UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA

Rodolfo Lottin Pereira

APLICAÇÃO DE APRENDIZAGEM POR REFORÇO PARA UM MODELO BAYESIANO DE CONSCIÊNCIA SITUACIONAL

Florianópolis

2017

Rodolfo Lottin Pereira

APLICAÇÃO DE APRENDIZAGEM POR REFORÇO PARA UM MODELO BAYESIANO DE CONSCIÊNCIA SITUACIONAL

Trabalho de Conclusão de Curso submetido ao curso de Sistemas de Informação para a obtenção do Grau de Bacharel em Sistemas de Informação. Orientador: Prof. Dr. Elder Rizzon Santos

Florianópolis

2017

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Pereira, Rodolfo Lottin
Aplicação de aprendizagem por reforço para um modelo bayesiano de consciência situacional / Rodolfo Lottin Pereira ; orientador, Elder Rizzon Santos, 2017.
119 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Sistema de Informação, Florianópolis, 2017.

Inclui referências.

1. Sistema de Informação. 2. Inteligência Artificial. 3. Aprendizagem de Máquina. 4. Aprendizagem por Reforço. 5. Redes Bayesianas. I. Santos, Elder Rizzon. II. Universidade Federal de Santa Catarina. Graduação em Sistema de Informação. III. Título.

Rodolfo Lottin Pereira

APLICAÇÃO DE APRENDIZAGEM POR REFORÇO PARA UM MODELO BAYESIANO DE CONSCIÊNCIA SITUACIONAL

Este Trabalho de Conclusão de Curso foi julgado aprovado para a obtenção do Título de “Bacharel em Sistemas de Informação".

Florianópolis, de 2017.

Prof. Dr. Cristian Koliver
Coordenador do Curso

Banca Examinadora:

Prof. Dr. Elder Rizzon Santos
Orientador

Prof. Dr. André Wüst Zibetti

Me. Thiago Ângelo Gelaim

À minha família.

AGRADECIMENTOS

Agradeço à minha família, especialmente aos meus pais, Claudinei e Jane, por sempre me incentivarem nos estudos e representarem, pra mim, grandes exemplos de pessoas a serem seguidos. Ao meu irmão, Leopoldo, pela imagem que representa para mim. Dentre meus outros familiares, os quais também foram peças chaves para mim durante essa formação, agradeço em específico à minha prima, Poline, pelo seu sempre disponível ombro amigo, bem como as conversas e incentivos que significam muito para mim.

Agradeço à minha namorada, Giselle, pela sua compreensão indescritível e conforto oferecido. Espero estar a altura para retribuir tanto amor e carinho.

Agradeço imensamente ao professor Elder, pela sua paciência e, acima de tudo, gentileza, durante a orientação desse trabalho e fase de minha vida.

Não poderia deixar de agradecer aos meus amigos pelas conversas, gargalhadas e todos momentos especiais que presenciamos, sem vocês essa caminhada não teria sido a mesma.

Agradeço aos membros da banca, Professor André e Thiago, pelas considerações e apontamentos realizados, assim como pelo acompanhamento nesse processo. Por fim, agradeço ao espaço disponibilizado pelo IATE nesses últimos meses de trabalho.

Even though you're fed up, you gotta keep your head up.

Tupac Shakur

RESUMO

Redes Bayesianas são modelos gráficos para raciocinar e representar um conhecimento sobre um meio incerto. Seus nós representam as variáveis, discretas ou contínuas, e os arcos são conexões diretas entre elas. São aplicáveis em diversos tipos de problemas, como: diagnósticos de doenças, aquisição de conhecimento em sistemas especialistas e até relações de causas em no disparo de um alarme. Tendo uma forte ligação com as redes neurais artificiais está o surgimento da área denominada aprendizagem de máquina, trazendo consigo novas pesquisas em métodos de aprendizagem automática: por exemplos, indução, reconhecimento de padrões, similaridade ou por reforço. Esses métodos podem ser aplicados em previsão de caracteres manuscritos, preços de casas, significados de palavras, assim como na aprendizagem de redes bayesianas. A enorme quantidade de acidentes que acontecem com pedestres trafegando próximos de vias urbanas está associada ao uso excessivo de dispositivos móveis, por acarretar na falta de consciência em relação ao meio. Tendo isso em mente, o projeto "Road Awareness" propõe um modelo de consciência situacional de usuários de smartphones próximo a vias urbanas, a fim de contruibuir com a diminuição da incidência desses acontecimentos. Com o objetivo de contribuir com o projeto, foi implementado e aplicado um algoritmo de aprendizagem por reforço na aprendizagem dos parâmetros do modelo em questão, a rede Bayesiana. O modelo foi implementado e testado, concluindo-se que o mesmo serve como uma maneira de se aplicar o reforço em um sistema especialista.

Palavras-chave: Inteligência Artificial, Aprendizagem de Máquina, Aprendizagem por Reforço, Redes Bayesianas

ABSTRACT

Bayesian networks are graphical models to reason and represent a knowledge about an uncertain environment. Their nodes represent as variables, discrete or continuous, and the arcs are direct connections between them. They are applicable in in several problems, such as: diagnoses of diseases, acquisition of knowledge in expert systems and even causes relationships in the triggering of an alarm. The emergence of the area known as machine learning is strongly linked to advances in artificial neural networks, bringing new research into automatic learning methods, such as: induction, pattern recognition, similarity, or reinforcement. These methods can be applied in prediction of handwritten characters, house prices, word meanings, as well as in learning of Bayesian networks. The huge amount of accidents that occur with pedestrians traveling near urban roads is associated with the excessive use of mobile devices, as it causes a lack of awareness of the environment. With this in mind, the Road Awareness project proposes a situational awareness model of smartphone users near urban roads, to build up with the decrease in the incidence of these events. To contribute to the project, an algorithm of reinforcement learning was implemented and applied in the parameter learning of the present model, a Bayesian network. The model was implemented and tested, concluding that it serves as a way to apply the reinforcement in a specialized system.

Keywords: Artificial Intelligence, Machine Learning, Reinforcement Learning, Bayesian Networks

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SUMÁRIO

1 INTRODUÇÃO

"A Inteligência Artificial (IA) pode ser definida como o ramo da ciência da computação que se ocupa da automação do comportamento inteligente" (LUGER, 2013, p. 21). A automação do comportamento inteligente, ou IA, pode ser feita com uma abordagem simbólica, fazendo referência a objetos e relações de um domínio com o uso de símbolos; e também com uma abordagem conexionista, por meio de conexões entre componentes simples e interativos, ajustados por um processo de aprendizado (LUGER, 2013).

Devido ao surgimento e desenvolvimento de algoritmos de aprendizado prático, houve um crescente interesse de pesquisadores na abordagem conexionista e assim nas redes neurais artificiais. Proporcionando a origem de uma área chamada de aprendizagem de máquina (AM), criada com o objetivo de resolver problemas de natureza prática, por meio de métodos e modelos da estatística e teoria da probabilidade (MITCHELL, 1997).

Responsável pelo surgimento de carros automáticos, recursos de reconhecimento de fala e em otimizações em busca na web, a AM está tão difundida que é utilizada diariamente sem que percebamos. O fato de envolver a criação e o manuseio de algoritmos e métodos que permitam o computador a aprender por experiências e dados analíticos, possibilita a um sistema a capacidade de aperfeiçoar-se sozinho e, por consequência, ser mais eficiente.

> "A solução óbvia é permitir que esses sistemas aprendam por conta própria, seja por sua própria experiência, por analogia, por exemplos, por um professor que lhes diga o que fazer, ou por recompensa ou punição, dependendo dos resultados" (LUGER, 2013, p. 23).

De acordo com a estrutura do problema a ser resolvido, diferentes formas de AM podem ser aplicadas. Cada uma delas possui características particulares e comuns que possibilitam ao sistema adquirir conhecimento sobre o seu domínio.

Um dos modelos de AM é o supervisionado. Nele existe uma espécie de professor a fim de ajudar o sistema no desenvolvimento da função responsável pela classificação. "Em aprendizagem supervisionada (AS), o agente observa alguns exemplos de pares de entrada e saída e aprende uma função que mapeia uma nova entrada para uma saída" (RUSSELL; NORVIG, 2003, p. 695, traduzido pelo autor). Em ou-

tras palavras, é manuseado um conjunto de dados categorizados, com a finalidade de treinar o classificador e então utilizá-lo com dados que ainda não foram processados. Portanto, de acordo com o que já foi visto tenta-se prever os que se seguem. Um exemplo de seu uso seria em calcular a probabilidade de certas operações de crédito serem fraudulentas de acordo com seu histórico.

Em situações onde não se possui bases rotuladas, é necessário uma abordagem diferente, conhecida como não-supervisionada. Embora o sistema não tenha um professor para guiá-lo, é responsabilidade do próprio algoritmo de aprendizado avaliar seus conceitos (LUGER, 2013), onde seu objetivo é explorar os dados e encontrar estruturas padronizadas ou relações nele. Por conseguinte, é possível agrupar os elementos com características similares e assim tratá-los de formas diferentes. Usualmente utilizado por aplicações de recomendação de itens, categorização de clientes com perfis semelhantes, essa abordagem também é mencionada como aprendizagem por observação e descoberta.

Diferindo-se das abordagens citadas acima, existe a aprendizagem por reforço (AR), sendo comumente utilizada com agentes que devem aprender sobre seu comportamento por interações de tentativa e erro (LUGER, 2013). É apresentado a ele a recompensa pela escolha feita e o estado avançado, porém não é dito qual seria a melhor opção de acordo com seus interesses a longo prazo (KAELBLING; LITTMAN; MOORE, 1996). Tem-se que uma das razões pelas quais essa maneira tornou-se popular foi por servir como uma ferramenta teórica para estudar os princípios de um agente aprendendo a agir. Suas aplicações vão desde robótica e produção industrial até problemas de busca combinatória, como o modo do computador em jogos.

Há casos em que se é necessário a utilização de um modelo semi-supervisionado, podendo serem utilizados dados categorizados, difíceis de conseguir e onde há a necessidade de um certo esforço humano; e não categorizados, de fácil obtenção, porém existem poucas técnicas para trabalhar com eles. Dessa forma, o agente aproveita os qualificados para treinar o método classificador e, em seguida, classificar os não qualificados (ZHU, 2005). Os elementos com maiores taxas de confiança são colocados juntos do primeiro conjunto e o processo é repetido. Essa técnica é conhecida como *self-teaching* ou *bootstrapping*.

Redes Bayesianas (RB) foram criadas para permitir uma eficiente representação sob a incerteza de um determinado conhecimento. Essa abordagem propicia uma aprendizagem por experiência, e combina alguns conceitos da IA clássica e das redes neurais. Uma RB é um grafo dirigido em que cada nodo tem uma informação probabilística

quantitativa correspondente. A topologia dessa rede (o conjunto de nodos e arestas) indica as relações condicionais que existem no domínio. Uma vez que a estrutura tiver sido estabelecida, é necessário especificar a distribuição de probabilidade condicional de cada variável, de acordo com seus parentes (PEARL, 2014).

É de comum acordo que a elaboração da estrutura de uma rede e seus parâmetros requer a necessidade de um especialista e uma análise criteriosa do domínio. Com o crescimento das pesquisas na área de IA meios de aplicar técnicas de aprendizagem em RB foram e têm sido desenvolvidos. Sendo assim, este trabalho de conclusão de curso atuará na análise, desenvolvimento e experimentação de uma técnica de aprendizagem por reforço com o propósito de aprender os parâmetros da rede.

1.1 OBJETIVOS

1.1.1 Objetivo Geral

Este trabalho de conclusão de curso tem como objetivo a implementação e avaliação de uma técnica de AM sob uma RB. A partir de um conjunto de dados provido por um projeto é implementado e aplicado um algoritmo de aprendizagem por reforço a fim de aprender os parâmetros probabilísticos de uma rede bayesiana para raciocínio sob atenção situacional.

1.1.2 Objetivos Específicos

1. Análise do estado da arte das técnicas de aprendizagem por reforço aplicáveis em modelos bayesianos.
2. Preparação, normalização e discretização dos dados.
3. Definição dos requisitos do modelo bayes.
4. Seleção de técnicas de aprendizagem por reforço e experimentação com o modelo bayesiano.
5. Estudo e aplicação das melhores abordagens elencadas no objetivo anterior.
6. Testar o modelo criado.

1.2 ORGANIZAÇÃO DO TEXTO

O trabalho é dividido em 5 capítulos: o primeiro traz a introdução, os objetivos, a metodologia de pesquisa e o tópico em questão. No capítulo seguinte é apresentada toda a fundamentação teórica realizada. O capítulo três apresenta os trabalhos relacionados, suas características e abordagens escolhidas.

O capítulo 4 apresenta o estudo de caso utilizado, assim como informações relevantes sobre o conjunto de dados, a RB criada e a proposta de modelo de aplicação e AR. Nesse capítulo também são apresentados os resultados e considerações sobre o modelo. Por último, o capítulo 5 apresenta a conclusão e os possíveis trabalhos futuros.

1.3 MOTIVAÇÃO E JUSTIFICATIVA

Esse trabalho contribui em parte do projeto de pesquisa *Computational Model of Situational Awareness for users of Smartphones*. O projeto propõe o desenvolvimento de um modelo bayesiano de atenção situacional de usuários que trafegam próximo a rodovias e áreas urbanas com dispositivos móveis (SANTOS; FAZENDA, 2016). Portanto, de acordo com dados provindos de análise e experimentação do projeto, será aplicada uma técnica com o intuito de alimentar essa rede previamente definida. Esse modelo poderá ser utilizado para ajudar a reduzir a incidência de acidentes com pedestres por conta da distração provocada por smartphones. Colaborando de tal forma com o projeto citado.

Objetivando contribuir com o desenvolvimento de um modelo computacional a aplicação de reforço se dará de forma a proporcionar ajustes no modelo gerado. Essa melhoria é fundamental para que se haja um modelo mais fidedigno e utilizável de acordo com as situações que lhe são apresentadas.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem como propósito explicar os fundamentos necessários para compreensão do trabalho desenvolvido. Primeiramente, na seção 2.1 é explicado brevemente o conceito de AM, seguido de uma explicação sobre os seguintes tipos: supervisionada, não supervisionada e por último, mais detalhadamente, da abordagem escolhida como aplicação para este trabalho, a AR. Essas também divididas em subseções dentro da seção 2.1.

Na seção 2.2 será explicado sobre RB's, os conceitos necessários para o seu entendimento e na seção seguinte, 2.3, será explicado sobre o estudo de caso utilizado para este trabalho, um estudo sobre consciência situacional.

2.1 APRENDIZAGEM DE MÁQUINA

AM tem como principal objetivo construir programas que automaticamente melhoram por experiência. Sendo um campo multidisciplinar, baseia-se em conceitos e resultados de campos como Estatística, Inteligência Artificial, Filosofia, Teoria da Informação, Teoria do Controle e Ciências Cognitivas (MITCHELL, 1997).

Segundo Mitchell (1997), uma aplicação aprende por meio de um conjunto de experiências em um domínio. Porém, apenas se, a partir de um conjunto de tarefas e com uma medida de desempenho, há uma melhoria desta medida de acordo com o aumento de sua experiência nas mesmas tarefas em que estava sendo aplicado.

Existem alguns modelos de aprendizagem que podem ser aplicados para cada natureza de um problema, assim como há algoritmos específicos para cada uma. Esta forma de organizar os algoritmos e técnicas de aprendizagem automática é útil por ajudar a organizar nas atribuições dos dados escolhidos para treinamento e também na preparação do modelo de processamento.

