual será melhorado, mediante a ocorrência de reforços positivos e negativos. Este ran-

king de características irá ponderar as características, valorizando as características mais importantes e permitindo que variações nas características menos importantes tenham pouca influência no cálculo da distância. A forma de atualização do ranking de características será apresentada na seção 4.2.2.

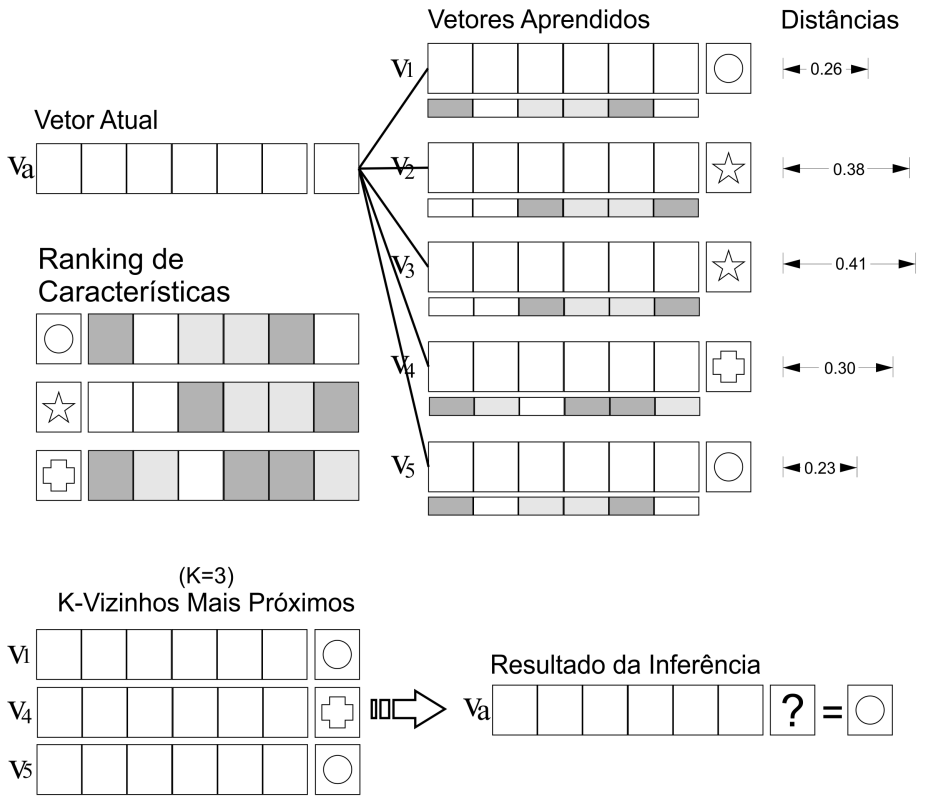


Figura 4.4: Esquema da Classificação usando o K-NN e distância de Gower.

A figura 4.4 ilustra o comportamento da etapa de classificação do CoRe-RL usando o K-NN, onde calcula-se a distância de Gower entre o vetor atual v_a (não rotulado) e os vetores aprendidos v_1, v_2, v_3, v_4, v_5 presentes na base de exemplos do CoRe-RL. As características desses vetores são ponderadas de acordo com o rótulo que possuem (os rótulos estão representados na figura por uma bola, uma estrela ou uma cruz), sendo que os pesos correspondem ao ranking de características aprendido para cada rótulo. Após o cálculo das distâncias, são identificados os K vizinhos mais próximos de v_a. No exemplo da figura 4.4 para o valor de $K = 3$ foram retornados os vetores v_1, v_4 e v_5 como os três vizinhos mais próximos (i.e., com menores distâncias). Em seguida, é possível obter o rótulo predominante dentre os rótulos dos vizinhos mais próximos, no exemplo ilustrado o resultado da inferência seria "bola".

A classificação usando o K-NN sempre retorna um rótulo de algum vetor presente

na base de exemplos. Portanto, até nos casos em que o vetor atual estiver muito distante dos seus vizinhos o K-NN irá retornar algum rótulo. Contudo, nem sempre este comportamento é desejável em SSC's, pois o sistema sempre retornará algum contexto, dentre aqueles cadastrados, mesmo quando o usuário não estiver em nenhum deles. Por este motivo, o CoRe-RL permite que as aplicações definam um limiar de corte baseado na similaridade ($S(x, y)$) ou distância de Gower ($D(x, y)$), a fim de que a classificação retorne o rótulo dos k vizinhos mais próximos cuja similaridade seja $S(x, y) \geq s$ ou cuja distância seja $D(x, y) \leq d$, onde $d = 1 - s$. Por exemplo, se o limiar de corte pela similaridade é $s = 0.95$, então o CoRe-RL só irá retornar algum contexto caso pelo menos 1 dos K vizinhos tenha similaridade $S(x, y) \geq 0.95$, ou seja, quando a distância for $D(x, y) \leq 0.05$.

O limiar de corte é determinado pelas aplicações no momento da solicitação de uma inferência. Desta forma, permite-se que as aplicações tenham certa confiança nos resultados obtidos, o que também possibilita o controle quanto ao que priorizar - desempenho ou exploração. Enquanto um limiar alto (de corte pela similaridade) valoriza o desempenho, a exploração é incentivada quando este limiar é baixo. Caso o limiar de corte não seja informado, o gerente de contexto considerará que $s = 0$ e $d = 1$, assim, não haverão cortes e o CoRe-RL sempre retornará algum resultado.

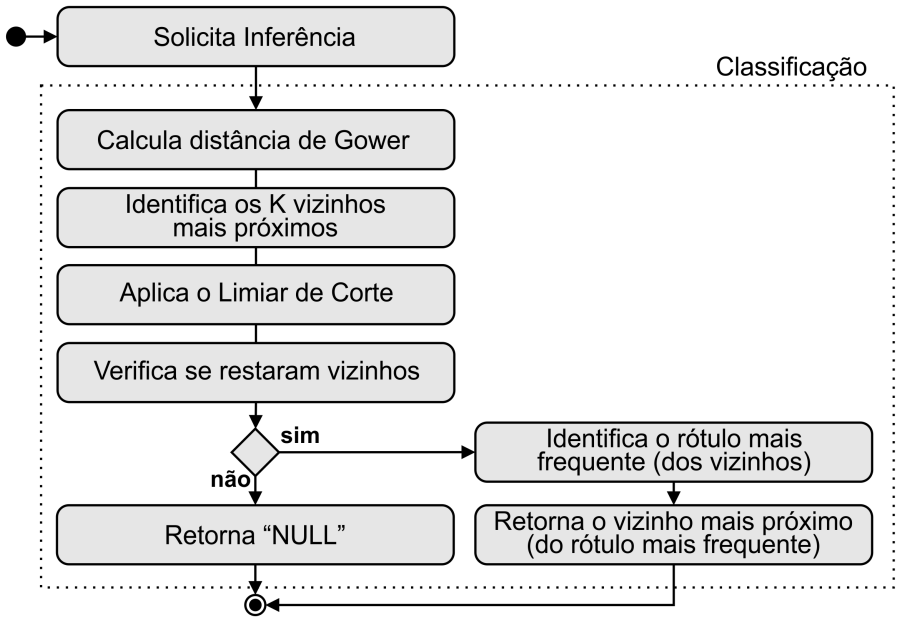


Figura 4.5: Fluxograma da etapa de Classificação do CoRe-RL.

O fluxograma da figura 4.5 apresenta o algoritmo usado no CoRe-RL para classificar um vetor de características não rotulado. A classificação ocorre mediante solicitações feitas ao Gerente de Contextos e repassadas ao CoRe-RL. Quando uma solicitação

de inferência é realizada, o Gerente de Contextos informa os parâmetros de entrada necessários afim de que a classificação ocorra (i.e., o vetor atual a ser rotulado e o limiar de corte a ser considerado). Em seguida, o CoRe-RL calcula as distâncias de gower do vetor atual para os demais vetores aprendidos. A identificação dos K-vizinhos mais próximos ordena os vetores pelas distâncias e retorna apenas os K vetores com as menores distâncias. A etapa seguinte aplica o limiar de corte aos k-vizinhos, eliminando aqueles cujas distâncias são superiores ao limiar de corte pela distância. Em seguida, é verificado se existem vizinhos que não foram cortados, para que seja identificado o rótulo mais frequente dentre os rótulos dos K vizinhos mais próximos. Por fim, a classificação do CoRe-RL retornará o vetor mais próximo que possui o rótulo mais frequente (identificado na etapa anterior). Além disso, o CoRe-RL também poderá retornar "Null" caso todos os vizinhos tenham sido cortados, indicando que os mesmos não atenderam à condição do limiar de corte.

É importante notar que, em vez de retornar apenas o rótulo mais frequente dos vizinhos mais próximos (conforme ilustrado na figura 4.4), o algoritmo proposto retorna o vetor mais próximo cujo rótulo é o mais frequente dentre os K vizinhos. Este vetor será usado durante a etapa de adaptação.

O CoRe-RL utiliza o K-NN na classificação de vetores de características. Entretanto, para usar o K-NN é importante definir um valor K de vizinhos mais próximos a serem considerados. A escolha do valor de K depende da relação entre o número de rótulos e o número de exemplos da base de vetores aprendidos. Geralmente, os classificadores que utilizam o K-NN sabem de antemão a base que será aprendida e consequentemente os rótulos que serão aprendidos dessa base, ainda assim o valor de K é determinado empiricamente. No CoRe-RL, a base usada vai sendo aprendida de acordo com os reforços positivos para as classificações, portanto, não é possível determinar o número de exemplos e de rótulos que serão aprendidos. Forster et al. [2010] apresenta uma forma de utilizar o K-NN em um classificador incremental, e propõe que o valor de K seja uma porcentagem do total de exemplos aprendidos. Mesmo assim, o valor da porcentagem utilizada é obtido empíricamente e não existe uma porcentagem que seja ótima para todos os casos. A heurística usada por Forster et al. [2010] não considera a possibilidade de que exemplos de novos rótulos sejam cadastrados incrementalmente. Por isso, neste trabalho, o valor de K é dado pela razão entre o número de exemplos aprendidos e o número de contextos cadastrados, sem exceder um K_{max}. Contudo, fica como sugestão de experimentos futuros a utilização de outras heurísticas para a determinação dinâmica do valor de K, ou experimentos com outros valores de K_{max}.

4.2.2 Adaptação

Os exemplos usados na classificação de contextos do CoRe-RL são vetores de características rotulados, que podem ser cadastrados diretamente pelas aplicações, ou aprendidos após o recebimento de reforços positivos para as inferências realizadas. Portanto, à medida que o sistema vai inferindo contextos corretamente, novos vetores vão sendo adicionados à base de vetores aprendidos, até o ponto que todos ou a maioria dos vetores de um determinado contexto, sejam aprendidos (adicionados à base). Desta forma, se a base de exemplos for suficientemente representativa para um contexto X, quando um novo vetor v deste mesmo contexto for ser rotulado, a distância de v para algum vetor $x \in X$ será nula (i.e., $v = x$) ou muito próxima de zero (i.e., $v \approx x$). Neste tipo de aprendizagem, sempre que uma classificação correta for realizada para um vetor de características que ainda não está na base, o sistema deverá adicioná-lo à base com o rótulo inferido, possibilitando que na próxima ocorrência deste vetor, ou de um vetor muito parecido a classificação retorne o mesmo rótulo do vetor aprendido.

A computação móvel sensível ao contexto extrai informações contextuais a partir de uma variedade de sensores, portanto existe um grande número de variáveis que caracterizam o ambiente físico e o usuário. Embora as combinações de todos os possíveis estados representáveis por essas características seja muito grande, a maioria desses estados não ocorre, ou ocorre com pouca frequência. Além disso, não são todos os estados que ocorrem que representam contextos úteis para as aplicações e usuários. Portanto, considerando que o CoRe-RL só aprende vetores que representam contextos úteis para aplicações e usuários, não é necessário que o sistema tenha capacidade de armazenar todas as combinações possíveis de estados.

A capacidade de esquecer é outro aspecto importante na sensibilidade ao contexto, principalmente em ambientes dinâmicos, pois nem tudo que se aprende é correto, e os conceitos aprendidos podem mudar ao longo do tempo, de acordo com as mudanças do ambiente. Por este motivo o CoRe-RL esquece (i.e., remove da base de vetores aprendidos) exemplos que produzem uma série de reforços negativos, afim de que estes exemplos ruins não comprometam o desempenho da inferência. Desta forma, quando um vetor v é inferido como do rótulo X, sendo x o seu vizinho mais próximo que possui o rótulo X, e ocorre uma série de reforços negativos (i.e., $v \notin X$), então x deverá ser esquecido. Desta forma, sem este vizinho, nas próximas inferências o resultado retornado poderá ser diferente, possibilitando que o sistema associe aquele vetor a outros rótulos. Assim, esquecer é importante para reaprender conceitos.

O esquecimento de exemplos no CoRe-RL considera que para cada vetor aprendido existe um campo τ que representa o nível de importância do vetor. Inicialmente,

quando um vetor $x \in X$ é aprendido ou cadastrado à base, o nível de importância é máximo (i.e., $\tau = \tau_{max}$). Conforme x é retornado na classificação e reforçado negativamente, o nível de importância de x vai sendo decrementado, enquanto $\tau > 0$. Por fim, o vetor x deverá ser esquecido quando o nível de importância do vetor for $\tau \leq 0$. Por outro lado, à medida que o vetor x recebe reforços positivos, para o contexto X, o seu nível de importância deverá ser incrementado, enquanto $\tau < \tau_{max}$, sem exceder τ_{max}.