2.1.1 Aprendizagem Supervisionada

Das diversas aplicações que existem para AM, uma das mais significantes é *data mining* (KOTSIANTIS; ZAHARAKIS; PINTELAS, 2007). Onde, segundo Mitchell (1997), algoritmos de *machine learning* vem

sendo utilizados para descobrir conhecimento de valor em grandes bases de dados, como: transações financeiras, pedidos de empréstimos, registros médicos, de manutenção de equipamentos e etc.

Pessoas estão frequentemente propensas a cometer erros durante a análise, ou mais especificamente, tentando estabelecer relações entre múltiplos atributos em instâncias de bases de dados. Tendo isso em vista a aprendizagem pode ser eficientemente aplicada nesses tipos de problemas (KOTSIANTIS; ZAHARAKIS; PINTELAS, 2007).

AS é um dos tipos de AM. Em específico é a aplicação de algoritmos que utilizam de uma análise de instâncias providas para criar hipóteses, as quais são utilizadas para predizer sobre futuras instâncias (KOTSIANTIS; ZAHARAKIS; PINTELAS, 2007). Sua função implica em aprender um mapeamento entre uma variável de entrada e uma de saída, e aplicar esse mapeamento para predizer a saída de dados não vistos até então (CUNNINGHAM; CORD; DELANY, 2008).

Barber (BARBER, 2012), define que tarefa de uma aplicação utilizando AS é, a partir de um conjunto de dados $D = \{(x^{\mathrm{n}}, y^{\mathrm{n}}), n = 1, ..., N\}$, aprender a relação entre x e y de tal maneira que, quando for recebido um novo registro contendo x, o valor predito de y seja preciso. O termo supervisionado indica que existe um supervisor com a tarefa de especificar o valor de saída de y para cada dado de entrada de x do conjunto de dados utilizado.

A Tabela 1 ilustra um exemplo de conjunto de dados. Neste caso, a partir das informações sobre os registros de clientes que pediram empréstimo a um banco, pode ser aplicado um modelo de AS a fim de predizer a qual classe (bom, regular, mau pagador) uma pessoa com determinadas características se encaixaria.

Dados de registros de clientes de empréstimos					
Cliente	Histórico	Primeiro	...	Tem empréstimo	Classe
1	Ruim	Não	...	Sim	Mau pagador
2	Bom	Sim	...	Não	Bom pagador
3	Ótimo	Não	...	Não	Bom pagador
...	...	...	...	...	...

Tabela 1 – Exemplo de conjunto de dados e a classe predita. Fonte: Elaborado pelo autor (2017)

Na Figura 1 descreve-se o processo de aplicação de aprendizagem em um problema do mundo real. De acordo com o problema, o primeiro passo é a coleta do conjunto de dados. Na presença de um especialista

no assunto, a identificação de quais campos são mais significantes reduzirá o esforço na etapa a seguir, a qual é o pré-processamento dos dados. Caso contrário, será necessário a investigação de quais atributos são mais relevantes (KOTSIANTIS; ZAHARAKIS; PINTELAS, 2007).

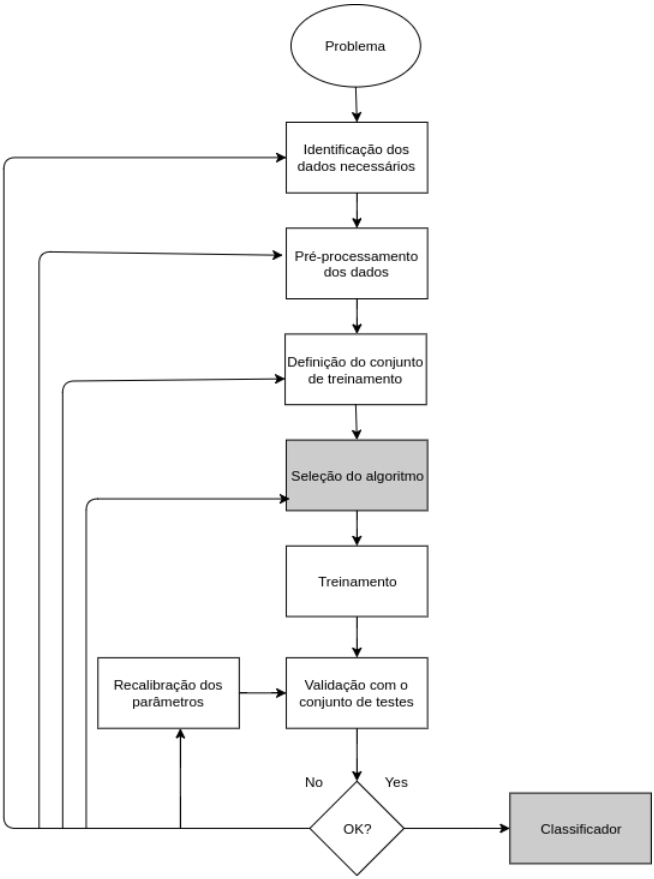

Figura 1 – Processo de aplicação de AS. Fonte: Kotsiantis, Zaharakis e Pintelas (2007)

.

Segundo Han, Pei e Kamber (2011), bases de dados são altamente suscetíveis a inconsistência, ausência e impureza nas suas instâncias devido ao seu grande volume. Sendo assim, para aumentar a qualidade do modelo, deve-se ter uma boa qualidade de dados. A aplicação de técnicas como: limpeza, integração, redução e transformação podem

melhorar a acurácia e eficiência dos algoritmos.

Acurácia de um modelo é a porcentagem das instâncias que foram corretamente classificadas pelo classificador, dado um conjunto de testes (HAN; PEI; KAMBER, 2011). "Se fossemos utilizar o conjunto de treinamento para medir a acurácia do classificador, essa estimativa iria ser bem otimista, uma vez que o classificador tende a se acostumar com o conjunto de dados" (HAN; PEI; KAMBER, 2011, p. 330, traduzido pelo autor). Ou seja, durante a construção do modelo este pode incorporar certos padrões do conjunto de treinamento que não estão presentes no conjunto total. Por isso a necessidade de separar o conjunto de dados em dois conjuntos independentes, sendo o de treino utilizado para construir o modelo e o de teste para avaliação.

Sendo crítica a etapa de seleção de algoritmo, a avaliação do modelo é frequentemente baseada na acurácia de predição de novas instâncias pelo classificador (KOTSIANTIS; ZAHARAKIS; PINTELAS, 2007). No entanto, de acordo com cada domínio, diferentes métricas de desempenho devem ser levadas em consideração (CARUANA; NICULESCU-MIZIL, 2006).

		Classe predita		
		Sim	Não	Total
Classe atual	Sim	VP	FN	P
	Não	FP	VN	N
	Total	P'	N	P + N

Tabela 2 – Matriz de confusão. Fonte: Han, Pei e Kamber (2011)

.

A Tabela 2 exemplifica uma matriz de confusão. Conforme Han, Pei e Kamber (2011), dado m classes, uma matriz de confusão é uma tabela de pelo menos m por m. Sendo uma ferramenta para avaliar o quanto um modelo consegue prever tuplas de diferentes classes, nela são introduzidos quatro termos:

- Verdadeiros positivos (VP): tuplas positivas que foram corretamente classificadas.
- Verdadeiros negativos (VN): tuplas negativas que foram corretamente classificadas.
- Falsos positivos (FP): tuplas negativas que foram classificadas erroneamente como positivas pelo classificador.

- Falsos negativos (FN): tuplas positivas as quais foram erroneamente classificadas como negativas.

Na Equação 2.1, precisão é a porcentagem de instâncias corretamente classificadas como positivas, também conhecida como uma medida de exatidão. Em outras palavras, mede a quantidade de tuplas que o modelo classificou como de uma específica classe, que realmente pertencem a essa classe. *Recall*, representado na Equação 2.2, é uma medida de completude: a porcentagem de tuplas positivas que foram corretamente categorizadas (HAN; PEI; KAMBER, 2011). Essas medidas são comumente utilizadas para avaliação de recuperação de informação. A Equação 2.3, representa a fórmula da acurácia, e ela é a taxa de predições corretas realizadas pelo modelo para um determinado conjunto de dados, tanto positivas quanto negativas.

$$precisao = \frac{VP}{VP + FP} \tag{2.1}$$

$$recall = \frac{VP}{VP + FN} \tag{2.2}$$

$$acuracia = \frac{VP + VN}{P + N} \tag{2.3}$$

De acordo com Han, Pei e Kamber (2011), classificação é o processo de encontrar um modelo ou função que descreve e distingue classes de dados. O modelo pode ser representado como: árvores de decisão, fórmulas matemáticas, regras de classificação ou redes neurais artificiais. Caso o atributo classe seja discreto, o problema é visto como de classificação; caso contrário, o problema de indução é conhecido como de regressão.

Uma árvore de decisão é um diagrama de atividades em formato de árvore, onde cada nodo representa um teste em um valor possível de um atributo. Os galhos caracterizam o valor de saída desse teste e as folhas a classe (HAN; PEI; KAMBER, 2011). A Figura 2 exemplifica uma árvore de decisão, seus nodos, galhos e folhas.

A Figura 3 ilustra o mesmo problema, antes representado como uma árvore de decisão, agora como regras de classificação. Já na Figura 4 está na forma de uma rede neural artificial. Onde, segundo Han *et. al.* (HAN; PEI; KAMBER, 2011), quando utilizada pra classificação, uma rede neural, nada mais é do que uma coleção de unidades de processamento (como neurônios), com conexões (com seus pesos e medidas) entre elas.

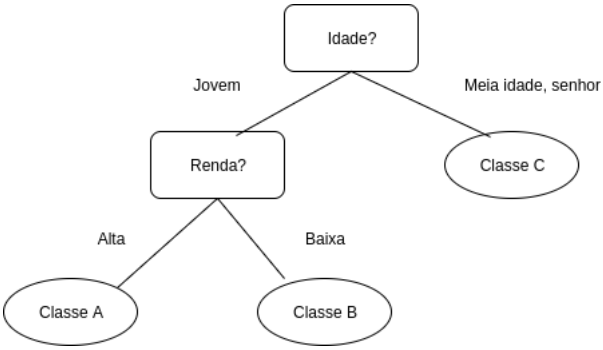

Figura 2 – Exemplo de árvore de decisão. Fonte: Han, Pei e Kamber (2011).

idade(X, “Jovem”) e renda(X, ”alta) $\rightarrow classe(X, \text{“}A\text{“})$
idade(X, “Jovem”) e renda(X, ”baixa) $\rightarrow classe(X, \text{“}B\text{“})$
idade(X, “Meia idade”) $\rightarrow classe(X, \text{“}C\text{“})$
idade(X, “Senhor”) $\rightarrow classe(X, \text{“}C\text{“})$

Figura 3 – Exemplo de regras de classificação. Fonte: Han, Pei e Kamber (2011).

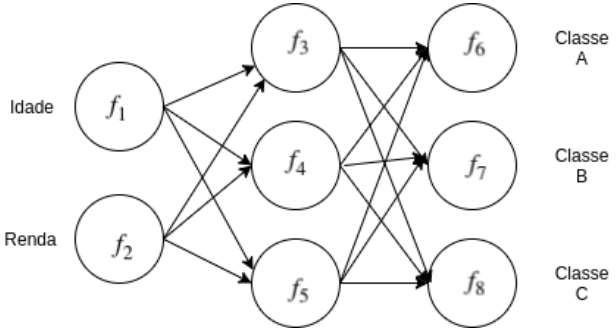

Figura 4 – Rede neural artificial. Fonte: Han, Pei e Kamber (2011)
.

2.1.2 Aprendizagem Não Supervisionada

O aprendizado não supervisionado (ANS), por sua vez, elimina a necessidade de um professor e demanda que o próprio algoritmo de

aprendizagem avalie os conceitos (LUGER, 2013). Segundo Russel (RUSSELL; NORVIG, 2003), o agente aprende padrões nos dados de entrada, mesmo sem um *feedback* fornecido. A tarefa de aprendizagem mais comum é conhecida como *clustering*.

Clustering é uma técnica utilizada para detectar, agrupar e distinguir diferentes categorias de uma coleção de instâncias. Em seu livro, Russel (RUSSELL; NORVIG, 2003), descreve um exemplo de sua aplicação como, a partir de dados sobre o espectro de milhares de estrelas, quais têm características mais em comum? Quantos tipos existem?

Em seu livro, Han *et. al.* (HAN; PEI; KAMBER, 2011) cita que existem diversos algoritmos de *clustering* na literatura e é uma tarefa difícil categorizá-los, uma vez que muitos utilizam técnicas de não apenas uma categoria. Entretanto, fundamentalmente os métodos podem ser separados da seguinte forma:

- De particionamento: organiza os objetos de um conjunto, a partir de uma medida de distância, em grupos exclusivos. Um critério para avaliar o particionamento é de que os objetos de cada um dos centróides são próximos entre si e os centróides estão longe ou são bem diferentes dos outros;

- Hierárquicos: tendo uma métrica de similaridade, são realizadas etapas para formar os agrupamentos de baixo para cima. Primeiramente, são examinados os objetos e formado agrupamentos com os pares de grau de maior similaridade. Posteriormente, definidas as características do agrupamento como uma função da média das características, são substituídos os componentes por essa nova definição. Tal processo é repetido sobre a coleção de objetos até que seja reduzido a um único grupo;

- Baseados em densidade: são métodos muito parecido com o de particionamento, porém não formam grupos em formatos esféricos. A ideia de um método em densidade é continuar aumentando um *cluster* até que chegue em um número de objetos (densidade). Esse método pode eliminar *outliers* e descobrir *clusters* de diferentes tamanhos;

- Baseados em redes: separam os objetos em um número finito de células que forma uma espécie de estrutura de rede. A maior vantagem destes métodos é o tempo de processamento, dependendo apenas da quantidade de células do espaço. Métodos baseados em redes podem ser integrados em modelos hierárquicos e baseados em densidade.

A técnica de agrupamento utiliza medidas de similaridade em uma coleção de objetos não classificados para organizá-los em classes que satisfazem algum padrão. Uma medida de similaridade bem conhecida e utilizada é a distância euclidiana (LUGER, 2013).

$$d(i,j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + ... + (x_{ip} - x_{jp})^2}. \quad (2.4)$$

$$d(i,j) = \mid x_{i1} - x_{j1} \mid + \mid x_{i2} - x_{j2} \mid + ... + \mid x_{ip} - x_{jp} \mid . \quad (2.5)$$

A Equação 2.4 representa a distância euclidiana. Sendo $i = (x_{i1}, x_{i2}, ..., x_{ip})$ e $j = (x_{j1}, x_{j2}, ..., x_{jp})$ dois objetos de p atributos numéricos, a equação trata de comparar a similaridade entre eles, de acordo com a diferença de valor de cada um de seus atributos. A Equação 2.5 é a fórmula da distância de *Manhattan* (HAN; PEI; KAMBER, 2011).

Um dos métodos mais bem conhecidos e comumente utilizada para *clustering* é o *k-means*, sendo um método de particionamento. O pseudocódigo ilustrado na Figura 5 explica o seu funcionamento básico (HAN; PEI; KAMBER, 2011).

Algoritmo 1 Algoritmo *k-means*

Entrada: Um número k de *clusters* e um conjunto de dados contendo n objetos
Saída: Um conjunto de k *clusters*
Método:
arbitrariamente escolha k objetos do *dataset* para serem os *clusters* iniciais
repita:
atribua cada objeto ao *cluster* em que ele é mais similar baseado nos centróides atuais do *cluster*
atualize os centróides dos *clusters*
até não haver mudanças

Figura 5 – Pseudocódigo *k-means*. Fonte: Han, Pei e Kamber (2011).