Conforme apresentado, a aprendizagem baseada em exemplos é bastante simples, pois requer apenas que se busque, entre os exemplos vistos, o que mais se aproxima daquele que será classificado. Embora seja possível utilizar indexação (e.g.,, tabelas hash) com o intuito de reduzir o tempo de busca, a aplicabilidade deste tipo de técnica é delimitada devido a sua capacidade limitada de generalização [Kazakov & Kudenko, 2001]. Por esse motivo, o CoRe-RL busca aprender, além de exemplos, um modelo para cada contexto cadastrado, o qual irá impactar na classificação de novos vetores.

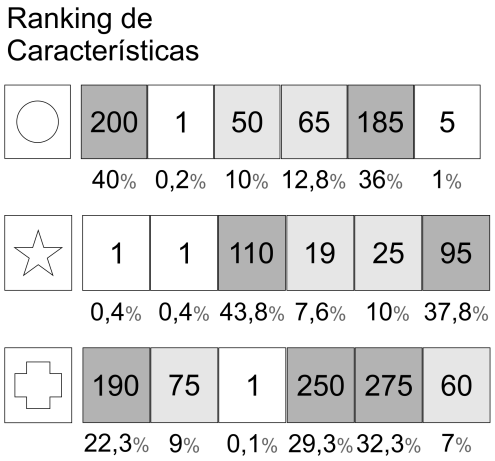


Figura 4.6: Exemplo de Ranking de Características e seus respectivos pesos.

O modelo aprendido pelo CoRe-RL consiste em um ranking de características, que indica a relevância de cada característica para a inferência de um determinado contexto. Ao utilizar este ranking na ponderação das características, durante o cálculo da distância de Gower, mesmo com uma pequena base de exemplos, é possível valorizar as características de maior importância e minimizar os efeitos daquelas que são pouco relevantes durante a classificação.

A figura 4.6 exemplifica os rankings de características para os contextos (rótulos) "bola", "estrela" e "cruz". Para cada rótulo cadastrado, é criado um ranking contendo as pontuações para as características dos vetores comparados, o qual é alterado de acordo com o recebimento de reforços para aquele rótulo. Os valores exibidos dentro dos rankings da figura 4.6 são as pontuações de cada característica para o respectivo contexto,

e abaixo estão as porcentagens referentes aos pesos de cada característica de acordo com as suas pontuações. Por exemplo, para o contexto "bola", a primeira característica tem 200 pontos, o que representa aproximadamente 40% do somatório da pontuação das características do contexto "bola", portanto, quando um vetor não rotulado vier a ser comparado com um vetor qualquer do tipo "bola", a primeira característica terá 40% de influência no cálculo da distância de Gower entre esses dois vetores.

No CoRe-RL os reforços φ podem ser números inteiros positivos ou negativos, indicando recompensas ou punições respectivamente. O CoRe-RL assume que os reforços avaliam a corretude da inferência. Portanto, se a classificação de um vetor v retorna o vetor $x \in X$, então cria-se a hipótese que $v \in X$. Caso ocorra um reforço $\varphi > 0$, então conclui-se que a hipótese $v \in X$ é verdadeira, caso $\varphi < 0$, então $v \notin X$. Por isso, quando $\varphi > 0$, adiciona-se o vetor v, com o rótulo X, à base de vetores aprendidos.

Como o objetivo do ranking de características é modelar a relevância de cada característica para os diferentes contextos na ocorrência de reforços, então a atualização do ranking deverá valorizar aquelas que mais contribuiram para o acerto (reforço positivo) ou erro (reforço negativo) da inferência. Portanto, de acordo com o reforço φ para o contexto inferido X, a atualização do ranking R_X^j para uma característica j, é dada pela seguinte fórmula:

$$R_X^j = R_X^j + s_{v,x}^j * \varphi \tag{4.1}$$

Onde $s_{v,x}^j$ é a similaridade entre os valores das características j dos vetores v e x (definida pelas fórmulas 2.9 e 2.10). O valor de $s_{v,x}^j$ varia entre 0, quando as características j dos vetores v e x diferem de forma máxima, e 1, quando possuem valores exatamente iguais. Assim, quanto maior for a similaridade das caracterítiscas j de v e x, maior será o incremento sobre R_X^j. Pois, conforme $s_{v,x}^j \to 1$, o incremento sobre o ranking tenderá a φ. Além disso, quando $\varphi > 0$ o incremento sobre o ranking R_X^j será positivo, valorizando as similaridades que levaram o sistema a acertar, e quando $\varphi < 0$ o incremento será negativo (i.e., ocorrerá um decremento), penalizando as similaridades que levaram o sistema ao erro. Desta forma, serão reforçadas com maior intensidade as características que mais contribuiram com o valor da similaridade (i.e., que menos influenciaram na distância entre os vetores).

A figura 4.7 contém um fluxograma que apresenta os passos envolvidos na etapa de adaptação. Após uma inferência, o CoRe-RL pode receber ou não um reforço. Caso não receba, não há adaptação, o sistema irá continuar inferindo contextos como se nada tivesse acontecido. Quando o gerente de contexto informar um reforço, para uma determinada inferência, o CoRe-RL fará a atualização do ranking de características de

acordo com o reforço e as similaridades entre as características do vizinho mais próximo (retornado na classificação) e o vetor que foi classificado. Após atualizar o ranking de características, o CoRe-RL reforça também o vetor do vizinho mais próximo, de modo que após uma série de reforços negativos, ele venha a ser esquecido. Em seguida, é verificado se o reforço é positivo ou negativo. Quando o reforço for positivo, o vetor classificado deverá ser adicionado à base de vetores aprendidos, caso ainda não tenha sido adicionado. Quando o reforço for negativo, é importante verificar se o vetor do vizinho mais próximo (retornado na classificação) precisa ser esquecido, caso o nível de importância deste vetor seja $\tau \leq 0$ então ele deverá ser removido da base de vetores aprendidos.

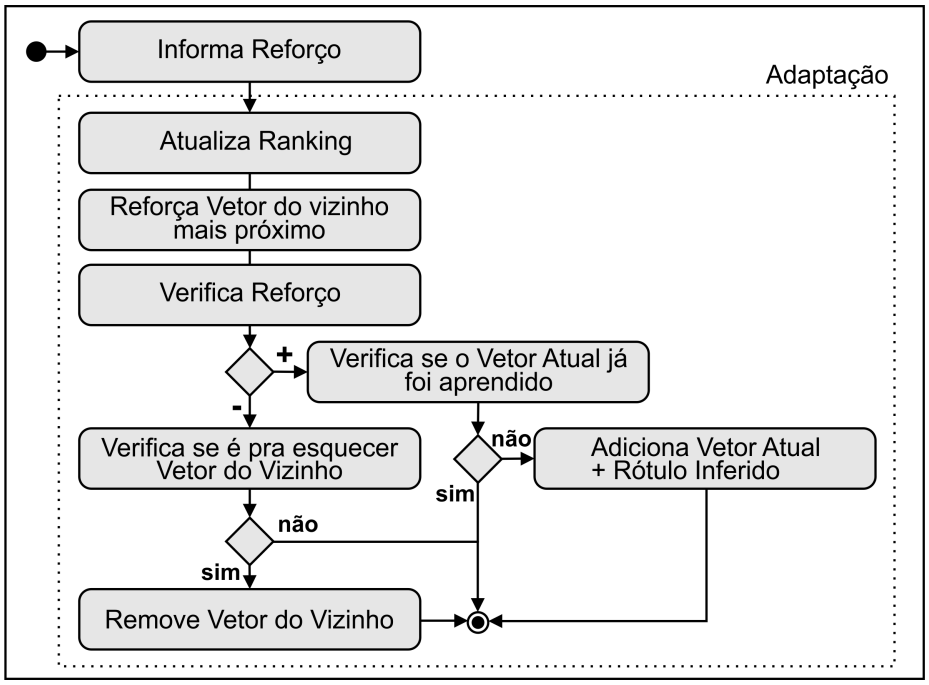


Figura 4.7: Fluxograma da etapa de Adaptação do CoRe-RL.

4.3 Considerações Sobre o Capítulo

Neste capítulo apresentou-se o Mob Context, que consiste em uma abordagem de arquitetura distribuída voltada para aplicações de dispositivos móveis e que suporta aprendizagem por meio de reforços. Esta abordagem dá suporte à adaptação na inferência de contexto, pois não utiliza um modelo estático na classificação, mas aprende a inferir contextos de modo incremental. Isto possibilita que o sistema aprenda modelos específicos para usuários específicos (ajustabilidade), que o desempenho da inferência melhore (adaptabilidade) com o passar do tempo (conforme ocorrem inferências e re-

forços) e que o sistema seja capaz de se adequar, caso ocorram mudanças nas condições do ambiente que podem influenciar na percepção dos contextos.

O Mob Context é um framework conceitual que estabelece os principais requisitos necessários para a aplicação do CoRe-RL, que consiste em uma técnica de inferência de contexto capaz tanto de classificar vetores de características não rotulados com base em exemplos aprendidos, como de adaptar-se de acordo com a ocorrência de reforços positivos ou negativos.

O CoRe-RL utiliza o método dos K vizinhos mais próximos para fazer a classificação de vetores não rotulados. Contudo, em vez de usar distância euclidiana, o CoRe-RL utiliza a distância ou similaridade de Gower, ponderando as características de acordo com as relevâncias das mesmas para cada contexto. A utilização do método de Gower permite que diferentes tipos de características sejam usados para compor os vetores de características que poderão ser aprendidos e cadastrados, suportanto características nominais além das características numéricas (i.e., com propriedades ordinais).

A partir da determinação de um limiar de corte, as aplicações podem estabelecer limites mínimos obrigatórios de similaridade para que um contexto seja retornado pela inferência. Assim, em vez de sempre retornar algum contexto mesmo quando o usuário não estiver em nenhuma contexto, a classificação também poderá retornar respostas negativas informando que não foi possível classificar o vetor observado para o limiar de corte indicado.

A utilização de um ranking de características (modelo) permite que a aprendizagem (que é baseada em exemplos) ocorra mesmo na ocorrência de reforços negativos, mas sem exigir que vetores reforçados negativamente sejam armazenados temporariamente para auxiliar nas próximas classificações. Além disso, os pesos atribuídos às características inserem um viés que aproximam vetores cujas características relevantes são idênticas ou muito próximas, enquanto distancia vetores entre si, quanto mais distantes forem as características relevantes.

Além disso, o CoRe-RL também é capaz de esquecer vetores que atrapalham na inferência e poderiam desbalancear significativamente (a longo prazo) o ranking de características aprendido. Desta forma, os conceitos aprendidos irão durar enquanto não estiverem sendo frequentemente reforçados negativamente. Permitindo assim, que o sistema esqueça vetores que não produzem mais reforços positivos e reaprenda a rotular estes vetores. Portanto, caso ocorra uma mudança de conceito de um contexto X, no pior caso, o sistema levará $n_{vet} * \tau_{max}/\varphi$ interações para esquecer todos os vetores associados ao conceito antigo, onde n_vet é a quantidade de vetores do contexto X, que devem ser esquecidos, τ_{max} é o nível máximo de importância de cada vetor e φ é o reforço que decrementa o nível de importância.

Capítulo 5

Estudo de Caso e Resultados

> Se você quer os acertos, esteja preparado para os erros.
>
> Carl Yastrzemski

Neste capítulo, a técnica de inferência proposta (i.e., CoRe-RL) foi implementada e avaliada em um estudo de caso da abordagem Mob Context. Assim, as principais funcionalidades das camadas de coleta e gerenciamento de contexto foram implementadas em uma aplicação móvel, cujo objetivo é testar e avaliar o CoRe-RL. Portanto, não foi desenvolvida uma aplicação móvel sensível ao contexto, mas sim um aplicativo que permite ao usuário interagir com o CoRe-RL de modo similar às interações apresentadas para a camada de aplicação do Mob Context (e.g., solicitar inferências, retornar sinais de reforço, gerenciar contextos).

Na seção 5.1, serão apresentados detalhes referentes à implementação do aplicativo desenvolvido. Em seguida, na seção 5.2, serão discutidos os cenários e as condições em que o CoRe-RL foi avaliado e a metodologia utilizada nos experimentos. Por fim, na seção 5.3, serão apresentados os resultados obtidos de acordo com os experimentos realizados.

5.1 Estudo de Caso

Mob Context App é o nome do aplicativo desenvolvido como estudo de caso deste trabalho, cujo propósito é servir como uma ferramenta para a validação e avaliação da técnica de inferência proposta. Desta forma, focou-se na implementação das principais funcionalidades das camadas de coleta e gerenciamento diretamente no próprio aplicativo, em vez de utilizar módulos independentes (como é proposto na seção 4.1).