Primeiro, o algoritmo aleatoriamente escolhe k objetos do conjunto de dados para serem os centróides de cada um dos *clusters*. Para cada um dos objetos restantes, um dos centróides é atribuído a eles de

acordo com uma medida de similaridade, a medida euclidiana. Após, o algoritmo computa os novos centróides de acordo com os objetos atribuídos a eles e atualiza os centróides dos objetos que haviam sido atribuídos no passo anterior, de acordo com os novos. A iteração continua ocorrendo até que não haja nenhuma modificação nos centróides (HAN; PEI; KAMBER, 2011).

A Figura 6 ilustra o processo de execução do algoritmo explicado. A partir dos grupos gerados é possível interpretar por um olhar diferente o conjunto de dados utilizado, como por exemplo: diferentes grupos de usuários de acordo com seu comportamento e o que compartilham em comum e outros problemas em que não se tem informações suficientes.

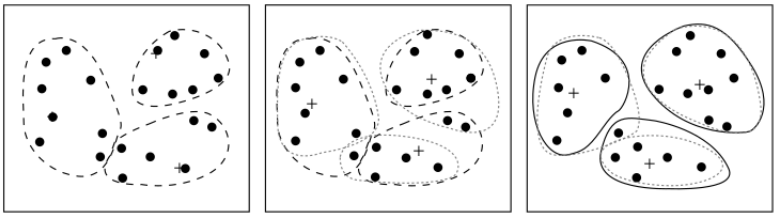

Figura 6 – Em ordem: agrupamento inicial, sua atualização e agrupamento final. Fonte: Han, Pei e Kamber (2011).

2.1.3 Aprendizagem por Reforço

O conceito que hoje vem a ser considerado como AR foi identificado em um artigo de 1950, juntamente com o teste de Turing, ideias sobre AM e algoritmos genéticos (RUSSELL; NORVIG, 2003). Envolvendo não apenas assuntos relacionados a IA, mas tópicos como configuração de objetivo, planejamento e percepção, estas características formam uma ponte com disciplinas da engenharia.

Um dos principais ramos que a AR se baseou para sua construção foi da aprendizagem por tentativa e erro. Originada na psicologia animal, segundo Sutton e Barto (1998), em essência dá-se que ações escolhidas num determinado instante, com suas respectivas recompensas, boas ou ruins, tem uma tendência a serem retomadas. Conhecida por lei de causa e efeito por descrever a consequência de eventos em reforço na tendência da escolha de certas ações.

Outro ramo que deu origem aos conceitos relacionados a essa

abordagem hoje, foi da programação dinâmica. Categorizando-se por propor métodos para resolver problemas de otimização de controle, é amplamente considerada a única maneira factível de resolver problemas estocásticos genéricos desta área (SUTTON; BARTO, 1998).

Mitchell (1997) cita que a grande diferença entre AR e programação dinâmica é que historicamente os problemas dessa assumem que o agente possui conhecimento sobre a função de transição de estados e a função de recompensa, em contraste, algoritmos de reforço tipicamente assumem que o aprendiz não tem tal conhecimento. Apesar de considerar parte do trabalho da programação dinâmica como da AR, Sutton e Barto (SUTTON; BARTO, 1998) frisam a grande diferença da natureza dos métodos de cada uma das áreas. Considerando que na primeira, a programação dinâmica, tem como principal requisito o total conhecimento sobre o sistema a ser controlado.

De acordo Sutton e Barto (1998), AR é aprender como mapear situações para ações, de forma a maximizar um valor numérico como sinal. O aprendiz deve descobrir quais ações geram maiores valores tentando inúmeras delas. Assim como deve estar preparado para compreender que suas tomadas de decisão não afetam apenas a sua próxima situação, mas também nos seus próximos estados.

Ainda conforme Sutton e Barto (1998), um modelo de AR consiste basicamente de:

- um conjunto discreto de estados do ambiente, S
- um conjunto discreto das ações do agente, A
- um conjunto de sinais de reforço, normalmente entre $\{-1, 1\}$

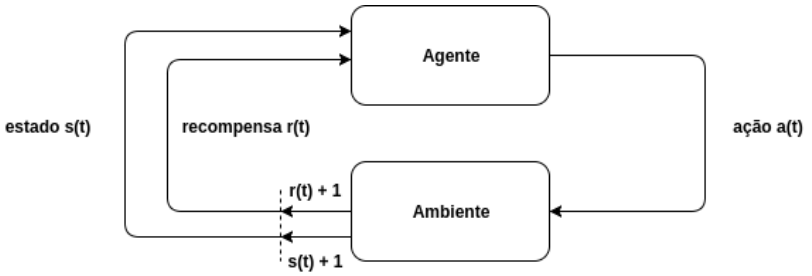

Figura 7 – Natureza de um agente e seu ambiente. Fonte: Sutton e Barto (1998)

.

A Figura 7 ilustra o comportamento básico de um agente interagindo com o seu ambiente em uma perspectiva de AR. Nele, o agente, tendo recebido informações que representam o estado atual do ambiente, executa ações possíveis a este momento e tem como retorno o seu valor de recompensa. Consequentemente, o ambiente se altera respondendo para estas ações e apresentando novas observações para o agente.

Onde um agente interage em um ambiente S, o qual representa o seu conjunto de possíveis estados, numa sequencia de etapas em um tempo discreto $t = 0, 1, 2, 3, ...$ Nele, o agente pode tomar possíveis ações de acordo com o estado no momento, $a_t \in A(s_t)$, onde $A(s_t)$ é o conjunto de ações disponíveis no estado s_t. Desta forma, cada vez que o agente executa uma ação a_t em um momento s_t, é retornado para o agente uma recompensa r_{t+1} indicando o valor imediato desta transição estado-ação. O ambiente se altera passando a se encontrar em um novo estado s_{t+1}.

Uma maneira intuitiva de entender a relação entre um agente e seu ambiente pode ser descrita da seguinte maneira:

Ambiente:	Você está no estado 65.
	Você tem 4 ações possíveis.
Agente:	Eu vou tomar a ação 2.
Ambiente:	Você recebeu um reforço de 7 unidades.
	Você está no estado 15.
	Você tem 2 ações possíveis.
Agente:	Eu vou tomar a ação 1.
Ambiente:	Você recebeu um reforço de -4 unidades.
	Você está no estado 65.
	Você tem 4 ações possíveis.
Agente:	Eu vou tomar a ação 2.
Ambiente:	Você recebeu um reforço de 5 unidades.
	Você está no estado 44.
	Você tem 5 ações possíveis.
...	...

Tabela 3 – Ecossistema de um agente e ambiente. Fonte: Adaptado de Kaelbling, Littman e Moore (1996)

Diferente da maioria dos métodos de AM, não é informado para a entidade quais ações tomar, porém ela deve ter a capacidade, até certo ponto, de perceber o estado do ambiente e de tomar decisões que irão afetá-lo (KAELBLING; LITTMAN; MOORE, 1996). Para tal percepção, o

aprendiz deve conseguir compreender o que se passa e buscar por um fim específico. Assim como o agente deve ter objetivos, um ou mais, que estejam relacionados ao ambiente inserido.

Em um geral, supõe-se um ambiente não-determinístico, ou seja, que tomando a mesma ação, no mesmo estado em duas ocasiões diferentes pode gerar estados disponíveis diferentes, assim como diferentes valores de recompensa (KAELBLING; LITTMAN; MOORE, 1996).

2.1.3.1 A tarefa de aprender

"A tarefa do agente é encontrar uma política π, mapeando estados a ações, que maximizem alguma medida de reforço a longo termo" (KAELBLING; LITTMAN; MOORE, 1996, p. 239, traduzido pelo autor). Como dito, a tarefa do agente é encontrar uma política, um mapeamento de estados para ações que sejam melhores em determinadas situações, não só de acordo com o valor numérico de recompensa imediato, mas também com o valor de longa distância.

Segundo Russell e Norvig (2003), são considerados três *designs* de agentes para AR. Um agente baseado em utilidade (*utility-based agent*) aprende uma função de utilidade durante sua execução e a utiliza para selecionar ações que maximizem o seu valor de retorno. Ele deve modelar e observar o ambiente para tomar suas decisões, pois deve conhecer os estados para os quais suas ações o levarão. Apenas desta forma ele poderá a aplicar a função de utilidade para os estados que lhe forem apresentados. Por exemplo, num programa que joga *backgammon*, esse agente precisa saber quais são os movimentos legais e como eles afetam as posições do tabuleiro.

Já um agente *Q-Learning* desenvolve uma função de utilidade de ação, também conhecida como *Q-function*, retornando um valor de tomada de decisão. É capacitado em comparar as utilidades esperadas das suas opções disponíveis sem conhecer os estados resultantes. Não depende de ter um modelo do ambiente, em contrapartida por não saber onde suas ações o levarão, um agente *Q-learning* não consegue olhar a frente, restringindo a sua habilidade em aprender (RUSSELL; NORVIG, 2003).

Por outro lado, um agente reflexo (*reflex agent*) aprende uma política que mapeia diretamente estados para ações. Subdivindo-se entre simples e baseado em modelo, a diferença característica entre ambos é que na última é mantido um estado interno responsável por salvar parte da história percebida (RUSSELL; NORVIG, 2003).

Ainda segundo Russell e Norvig (2003), agentes de reflexo simples (*simple reflex agents*) são considerados simples, porém de inteligência limitada. Fatos inobserváveis podem causar sérios problemas. Isto se dá por conta de implementarem regras de ação de acordo com uma condição. Como por exemplo, um agente simples como um motorista inserido no trânsito. Neste caso poderia-se criar uma regra em que caso o carro da frente freie, o agente comece a freiar também. Desta forma é imprescindível que tal percepção, como um único *frame* de vídeo, resuma as mudanças necessárias para o agente. Infelizmente nem sempre é possível em uma imagem determinar se um carro está freiando ou não, levando a um comportamento inadequado do agente, podendo vir a freiar continuamente ou até mesmo nunca. A Figura 8 a seguir resume a estrutura de um agente de reflexo simples.

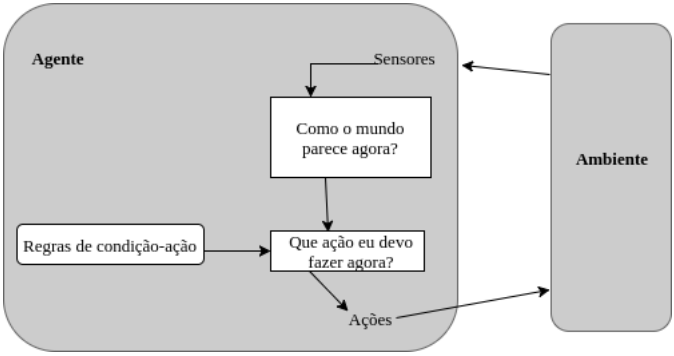

Figura 8 – Interface de um agente simples. Fonte: Russell e Norvig (2003)

.

Uma solução para o suposto problema de limitação de um *simple reflex agent* seria a utilização de um estado interno que estaria refletindo parte dos aspectos não observáveis no estado atual. Nesse quesito apenas o *frame* anterior capturado pelo sensor já seria o suficiente. Habilitando o agente a detectar quando duas luzes vermelhas no veículo da frente ligam ou desligam simultaneamente (RUSSELL; NORVIG, 2003).

2.1.3.2 Função de recompensa

Em jogos ou situações em que não se tem o feedback de um professor, um agente precisa saber quão bom ou ruim uma ação sua é em

relação ao ambiente em que ele está inserido. Por exemplo um jogo de xadrez, onde um sistema deve aprender a jogá-lo. Sem o devido feedback sobre suas decisões, um agente não vai ter o embasamento necessário sobre qual movimento obter. Mesmo utilizando um modelo transicional para seus movimentos e para prever os do oponente, é preciso haver um modelo responsável por informar quando algo bom aconteceu devido a um xeque-mate no oponente ou algo ruim aconteceu por seu xeque-mate e vice-versa. Esse tipo de informação, uma recompensa, é também chamado de reforço e a sua utilização pode ser tanto no fim de um jogo, quanto mais frequentemente, a cada passo (RUSSELL; NORVIG, 2003).

Usualmente utilizado em problemas interativos, onde, na maioria dos casos, não é prático obter exemplos representativos para todas situações e estados do ambiente que o agente deve agir. Desse modo é necessária a habilidade do agente de poder aprender por suas próprias experiências, de acordo com o feedback colhido pelas alterações no local inserido por conta das suas ações (SUTTON; BARTO, 1998).

Uma função de recompensa mapeia cada par estado-ação do ambiente e do agente em um número indicando o quão interessante, de acordo com os objetivos, é a situação daquele estado em específico. As ações do agente determinam não apenas o seu valor de recompensa imediato, mas também seu próximo estado. Dessa forma o agente deve ter a habilidade de aprender por reforço atrasado[1], pois pode levar uma sequencia de ações com valores baixos de recompensa, até obter um estado com um alto valor de reforço. E é exatamente o seu modelo de longo prazo de otimização que determina como o agente lida com os futuros valores de recompensa (KAELBLING; LITTMAN; MOORE, 1996)

Sua natureza de tomadas de decisão sequenciais exprime a ideia de consequencias de longo termo. Objetivando escolher ações que maximizem uma recompensa total futura, essas ações tem a possibilidade de levar a estados mais interessantes apenas no futuro. Assim há o cenário de ser melhor sacrificar a recompensa imedita por um valor maior em longo termo (SUTTON; BARTO, 1998). Um exemplo prático é um investimento financeiro que pode levar até meses para maturar.

Apesar de parecer limitado, na prática a ideia de um sinal de recompensa se provou flexível e amplamente aplicada. Por exemplo, para um robô aprender a andar, cada passo completado para frente poderia receber um sinal positivo. No caso de um robô aprendendo a escapar de uma prisão, a recompensa seria -1 até escapar, onde passaria a ser 1. No mesmo exemplo, porém com outra abordagem, afim do robô

[1] termo traduzido de delayed reward.

aprender a escapar o mais rápido possível, cada passo que o distanciaria mais de sua fuga, receberia um sinal negativo (SUTTON; BARTO, 1998). A maneira de aplicar este sinal de reforço depende de como se quer modelar o problema.

2.1.3.3 Politica de Controle

Segundo Sutton e Barto (1998), a política de controle define o comportamento de um agente em um determinado momento. A cada passo, o agente implementa um mapeamento dos possíveis estados para probabilidades de selecionar cada possível ação. Sendo considerada a principal característica desta abordagem, é dito que ela, por sozinha, é suficiente para determinar o comportamento de uma aplicação.

Há casos em que o ambiente é completamente conhecido pelo agente a priori. Não sendo necessário perceber ou aprender sobre. Nestes casos, tais agentes são frágeis por não estarem aptos a perceber mudanças e alterações de estados, que caso ocorram, o aprendiz não perceberá e tomará ações que ele ainda julga corretas, correndo o risco de falhar inúmeras vezes sem notar (RUSSELL; NORVIG, 2003).

Estando inserido em um ambiente, de começo, desconhecido, não necessariamente uma aplicação de AR tem um estágio de aprendizagem no início. Podendo se encarregar de aprender e explorar concorrentemente. As experiências coletadas em contatos prévios com o ambiente podem ser utilizadas assim quando obtidas (ROVIRA; SLATER, 2017). Assim como para qualquer mudança no local inserido, a aplicação tem a capacidade de adaptar sua estratégia.

2.1.3.4 Aprendizagem por Reforço Passiva

Em AR passiva[2], a política do agente é fixa, ou seja, no estado s o agente sempre executa a ação $\pi(s)$. Seu objetivo é simplesmente aprender o quão boa a sua política de controle é: as utilidades de cada estado (ou o par estado - ação). Deste modo podendo também envolver aprender um modelo do ambiente (RUSSELL; NORVIG, 2003).

Um agente de aprendizagem passiva não tem conhecimento sobre o modelo de transição. Tal modelo representaria a probabilidade de alcançar um determinado estado a partir de outro, depois de escolher uma ação específica. Nem conhece a função de recompensa, a qual

[2]termo traduzido de passive reinforcement learning.

especifica o valor para cada estado (RUSSELL; NORVIG, 2003).

2.1.3.5 Aprendizagem por Reforço Ativa

De acordo com Russell e Norvig (2003), um agente por AR ativo[3], ao contrário do passivo, deve decidir quais ações tomar. Para isso, primeiramente esse precisará aprender um modelo completo com as probabilidades respectivas para cada ação, do que apenas um modelo de política fixa. Assim ele deve, em particular, aprender a melhor ação para cada estado.

Porém, nem sempre escolher a melhor ação para o modelo aprendido poderá levar aos melhores resultados. Uma vez que esse modelo não é verdadeiramente o mesmo que o ambiente. O que é ótimo no modelo pode ser considerado subótimo no ambiente. Simultaneamente, as escolhas de ações não acarretam apenas em valores de retorno como recompensas, elas também contribuem em aprender um modelo mais fiel ao ambiente, afetando no que está sendo percebido e aprendido (RUSSELL; NORVIG, 2003).

2.1.3.6 Investigar ou Explorar

É de fácil compreensão que a sequência de ações escolhida pelo agente influencia diretamente na distribuição dos possíveis estados do ambiente. Isto levanta uma questão sobre qual estratégia de experimentação produz um modelo de aprendizagem mais eficiente. Logo, o aprendiz enfrenta um dilema em escolher entre investigar[4] estados e ações desconhecidos, em favor de ganhar novas informações e descobrir melhores valores de recompensa, ou explorar[5] os estados e ações que ele já sabe que irão gerar um maior valor de recompensa, inclusive maximizando o total acumulado (MITCHELL, 1997).

Também conhecido como o problema do bandido multi armado[6]. Cada escolha de ação seria puxar uma das alavancas, e a recompensa o prêmio por acertar a combinação. Ao longo das interações, ou seja, ações do agente em escolher qual alavanca puxar, o objetivo seria maximizar o valor de recompensa concentrando-se nas que seriam consi-

[3] termo traduzido de active reinforcement learning.
[4] termo traduzido de exploration.
[5] termo traduzido de explotation.
[6] termo traduzido de n-armed bandit problem, o seu nome faz referência a máquinas caça-níqueis comumente ligadas a jogos de azar.

deradas as melhores alavancas (SUTTON; BARTO, 1998).

Considerando que o agente não saiba precisamente os valores de cada ação, embora tenha uma estimativa, uma vez que estivesse mantendo os valores de recompensa, a cada iteração existe pelo menos uma ação em que o seu valor estimado é o maior de todos. Esta é uma *greedy action* e a sua seleção seria a escolha do agente por explorar o seu conhecimento atual das recompensas de cada ação. Caso seja selecionada uma outra ação possível, é dito por estar explorando, porque desta maneira o agente pode melhorar as suas estimativas de valores das *nongreedy actions* (SUTTON; BARTO, 1998).

Com o exposto conclui-se que não é possível explorar e investigar um par de estado-ação ao mesmo tempo. Deste modo a necessidade por um balanceamento entre as políticas se vê como um desafio diferencial que aparece em AR. Ainda em sua obra Sutton e Barto (1998) levantam técnicas possíveis de se buscar este balanço. "Uma solução exata do problema de exploração é inviável, porém simples funções de heurísticas conseguem chegar a um resultado razoável" (RUSSELL; NORVIG, 2003, p. 853, traduzido pelo autor).

2.1.3.7 Aplicações

Um dos motivos pelos quais AR se tornou popular, é por servir como uma ferramenta teórica para estudar os princípios de um agente aprendendo a agir. No entanto, é também utilizada como uma ferramenta computacional prática para construir sistemas autônomos que melhoram o seu desempenho a medida que adquirem experiência (KAELBLING; LITTMAN; MOORE, 1996). Com aplicações em robótica, um de seus problemas é o do balanceamento da carreta (*cart-pole*), ilustrado na Figura 9.

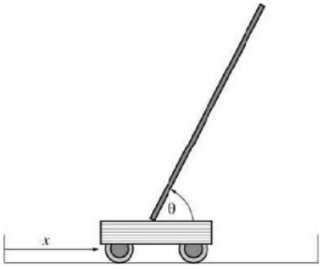

Figura 9 – *Cart-pole*. Fonte: Russell e Norvig (2003)

O carro pode ser movido para a esquerda ou direita por um controlador que observa θ e x. O objetivo é controlar o carro de forma que a estaca fique exatamente na vertical. Diversos artigos em AR e teoria do controle tem sido publicados nesse, aparentemente, simples problema (RUSSELL; NORVIG, 2003).

Sutton e Barto (1998) descrevem outro problema em robótica em que pode ser aplicado. Considerando um robô que tem como tarefa coletar garrafas vazias em um escritório, este tem sensores para detectá-las, um braço e um pegador para colocá-las em uma prateleira acoplada, e roda utilizando baterias recarregáveis. Há também componentes para interpretar informações dos sensores, para navegação e para controlar o braço e a pinça. As decisões sobre qual estratégia utilizar para procurar pelas latas são feitas por um agente utilizando AR, baseado na atual carga da bateria.

O agente decide quando o robô deve ativamente procurar por uma lata por um certo período de tempo; ficar parado a fim de esperar por alguém trazer uma lata; ou voltar para o local para recarregar suas baterias. As recompensas podem ser 0 por boa parte do tempo, porém positivas quando o robô encontra por uma garrafa a coletar e negativas quando descarrega por completo a sua energia.

Desde o surgimento do campo da IA a aplicação em jogos é um de seus principais domínios (KAELBLING; LITTMAN; MOORE, 1996). A primeira aplicação significativa de AR foi também o primeiro programa expressivo de aprendizagem de qualquer tipo: um jogo de damas desenvolvido por Arthur Samuel em 1959 (RUSSELL; NORVIG, 2003).

2.2 REDES BAYESIANAS

Ao tomar uma decisão, um agente precisa levar em consideração as informações que ele tem disponível sobre o seu mundo, o local em que está inserido. O raciocínio deve ser feito com base em suas percepções, as informações captadas ao seu redor, e muitas vezes se requer que algumas simplificações sejam feitas. O simples fato de tentar mapear um conhecimento ou comportamento em regras, pode se tornar muito complexo pela quantidade de exceções que elas podem ter (PEARL, 2014).

2.2.1 Incerteza e Probabilidade

Uma das alternativas para se ter mais precisão e confiabilidade ao raciocionar sobre um domínio é tentar sumarizá-lo de tal forma atribuindo valores númericos a cada proposição dispostos em uma distribuição (PEARL, 2014). Tal maneira de raciocínio é conhecido como probabilístico e suas aplicações são principalmente em domínios em que as relações de causa e efeito não são captadas tão facilmente. A informação probabilística pode indicar e priorizar causas e possíveis explicações para uma evidência, por exemplo: a evidência de quais sintomas são mais prováveis de levar um paciente a ter uma certa doença (LUGER, 2013).

De acordo com Costa (2013) a teoria da probabilidade possibilita uma alternativa ao raciocínio lógico, representnado a incerteza por aleatoriedade. Ela proporciona um método para resumir uma determinada incerteza, utilizando de uma estrutura rigorosa para representação de eventos aleatórios, atribuindo a cada sentença um grau entre 0 e 1. Dela, três conceitos são importantes para o entendimento de RB's:

- probabilidade a priori: é a probabilidade atribuída a um evento antes da realização do experimento;
- probabilidade condicional: probabilidade condicionada. Dado a ocorrência de um evento, probabilidade de que outro aconteça;
- probabilidade a posteriori: é a probabilidade condicional de acordo com a constatação de evidência de um evento.

A probabilidade a posteriori ou condicional proporciona que um agente responda proposições de uma consulta através da computação de evidências observadas. Ela é obtida a partir de informações sobre as variáveis aleatórias. Na ausência de quaisquer outras informações, é chamada de probabilidade a *priori* ou incondicional. Estes valores de probabilidade devem somar 1 (100%) e para isso é preciso normalizá-las[7], de forma que as probabilidades a *posteriori* sejam divididas por um valor fixo.

2.2.2 Teorema de Bayes

O Teorema de Bayes, apresentado na Equação 2.6 a seguir, é um meio de calcular a probabilidade de uma hipótese, de acordo com as

[7] reescalar no universo onde a condição é satisfeita.

probabilidades de outras hipóteses que venham a ser causa. Em outras palavras ele é utilizado para determinar qual hipótese de um conjunto de hipóteses é a mais provável, a partir de um conjunto de evidências (RUSSELL; NORVIG, 2003).

$$P(B|A) = \frac{P(A|B).P(B)}{P(A)} \tag{2.6}$$

Onde:

- $P(B|A)$: probabilidade de ocorrer o evento B dado a ocorrência do evento A
- $P(A|B)$: probabilidade de ocorrer o evento A dado a ocorrência do evento B
- $P(B)$: probabilidade de ocorrência do evento B
- $P(A)$: probabilidade de ocorrência do evento A

"O formalismo de RB foi inventado para permitir uma representação eficiente e um raciocínio rigoroso ao lidar com um conhecimento incerto." (RUSSELL; NORVIG, 2003, p. 26, traduzido pelo autor). Segundo Coppin (2010 apud COSTA, 2013), uma RB é um grafo orientado acíclico, em que cada um de seus nós representam evidências ou hipóteses e as arestas que conectam dois nodos representam uma dependência entre eles. Caso não haja dependência entre dois eventos, é dito que a probabilidade de um evento ocorrer não interfere na de outro. As RB são comumente utilizadas a fim de representar graficamente um conhecimento de um domínio por meio de relações de dependência entre variáveis e, matematicamente, probabilidades a priori e probabilidades condicionais entre variáveis.

Na construção de uma RB, após definir a sua estrutura, nodos e arestas é preciso especificar a distribuição de probabilidade para a rede. É necessário atribuir a probabilidade a priori para todos os nodos raízes (nodos sem predecessores) e os valores de probabilidade condicional para todos os outros nodos, de forma a considerar todas as possíveis combinações de acordo seus predecessores Charniak (1991).

A Figura 10 é uma RB muito utilizada para fins didáticos, encontrada em vários trabalhos. Neste domínio, pela RB é possível inferir qual a probabilidade de Maria ou João ligarem quando o alarme tocar, seja por um terremoto e/ou ladrão. Porém, também é possível inferir, por exemplo, qual a probabildiade do alarme ter tocado, dado que Maria e/ou João ligaram. Apesar da ordem do arco significar uma razão

de causa, o inverso também pode ser computado por uma definição inversa do Teorema de Bayes.

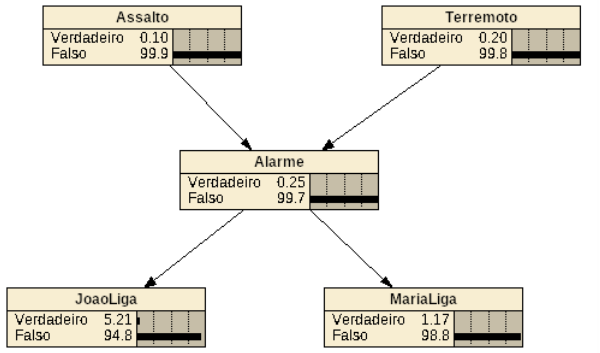

Figura 10 – Exemplo de RB para um domínio específico. Fonte: Marques e Dutra (2002)

Ladrão	Terremoto	*P(Alarme | Ladrão, Terremoto)*	
		Verdadeiro	Falso
Verdadeiro	Verdadeiro	0.95	0.050
Verdadeiro	Falso	0.95	0.050
Falso	Verdadeiro	0.29	0.71
Falso	Falso	0.001	0.999

Tabela 4 – Tabela de probabilidades condicionais. Fonte: Marques e Dutra (2002)

Uma das maiores dificuldades práticas está na atribuição dos valores de probabilidade incondicional (probabilidade a priori), requisitando um conhecimento sobre o domínio. Nestes casos, muitas vezes os valores são estimados com base no conhecimento de especialistas (MITCHELL, 1997). No preenchimento das tabelas de probabildiades condicionais, caso a relação entre os nodos pais e filhos seja arbitrário, pode ser uma tarefa difícil e que requeira um certo grau de experiência (MARQUES; DUTRA, 2002).

Costa (2013) ainda define que uma RB representa de um modo sistemático, através de uma estrutura de dados, relações de independência condicional entre variáveis. Cada um de seus nós têm informações de probabilidade quantitativa e os arcos representam a existência de

uma influência causal direta entre os nodos ligados. Ainda segundo Costa, redes bayesianas permitem, eficientemente, o cálculo de probabilidades a posteriori de qualquer variável aleatória por meio de uma definição recursiva do teorema de Bayes.

2.2.3 Inferência em Redes Bayesianas

Segundo Russell e Norvig (2003) RB's podem ser utilizadas para: tomadas de decisões baseadas em probabilidades; para definir quais evidências devem ser analisadas para obter um conhecimento sobre um domínio; realizar análises para descobrir quais são causas de maior impacto sobre as variáveis e para explicar resultados de uma inferência a um usuário. Desta maneira, as inferências podem ser realizadas de quatro maneiras diferentes:

1. Diagnóstico: a partir de efeitos descobrir causas;
2. Causa: a partir das causas descobrir os efeitos;
3. Intercasual: entre as causas de um efeito comum;
4. Misto: combinar dois ou mais dos tipos descritos.

2.2.4 Aprendizagem em Redes Bayesianas

A aprendizagem bayesiana está dividida entre aprendizagem de parâmetros e de aprendizagem de estruturas. A aprendizagem dos parâmetros se remete a aprender os valores dos parâmetros de cada nodo da rede, ou seja, as distribuições de probabilidade condicional. Já a aprendizagem de estruturas é a aprendizagem do grafo (COSTA, 2013).

Segundo Cheng e Greiner (2001) existem diversas maneiras utilizadas para aprendizagem de estrutura de uma RB. Uma delas, utiliza testes estatísticos como Qui-quadrado e Informação Mútua como uma medida de pontuação para escolher a melhor rede. Estes testes procuram avaliar as relações de independência condicional existentes entre os nodos.

Já de acordo com Russell e Norvig (2003) uma abordagem que pode ser utilizada para a aprendizagem de estruturas de rede é a partir de um modelo ainda simples, sem vínculos, avaliar através de uma métrica escolhida a adição de nós pais e assim ajustando os parâmetros. Através dessas avaliações se decidiria sobre modificações como: inversão, adição ou eliminação dos arcos adicionados.

Em ambos tipos de aprendizagem, a aprendizagem pode ocorrer de um modo indutivo ou por meio de um especialista. No primeiro caso se parte apenas de um conjunto de informações, uma amostra, e a partir deste a rede é construída. Quanto maior o banco de exemplos, mais fidedigna será a rede. Já no último o conhecimento do especialista será integrado durante a construção da rede, ao supervisionar a rede de acordo com o seu conhecimento sobre o domínio.

Segundo Costa (2013) o método de estimação de máxima verossimilhança é uma das técnicas mais populares para se derivar estimadores. Este estimador é o valor que a amostra observada é a mais provável. Em outras palavras, é o valor mais provável de um parâmetro para os dados observados.