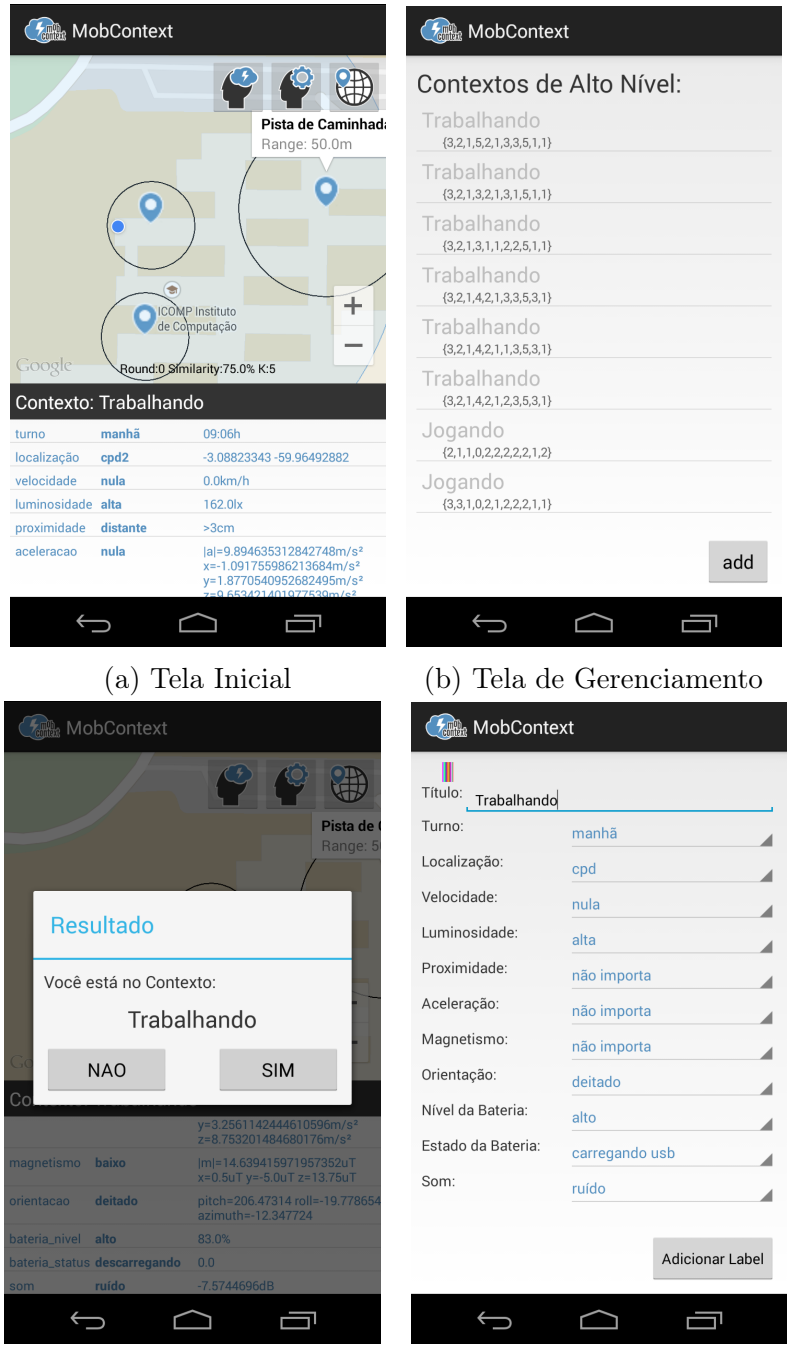
O Mob Context App não é uma aplicação sensível ao contexto, pois, embora faça inferência de contextos do ambiente, não modifica seu comportamento de acordo com os contextos inferidos. Por outro lado, o aplicativo transfere para o usuário as atribuições da camada de aplicação, ao permitir que o usuário interaja com o CoReRL de modo similar às interações entre camada de aplicação e gerenciamento, propostas na abordagem Mob Context. Ou seja, o usuário é quem deverá ser capaz de criar e remover rótulos para os contextos de interesse; solicitar informações contextuais; e informar os reforços adequados para os contextos inferidos.

Em resumo, o Mob Context App utiliza os sensores disponíveis para coletar dados brutos do ambiente; realiza a extração de características por meio de regras; utiliza essas informações do ambiente para inferir contextos de alto nível, cadastrados pelo usuário; após a inferência, o aplicativo exibe o resultado e pergunta se a inferência está correta; e informa o sinal de reforço para a adaptação da inferência, de acordo com a resposta do usuário. Além disso, o aplicativo também exibe os dados brutos coletados e os valores das características extraídas; permite o cadastro (i.e., adição), remoção, renomeação e visualização dos contextos cadastrados e seus vetores de características; possibilita que, caso necessário, o usuário adicione rótulos a locais de interesse para a inferência, os quais poderão ser usados como valores da característica "localização".

A figura 5.1a apresenta a tela inicial que possui: um mapa, três botões no canto superior direito, uma lista contendo as características e valores de características extraídas dos dados brutos coletados em determinado instante, e o contexto de alto nível inferido (caixa com fundo preto, entre o mapa e os dados coletados). O mapa serve para que o usuário visualize, além da sua própria localização (coletada pelo GPS), as localizações cadastradas na base de localizações do dispositivo. Através do mapa também é possível remover e cadastrar novos locais. Os botões do canto superior servem para: solicitar uma inferência de contexto que poderá receber reforço (botão da esquerda); acessar a tela de gerenciamento de contextos (botão do meio); e atualizar a lista de dados coletados (botão da direita).

A tela de gerenciamento de contextos é apresentada na figura 5.1b, nela é exibida a lista de vetores de características da base de vetores aprendidos. No caso, os números exibidos em baixo de cada rótulo são os valores armazenados das características de cada vetor. Esta tela permite que se cadastrem novos vetores ou rótulos a serem aprendidos (clicando no botão "add" no canto inferior direito); que vetores específicos ou rótulos sejam removidos da base de vetores (clicando e segurando sobre o vetor que deverá ser excluído); que sejam visualizados, por escrito, os valores das características de um determinado vetor (através de um clique simples sobre o vetor que se deseja visualizar).

A tela de cadastro de contextos (apresentada na figura 5.1d) permite que o usuá-



(a) Tela Inicial (b) Tela de Gerenciamento

(c) Resultado da Inferência (d) Tela de Cadastro

Figura 5.1: Exemplos de Telas do Mob Context App.

rio informe o valor de uma ou mais características para um contexto específico. Desta forma, é possível inserir tanto vetores completos (i.e., com todos os valores de características preenchidos), permitindo que 1 ou mais vetores de exemplo sejam cadastrados; como vetores incompletos (i.e., com alguns campos não preenchidos, na figura 5.1d consta o termo "não importa"), informando apenas os valores das características rele-

vantes (a priori). Quando o primeiro vetor cadastrado (para um contexto específico) é um vetor completo, o Mob Context App assume que todas as características são igualmente relevantes, e atribui pesos iguais para cada característica. Quando o primeiro vetor cadastrado for incompleto, o Mob Context App atribuirá pesos maiores para as características que foram informadas e menores para as características não preenchidas. Além disso, após uma inferência bem-sucedida (i.e., que recebeu reforço positivo), onde um vetor incompleto é retornado, os campos que faltavam são preenchidos com as características do novo vetor.

A figura 5.1c apresenta o resultado de uma inferência e as opções de reforço disponíveis: "NÃO" (reforço negativo) e "SIM" (reforço positivo). Após o usuário informar o tipo de reforço, o sistema irá realizar a etapa de adaptação do CoRe-RL.

Tabela 5.1: Lista de Possíveis Valores das Características Extraídas

Características	Tipo	*Valores das Características*					
Turno	Ordinal	*Madrugada*	*Manhã*	*Tarde*	*Noite*		
Localização	Nominal	*Outro*	*CPD*	*Casa*	*Pista de Caminhada*		
Velocidade	Ordinal	*Nula*	*Caminhando*	*Bicicleta*	*Média*	*Alta*	*Altíssima*
Nível de Luminosidade	Ordinal	*Nula*	*Baixa*	*Média*	*Alta*	*Altíssima*	
Proximidade	Nominal	*Próximo*	*Distante*				
Nível de Aceleração	Ordinal	*Nulo*	*Baixo*	*Médio*	*Alto*	*Altíssimo*	
Nível de Magnetismo	Ordinal	*Baixo*	*Normal*	*Médio*	*Alto*	*Altíssimo*	
Orientação	Nominal	*Em Pé*	*De Lado*	*Deitado*			
Intensidade do Som	Ordinal	*Ruído*	*Baixo*	*Médio*	*Alto*	*Altíssimo*	
Nível de Bateria	Ordinal	*Nulo*	*Baixo*	*Médio*	*Alto*	*Carregado*	
Estado da Bateria	Nominal	*Descarregando*	*Carregando AC*	*Carregando USB*	*Carregado AC*	*Carregado USB*	*Desconhecido*

O aplicativo criado foi implementado sobre a plataforma Android de dispositivos móveis, e os dados coletados são os dados produzidos pelos sensores físicos de um aparelho Moto G (da primeira geração). Sendo que, foram usadas as seguintes fontes de dados: gps, luminosidade, proximidade, acelerômetro, magnetismo, orientação, áudio, bateria e relógio. Após a aplicação de regras sobre os dados brutos, foram extraídas as seguintes características: turno, localização, velocidade, nível de luminosidade, proximidade, nível de aceleração, nível de magnetismo, orientação, intensidade do som, nível

de bateria, estado da bateria. Os possíveis valores dessas características (listados na tabela 5.1) são mapeados e tratados como números inteiros na aplicação. Além disso, como o conjunto de características usado não muda, a diferenciação entre características nominais e ordinais, também apresentada na tabela 5.1, foi feita diretamente em código.

A extração de características produz vetores de características similares ao exemplo de vetor atual (não rotulado) da tabela 5.2. Esses vetores são armazenados temporariamente em uma tabela de vetores coletados. No processo de classificação, o vetor coletado será comparado com os vetores da base de vetores aprendidos. Esta comparação é feita a partir do cálculo da distância de Gower, atribuindo como pesos para as características os valores armazenados na tabela de ranking de características. A tabela 5.2 exemplifica um vetor coletado não rotulado (chamado como "Vetor atual"), uma base de vetores aprendidos (i.e., que foram cadastrados ou aprendidos mediante reforços positivos) e os rankings de características associados aos contextos cadastrados (i.e., "Dirigindo" e "Trabalhando").

Tabela 5.2: Um vetor não rotulado, uma base de vetores aprendidos e o ranking de características

Vetor Atual (não rotulado)

id	rótulo	turn	loca	velo	lumi	prox	acel	magn	orie	som	nbat	ebat
1	????	manhã	CPD	nula	baixa	distante	baixo	médio	de lado	ruído	carregado	descarregando

Base de Vetores Aprendidos

id	rótulo	turn	loca	velo	lumi	prox	acel	magn	orie	som	nbat	ebat
1	Dirigindo	manhã	outro	alta	alta	distante	médio	médio	deitado	médio	médio	descarregando
2	Dirigindo	manhã	outro	alta	altíssima	distante	baixo	alto	deitado	alto	alto	descarregando
3	Dirigindo	tarde	outro	média	nula	próximo	baixo	normal	deitado	médio	baixo	descarregando
4	Trabalhando	manhã	CPD	nula	alta	distante	nulo	normal	deitado	baixo	médio	carregando_ac
5	Trabalhando	manhã	CPD	nula	média	distante	nulo	normal	em pé	ruído	alto	carregando_usb

Ranking de Características

rótulo	turn	loca	velo	lumi	prox	acel	magn	orie	som	nbat	ebat
Dirigindo	20	150	120	5	5	1	70	75	80	20	5
Trabalhando	50	200	100	60	15	5	20	10	70	1	55

O exemplo da tabela 5.2 apresenta os valores das características por escrito, como se todas as características fossem nominais. Contudo, na prática, o Mob Context App mapeia esses valores de características em números inteiros positivos. Assim, as características nominais continuam sendo tratadas como nominais (i.e., calcula-se $s^j_{x,y}$ pela equação 2.9) e as que possuem propriedades ordinais podem receber o devido tratamento (i.e., calcula-se $s^j_{x,y}$ pela equação 2.10) no cálculo da distância de

Gower. Desta forma, a tabela 5.3 representa um mapeamento em inteiros dos valores apresentados na tabela 5.2.

Tabela 5.3: Representação numérica da tabela 5.2

Vetor Atual (não rotulado)

id	rótulo	turn	loca	velo	lumi	prox	acel	magn	orie	som	nbat	ebat
1	????	2	2	1	2	2	2	3	2	1	5	1

Base de Vetores Aprendidos

id	rótulo	turn	loca	velo	lumi	prox	acel	magn	orie	som	nbat	ebat
1	Dirigindo	2	1	5	4	2	3	3	3	3	3	1
2	Dirigindo	2	1	5	5	2	2	4	3	4	4	1
3	Dirigindo	3	1	4	1	1	2	2	3	3	2	1
4	Trabalhando	2	2	1	4	2	1	2	3	2	3	2
5	Trabalhando	2	2	1	3	2	1	2	1	1	4	3

Ranking de Características

rótulo	turn	loca	velo	lumi	prox	acel	magn	orie	som	nbat	ebat
Dirigindo	20	150	120	5	5	1	70	75	80	20	5
Trabalhando	50	200	100	60	15	5	20	10	70	1	55

Para os vetores da tabela 5.3, o vizinho mais próximo será o vetor com id=5 da base de vetores aprendidos. O cálculo da distância entre o vetor atual (não rotulado) e os vetores da base de vetores aprendidos depende principalmente de dois elementos: as similaridades individuais para cada característica dos vetores (i.e., $s^j_{x,y}$); e os pesos atribuídos à estas similaridades (i.e., ranking de características). Como os pesos das características para o vetor 5 já são conhecidos (ranking para o contexto "Trabalhando"), então resta calcular as similaridades individuais $s^j_{x,y}$ entre as características do vetor atual e do vetor 5. As características que possuem valores idênticos entre os vetores terão $s^j_{x,y} = 1$ (e.g., turno, localização, velocidade, proximidade e som), as que possuem valores distintos e são nominais terão $s^j_{x,y} = 0$ (e.g., orientação, estado da bateria), as demais são características ordinais que possuem valores distintos (e.g., luminosidade, aceleração, magnetismo, nível da bateria) e deverão ser calculadas pela fórmula da equação 2.10.