Apesar do método ser simples de executar, enfrenta alguns problemas em conjuntos de dados pequenos. Por ser um contador da frequencia relativa de observação de um parâmetro, tende a se acostumar com os dados observados. Por exemplo: após observar um tipo de doce específico, a sua hipótese seria que 100% dos doces na bolsa são deste tipo. É muito mais provável que uma bolsa contenha vários tipos de doces, não um ou outro (RUSSELL; NORVIG, 2003).

O algoritmo *Expectation-Maximization* (EM), baseado na teoria de estimação, também é utilizado para a estimação de parâmetros de modelos probabilísticos. Ele consiste de basicamente dois passos: *expectation* (passo E) que calcula a probabilidade posterior para cada possível atribuição e *maximization* (passo M) que calcula a máxima da distribuição de probabilidade. O algoritmo recebe como entrada uma estrutura inicial de RB e amostras de dados das variáveis desta rede e retorna a rede com seus valores de distribuição de probabilidade condicional atualizados (GUO; LI, 2009).

2.2.5 Aplicações

Como já dito, RB's vem sendo aplicadas em problemas em que se deseja lidar com a incerteza por aleatoriedade de um certo domínio e utilizá-lo para uma tomada de decisão. Charniak (1991) cita um exemplo de aplicação no diagnóstico médico de sintomas. Em *Pathfinder* (HECKERMAN, 1990 apud CHARNIAK, 1991), um programa para ajudar no diagnóstico de doenças no nódulo linfático, uma RB é é utilizada para escolher qual o próximo teste quando os testes já feitos não levantam informações e hipóteses o suficiente para um diagnóstico.

Outro exemplo prático é a sua utilização por um robô apren-

dendo a navegar em um mapa. A medida em que segue, o robô atualiza as probabilidades referentes a novos corredores disponíveis, de acordo com informações captadas por seus sensores. Neste caso o robô poderia optar por desviar do seu caminho já planejado e explorar outras porções do mapa (CHARNIAK, 1991).

RB's também podem ser utilizadas como um classificador, como o *Naive Bayes*. Suas aplicações estão em: classificação textual, identificação de genes, filtros *anti-spam*. Mesmo simples é um modelo que vem obtendo sucesso comparado à outros classificadores mais sofisticados.

Concluindo, em problemas em que será difícil ter conclusões sob um olhar lógico, a abordagem probabilística proporcionada pela utilização de RB oferece a seus utilizadores um jeito conveniente de se obtê-las (CHARNIAK, 1991).

2.3 CONSCIÊNCIA SITUACIONAL

"Consciência situacional é o entendimento do estado do ambiente, assim como os parâmetros relevantes do sistema" (ENDSLEY, 1995a, p. 65, traduzido pelo autor). Em ambientes dinâmicos e complexos operados por humanos, muitas decisões devem ser tomadas em um curto período de tempo e de acordo com uma análise atualizada do ambiente. Nesse sentido, a CS dos operadores é um detalhe crucial que impacta diretamente na performance em tais ambientes (ENDSLEY, 1995b).

Endsley (1995b) afirma que a CS passou a ser reconhecida como uma grande utilidade para pilotos de aeronaves em um período próximo da Primeira Guerra Mundial. Tal que, as operações executadas pelo piloto da aeronave devem estar completamente consistente com os objetivos do piloto e as situações apresentadas a maquina. Sem informações como: condições externas, fatores hostis, informações de navegação e sobre outras aeronaves próximas, pilotos não serão aptos a executarem as suas funções e objetivos efetivamente.

Ainda em seu outro trabalho, Endsley (1995a) diz que a CS pode ser formalmente definida como a percepção de uma pessoa por elementos de um ambiente, de acordo com tamanho e espaço, assim como o significado dessa percepção e a projeção de quanto impactaria no futuro.

A melhoria da CS passou a ser uma preocupação para *designers* de interfaces de operadores, conceitos de automação e programas de treinamento em uma variedade de campos. Assim se faz necessária a

avaliação de novos conceitos e tecnologias e se melhoram ou pioram a atenção de um operador (ENDSLEY, 1995a).

2.4 CONCLUSÃO

Os conceitos até então citados, como: AM, RB e CS estão diretamente relacionados a proposta do presente trabalho. Como forma de modelar a CS de um usuário de *smartphone* trafegando próximo a uma via urbana, será criado um modelo conexionista, uma RB, para raciocinar sobre a incerteza relacionada a este meio. Já a AM, em especificamente a AR, será utilizada na aprendizagem dos parâmetros do modelo.

3 TRABALHOS RELACIONADOS

Neste capítulo serão citados e explicados os principais trabalhos relacionados a proposta deste trabalho. O capítulo se divide em trabalhos que aplicam AM em RB e AR em diferentes domínios de problemas.

3.1 APRENDIZAGEM ESTRUTURAL

Costa (2013), em seu trabalho, tem como principal objetivo a criação de um método para aprendizagem estrutural de redes Bayesianas, utilizando a simulação de Monte Carlo e Cadeias de Markov (MCMC) , de forma não supervisionada. Tem como seus objetivos específicos a investigação das técnicas necessárias para moldar o relacionamento entre nós de uma rede, assim como a construção de um algoritmo que utilize tais conceitos e a uma posterior avaliação de sua adequação para o algoritmo MCMC.

De acordo com Costa (2013) o método de MCMC tem como objetivo visitar um ponto específico com uma probabilidade proporcional a alguma função de distribuição dada. O método MCMC procura simular um experimento, a partir de uma amostra aleatória, com o objetivo de determinar as propriedades probabilísticas de uma população.

Buntine (1991) introduz, em seu trabalho, uma metodologia de refinamento de RB's (ajustes/atualização da rede conforme a disponibilidade de novos dados). Começando por perguntar ao especialista uma ordenação das variáveis, esse deve também explicitar o quanto ele acredita que uma ligação entre dois vértices realmente é fiel, atribuindo níveis de confiabilidade a estas arestas. Na disponibilidade destas informações, é feita a atualização dos valores de cada vértice de acordo com o Teorema de Bayes.

Cano, Masegosa e Moral (2011) propõem uma metodologia interativa para integrar o conhecimento do especialista. Nela, o usuário submete o seu conhecimento sobre o domínio enquanto o processo de aprendizagem estará ocorrendo. A justificativa da criação e utilização de tal metodologia se dá por conta de alguns motivos, como: limitação do número de perguntas ao especialista, perguntando a ele apenas os pontos da estrutura que mais se tem incerteza sobre, ou seja, quando os dados não dão evidências o suficiente sobre a presença ou ausência de uma aresta e também em ajudar o especialista em integrar o seu

conhecimento, mostrando a ele as informações encontradas nos dados.

A integração pode ocorrer de duas formas. Em uma delas é calculado o ganho de informação relativo a cada possível aresta e perguntado a ele sobre a de maior valor (o valor de probabilidade a *posteriori* pode ser mostrado). Caso ele não tenha confiança o suficiente para tomar tal decisão, pode pedir ao sistema para perguntá-lo sobre outra aresta. Sendo confirmado são atualizados os valores de probabilidade. Na segunda, são computados os valores de entropia de cada aresta e perguntado a ele sobre as de maior valor. Se o especialista tiver total certeza quanto a sua resposta, a partir de agora estas arestas serão fixas ou descartadas. A Figura 11 ilustra um dos métodos de integração.

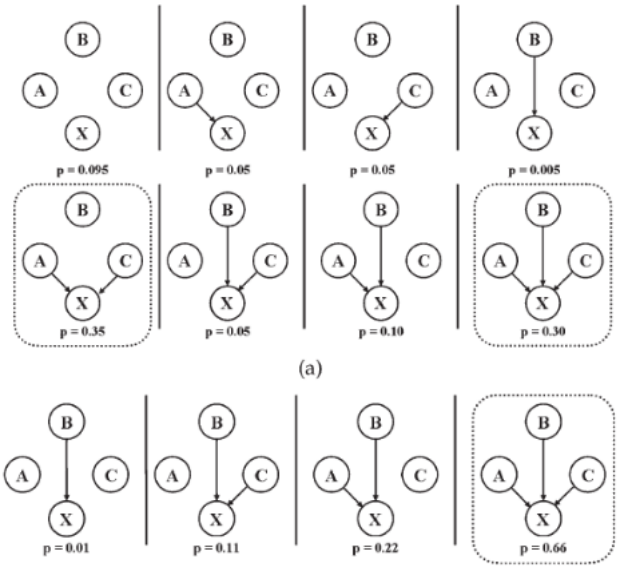

Figura 11 – Um dos métodos de integração proposto por Cano *et. al.*.
Fonte: Cano, Masegosa e Moral (2011)

3.2 APRENDIZAGEM DOS PARÂMETROS

Guo e Li (2009) propõe a utilização do algoritmo EM para a aprendizagem do parâmetros de uma RB e o algoritmo de MCMC é utilizado para a geração de amostras de estruturas de rede. A sua diferença para outros trabalhos está também na utilização de *Accepting*

Rejection Sampling na função de aceitação de MCMC.

A integração de ambos métodos, EM e MCMC trouxe, segundo o autor, resultados satisfatórios. Em iterações limitadas e RB's simples o algoritmo proposto saiu melhor que o utilizado para comparação EM-EM, porém, apesar de superá-lo em casos mais complexos, leva mais tempo para execução. Uma maneira de melhorar o seu modelo seria a utilização de programação paralela em MCMC ou integrar otimizações na aprendizagem estrutural.

3.3 APRENDIZAGEM POR REFORÇO

Taylor e Wolf (2004) propõe um *framework* para aplicar AR no controle de parâmetros em um algoritmo para detectação de textos em imagens retiradas de vídeos. Seu trabalho apresentou uma melhoria de 83% na seleção dos parâmetros comparado aos que haviam sido selecionados previamente. Como modelagem, definiram a recompensa como médias harmônicas entre os valores de *recall* e precisão de classificações do modelo. As ações correspondem aos parâmetros escolhidos em uma etapa do algoritmo e o estado os parâmetros anteriores que seriam selecionados.

Apesar de seu trabalho não estar relacionado a RB's, Adam e Smith (2008) aplicaram AR no controle de uma estrutura de *tensegrity*[1]. O trabalho propõe um algoritmo que, por tentativa e erro, aprende ajustes a serem feitos nos comandos de controle da estrutura. O autor conclui que AR tem um potencial para melhorar o controle ativo de estruturas do *tensegrity* e suas aplicações podem se estender a outros tipos mais adaptivos.

Brockman *et al.* publicou em seu artigo *OpenAI Gym* sobre a biblioteca *OpenAI Gym*[2], a qual possibilita que usuários treinem seus agentes, fazendo *benchkmarks* de algoritmos de AR em diversos ambientes. A ferramenta possui jogos clássicos de *Atari*, de tabuleiro (como por exemplo Go), possibilita controlar robôs em simulações *2D* e *3D*, assim como executar algoritmos e problemas clássicos da literatura (BROCKMAN et al., 2016).

[1] uma estrutura em que a forma é mantida por uma malha em um estado contínuo de tensão.

[2] https://github.com/openai/gym

3.4 CONCLUSÃO

A Tabela 5 compara os trabalhos até então citados como relacionados, de acordo com o tipo de aprendizagem bayesiana proposta e também o tipo de AM.

	(COSTA, 2013)	(BUNTINE, 1991)	(GUO; LI, 2009)
Parâmetros	X	X	X
Estrutural	X	X	X
Supervisionado			X
Não supervisionado	X		
Por reforço			

Tabela 5 – Levantamento de trabalhos relacionados. Elaborado pelo autor (2017)

Já a Tabela 6 compara os trabalhos citados de acordo com a modelagem gerada de acordo com o domínio. Por não ter sido encontrado algum trabalho que aplicasse AR em uma RB, foram utilizados trabalhos que aplicassem AR em diferentes domínios para se ter uma visão melhor sobre as tomadas de decisões para a criação do modelo.

	(TAYLOR; WOLF, 2004)	(ADAM; SMITH, 2008)
Ação	Ajustes nas configurações de controle da estrutura	Ajustes nos parâmetros
Estado	Parâmetros escolhidos	Atual configuração
Recompensa	Média harmônica entre precisão e recall	Resposta da estrutura ao evento
Domínio	Algoritmo para detectação de texto sem imagens retiradas de vídeos	Controle de estrutura de tensegrity

Tabela 6 – Levantamento de trabalhos que aplicam AR em modelos. Elaborado pelo autor (2017)

Dessa forma, este trabalho se diferencia dos relacionados apresentados na Tabela 5 por tentar integrar o conhecimento do especialista por um reforço enviado a aplicação, que por sua vez é uma RB.

4 DESENVOLVIMENTO

Essa capítulo trata do desenvolvimento da proposta de aplicação de AR em uma RB. Essa combinação se dará pela integração do conhecimento do especialista ao reforçar - avaliar entre um sinal positivo e negativo - o caminho o qual o processo de aprendizagem está tomando.

4.1 ESTUDO DE CASO

Essa seção tem como objetivo descrever o caso de estudo utilizado, o experimento, informações relevantes sobre o conjunto de dados levantado e a RB criada.

Como estudo de caso foi utilizado o experimento realizado pelo projeto *Road Awareness*. O objetivo do experimento foi coletar informações que pudessem ajudar a entender quais os principais fatores que afetam a consciência situacional de um usuário de *smartphone* trafegando próximo a vias urbanas. Para isso, foram realizados experimentos em um laboratório que lograva de uma plataforma de imersão em realidade virtual capaz de simular ambientes urbanos.

Essa plataforma de imersão é chamada de *Octave*, que é um ambiente completamente configurável, de uma experiência de imersão holográfica que projeta visões 3D ao redor do usuário, recria som e toques no próprio ambiente. Proporcionando ao usuário ficar imerso em uma realidade virtual, com a habilidade de ver, ouvir, mover e manipular objetos.

Ela está localizada na Universidade de Salford - UK e suas utilizações estão em fins acadêmicos, pesquisa médica, prototipação de veículos, recriação segura de ambientes perigosos, entre outros. Um sistema de acústica com 264 altofalantes dá ao usuário uma experiência quase real, enquanto imerso no ambiente, seja no protótipo de um carro ou na superfície de Marte.

A Figura 12 é uma foto do ambiente sendo simulado. O ambiente contém quatro vias, em quatro direções: a frente, trás, direita e a esquerda do usuário, onde, em apenas uma por vez, surge um carro que irá trafegar até o final do seu sentido oposto. Enquanto isso, existem pequenos animais, chamados de *lemmings*, que aparecem e atravessam a faixa de pedestres que é possível visualizar na figura. O usuário deve ficar atento a quando um carro aparece no cenário e sinalizar de que ficou consciente a ele clicando em um botão no seu *smartphone*,

assim o semáforo da faixa de pedestres fecha e os *lemmings* param de atravessá-la.

Figura 12 – Usuário imerso no ambiente da simulação. Fonte: Primeiro experimento realizado no laboratório na Universidade de Salford - UK.

A análise dos fatores que afetam a atenção do usuário não está apenas nas direções dos carros e nos *lemmings*. Foram feitos três tipos de simulações: com um aplicativo fazendo perguntas a fim de tomar a sua atenção e distraí-lo, de cunho matemático em sua maioria, que o usuário deveria responder; com as perguntas e com fones de ouvido; e simulações em que o objetivo era apenas apertar o botão de consciência em relação ao carro.

Além das diferentes simulações, os carros que eram simulados no ambiente poderiam vir sem ou com som. Os carros sem som vem a ser útil a fim de simular o efeito da diminuição de sons emitidos por carros elétricos nos dias de hoje e o quanto a percepção do usuário é afetada.