A tabela 5.4, apresenta os valores de $s^j_{x,y}$ usados no cálculo da distância de gower entre o vetor 5 e o vetor atual da tabela 5.3. Multiplicando-se as similaridades da tabela 5.4 pelos respectivos pesos do ranking de características para o contexto associado ao vetor 5 ("Trabalhando") e dividindo o resultado pelo somatório de todos os pesos,

Tabela 5.4: Similaridades parciais entre o vetor atual e o vetor 5 da tabela 5.3

	turn	**loca**	**velo**	**lumi**	**prox**	**acel**	**magn**	**orie**	**som**	**nbat**	**ebat**
$s^j_{x,y}$	1	1	1	0,75	1	0,75	0,75	0	1	0,75	0

obtém-se a similaridade de gower $S(x, y) = 0,8524$ e a distância de gower $D(x, y) = 0,1476$ entre os dois vetores. Desta forma, caso o limiar de corte pela similaridade, setado no aplicativo, seja inferior a $S(x, y)$, então a classificação retornará o vetor 5 como resultado da inferência, caso contrário, nenhum resultado será retornado.

Na ocorrência de reforço sobre o vetor 5, as similaridades da tabela 5.4 irão ponderar o reforço sobre o ranking de características (conforme apresentado na equação 4.1). Assim, se o vetor retornado for reforçado negativamente ($\varphi < 0$), então as características do rótulo associado a este vetor serão decrementadas em $\varphi * s^j_{x,y}$, caso o reforço seja positivo, estas características serão incrementadas da mesma maneira. Por exemplo, considerando um reforço $|\varphi| = 50$ para o vetor 5, retornado como resultado da inferência do vetor atual da tabela 5.3, as características do contexto "Trabalhando" no ranking de características serão incrementadas ou decrementadas dos seguintes valores: $turn = \pm 50$; $loca = \pm 50$; $velo = \pm 50$; $lumi = \pm 37,55$; $prox = \pm 50$; $acel = \pm 37,5$; $magn = \pm 37,5$; $orie = \pm 0$; $som = \pm 50$; $nbat = \pm 37,5$; $ebat = \pm 0$;

Os passos envolvidos na classificação e na adaptação são os mesmos descritos pelos fluxogramas das figuras 4.5 e 4.7. No estudo de caso, o valor padrão dos reforços foi $\varphi = \pm 25$. Este valor também foi usado para incrementar e decrementar o nível de importância τ dos vetores aprendidos, cujo valor máximo é $\tau_{max} = 100$. Portanto, para cada reforço negativo o nível de importância do vetor é decrementado de 25 (enquanto τ for maior que zero), e para cada reforço positivo τ será incrementado de 25 (sem exceder $\tau_{max} = 100$).

5.2 Experimentos

5.2.1 Cenários Avaliados

Os experimentos realizados neste trabalho avaliam o CoRe-RL com foco em dois aspectos principais: 1) capacidade de classificação; 2) capacidade de adaptação. A classificação foi avaliada considerando os acertos e erros para diferentes limiares de corte, enquanto que, para a adaptação, foram avaliados os acertos e erros para um número crescente de rodadas. Além disso, esses aspectos foram avaliados tendo em vista dois cenários principais: no primeiro cenário, a inferência considera apenas um único con-

texto, ou seja, o CoRe-RL deverá responder se um determinado vetor de características pertence ou não à classe do contexto cadastrado (de acordo com o limiar de corte); no segundo cenário, foram cadastrados três contextos distintos, neste caso a inferência deve ser capaz de diferenciar os vetores de entrada em três classes de contextos diferentes e, de acordo com o limiar de corte, indicar se o vetor pertence ou não a alguma das classes cadastradas.

Desta forma, foram realizados quatro tipos de experimentos, tendo em vista os aspectos levantados e os cenários propostos:

- Experimento 01 - Avaliação da Classificação para 1 Contexto
- Experimento 02 - Avaliação da Classificação para 3 Contextos
- Experimento 03 - Avaliação da Adaptação para 1 Contexto
- Experimento 04 - Avaliação da Adaptação para 3 Contextos

5.2.2 Metodologia

Os experimentos práticos foram executados por um usuário pesquisador, cujo interesse era testar e avaliar o CoRe-RL em contextos específicos do seu dia a dia. Os contextos considerados nos experimentos deste trabalho foram: "Trabalhando", "Caminhada" e "Jogando". O contexto "Trabalhando" foi associado à localização do Centro de Processamento de Dados (CPD) da Universidade Federal do Amazonas (UFAM), onde trabalha o usuário que realizou os experimentos. O contexto "Caminhada" foi associado a uma área chamada de "Pista de Caminhada", na qual o usuário praticava caminhada diariamente. O contexto "Jogando" representa as situações que o usuário jogava algum jogo em seu smartphone (fora do local de trabalho e da pista de caminhada) com a orientação do aparelho colocada de lado.

Em todos os experimentos, foram cadastrados vetores incompletos para os contextos utilizados, ou seja, contendo apenas algumas pistas do que seria cada contexto (conforme apresentado na tabela 5.5), por exemplo, para o contexto "Trabalhando" as únicas informações passadas foram a localização ("CPD") e velocidade ("nula"). Portanto, as características que tiveram seus valores informados receberam pontuações maiores nos rankings de características de seus respectivos contextos. Por exemplo, as características localização e velocidade do contexto "Trabalhando" receberam 100 pontos, enquanto as demais características receberam apenas 1 ponto. Partindo de vetores incompletos, sem precisar de uma base com vários exemplos, foi possível classificar os vetores coletados tendo em vista o modelo inicial utilizado. Além disso, justifica-se a

utilização de vetores incompletos, pois é mais provável que uma aplicação informe dicas do contexto de interesse, ao invés de cadastrar um exemplo ou uma base de exemplos completos para cada contexto.

Nos experimentos que somente 1 contexto é avaliado, cadastrou-se apenas o contexto "Trabalhando", enquanto que, nos demais experimentos o estado inicial dos rankings de características e da base de vetores aprendidos ficou exatamente conforme apresentado na tabela 5.5.

Tabela 5.5: Vetores cadastrados para cada contexto e os rankings de características iniciais dos experimentos

Vetores Cadastrados

id	rótulo	turn	loca	velo	lumi	prox	acel	magn	orie	som	nbat	ebat
1	Trabalhando	–	CPD	nula	–	–	–	–	–	–	–	–
2	Caminhada	–	Pista de Caminhada	caminhando	–	–	–	–	–	–	–	–
3	Jogando	–	–	–	–	–	–	–	de lado	–	–	–

Inicialização do Ranking de Características

rótulo	turn	loca	velo	lumi	prox	acel	magn	orie	som	nbat	ebat
Trabalhando	1	100	100	1	1	1	1	1	1	1	1
Caminhando	1	100	100	1	1	1	1	1	1	1	1
Jogando	1	1	1	1	1	1	1	100	1	1	1

Durante a elaboração dos experimentos surgiu o seguinte questionamento: "Como avaliar se o CoRe-RL melhora seus resultados com o número de interações, uma vez que é necessário haver um controle sobre os tipos de interações que podem ocorrer?". De acordo com o tipo, ordem e frequência de interações o sistema irá aprender de formas diferentes. Uma vez que as interações realizadas tem influência sobre o que é aprendido, então o desempenho da técnica proposta também será influenciado por estas interações.

Como os experimentos realizados são práticos, não é possível avaliar o comportamento do CoRe-RL para todas as possíveis combinações de vetores de entrada e reforços (i.e., interações). Por isso, foram estabelecidos alguns roteiros que direcionam os tipos de interações que devem ocorrer em cada experimento. Estes roteiros não especificam completamente as interações, mas indicam as condições em que elas irão ocorrer. Desta forma, foi possível avaliar o comportamento da técnica proposta em cenários específicos, de acordo com os contextos usados. A tabela 5.6 apresenta os

roteiros para as interações nos experimentos com um (na parte superior da tabela) e três contextos (na parte inferior da tabela).

Tabela 5.6: Roteiros de Interações

Experimentos com 1 contexto

Repetições	**Interações**	**Rótulo Esperado**
6x	loca.CPD & velo.nula	Trabalhando
3x	loca.CPD & velo.caminhando	<NULL>
3x	loca.outro & velo.nula	<NULL>
3x	loca.outro & velo.caminhando	<NULL>
3x	loca.CPD & velo.nula	Trabalhando

Experimentos com 3 contextos

Repetições	**Interações**	**Rótulo Esperado**
4x	loca.CPD & velo.nula	Trabalhando
4x	loca.outro & orie.'de lado'	Jogando
4x	loca.'Pista de Caminhada' & velo.caminhando	Caminhada
4x	loca.'Pista de Caminhada' & velo.nula	<NULL>
4x	loca.outro & velo.caminhando	<NULL>
4x	loca.CPD & velo.caminhando	<NULL>

Em ambos os roteiros, metade das interações devem produzir alguma classificação e a outra metade não deverá retornar nenhum contexto (i.e., retornam <NULL>). Os experimentos que avaliam a classificação utilizam os vetores cadastrados e o modelo inicial do ranking para inferir contextos em uma única execução dos roteiros, demonstrando que é possível obter resultados melhores ou piores de acordo com os limiares de corte escolhidos. Os experimentos de aprendizagem (i.e., que avaliam a adaptação) fixam o limiar de corte e executam os roteiros em várias rodadas, a fim de demonstrar que o CoRe-RL melhora suas inferências e converge para o desempenho ótimo. Ou seja, na metade das vezes que o sistema deve retornar algum contexto, o CoRe-RL deverá retornar o contexto correto, e na metade que o usuário não estiver em nenhum contexto, o sistema deverá retornar <NULL>.

Seguindo os roteiros propostos, cada experimento foi repetido 3 vezes. Experimentos que consideram limiares diferentes foram avaliados em execuções diferentes, pois produzem taxas diferentes de acertos e erros de acordo com os limiares. A tabela

5.7 sumariza as especificações dos experimentos de acordo com o tipo de avaliação, roteiro, limiar de corte, número de rodadas e número de repetições de cada experimento.

Tabela 5.7: Especificação dos Experimentos

Experimento	**Tipo**	**Roteiro**	**Limiar de Corte**	**Número de Rodadas**	**Número de Experimentos**
01	Avaliação da Classificação	1 contexto	0%, 25%, 50%, 75%, 90%, 95%, 100%	1	3
02	Avaliação da Classificação	3 contextos	0%, 25%, 50%, 75%, 90%, 95%, 100%	1	3
03	Avaliação da Adaptação	1 contexto	90%	até o desempenho ótimo	3
04	Avaliação da Adaptação	3 contextos	90%	até o desempenho ótimo	3

Neste trabalho, tanto o número de vetores aprendidos, como o de contextos usados nos experimentos são limitados. Conforme apresentado na seção 4.2.1, o valor de K (i.e., quantidade de vizinhos mais próximos considerados na classificação) é dado pela razão entre o número de exemplos aprendidos e o número de contextos cadastrados, sem passar de um K_{max}. Assim, nos experimentos realizados foi escolhido o valor de $K_{max} = 5$. Contudo, como o número de interações diferentes em cada experimento é pequeno, em todos os experimentos o valor de K considerado nas inferências nunca chegou a ser igual a $K_{max} = 5$. Não é objetivo deste trabalho fazer uma análise a respeito da influência do valor de K no K-NN, uma vez que tal análise estaria restrita ao cenário destes experimentos, e ainda poderia sofrer influência de outras variáveis (e.g. limiar de corte). Portanto, foi estabelecida uma heurística com parâmetros fixos ($K_{max} = 5$), a qual foi usada em todos os experimentos, de modo que, todos os resultados obtidos foram influenciados de igual modo por esta heurística. Uma sugestão de trabalhos futuros é que seja feito um estudo com diferentes heurísticas para determinação dinâmica do valor de K, tendo em vista a aplicação do K-NN em abordagens incrementais.

5.3 Resultados

Os experimentos realizados foram analisados a partir das quantidades de acertos e erros na inferência de contexto. Assim, para cada execução, foram obtidas as quantidades de verdadeiros positivos (V_P), falsos positivos (F_P), verdadeiros negativos (V_N) e falsos negativos (F_N), e a partir desses valores, também foram calculadas as seguintes medidas de desempenho (apresentadas na seção 2.3.1):

- Acurácia Total (*acur*) - Representa a proporção de acertos com relação ao total de inferências.
- Taxa de Erro (*erro*) - Representa a proporção de erros com relação ao total de inferências.
- Completude (*comp*) - avalia se todos os exemplos, pertencentes a alguma classe, são classificados (exemplos sem classe não devem ser classificados).
- Consistência (*cons*) - avalia se os exemplos classificados foram classificados corretamente.
- Precisão (*prec*) - avalia a porcentagem de acertos (classificações corretas) V_P dentre as classificações que retornaram algum resultado positivo ($V_P + F_P$).
- Valor Preditivo Negativo (*vpn*) - avalia a porcentagem de acertos (V_N) dentre as classificações que retornaram que o exemplo não pertencia a nenhum contexto ($V_N + F_N$).

Avaliação da Classificação

Tabela 5.8: Experimento de classificação de 1 contexto, variando o limiar de corte pela similaridade

Limiar	**VP**	**FP**	**VN**	**FN**	**erro**	**acur**	**comp**	**cons**	**prec**	**vpn**
0%	9.00	9.00	0.00	0.00	0.50	0.50	1.00	0.50	0.50	-
25%	9.00	9.00	0.00	0.00	0.50	0.50	1.00	0.50	0.50	-
50%	9.00	6.33	2.67	0.00	0.35	0.65	1.00	0.65	0.59	1.00
75%	9.00	3.00	6.00	0.00	0.17	0.83	1.00	0.83	0.75	1.00
90%	9.00	3.00	6.00	0.00	0.17	0.83	1.00	0.83	0.75	1.00
95%	4.67	0.00	9.00	4.33	0.24	0.76	0.76	1.00	1.00	0.68
100%	1.33	0.00	9.00	7.67	0.43	0.57	0.57	1.00	1.00	0.54
desempenho ótimo	9.00	0.00	9.00	0.00	0.00	1.00	1.00	1.00	1.00	1.00

A tabela 5.8 apresenta as médias dos resultados (após 3 repetições) do experimento 01, que avalia o CoRe-RL quanto à classificação de um único contexto. Nesta tabela é possível observar as quantidades de acertos e erros obtidos, bem como, as diferentes medidas de desempenho calculadas, tendo em vista diferentes limiares de

cortes pela similaridade. Além disso, também são exibidos os valores esperados para um comportamento ótimo, na última linha da tabela. Para o limiar de corte de 0% e de 25% todas as classificações realizadas retornaram o contexto trabalhando (que era o único contexto cadastrado), pois não ocorreram cortes, uma vez que a similaridade do vetor atual com os k-vizinhos sempre foi superior a 25%. Além disso, quando o limiar de corte é 100% o CoRe-RL só irá retornar um rótulo, caso o vetor atual seja exatamente igual a algum vetor rotulado da base. Como na primeira interação a comparação é feita entre o vetor atual e o vetor incompleto que foi cadastrado, se as características informadas forem iguais nos dois vetores, a inferência retornará o contexto do vetor cadastrado com 100% de similaridade. Os espaços vazios na coluna "vpn" indicam que o cálculo do Valor Preditivo Negativo não é aplicável às quantidades de acertos e erros obtidas, uma vez que a soma de V_N com F_N é nula e produz divisão por zero no cálculo dessa medida de desempenho.

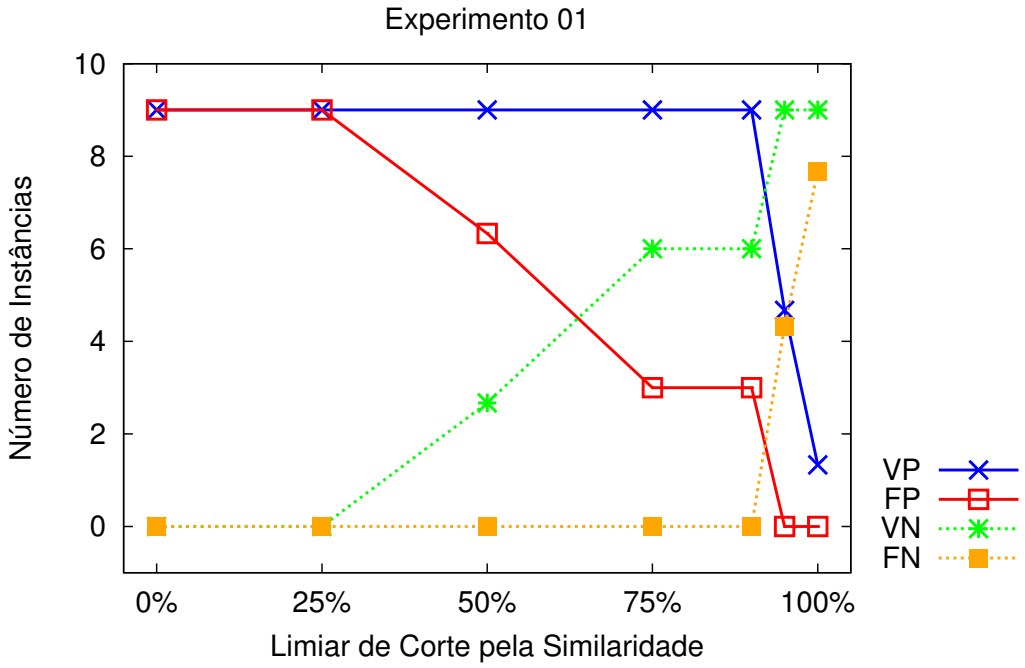


Figura 5.2: Acertos e erros para os respectivos limiares de corte para a inferência de um único contexto.

O gráfico da figura 5.2 apresenta o comportamento dos acertos (V_P, V_N) e erros (F_P,F_N) apresentados na tabela 5.8, tendo em vista os diferentes limiares testados. Neste gráfico é possível observar que a quantidade de classificações corretas (V_P) é máxima para similaridades de 0% a 90%, e diminui quando a similaridade de corte é muito alta para as comparações entre vetores observados e vetores aprendidos. Neste caso, o que acontece é que o CoRe-RL deixa de retornar contextos pois não possui a

"confiança" (similaridade) exigida pela aplicação. Desta forma, aumentando o limiar de corte, o CoRe-RL aumenta o número de verdadeiros e falsos negativos. O pedaço da curva em que o número de falsos negativos cresce, é o mesmo momento em que os verdadeiros positivos decrescem. Por outro lado, conforme se aumenta a similaridade, o número de falsos positivos descresce, ou seja, o CoRe-RL deixa de retornar o contexto "Trabalhando" para vetores que não são tão parecidos (i.e., similares) com os vetores aprendidos. Na mesma proporção que os falsos positivos diminuem, os verdadeiros negativos aumentam.

Existem dois aspectos que permitem que o CoRe-RL tenha um desempenho ótimo: a quantidade de vetores na base - quanto maior a base, maiores as chances de que o vetor observado esteja presente na base de vetores aprendidos (resultando em uma similaridade de 100%); e o modelo aprendido para o ranking de características - quanto mais especializado for o modelo para cada contexto, as comparações entre os vetores irão atribuir pesos maiores para as características de maior relevância, aproximando vetores cujas características de menor relevância são diferentes entre si (resultando em uma similaridade próxima a 100%).

A partir do gráfico da figura 5.2 é possível observar claramente que, para as condições iniciais estabelecidas, não existe nenhum valor de limiar que faça a classificação de contextos ser ótima no cenário testado. Mesmo assim, os resultados obtidos são interessantes, pois, o CoRe-RL teve um desempenho bom mesmo com um vetor inicial incompleto e utilizando um modelo não especializado, sem exigir uma longa etapa de treinamento do classificador. Neste gráfico, os melhores limiares observados são os de 75%, 90% e 95%. No primeiro caso (similaridade de 75% e 90%), o CoRe-RL apresentou a menor quantidade de falsos positivos (F_P), conseguindo classificar corretamente (V_P) todos as vezes que o usuário estava no contexto "Trabalhando", por outro lado, no segundo caso (95% de similaridade), não houveram falsos positivos, ou seja, o sistema conseguiu classificar corretamente (V_P) vários vetores do contexto "Trabalhando" e não fez nenhuma classificação errada ($F_P = 0$), ou seja, obteve a precisão máxima com uma completude relativamente alta.

É possível avaliar o desempenho da classificação no CoRe-RL através do gráfico da figura 5.3. A acurácia máxima, de 83%, foi obtida para os limiares de 75% e 90%. Conforme são utilizados limiares maiores, observa-se que a acurácia (acur) total aumenta até certo ponto e, em seguida, diminui. Entretanto, não há sentido em fixar um limiar de corte para obter a maior acurácia em cenários diferentes, pois o limiar será específico para o cenário testado, uma vez que esse "ponto de máxima" varia de acordo com a base de vetores e o modelo utilizado, e ambos mudam conforme ocorrem as interações (i.e., com os vetores observados e seus reforços). Além disso, observa-

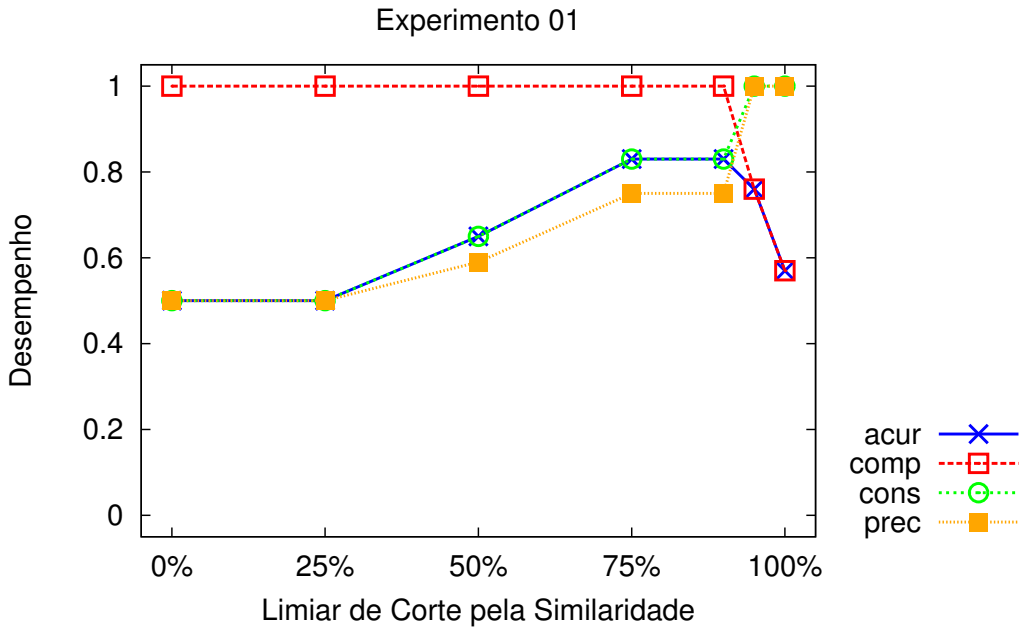


Figura 5.3: Desempenho de acordo com os limiares de corte, na inferência de um único contexto.

se que a precisão e a consistência da classificação cresce até o maior valor possível (i.e., 1 ou 100%), conforme são utilizados limiares maiores, enquanto que a completude diminui quando são usados limiares muito altos. O comportamento da consistência e da precisão indicam que quanto maior o limiar menos o CoRe-RL irá retornar classificações erradas. Entretanto, a diminuição da completude indica que para limiares muito altos o CoRe-RL irá, erroneamente, deixar de retornar rótulos para os vetores observados.

A tabela 5.9 apresenta os resultados obtidos para o experimento 02, que avalia a classificação para um cenário em que o CoRe-RL deverá diferenciar três contextos durante a inferência. Para os limiares de 0%, 25% e 50%, o classificador sempre retornou algum rótulo, inclusive nos casos em que o usuário não estava em nenhum dos contextos cadastrados. Como o modelo utilizado ainda não estava totalmente balanceado e os vetores de entrada possuiam certas semelhanças com os vetores aprendidos, foi necessário um limiar um pouco mais alto para que o CoRe-RL fosse capaz de diferenciar os vetores que estavam associados a algum contexto, dos que não estavam.

A comparação entre um vetor observado e os vetores cadastrados (incompletos) retorna algum contexto com similaridade 100% quando as características informadas são idênticas entre vetores. Desta forma, justifica-se os quatro verdadeiros positivos para a similaridade de 100%, uma vez que, pelo menos para o primeiro vetor observado de cada contexto (no caso são três contextos) a similaridade será de 100%, podendo

Tabela 5.9: Experimento de classificação de 3 contextos, variando o limiar de corte pela similaridade

Limiar	**VP**	**FP**	**VN**	**FN**	**erro**	**acur**	**comp**	**cons**	**prec**	**vpn**
0%	12.00	12.00	0.00	0.00	0.50	0.50	1.00	0.50	0.50	-
25%	12.00	12.00	0.00	0.00	0.50	0.50	1.00	0.50	0.50	-
50%	12.00	12.00	0.00	0.00	0.50	0.50	1.00	0.50	0.50	-
75%	12.00	8.00	4.00	0.00	0.33	0.67	1.00	0.67	0.60	1.00
90%	11.67	6.67	5.33	0.33	0.29	0.71	0.99	0.72	0.64	0.94
95%	11.00	1.67	10.33	1.00	0.11	0.89	0.96	0.93	0.87	0.91
100%	4.00	0.00	12.00	8.00	0.33	0.67	0.67	1.00	1.00	0.60
desempenho ótimo	12.00	0.00	12.00	0.00	0.00	1.00	1.00	1.00	1.00	1.00

haver outras ocorrências de vetores idênticos ao primeiro vetor aprendido.

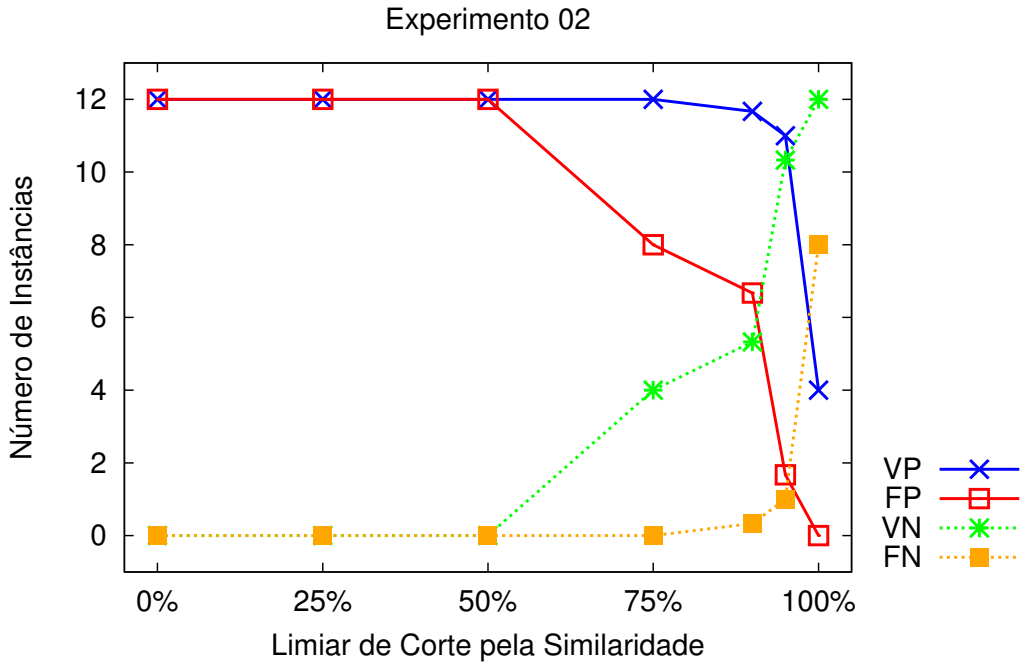


Figura 5.4: Acertos e erros para os respectivos limiares de corte para a inferência de três contextos.