Na Figura 13 aparece o laboratório em que o ambiente estava sendo simulado.

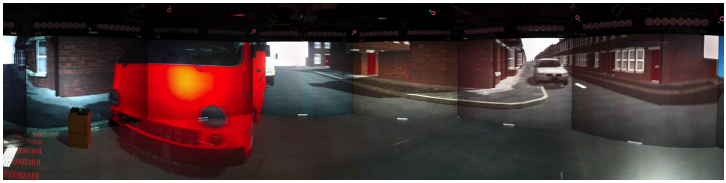

Figura 13 – Foto do ambiente em que o experimento foi simulado.

A indicação de que o usuário havia ficado consciente em relação

ao carro deveria ocorrer antes de um determinado tempo, caso contrário, significaria que o usuário não ficou atento ao carro. A ideia desse tempo demarca uma zona de segurança em que, se ultrapassada, sugere que o usuário ficou inseguro em relação ao carro.

Assim como o projeto *Road Awareness* foi utilizado como estudo de caso para este trabalho, outros trabalhos utilizaram-no em suas pesquisas. Como por exemplo: o desenvolvimento de uma arquitetura para atuação de agentes em ambientes de simulação virtual, que teve como proposta a criação de um modelo de interação entre o agente e o ambiente, objetivando se tornar o mais próximo do visto em um mundo real; integração de AM no desenvolvimento de uma arquitetura baseada em agentes BDI, a fim de realizar o processamento de percepções por meio da união entre os componentes pró-ativo e reativo.

Neste trabalho, foi utilizado como estudo de caso o conhecimento adquirido pelo experimento, assim como o conjunto de dados, possibilitando a criação de um modelo de consciência situacional, uma RB. Posteriormente propiciando a aplicação de AR no modelo gerado como forma de aproximá-lo do mundo real.

4.1.1 O experimento

O experimento coletou dados sobre o comportamento de 20 usuários nos 3 tipos de simulações. O experimento teve como resultado um conjunto de dados com informações como: identificação do objeto (*lemming*, carro), assim como o tempo em que foi adicionado na simulação, sua posição, direção, tipo (som ligado ou desligado para os carros), qual era o tipo de simulação que estava ocorrendo, perguntas feitas para o usuário e suas respectivas respostas, assim como informações relativas a sua percepção e atenção relacionada aos carros. A Tabela 7 mostra um exemplo de parte do conjunto e quais informações estão contidas.

id	Consciente	Simulacao	SomCarro	DirecaoCarro	Tempo
6	Sim	Fones	Não	Esquerda	12.055s
6	Não	Fones	Sim	Esquerda	16.260s
6	Sim	Perguntas	Não	Atrás	1.383s
6	Não	Perguntas	Não	Esquerda	11.582s
...	...	...	...	...	...

Tabela 7 – Exemplo de parte do conjunto de dados utilizado. Fonte: Do autor (2017).

Esse conjunto de dados, apresentado na Tabela 7 é uma abstração das informações que foram coletadas durante o experimento para que mais se adequem ao modelo desejado e foi gerada em um esforço colaborativo do grupo envolvido com a pesquisa. O novo conjunto contém 962 registros, dos quais 40 foram removidos por não estarem completos.

Para cada um dos carros adicionados ao evento foram consideradas informações como: identificação do usuário, o tipo de simulação que estava ocorrendo, se o som do carro estava ligado ou desligado, a direção do carro, quanto tempo o usuário levou para sinalizar de que percebeu o carro no ambiente e se, a partir desse tempo, o usuário estaria consciente ou não em relação ao carro foram obtidas e utilizadas para preencher o conjunto de dados utilizado.

Para entender melhor como os dados se comportam, foi utilizado a biblioteca Pandas[1], principalmente por sua performance e produtividade. Como é possível visualizar na Tabela 7 há um atributo que está em valores contínuos e para facilitar a análise e aplicação do trabalho proposto foi decidido discretizá-lo.

Apesar de não estar refletido nos dados, há de se considerar que existem três casos de um usuário ter ficado consciente em relação a aproximação de um carro, que são: o fato dele realmente ter ficado consciente e apertado o botão do aplicativo em tempo suficiente, o fato dele não ter apertado o botão em momento algum desde a criação e destruição do carro no ambiente virtual e o fato dele ter apertado o botão porém o carro já ter passado de um limite relativo a zona de segurança.

Além disso, embora todos valores desse tenham o mesmo significado: o tempo desde que o carro apareceu no ambiente até o usuário percebê-lo ou não e sinalizar no seu *smartphone*, foi preciso lidar com alguns fatores técnicos que impossibilitaram definir, para as diferentes direções e opções de som do carro, faixas iguais para a discretização do atributo Tempo. Assim como problemas relacionados ao ambiente e a sua sincronização com o aplicativo.

Levando esses fatores em conta e a nacessidade de discretizar o atributo Tempo, agora renomeado para Percepção, foi gerado o histograma apresentado na Figura 14 dividindo o atributo em 8 intevalos de tamanhos iguais. Um dos problemas relatados que é possível notar na Figura 14 é o fato de ter números que não condizem com a quantidade de tempo que leva para um carro ser criado ou destruído no ambiente. Para solucionar esse problema foi decidido eliminar os registros que tivessem um tempo maior ou igual a 26 segundos (um total de 4

[1] http://pandas.pydata.org/

tuplas).

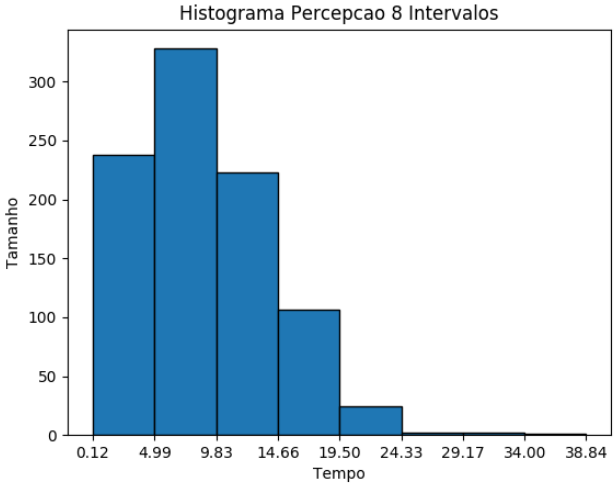

Figura 14 – Histograma atributo Percepção dividido em 8 faixas. Fonte: Do autor (2017).

Outro problema identificado foi no tempo de criação dos carros pelo Unity[2]. Um carro que continha som levava um tempo a mais para ser processado pelo motor de jogo até aparecer no ambiente e isso gerou um atraso em alguns casos para sua criação e registro. Portanto não se pôde considerar que valores maiores que apenas um número em específico corresponderiam a uma classe.

A Figura 15 apresenta um gráfico de caixas que representa o tempo em média em que os usuários levaram para apertar o botão do aplicativo para as diferentes direções e em relação ao som dos carros, assim como as suas medidas discrepantes. É possível constatar que as direções da Direita, Esquerda e Frente, especialmente as duas últimas, tiveram vários valores discrepantes. Assim como que o o som do carro afetou diretamente o tempo para resposta do usuário para todas direções, com médias, em um geral, bem diferentes do caso com som. Exceto da direção da Frente. É possível também observar a diferente dispersão dos dados representada pela amplitude de cada uma

[2]um motor de jogo que contém um ferramental necessário que possibilita e facilita o desenvolvimento de jogos. O software possibilita a adição de trilhas sonoras, adição de física a objetos, funções gráficas e outras ações.

das caixas.

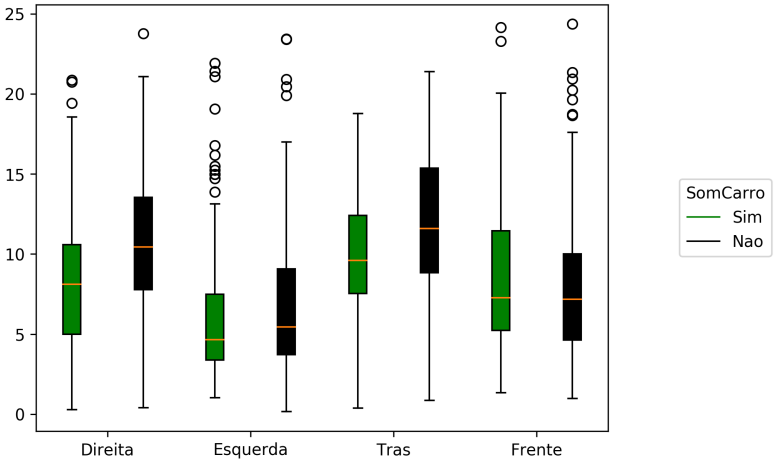

Figura 15 – Gráficos de caixa da distribuição do atributo Percepcao para os tipos de som de acordo com a direção do carro. Fonte: Do autor (2017).

A Tabela 8 apresenta os exatos valores que foram considerados para a discretização do atributo. Também é possível constatar que carros vindos da esquerda e da direita tem um tempo médio maior para percepção. Isto pode ser explicado pelo nível de oclusão visual do usuário para os carros da direita e ao fato de que, para aumentar o seu campo de visão em relação a via da esquerda, o usuário precisaria se locomover um pouco à frente.

Som \ Direcao	Trás	Frente	Esquerda	Direita
Sim	6.279	7.069	10.635	8.297
Não	8.518	8.160	11.629	9.646

Tabela 8 – Média de tempo de Consciente para direções e sons dos carros. Fonte: Do autor (2017).

Para a média geral, de todos tipos de carro e direções, foi encontrado o seguinte intervalo de confiança para 95%: [8.508, 9.033], sendo 8.7705 ± 0.2625. Assim, podendo afirmar que se forem construídos intervalos de confiança, nestas condições, para cada um dos dados, 95%

conterão estes valores.

A Figura 16 corrobora a ideia da influência do posicionamento do usuário em relação a essas vias, podendo constatar que houveram mais casos de incosciência em relação ao carro para os carros vindos dessas direções.

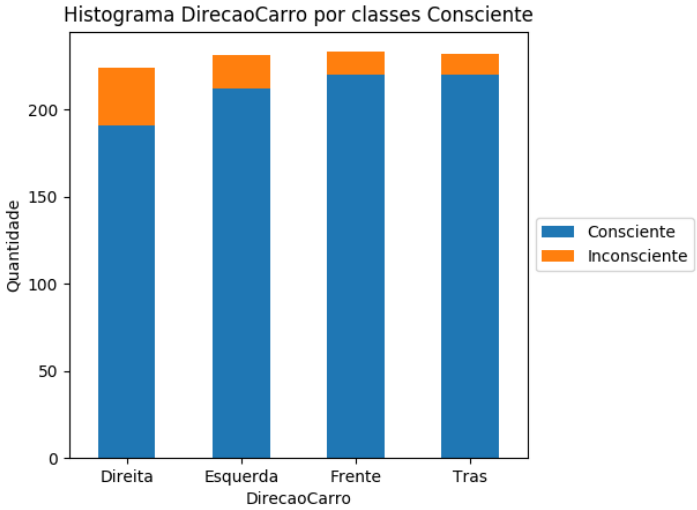

Figura 16 – Histograma relação classes Consciente por DirecaoCarro. Fonte: Do autor (2017).

Apesar das informações relativas ao tempo médio de carro, considerarem que todos eventos acima destes valores seriam uma classe única resultaria em um erro, uma vez que existem os casos de falsa consciência (casos em que o usuário ficou consciente ao carro porém com atraso). Logo, foi considerado que os registros em que o usuário sinalizou consciência, com tempo de percepção maior que a média da sua direção e som do carro, pertencem a classe de percepção Tardia. Eventos em que o usuário não sinalizou foram categorizados como Ausente e o restante como Presente.

Na Figura 17 é possível visualizar como ficou a distribuição dos dados de acordo com as classes criadas. Existe uma quantidade muito maior de itens pertencente às classes Presente e Tardia em relação a Ausente, isso pode ser explicado pelo fato de que durante o experimento houveram poucos casos de Inconsciente, como também é possível cons-

tatar na própria figura.

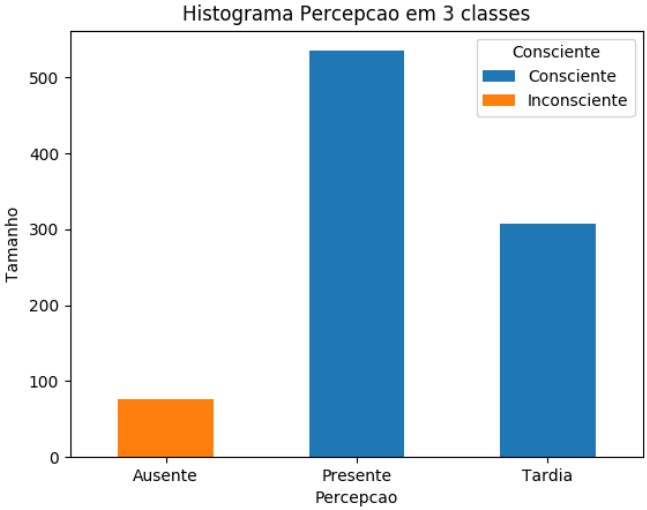

Figura 17 – Histograma atributo Percepção após discretização. Fonte: Do autor (2017).

No que tange aos outros atributos, houve um processo de discretização apenas no atributo Simulacao, renomeado para DistracaoApp. As simulações com fones de ouvido e que havia perguntas foram agrupadas na mesma classe, a de que há a distração do aplicativo.

4.2 A REDE BAYESIANA

Uma vez que RB's são mecanismos eficientes para representar e raciocionar sobre um modelo de incerteza, seu formalismo é facilmente aplicável para este domínio.

A definição do modelo partiu do objetivo de se possibilitar analisar o quanto cada um dos fatores influeciam diretamente na consciência do usuário. Para isso, ela foi elaborada de acordo com as situações que lhe são apresentadas durante a simulação.

Com o conjunto de dados gerado e após o processo de discretização, além dos estados de consciência do usuário, obtem-se as situações que lhe foram evidenciadas, como: direção do carro, tempo de percep-

ção, som do carro e o tipo de simulação. Tais situações influenciam diretamente na atenção do usuário em relação ao seu meio.

Os valores dos parâmetros dos nodos da RB foram obtidos a partir da utilização de um estimador bayesiano, utilizando a biblioteca pgmpy[3], implementada em Python. O conjunto de dados foi separado em 2 conjuntos menores, dos quais: o conjunto maior, com 70% dos dados, foi utilizado para a aprendizagem dos parâmetros; e o menor, com 30%, será utilizado para a aplicação de AR para ajustar os valores da RB.

Portanto, a partir de um esforço colaborativo pelo grupo de pesquisa, que inclui o autor, relacionado ao projeto, foi construído manualmente um modelo de RB, retratado na Figura 18. No modelo construído é possível analisar, por exemplo, o quanto as diferentes direções dos carros influenciam na consciência do usuário, assim como os tipos de simulação e os fatores restantes.

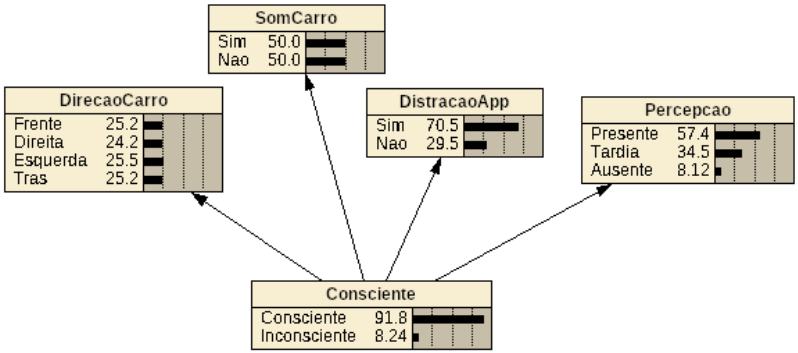

Figura 18 – RB original no software Netica. Fonte: Do autor (2017).

De acordo com as evidências apresentadas à RB, é possível inferir qual seria o grau de consciência situacional do usuário, como é possível analisar em um exemplo de inferência na Figura 19. O caso simulado, com um carro vindo da direita, sem som e em uma simulação do tipo em que há distração no aplicativo, é o caso que mais interferiu na atenção do usuário em relação ao carro.

[3]https://github.com/pgmpy/pgmpy

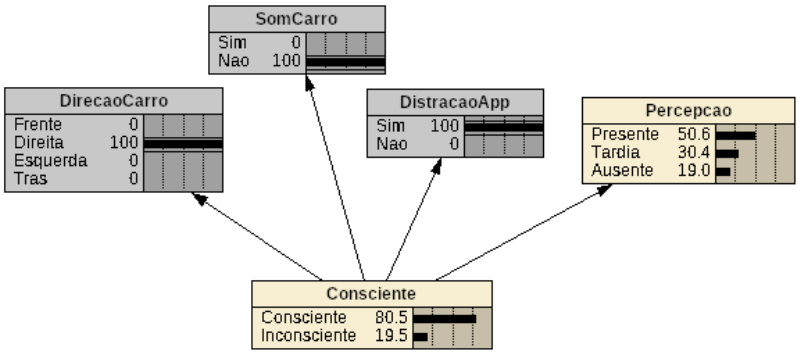

Figura 19 – Inferência RB com evidência de DirecaoCarro, SomCarro e DistracaoApp. Fonte: Do autor (2017).

4.2.1 O modelo

A aplicação de AR acontecerá no intuito de reforçar a rede e aproximá-la do que se observou e estudou referente a consciência situacional do usuário ao ambiente, tentando aproximar os parâmetros da RB de situações mais reais que, por conta da quantidade de dados, podem não estar retratadas no modelo. Em uma visão geral o processo de AR na RB será conforme a Figura 20

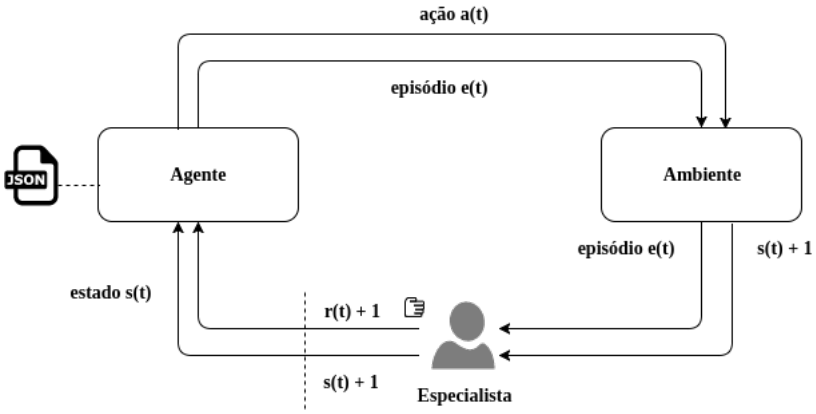

Figura 20 – Modelo proposto. Fonte: Do autor (2017).

Serão feitas simulações com base no conjunto de testes e de acordo com o conhecimento do especialista sobre o domínio, a RB será reforçada. A análise será de acordo com os resultados da rede para os respectivos cenários.

No processo estabelecido, o agente propõe ações de ajustes nos valores de probabilidade condicional da RB e envia esta ação ao ambiente juntamente com o episódio da iteração. O ambiente, que contém a RB, recebe o episodio[4], e utiliza a estrutura atual da RB para inferir a probabilidade a posteriori de diagnóstico do nodo Consciente conforme os valores contidos no episódio. Em seguida, utiliza a ação de ajuste proposta pelo agente para alterar os valores de probabilidade condicional do nodo Consciente, propagando os ajustes para os seus nodos filhos e assim construindo uma RB com novos valores de probabilidade condicional e a priori.

A figura do Especialista, apresentada na Figura 20, representa o papel do especialista sobre o domínio da RB. Como escopo deste trabalho, a recompensa relacionada a ação e como o ambiente respondeu a ela, é manualmente inserida pelo especialista. O especialista, conforme o seu conhecimento, analisa a ação e determina o novo valor de probabilidade a posteriori para a RB ajustada e retorna a aplicação um valor numérico entre -1 e 1 simbolizando uma punição ou recompensa.

O agente recebe a recompensa, o novo estado do nodo Consciente da RB, assim como os valores dos atributos do episódio que foram utilizados como evidência para a consulta e armazena essas informações conforme o seu algoritmo de aprendizagem.

4.3 A APRENDIZAGEM

Neste trabalho foi utilizado o algoritmo *Q-learning* para implementar a aprendizagem do agente para os estados passíveis de serem atingidos pelo ambiente e as respectivas recompensas, qual o ajuste mais provável para determinada situação.

A implementação do agente foi adaptada do repositório *reinforcement learning*[5], também escrito em Python, que implementa exemplos dos agentes mais conhecidos na literatura de AR.

Primeiramente é construída a RB e, com o conjunto de dados de treino, aplicado o método de estimação bayesiana. Posteriormente,

[4] o episódio é uma tupla do conjunto de dados que fora separado para testes, e aqui, neste trabalho, estarão sendo utilizados para ajustar os parâmetros da RB.

[5] https://github.com/rlcode/reinforcement-learning/

para cada uma das 10 iterações de um laço, a RB será reiniciada, ou seja, voltará aos valores de probabilidade iniciais. O fato de voltar a RB ao seu estado original, proporciona que os as ações sugeridas pelo agente passem novamente pelos mesmos processos e, de acordo com o aprendido, sugira ações que maximizem o valor de recompensa em longo termo. Em cada uma dessas iterações, os episódios contidos no conjunto de dados de teste serão aplicados ao modelo proposto. Para cada um desses episódios, o agente fará ajustes de acordo com o que já sabe sobre o ambiente e a recompensa guiará o processo de aprendizagem, numa forma de tentativa e erro.

Para este trabalho foram considerados como estados do ambiente, os valores contidos no epísodio e os parâmetros do nodo Consciente. A sugestão de ajustes pelo algoritmo se dará da seguinte forma: caso não sejam utilizadas informações salvas de outras iterações, o agente irá propor de maneira aleatória a escolha de um ajuste para o estado inicial. A decisão sobre escolher entre explorar novas ações ou escolher entre as ações que já se sabe que são boas, se dá por um processo de verificar um número aleatório gerado e compará-lo com uma taxa, caso seja maior, serão exploradas novas ações, caso contrário se escolherá entre os ajustes que já se conhece terem resultados satisfatórios.

Algoritmo 2 Algoritmo de aprendizagem para o agente Q-learning

Entrada: O atual estado, a ação escolhida, o valor de recompensa e o próximo estado

Método:

```
q_1 = tabela_q[estado][acao]
aux = max(tabela_q[proximo_estado])
q_2 = recompensa + fatorDesconto * aux
tabela_q += taxaAprendizagem * (q_2 - q_1)
```

Figura 21 – Código da função de aprendizagem. Fonte: Do autor (2017).

Em se tratando de sua função de aprendizagem, o algoritmo, apresentado na Figura 21, recebe como entrada o estado apresentado, a sua ação, o valor dado de recompensa e o próximo estado obtido. Nele, seleciona a atual pontuação utilizando como chave para busca o estado anterior e a ação escolhida. Em seguida, soma a recompensa

atribuída com um fator de desconto multiplicado pelo valor máximo de pontuação apresentado para as ações escolhidas para o novo estado. É feita uma subtração da última com a primeira variável, multiplicado por um fator de aprendizagem e somado a pontuação que se tem para a respectiva ação no determinado estado.

Diante da dificuldade de se obter o valor exato que é propagado para os outros nodos enquanto se altera a distribuição de probabilidade do nodo Consciente, foi encontrado uma solução aproximada. Os valores de ajustes estabelecidos para cada um dos nodos foram estimados de acordo como o modelo se comportou em ajustes manuais no software Netica. Portanto, os ajustes realizados no nodo Consciente serão propagados para os seus nodos filhos e os seus valores de probabilidade serão somados aos valores obtidos de acordo com o ajuste proposto pelo agente em razão do ajuste de cada nodo, em seguida estes valores serão normalizados. As tabelas a seguir apresentam os ajustes para cada porcentagem que for diminuída no estado de Consciente do nodo Consciente.

DirecaoCarro	Tras	Direita	Esquerda	Frente
Valor	-0.0011	0.0016	0.0022	-0.0088

Tabela 9 – Ajustes no nodo DirecaoCarro. Fonte: Do autor (2017).

DistracaoApp	Não	Sim
Valor	0.0017	-0.0017

Tabela 10 – Ajustes no nodo DistracaoApp. Fonte: Do autor (2017).

SomCarro	Não	Sim
Valor	0.0015	-0.0015

Tabela 11 – Ajustes no nodo SomCarro. Fonte: Do autor (2017).

Percepcao	Presente	Tardia	Ausente
Valor	-0.0062	-0.0037	0.0099

Tabela 12 – Ajustes no nodo Percepcao. Fonte: Do autor (2017).

Os valores dos ajustes podem ser: 0, 1%, 2%, 3%, 4%, 5% tanto positivos quanto negativos. Dessa forma, serão ajustados os valores de

probabilidade do nodo conforme os ajustes, não podendo haver ajustes inválidos que ultrapassem a margem de 100% ou 0%. A escolha dessa variedade de ações se deu pela natureza do domínio, uma vez que valores maiores de ajustes poderiam levar a números que não fossem tão fidedignos.

Ao iniciar a aplicação de AR na RB pode se escolher pela opção de utilizar um arquivo JSON[6] que contém os valores das pontuações de cada ação para os estados do ambiente. Essas informações são salvas sempre que a aplicação é encerrada. Dessa forma é possível continuar o processo de aprendizagem a partir do ponto em que foi encerrada.

Na dificuldade de se modelar os estados, ações e recompensas durante o processo de aprendizagem dos parâmetros de RB pelos algoritmos convencionais, se decidiu por aplicar a AR como uma segunda etapa.

Um dos problemas encontrados durante o desenvolvimento foi de como seria obtido o valor de recompensa. Visto que o ambiente é quem deve informá-la, a recompensa é dada conforme o estado do ambiente, a ação e como o estado respondeu a ação (o próximo estado do ambiente). Como o ambiente se trata de uma RB, uma avaliação simplesmente sistemática poderia não obter resultados satisfatórios.

Diante disso decidiu-se utilizar o valor de recompensa como uma maneira de integrar o conhecimento do especialista. A principal desvantagem desta decisão se deu no tempo de execução da aplicação, já que para todo episódio se torna necessário que o especialista avalie as alterações que a RB sofreu.

Uma das limitações do trabalho também está no nodo em que o agente propõe as ações de ajustes. Como é apenas no nodo Consciente, limitou-se a apenas analisar como a RB reflete a essas mudanças.

4.4 TESTES

O critério utilizado para os valores de recompensas atribuído para cada um dos ajustes propostos pelo agente, foi de acordo com o ajuste proposto para a situação apresentada. Ou seja, no caso de um episódio em que não se há som no carro, sua direção é da esquerda ou da direita e há a distração do aplicativo, ajustes que aumentem o nível de consciência são passíveis de receberem punição, uma vez que estes casos foram os que mais apresentaram atrapalhar o usuário em questão do seu nível de atenção.

[6] uma maneira compacta de armazenar e transmitir informações.

Enquanto em casos com percepção presente, carros vindo da frente, com som ligado e não há distração do aplicativo, foram dados valores de recompensas maiores a ajustes pequenos negativos ou que mantivessem o atual estado de consciência, já que para estes casos não há um nível de oclusão significante a ser levado em questão, assim como pelo som do carro despertar a atenção em relação a sua aproximação.

Embora a maioria dos casos houve percepção do usuário em relação ao carro, significando que ele ficou consciente a ele, houveram casos em que a sinalização de consciência pôde ser considerada como precoce. Em tais situações, na ausência ou demora de geração de um carro no cenário, o usuário por muitas vezes precipitou-se em apertar o botão para sinalizar. Estes casos seriam retratados pela RB como uma probabilidade a posteriori próxima de 100%, porém erroneamente. Diante disso, pequenos ajustes em alguns episódios já aproximariam o modelo de valores mais exatos.

As análises feitas durante as iterações também partiram do princípio de não tentar fugir muito dos valores de parâmetros apresentados pela RB após a estimação com o método utilizado, principalmente por tentar manter a RB o mais próximo do que foi apresentado no experimento, apenas tentando ajustá-la às situações mais críticas observadas durante a pesquisa e desenvolvimento do modelo.

4.5 EXPERIMENTOS

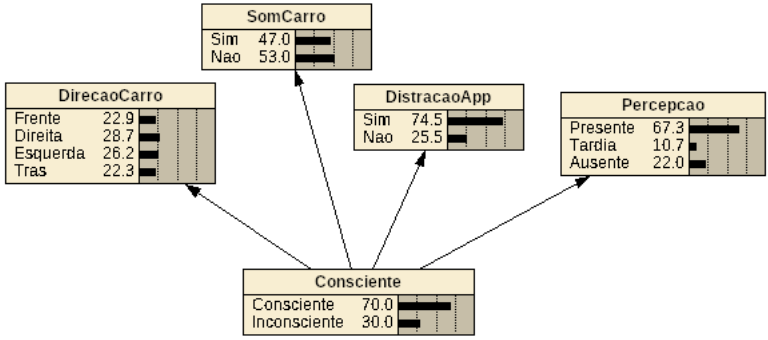

Figura 22 – RB ajustada com reforços aleatórios. Fonte: Do autor (2017).

Com o objetivo de testar o quanto o reforço influencia no rumo

em que os ajustes são dados as redes, foram criados 2 casos de testes. No primeiro, o qual gerou a RB retratada na Figura 22, foram retornados valores de recompensa aleatórios entre -1, 0 e 1. Para ele foram executadas todas as iterações do algoritmo.

Enquanto no segundo, apresentado na Figura 23, foram retornados valores de recompensas mais fidedignos ao domínio. Por conta do fator tempo, para esse segundo, foram executadas apenas três iterações do algoritmo, o que seria no total, aproximadamente, 970 ajustes.

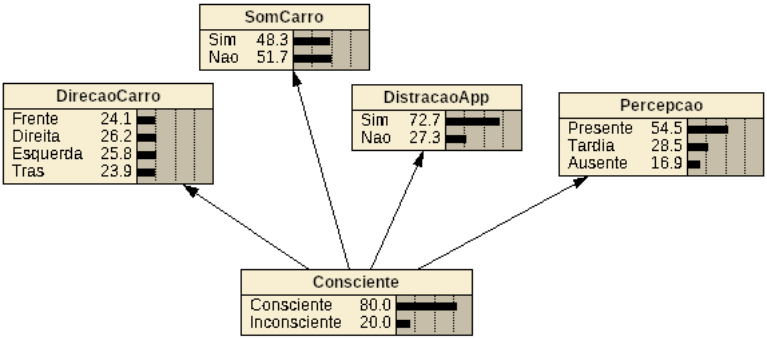

Figura 23 – RB ajustada. Fonte: Do autor (2017).