O gráfico da figura 5.4 apresenta os acertos e erros (listados na tabela 5.9) obtidos no experimento 02. Aparentemente, com mais contextos, são necessários limiares maiores (comparados com os resultados do experimento 01) para que as inferências tenham menos falsos positivos, isso pode ser decorrente das condições testadas (i.e.,

roteiro utilizado e as condições iniciais da base de vetores e ranking de características), ou da maior complexidade inerente à classificação com mais de 1 contexto. Além disso, de modo similar ao experimento 01, observa-se (no gráfico da figura 5.4) que as quantidades de falsos positivos decrescem conforme os verdadeiros negativos aumentam, e a quantidade de verdadeiros positivos diminui (para limiares muito altos) a medida que aumentam os falsos negativos.

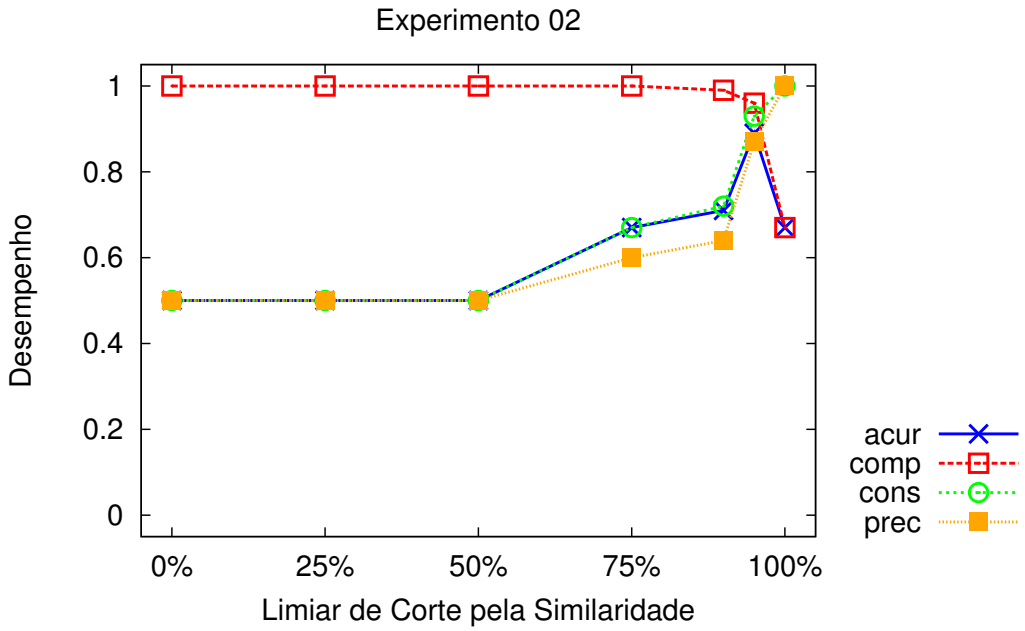


Figura 5.5: Desempenho de acordo com os limiares de corte, na inferência de três contextos.

Na figura 5.5 é apresentado o gráfico com as medidas de desempenho calculadas para cada limiar testado no experimento 02 (conforme a tabela 5.9). Novamente, não há nenhum valor de limiar que permita que o CoRe-RL tenha um comportamento ótimo (acurácia máxima igual a 100%) para as condições iniciais estabelecidas e para o roteiro utilizado. No gráfico, o ponto em que a acurácia é máxima (89%) é referente ao experimento que considerou 95% de similaridade. Neste ponto, tanto a completude como a consistência e a precisão são maiores que 85%, o que indica que a classificação é boa para este limiar, mesmo com uma base pequena e um modelo ainda imperfeitamente ajustado ao cenário.

Avaliação da Adaptação

A etapa de adaptação do CoRe-RL é responsável pela aprendizagem da inferência de contexto. Para demonstrar que o CoRe-RL aprende a inferir contextos de forma

Tabela 5.10: Experimento de aprendizagem de 1 contexto, para o limiar de 90% de corte pela similaridade

Rodadas	**VP**	**FP**	**VN**	**FN**	**erro**	**acur**	**comp**	**cons**	**prec**	**vpn**
1	9.00	3.00	6.00	0.00	0.17	0.83	1.00	0.83	0.75	1.00
2	9.00	3.00	6.00	0.00	0.17	0.83	1.00	0.83	0.75	1.00
3	9.00	3.00	6.00	0.00	0.17	0.83	1.00	0.83	0.75	1.00
4	9.00	2.00	7.00	0.00	0.11	0.89	1.00	0.89	0.82	1.00
5	9.00	2.00	7.00	0.00	0.11	0.89	1.00	0.89	0.82	1.00
6	9.00	1.33	7.67	0.00	0.07	0.93	1.00	0.93	0.87	1.00
7	9.00	0.33	8.67	0.00	0.02	0.98	1.00	0.98	0.96	1.00
8	9.00	0.00	9.00	0.00	0.00	1.00	1.00	1.00	1.00	1.00

incremental para um conjunto de vetores de entrada, foi fixado o valor do limiar de corte, no caso, em 90%, e o roteiro utilizado foi repetido uma série de vezes. O objetivo de fixar o limiar de corte foi para que o limiar não tivesse influência no desempenho da técnica, tendo em vista que diferentes limiares produzem diferentes desempenhos na classificação (conforme demonstrado nos experimentos 01 e 02). Além disso, se o valor utilizado fosse muito alto, ou muito baixo, o sistema iria precisar de um número maior de rodadas para alcançar o desempenho ótimo.

Na tabela 5.10 são apresentados os resultados obtidos no experimento 03, cujo propósito é avaliar a etapa de adaptação do CoRe-RL em um cenário com um único contexto. Como o roteiro utilizado é o mesmo do experimento 01, observa-se que os valores obtidos na primeira rodada correspondem aos valores obtidos no experimento 01 para a similaridade de 90% (apresentados na tabela 5.8). Além disso, observa-se também que todos os valores da última linha da tabela 5.10 são idênticos aos valores esperados para o desempenho ótimo da tabela 5.8.

O gráfico da figura 5.6 exibe as quantidades de erros e acertos obtidas em cada rodada do experimento e listadas na tabela 5.10, demonstrando como as classificações erradas (F_P) foram deixando de ocorrer, e como o CoRe-RL foi aprendendo a identificar as situações que não configuravam o contexto "Trabalhando" (V_N).

A figura 5.7 contém o gráfico da curva de aprendizagem do CoRe-RL para as condições iniciais impostas, e para o cenário do roteiro utilizado. A partir deste gráfico, verifica-se a melhoria do desempenho do CoRe-RL ao longo das rodadas. A completude foi ótima (i.e., $comp = 100\%$) do ínicio ao fim, enquanto a acurácia total, a consistência e a precisão da inferência foram melhorando até atingir o desempenho ótimo.

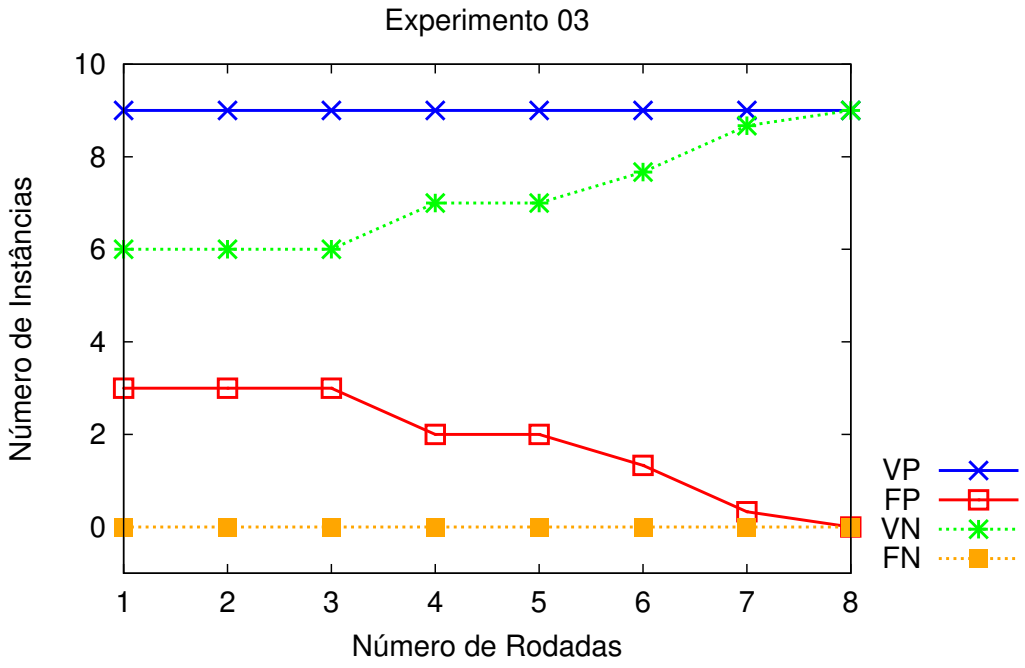


Figura 5.6: Acertos e erros para o limiar de corte de 90% em várias rodadas com apenas um contexto.

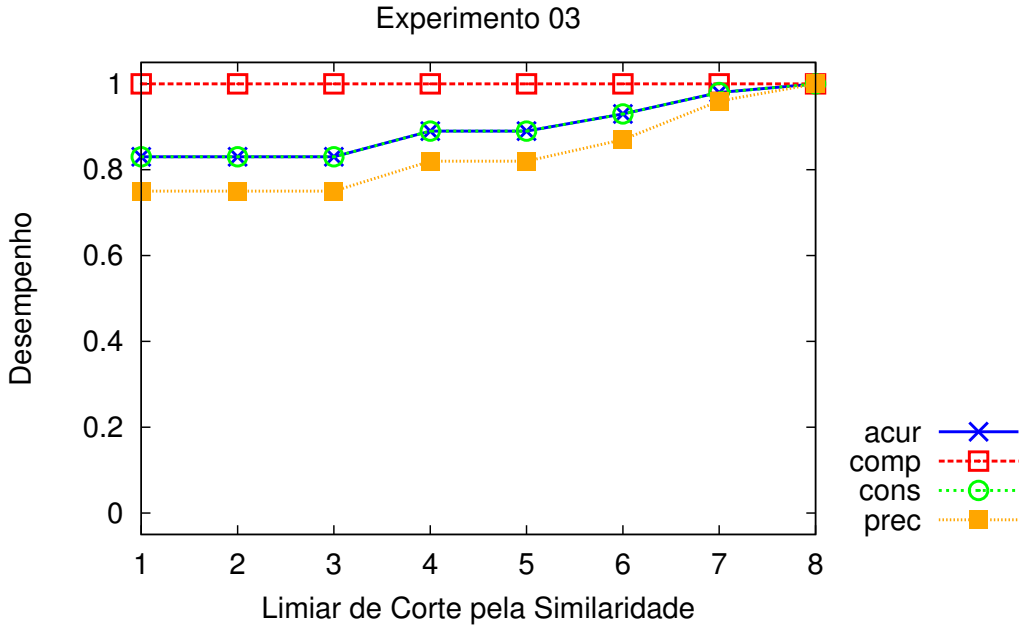


Figura 5.7: Melhoria do desempenho ao longo das rodadas, na inferência de um único contexto.

A tabela 5.11 apresenta os resultados obtidos no experimento 04, que avalia a adaptação do CoRe-RL em um cenário com três contextos diferentes. Os valores da

Tabela 5.11: Experimento de aprendizagem de 3 contextos, para o limiar de 90% de corte pela similaridade

Rodadas	**VP**	**FP**	**VN**	**FN**	**erro**	**acur**	**comp**	**cons**	**prec**	**vpn**
1	11.67	6.67	5.33	0.33	0.29	0.71	0.99	0.72	0.64	0.94
2	12.00	6.00	6.00	0.00	0.25	0.75	1.00	0.75	0.67	1.00
3	12.00	4.00	8.00	0.00	0.17	0.83	1.00	0.83	0.75	1.00
4	12.00	3.33	8.67	0.00	0.14	0.86	1.00	0.86	0.78	1.00
5	12.00	2.33	9.67	0.00	0.10	0.90	1.00	0.90	0.84	1.00
6	12.00	2.33	9.67	0.00	0.10	0.90	1.00	0.90	0.84	1.00
7	12.00	2.33	9.67	0.00	0.10	0.90	1.00	0.90	0.84	1.00
8	12.00	2.00	10.00	0.00	0.08	0.92	1.00	0.92	0.86	1.00
9	12.00	2.00	10.00	0.00	0.08	0.92	1.00	0.92	0.86	1.00
10	12.00	1.33	10.67	0.00	0.06	0.94	1.00	0.94	0.90	1.00
11	12.00	1.00	11.00	0.00	0.04	0.96	1.00	0.96	0.92	1.00
12	12.00	0.67	11.33	0.00	0.03	0.97	1.00	0.97	0.95	1.00
13	12.00	0.33	11.67	0.00	0.01	0.99	1.00	0.99	0.97	1.00
14	12.00	0.00	12.00	0.00	0.00	1.00	1.00	1.00	1.00	1.00

primeira rodada são idênticos aos valores do experimento 02 para 90% de similaridade (que constam na tabela 5.9), enquanto que os valores da rodada 14 consistem no desempenho ótimo para o roteiro utilizado. Além disso, observa-se um maior número de rodadas necessário para alcançar o desempenho ótimo no cenário contendo três contextos. Este número maior é decorrente da maior complexidade envolvida na classificação de mais de um contexto.

A figura 5.8 demonstra como os erros diminuiram e os acertos aumentaram ao longo das rodadas do experimento 04, enquanto que na figura 5.9 é exibida a curva de aprendizagem para o cenário do roteiro que envolve três contextos.

Considerando que, para cada rodada, repete-se o mesmo roteiro (de acordo com o experimento), após a primeira rodada quase não serão aprendidos novos vetores, pois praticamente todos os vetores do roteiro, pertencentes a algum contexto, já terão sido aprendidos (adicionados à base) na primeira rodada. Isto demonstra que o CoRe-RL é capaz de melhorar as classificações à medida que os rankings de características vão sendo ajustados aos seus respectivos contextos, sem que seja necessário adicionar outros vetores à base. Desta forma, utilizando o CoRe-RL é possível obter um bom desempenho na inferência de contexto, sem a necessidade de uma base muito grande

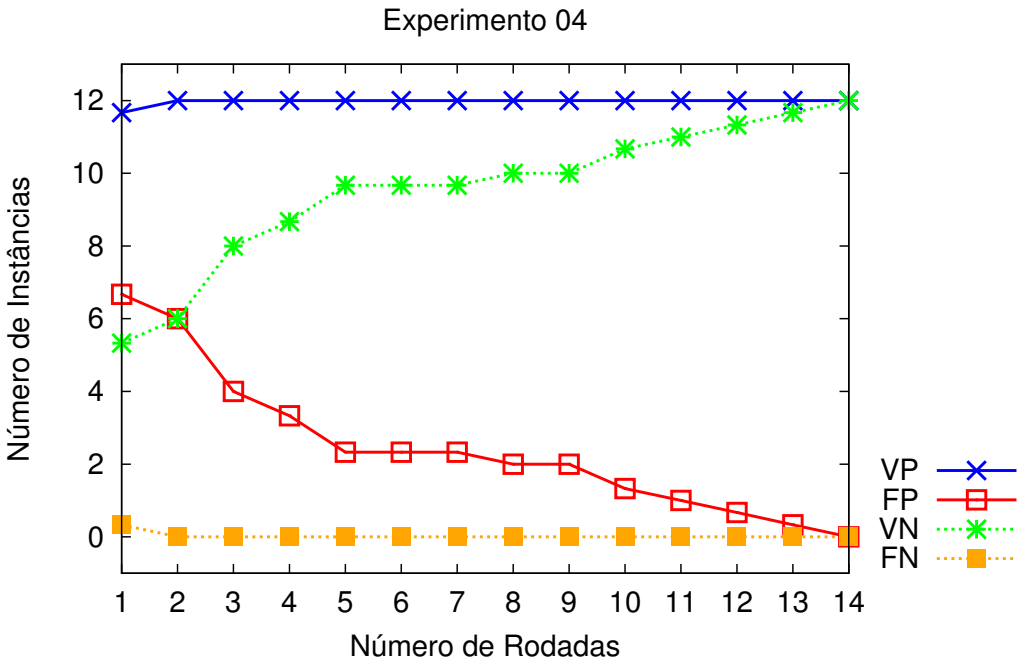


Figura 5.8: Acertos e erros para o limiar de corte de 90% em várias rodadas com três contextos.

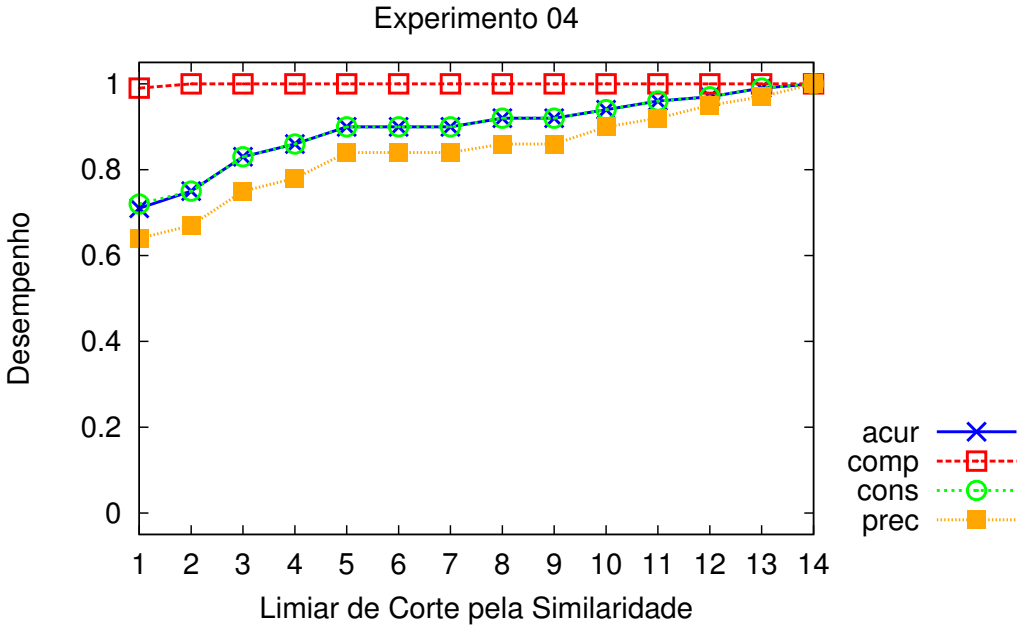


Figura 5.9: Melhoria do desempenho ao longo das rodadas, na inferência três contextoss.

de vetores aprendidos, e sem desperdiçar espaço com exemplos que receberam reforços negativos, como ocorre em outras abordagens baseadas em exemplos.

5.4 Considerações Sobre o Capítulo

Neste capítulo foi apresentado o Mob Context App, que é um aplicativo sobre a plataforma android de dispositivos móveis, que implementa o CoRe-RL diretamente em uma aplicação móvel. Esta aplicação foi criada com o objetivo de validar e avaliar a técnica de inferência proposta. Por isso, ela foi desenvolvida para funcionar como uma interface entre o CoRe-RL e o usuário que iria avaliar a técnica.

O CoRe-RL foi avaliado através de experimentos práticos, utilizando o Mob Context App, cujos propósitos eram: demonstrar que o CoRe-RL é capaz de inferir contextos (avaliação da classificação); e demonstrar que o CoRe-RL também pode melhorar seu desempenho de acordo com as interações realizadas (i.e., vetores observados e reforços às classificações). Ambos os experimentos foram avaliados em dois cenários com diferentes quantidades de rótulos a serem inferidos: no primeiro cenário havia apenas um único rótulo a ser atribuído ou não aos vetores observados, enquanto que no segundo cenário foram cadastrados três rótulos.

Através dos experimentos de avaliação da classificação, foram obtidos diferentes quantidades de acertos e erros para diferentes valores de limiares de corte. Contudo, mesmo com uma base pequena e um modelo pouco ajustado à realidade do usuário (devido às poucas interações), quase sempre que um vetor estava associado a um contexto a classificação retornava o contexto correto. Validando a classificação do CoRe-RL. Observou-se também que, tanto para um rótulo como para três, à medida que são utilizados limiares maiores, a precisão aumenta. Contudo, a partir de certo ponto, a completude (i.e., a capacidade de retornar uma classificação para o maior número possível de amostras que devem ter um rótulo) diminui, impactando também na acurácia do CoRe-RL. De outra forma, o número de positivos, tanto verdadeiros como falsos, é maior quanto menor for o limiar, e o número de negativos aumenta quanto maior for o limiar. Portanto, dependendo da base e dos modelos aprendidos, quanto maior for a confiabilidade exigida nas classificações, menor será a chances de que o classificador retorne algum rótulo. Por outro lado, quanto menores forem as exigências, mais fácil será para o classificador retornar algum rótulo (tanto correto como incorreto).

Assim, conclui-se que o limiar de corte pode sim influenciar no desempenho do CoRe-RL. Portanto, é importante que se utilize um limiar de corte alto a ponto de se obter uma boa precisão, mas não tão alto a ponto de comprometer a completude da técnica. O controle sobre o limiar de corte é responsabilidade das aplicações, portanto elas poderão aumentar ou diminuir o limiar de acordo com as demandas de informações contextuais mais, ou menos, confiáveis. Uma possibilidade para trabalhos futuros é a sugestão e avaliação de heurísticas para a determinação dinâmica do limiar, a fim de

obter desempenhos melhores usando o CoRe-RL.

Nos experimentos de avaliação da etapa de adaptação do CoRe-RL fixou-se um limiar (valor constante) e foram repetidas as interações de cada cenário testado na avaliação da classificação. Desta forma, anulando a influência do limiar de corte, observaram-se melhorias nas medidas de desempenho avaliadas, à medida que as interações iam sendo repetidas. Além disso, observou-se que o desempenho da classificação converge para o desempenho ótimo à medida que as interações são realizadas em um ambiente estático (i.e., onde as condições observadas para os contextos são sempre as mesmas). Desta forma, demonstrou-se que o CoRe-RL é capaz de aprender a inferir contextos, e que a técnica é ajustável e adaptável ao ambiente em que estiver inserida.

Todos os resultados obtidos consideram que os reforços são sempre corretos. Entretanto, na prática, podem ser atribuídos reforços errados para as inferências realizadas. Assim, se um reforço positivo é atribuído de maneira errada para uma inferência, então o CoRe-RL aprenderá erroneamente aquele vetor, o qual poderá produzir uma série de reforços negativos para vetores parecidos, ocasionando no eventual esquecimento deste vetor, pois seu nível de importância será decrementado até que ele seja removido da base. Caso seja retornando um reforço negativo de maneira errada para uma inferência, a base de vetores poderá esquecer um vetor que não deveria ser esquecido, contudo isso só ocorre no pior caso, em que aquele vetor já havia recebido uma série de reforços negativos e, por acaso, recebeu um reforço negativo errado. Neste caso, é possível afirmar que, como a importância daquele vetor era baixa, o esquecimento daquele vetor não terá um impacto muito grande nas próximas inferências.

Capítulo 6

Considerações Finais

Talvez não tenha conseguido fazer o melhor, mas lutei para que o melhor fosse feito. Não sou o que deveria ser, mas Graças a Deus, não sou o que era antes.

Marthin Luther King

6.1 Conclusões

Neste trabalho, foi apresentada uma ampla visão a respeito dos sistemas sensíveis ao contexto, sempre levando em consideração o domínio das aplicações móveis. A revisão da literatura possibilitou o entendimento dos principais aspectos envolvidos em sensibilidade ao contexto, bem como as principais arquiteturas, formas de modelagem de contexto e técnicas de inferência usadas em SSC's. Além disso, a partir desta revisão, observou-se que mesmo em ambientes dinâmicos como nos da computação móvel, as técnicas usadas são pouco flexíveis e não levam em conta se as inferências foram realmente adequadas para o contexto do usuário.

Com base na limitação identificada durante a revisão bibliográfica, neste trabalho, foi proposta e avaliada uma nova técnica de inferência de contexto, voltada para aplicações móveis, a qual utiliza aprendizagem por reforço e permite um gerenciamento flexível dos contextos, que serão aprendidos de modo incremental, conforme o usuário interage com o sistema. Utilizando aprendizagem por reforço, os conceitos aprendidos são afetados pelos erros e acertos da classificação, possibilitando que as inferências

melhorem enquanto ocorrerem interações em um ambiente estático. Caso o ambiente mude, o desempenho da classificação também sofre com a mudança, contudo, a aprendizagem por reforço permite que o sistema se adapte às novas condições, mediante sinais de reforço.

A técnica proposta opera em duas etapas de funcionamento – classificação e adaptação. Na etapa de classificação, o vetor observado é classificado de acordo com os K vizinhos mais próximos, sendo que a distância entre os vizinhos é calculada com base no método de Gower, utilizando um modelo aprendido para ponderar as características mais relevantes. A adaptação ocorre quando um reforço é informado para determinado vetor inferido. Quando o reforço for negativo o CoRe-RL deverá reforçar negativamente os pesos das características que mais influenciaram no erro. Quando o reforço for positivo, as características mais relevantes são reforçadas positivamente e o vetor observado é aprendido. Além disso, vetores também podem ser esquecidos mediante uma série de reforços negativos. Desta forma, é possível reaprender a inferir contextos, o que pode ser particularmente interessante no caso de aprendizagem de vetores incorretos, ou quando mudam as condições do contexto em questão.

Por se tratar de uma técnica incremental, como estudo de caso deste trabalho, foi desenvolvido um aplicativo que implementou todas as funcionalidades da técnica proposta (CoRe-RL). O aplicativo foi utilizado na realização de experimentos práticos de avaliação das duas etapas envolvidas no CoRe-RL. Os experimentos realizados demonstram que o CoRe-RL é capaz de inferir contextos corretamente, e de melhorar suas inferências de acordo com as interações que ocorrem.

Como contribuição secundária, foi proposta uma nova abordagem de SSC voltada para a computação móvel, a qual foi chamada de Mob Context. O objetivo do Mob Context é prover informações contextuais de alto nível às aplicações, dando suporte à utilização de reforços, a fim de que a técnica de inferência usada possa aprender com as experiências passadas. Além disso, esta abordagem também tem como diferencial a flexibilidade no gerenciamento de contextos, pois permite que, ao longo das interações, as aplicações cadastrem, removam ou renomeiem os rótulos que deverão ser aprendidos. Além disso, de modo similar às demais abordagens propostas na literatura (estudadas no capítulo 3), o Mob Context auxilia no desenvolvimento de aplicações sensíveis ao contexto, cada vez mais autônomas e proativas, abstraindo questões de hardware e software referentes aos sensores e dados brutos coletados nos dispositivos.

6.2 Limitações e Peculiaridades do CoRe-RL

A técnica proposta utiliza o método de Gower para calcular distâncias entre vetores de características, e considera o ranking de características aprendido na determinação dos pesos associados às características que serão comparadas. De acordo com os dados que serão coletados e os tipos de características extraídas, é possível utilizar métodos mais adequados para calcular a distância entre os vetores. Contudo, tais métodos deverão considerar a atribuição de pesos às características, e normalizar as distâncias/similaridades para que a adaptação funcione bem.

Como a técnica proposta é incremental e utiliza o K-NN como método de classificação, não há conhecimento prévio da quantidade de vetores e de contextos que deverão ser aprendidos. Desta forma, não é interessante utilizar um valor fixo para K, uma vez que este valor poderá vir a ser muito pequeno à medida que a base de exemplos cresce, ou muito grande caso a quantidade de vetores aprendidos diminua. Assim, a fim de que o valor de K seja modificado conforme a base cresce ou decresce, faz-se necessário a determinação dinâmica do valor de K (através de heurísticas).

Uma limitação do estudo de caso implementado, foi a utilização de reforços com intensidades (módulos) constantes. Desta forma, o modelo do ranking de características vai sendo balanceado, com incrementos ou decrementos com as mesmas intensidades. Contudo, é possível que, para algumas aplicações, seja possível aplicar reforços com diferentes módulos, o que provavelmente modificaria a curva de aprendizagem dos ranking de características.

A partir dos experimentos de avaliação da classificação, observou-se que a escolha do limiar de corte influencia no desempenho do CoRe-RL. Ou seja, o desempenho da classificação é determinado pela escolha do limiar de corte, e isto também irá influenciar na curva de aprendizagem para a adaptação, uma vez que inferências que retornam algum resultado (i.e., verdadeiros positivos e falsos positivos) modificam o modelo e a base de vetores aprendidos, enquanto inferências que não retornam contextos (i.e., verdadeiros negativos e falsos negativos) não alteram os conceitos aprendidos, retardando a aprendizagem. Por outro lado, caso sejam criadas heurísticas para a determinação dinâmica do limiar de corte, será possível tomar proveito da influência do limiar de corte para se obter um bom desempenho tanto na classificação como na adaptação.

Outro problema que não foi resolvido neste trabalho, está relacionado com o crescimento da base de vetores aprendidos. O K-NN possui desempenho ótimo quando todos os possíveis vetores que possuem uma classe estão presentes na base de exemplos. Desta forma, a base do CoRe-RL deverá crescer enquanto houverem vetores a serem classificados, das classes cadastradas. Entretanto, os dispositivos móveis possuem res-

trições de espaço que podem tornar este crescimento inviável.

6.3 Trabalhos Futuros

Os resultados obtidos através dos experimentos práticos servem como um indicativo forte do bom desempenho da solução proposta. Entretanto, estes podem ser consideravelmente melhorados se forem observadas as peculiaridades da técnica proposta bem como suas limitações, apresentadas na seção 6.2. Assim, é possível listar as seguintes sugestões de trabalhos futuros:

- Pesquisar e avaliar a influência da utilização de métodos diferentes (aplicáveis ou adaptáveis ao CoRe-RL) para calcular as distâncias entre vetores. Desta forma, seria possível identificar o método mais adequado ao CoRe-RL.
- Avaliar o desempenho da classificação e da aprendizagem considerando diferentes heurísticas para a determinação dinâmica do valor de K.
- Avaliar a influência na aprendizagem da atribuição de reforços com diferentes intensidades (módulos).
- Propor e avaliar a utilização de heurísticas para determinação dinâmica de um bom limiar de corte. Estas heurísticas seriam usadas pelas aplicações no Mob Context.
- Propor e avaliar variações do CoRe-RL limitando o número de vetores aprendidos para cada contexto. Por exemplo, para cada contexto seria possível aprender uma quantidade fixa de vetores, sendo que, apesar de fixa, a base poderia mudar de acordo com a relevância dos vetores. Desta forma, haverá um limite no tamanho da base de vetores aprendidos tornando os custos com memória e processamento praticamente constantes.
- Avaliar o CoRe-RL em cenários dinâmicos, onde os contextos mudam e precisam ser reaprendidos ao longo do tempo.
- Propor e avaliar outros métodos para manter ou esquecer vetores de acordo com o nível de importância dos mesmos.

Referências Bibliográficas

Abowd, G. D.; Atkeson, C. G.; Hong, J.; Long, S.; Kooper, R. & Pinkerton, M. (1996). Cyberguide : A Mobile Context-Aware Tour Guide. *Baltzer Journals*, 3:1–21. ISSN 10220038.

Araujo, R. B. D. (2003). Computação Ubíqua Princípios, Tecnologias e Desafios. *XXI Simpósio Brasileiro de Redes de Computadores*, pp. 45–115.

Baldauf, M.; Dustdar, S. & Rosenberg, F. (2007). A survey on context-aware systems.

Bellavista, P.; Corradi, A.; Fanelli, M. & Foschini, L. (2012). A survey of context data distribution for mobile ubiquitous systems. *ACM Computing Surveys*, 44:1–45. ISSN 03600300.

Bettini, C.; Brdiczka, O.; Henricksen, K.; Indulska, J.; Nicklas, D.; Ranganathan, A. & Riboni, D. (2010). A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing*, 6(2):161–180. ISSN 15741192.

Biegel, G. & Cahill, V. (2004). A framework for developing mobile, context-aware applications. Em *Proceedings - Second IEEE Annual Conference on Pervasive Computing and Communications, PerCom*, pp. 361–365.

Bikakis, a.; Patkos, T.; Antoniou, G. & Plexousakis, D. (2008). A Survey of Semantics-Based Approaches for Context Reasoning in Ambient Intelligence. Em *Constructing Ambient Intelligence*, Communications in Computer and Information Science, pp. 14–23. Springer Berlin Heidelberg.

Capra, L.; Emmerich, W. & Mascolo, C. (2003). CARISMA: Context-Aware Reflective middleware System for Mobile Applications. *IEEE Transactions on Software Engineering*, 29(10):929–945. ISSN 00985589.

Chen, G. & Kotz, D. (2000). A Survey of Context-Aware Mobile Computing Research. *Time*, 3755:1–16. ISSN 15277755.

Cover, T. & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13. ISSN 0018-9448.

CraftyAppsEU (2015). Tasker - Android Apps on Google Play. Disponível em: <https://play.google.com/store/apps/details?id=net.dinglisch.android.taskerm>. Acesso em: 09/03/2015.

Devaraju, A.; Hoh, S. & Hartley, M. (2007). A context gathering framework for context-aware mobile solutions. Em *Proceedings of the 4th international conference on mobile technology, applications, and systems and the 1st international symposium on Computer human interaction in mobile technology*, pp. 39–46. ACM.

Dey, A. K. & Abowd, G. D. (1999). Towards a Better Understanding of Context and Context-Awareness. *Computing Systems*, 40:304–307. ISSN 00219266.

Egomotion (2014). Trigger - Android Apps on Google Play. Disponível em: <https://play.google.com/store/apps/details?id=com.jwsoft.nfcactionlauncher>. Acesso em: 09/03/2015.

Fahy, P. & Clarke, S. (2004). CASS - Middleware for Mobile Context-Aware Applications. Em *Workshop on Context Awareness, MobiSys*, p. 6.

Forster, K.; Monteleone, S.; Calatroni, A.; Roggen, D. & Troster, G. (2010). Incremental kNN classifier exploiting correct-error teacher for activity recognition. Em *Proceedings of the 9th International Conference on Machine Learning and Applications*, pp. 445–450.

Garlan, D.; Siewiorek, D. P. & Steenkiste, P. (2002). Project Aura: Toward Distraction-Free Pervasive Computing. *Pervasive Computing, IEEE*, 1:22–31. ISSN 1536-1268.

Guo, B. & Zhang, D. (2010). The architecture design of a cross-domain context management system. Em *Proceedings of the 8th International Conference on Pervasive Computing and Communications Workshops*, pp. 499–504. IEEE.

Hofer, T.; Schwinger, W.; Pichler, M.; Leonhartsberger, G.; Altmann, J. & Retschitzegger, W. (2003). Context-awareness on mobile devices - the hydrogen approach. *36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the*, 43(7236):292–302.

IFTTT (2015). About IFTTT - IFTTT. Disponível em: <https://ifttt.com/wtf>. Acesso em: 09/03/2015.

J. C. Gower (1971). A General Coefficient of Similarity and Some of Its Properties. *Biometrics*, 27(4):857–871. ISSN 0006341X.

Kabir, M. H.; Hoque, M. R. & Yang, S.-h. (2015). Development of a Smart Home Context-aware Application: A Machine Learning based Approach. *International Journal of Smart Home*, 9(1):217–226.

Kanungo, T.; Mount, D.; Netanyahu, N.; Piatko, C.; Silverman, R. & a.Y. Wu (2002). An efficient k-means clustering algorithm: analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):881–892. ISSN 0162-8828.

Kazakov, D. & Kudenko, D. (2001). Machine Learning and Inductive Logic Programming for Multi-agent Systems. *Lecture Notes in Computer Science*, pp. 246–270.

Kohonen, T. (2001). *Self-organizing maps*, volume 30. Springer Science & Business Media.

Korpipaa, P.; Mantyjarvi, J.; Kela, J.; Keranen, H. & Malm, E. (2003). Managing context information in mobile devices. *IEEE Pervasive Computing*, 2(3):42–51. ISSN 1536-1268.

Kotsiantis, S. B. (2007). Supervised Machine Learning: A Review of Classification Techniques. 31:249–268.

Mayrhofer, R.; Radi, H.; Ferscha, A.; Kepler, J. & Linz, U. (2003). Recognizing and Predicting Context by Learning from User Behavior. Em *The International Conference On Advances in Mobile Multimedia (MoMM2003)*, volume 171, pp. 25–35.

Mayrhofer, R. M. (2004). *An Architecture for Context Prediction*. Tese de doutorado, Universitat Linz.

Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edição. ISBN 0070428077, 9780070428072.

Nurmi, P. & Floréen, P. (2004). Reasoning in Context-Aware Systems. *Position Paper. Department of Computer Science, University of Helsinki*, (1):1–6.

Park, H.-S.; Oh, K. & Cho, S.-B. (2011). Bayesian Network-Based High-Level Context Recognition for Mobile Context Sharing in Cyber-Physical System. *International Journal of Distributed Sensor Networks*, 2011:1–10. ISSN 1550-1329.

Perera, C.; Zaslavsky, A.; Christen, P. & Georgakopoulos, D. (2013). Context Aware Computing for The Internet of Things : A Survey. *IEEE Communications Surveys & Tutorials*, pp. 1–5.

Rai, P. & Singh, S. (2010). A Survey of Clustering Techniques. *International Journal of Computer Applications*, 7(12):1–5.

Ranganathan, A. & Campbell, R. H. (2003). A Middleware for Context-Aware Agents in Ubiquitous Computing Environments. Em *Middleware 2003*, pp. 143–161. Springer Berlin Heidelberg.

Rezende, S. (2003). *Sistemas inteligentes: fundamentos e aplicações*. Editora Manole Ltda. ISBN 9788520416839.

Riboni, D. & Bettini, C. (2010). COSAR: hybrid reasoning for context-aware activity recognition. *Personal and Ubiquitous Computing*, 15(3):271–289. ISSN 1617-4909.

Rizou, S.; Haussermann, K.; Durr, F.; Cipriani, N. & Rothermel, K. (2010). A system for distributed context reasoning. *6th International Conference on Autonomic and Autonomous Systems, ICAS 2010*, pp. 84–89.

Russel, S. J. & Norvig, P. (1995). *Artificial Intelligence - A Modern Approach*. New Jersey. ISBN 0131038052.

Soylu, A.; de Causmaecker, P. & Desmet, P. (2009). Context and adaptivity in pervasive computing environments: Links with software engineering and ontological engineering. *Journal of Software*, 4(9):992–1013. ISSN 1796217X.

Strang, T. & Linnhoff-Popien, C. (2004). A Context Modeling Survey. *Graphical Models*, Workshop o:1–8.

Tasker (2015). Tasker For Android. Disponível em: <http://tasker.dinglisch.net/index.html>. Acesso em: 09/03/2015.

Vallim, R. M. M. (2009). *Sistemas classificadores evolutivos para problemas multirrótulo*. Tese de doutorado, Universidade de São Paulo.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. Tese de doutorado, King's College.

Watkins, C. J. C. H. & Dayan, P. (1992). Q-learning. *Machine Learning*, 8:279–292. ISSN 0885-6125, 1573-0565.

Weinberger, K.; Blitzer, J. & Saul, L. (2006). Distance metric learning for large margin nearest neighbor classification. *Advances in neural information processing systems*, 10:207–244. ISSN 1532-4435.

Zweigle, O.; Haussermann, K.; Kappeler, U. P. & Levi, P. (2009). Extended TA Algorithm for Adapting a Situation Ontology. Em *Progress in Robotics*, volume 44 of *Communications in Computer and Information Science*, pp. 364–371. Springer Berlin Heidelberg.