Como é um processo supervisionado, o caminho o qual a RB levou durante o segundo teste e o resultado final, estarão, na maioria dos casos, cumprindo o desejado em questão de modificações na RB. Portanto, comparar o quanto os modelos se diferenciam um do outro e atribuir que um está certo ou errado, sem o levantamento desses apontamentos, não seria justo.

Como resultado final, para se ter uma ideia do quanto pequenos ajustes alteram o comportamento da RB em relação a novas inferências, a Figura 24 apresenta a mesma consulta realizada no novo modelo. É possível notar que pequenos ajustes em cada um dos nodos alteram a probabilidade a posteriori para este caso em aproximadamente 21%.

As tabelas de probabilidade condicional e a priori de ambos modelos, o primeiro estimado com o método e o segundo com a adição da aplicação de reforço, constam no apêndice que segue ao trabalho.

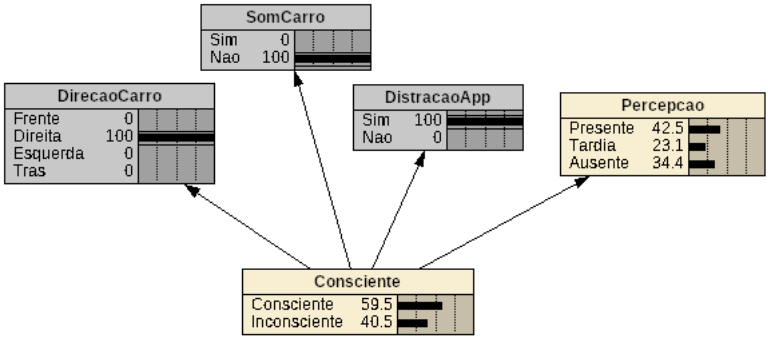

Figura 24 – Inferência nova RB com evidência de DirecaoCarro, SomCarro e DistracaoApp. Fonte: Do autor (2017).

4.6 CONSIDERAÇÕES FINAIS

4.6.1 Plugabilidade

Como a aplicação salva a política de controle encontrada para cada um dos estados do modelo, na disponibilidade de novos dados para a mesma estrutura de RB, o algoritmo desenvolvido pode ser novamente testado para reforça-la. Sendo asim, considerando uma política de controle com base no conhecimento do especialista, os ajustes propostos serão fieis aos utilizados em outras iterações.

4.6.2 Técnicas de amostragem

Como dito, uma vez que a quantidade dados obtida pelo experimento é relativamente baixa apenas 77 dos episódios para o conjunto de dados já tratado, o modelo desenvolvido possibilita que sejam utilizadas técnicas de amostragem para continuar integrando o conhecimento do especialista na RB ou para testes. Tais algoritmos de amostragem, como por exemplo de Gibbs, geram um grande número de registros coerentes com a distribuição original, preservando a natureza do conjunto de dados. Dessa forma, poderia-se continuar testando a RB em questão e aplicando o reforço afim de se obterem resultados mais satisfatórios.

5 CONCLUSÃO E TRABALHOS FUTUROS

Após a realização do trabalho e a avaliação dos resultados obtidos, é possível concluir que os objetivos específicos levantados foram alcançados, ainda que de certa forma limitados, visto a não identificação de trabalhos relacionados que propusessem a aplicação de aprendizagem por reforço em modelos bayesianos, não cumprindo o Objetivo 1. Durante a execução do projeto foram percebidos oportunidades de possíveis trabalhos futuros.

A implementação do modelo bayesiano de atenção situacional, assim como a preparação e discretização dos dados levantados pelo experimento, permitiram a modelagem de uma rede bayesiana capaz de mensurar o nível de consciência e percepção de um usuário de smartphone trafegando próximo a vias urbanas. Tal qual, a análise e diagnóstico dos fatores mais influentes nos níveis de atenção. Dessa forma, sendo possível a conclusão do Objetivo 2 e 3.

As dificuldades enfrentadas durante o desenvolvimento e a integração da AR na aprendizagem bayesiana do modelo, se deram principalmente diante da dificuldade de se modelar as recompensas, ações e estados de um ambiente de RB utilizando os algoritmos mais conhecidos de aprendizagem bayesiana de parâmetros. A modelagem realizada levou ao cumprimento do Objetivo 4. No entanto, a definição de utilizá-lo como uma segunda etapa proporcionando o refinamento dos parâmetros, trouxe resultados satisfatórios.

Esse trabalho também mostra que a aprendizagem por reforço aplicada a sistemas especialistas cumpre seu papel como maneira de ajustá-los de forma a serem mais congruentes com o estudado sobre o domínio, mesmo na ausência de uma quantidade de dados significativa. Embora a aplicação proposta tenha suas limitações referentes ao local de ajuste e inserção manual de valor de recompensa, os resultados foram considerados satisfatórios para o domínio aplicado, o que leva ao cumprimento do Objetivo 5 e 6.

No que tange as diferenças entre um reforço manual na rede e a aplicação de um processo de aprendizagem por reforço sobre os ajustes a serem feitos de acordo com determinadas situações, a principal desvantagem ainda se dá na necessidade da inserção de manual do valor de recompensa, porém se tem como vantagem da abordagem a recomendação de ajustes mais prováveis para cada determinada situação de acordo com o conhecimento previamente adquirido pela aplicação.

A partir do trabalho de pesquisa e desenvolvimento aqui reali-

zado pôde-se observar a oportunidade de trabalhos a serem desenvolvidos futuramente, como por exemplo:

- Estudo, implementação e comparação entre agentes convencionais da literatura de aprendizagem por reforço no escopo de um modelo bayesiano.
- Um estudo e implementação da análise da rede bayesiana de forma a automatizar o valor de recompensa;
- Um comparativo entre a rede bayesiana original e a desenvolvida na presença de um novo conjunto de dados proveniente de novas simulações;
- A adaptação do modelo desenvolvido de aplicação de aprendizagem por reforço na alteração da rede bayesiana utilizada para este domínio;
- Aplicação da rede bayesiana desenvolvida em aplicativos de smartphones para controle de informações sensíveis de acordo com informações do ambiente;
- Portar a aplicação para o ambiente de simulação (Google VR e Play Store) a fim de avaliar o comportamento de avatares ou de simulações, podendo utilizá-la tanto para coletar os dados quanto para avaliar as simulações e propor os ajustes na RB.

REFERÊNCIAS

ADAM, B.; SMITH, I. F. Reinforcement learning for structural control. *Journal of Computing in Civil Engineering*, American Society of Civil Engineers, v. 22, n. 2, p. 133–139, 2008.

BARBER, D. *Bayesian reasoning and machine learning*. [S.l.]: Cambridge University Press, 2012.

BROCKMAN, G. et al. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

BUNTINE, W. Theory refinement on bayesian networks. In: MORGAN KAUFMANN PUBLISHERS INC. *Proceedings of the Seventh conference on Uncertainty in Artificial Intelligence*. [S.l.], 1991. p. 52–60.

CANO, A.; MASEGOSA, A. R.; MORAL, S. A method for integrating expert knowledge when learning bayesian networks from data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, IEEE, v. 41, n. 5, p. 1382–1394, 2011.

CARUANA, R.; NICULESCU-MIZIL, A. An empirical comparison of supervised learning algorithms. In: ACM. *Proceedings of the 23rd international conference on Machine learning*. [S.l.], 2006. p. 161–168.

CHARNIAK, E. Bayesian networks without tears. *AI magazine*, v. 12, n. 4, p. 50, 1991.

CHENG, J.; GREINER, R. Learning bayesian belief network classifiers: Algorithms and system. *Advances in artificial intelligence*, Springer, p. 141–151, 2001.

COSTA, F. S. *Aprendizagem estrutural de redes bayesianas pelo método de Monte Carlo e cadeias de Markov*. Tese (Doutorado) — Universidade Federal de Santa Catarina, 2013.

CUNNINGHAM, P.; CORD, M.; DELANY, S. J. Supervised learning. *Machine learning techniques for multimedia*, Springer, p. 21–49, 2008.

ENDSLEY, M. R. Measurement of situation awareness in dynamic systems. *Human factors*, SAGE Publications Sage CA: Los Angeles, CA, v. 37, n. 1, p. 65–84, 1995.

ENDSLEY, M. R. Toward a theory of situation awareness in dynamic systems. *Human factors*, SAGE Publications Sage CA: Los Angeles, CA, v. 37, n. 1, p. 32–64, 1995.

GUO, P.; LI, N. An em-mcmc algorithm for bayesian structure learning. In: IEEE. *Computer Science and Information Technology, 2009. ICCSIT 2009. 2nd IEEE International Conference on*. [S.l.], 2009. p. 158–162.

HAN, J.; PEI, J.; KAMBER, M. *Data mining: concepts and techniques*. [S.l.]: Elsevier, 2011.

KAELBLING, L. P.; LITTMAN, M. L.; MOORE, A. W. Reinforcement learning: A survey. *Journal of artificial intelligence research*, v. 4, p. 237–285, 1996.

KOTSIANTIS, S. B.; ZAHARAKIS, I.; PINTELAS, P. *Supervised machine learning: A review of classification techniques*. 2007.

LUGER, G. F. *Inteligência Artificial*. 6. ed. [S.l.]: Pearson Education, 2013.

MARQUES, R. L.; DUTRA, I. Redes bayesianas: o que são, para que servem, algoritmos e exemplos de aplicações. *Coppe Sistemas–Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil*, 2002.

MITCHELL, T. M. *Machine learning*. [S.l.]: McGraw-Hill Science/Engineering/Math, 1997.

PEARL, J. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. [S.l.]: Morgan Kaufmann, 2014.

ROVIRA, A.; SLATER, M. Reinforcement learning as a tool to make people move to a specific location in immersive virtual reality. *Int. J. Hum.-Comput. Stud.*, Academic Press, Inc., p. 89–94, 2017.

RUSSELL, S. J.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. 2. ed. [S.l.]: Pearson Education, 2003.

SANTOS, E.; FAZENDA, B. *Computational Model of Situational Awareness for users of smartphones in the vicinity of traffic*. [S.l.], 2016.

SUTTON, R. S.; BARTO, A. G. *Reinforcement learning: An introduction*. [S.l.]: MIT press Cambridge, 1998.

TAYLOR, G. W.; WOLF, C. Reinforcement learning for parameter control of text detection in images from video sequences. In: IEEE. *Information and Communication Technologies: From Theory to Applications, 2004. Proceedings. 2004 International Conference on.* [S.l.], 2004. p. 517–518.

ZHU, X. Semi-supervised learning literature survey. Citeseer, 2005.

APÊNDICE A – Tabelas de probabilidade condicional e a priori

A.1 REDE BAYESIANA ORIGINAL

P (Consciente)	*P (Inconsciente)*
0.91757	0.08243

Tabela 13 – Probabilidade a priori nodo Consciente. Fonte: Do autor (2017).

Consciente	*P (DirecaoCarro / Consciente)*			
	Frente	Direita	Esquerda	Trás
Consciente	0.25798	0.22943	0.25294	0.25966
Inconsciente	0.17991	0.38551	0.27336	0.16121

Tabela 14 – Tabela de probabilidade condicional nodo DirecaoCarro. Fonte: Do autor (2017).

Consciente	*P (SomCarro / Consciente)*	
	Sim	Não
Consciente	0.51259	0.48741
Inconsciente	0.35981	0.64019

Tabela 15 – Tabela de probabilidade condicional do nodo SomCarro. Fonte: Do autor (2017)

Consciente	*P (DistracaoApp / Consciente)*	
	Sim	Não
Consciente	0.69228	0.30772
Inconsciente	0.84579	0.15421

Tabela 16 – Tabela de probabilidade condicional do nodo DistracaoApp. Fonte: Do autor (2017)

Consciente	*P (Percepcao / Consciente)*		
	Presente	Tardia	Ausente
Consciente	0.62441	0.3742	0.14
Inconsciente	0.01588	0.01588	0.96885

Tabela 17 – Tabela de probabilidade condicional do nodo Percepcao. Fonte: Do autor (2017)

A.2 REDE BAYESIANA APÓS APLICAÇÃO DE REFORÇO

P (Consciente)	*P (Inconsciente)*
0.80	0.20

Tabela 18 – Probabilidade a priori nodo Consciente. Fonte: Do autor (2017).

Consciente	*P (DirecaoCarro / Consciente)*			
	Frente	Direita	Esquerda	Trás
Consciente	0.25866	0.22767	0.25319	0.26048
Inconsciente	0.17175	0.40128	0.27608	0.15089

Tabela 19 – Tabela de probabilidade condicional nodo DirecaoCarro. Fonte: Do autor (2017).

Consciente	*P (SomCarro / Consciente)*	
	Sim	Não
Consciente	0.51308	0.48692
Inconsciente	0.3649	0.6351

Tabela 20 – Tabela de probabilidade condicional do nodo SomCarro. Fonte: Do autor (2017)

Consciente	*P (DistracaoApp / Consciente)*	
	Sim	Não
Consciente	0.70076	0.29924
Inconsciente	0.83153	0.16847

Tabela 21 – Tabela de probabilidade condicional do nodo DistracaoApp. Fonte: Do autor (2017)

Consciente	*P (Percepcao / Consciente)*		
	Presente	Tardia	Ausente
Consciente	0.66273	0.33727	0
Inconsciente	0.07633	0.07633	0.84734

Tabela 22 – Tabela de probabilidade condicional do nodo Percepcao. Fonte: Do autor (2017)

ANEXO A – Aplicação de aprendizagem por reforço

dataset_analysis.py

```
import numpy as np
import pandas as pd
import seaborn as sns
from scipy import stats
import matplotlib.pyplot as plt
from pgmpy.models import BayesianModel


nodes = {
    'Aware',
    'AppDistraction',
    'CarSound',
    'CarDirection',
    'AwareTime'
}

header = [
    'id',
    'car_id',
    'Aware',
    'AppDistraction',
    'CarSound',
    'CarDirection',
    'AwareTime',
    'Questions',
    'CorrectQuestions'
]

dataset = pd.read_csv('datasets/behaviour.csv', sep=';',
    names=header)
dataset.drop(list(set(header) - nodes), axis=1, inplace=True)
dataset.rename(
    columns={
        'Aware': 'Consciente',
        'AppDistraction': 'DistracaoApp',
        'CarSound': 'SomCarro',
        'CarDirection': 'DirecaoCarro',
        'AwareTime': 'Percepcao'
    }, inplace=True
)

dataset = dataset.replace('-', np.nan).dropna()
```

```
dataset.DistracaoApp.replace(['Trivia', 'Headphone'], 'Sim',
    inplace=True)
dataset.DistracaoApp.replace('Button', 'Nao', inplace=True)
dataset.Consciente.replace(['Aware', 'Unaware'], ['Consciente',
    'Inconsciente'], inplace=True)
dataset.SomCarro.replace(['On', 'Off'], ['Sim', 'Nao'],
    inplace=True)
dataset.DirecaoCarro.replace(['Back', 'Front', 'Left', 'Right'],
    ['Tras', 'Frente', 'Esquerda', 'Direita'], inplace=True)

def get_mean_aware_value_by_direction_soundtype(dataset):
    mean_aware_by_type = {}

    df = dataset
    for direction in ('Direita', 'Esquerda', 'Tras', 'Frente'):
        mean_aware_by_type[direction] = {}

        for sound in ('Sim', 'Nao'):
            df_split = df[(df.DirecaoCarro == direction) &
                (df.SomCarro == sound)]
            mean_aware_by_type[direction][sound] =
                df_split.Percepcao.mean()

    return mean_aware_by_type


def generate_awaretime_bins(dataset):
    df = dataset
    df['Percepcao_2'] = ''

    mean_aware =
        get_mean_aware_value_by_direction_soundtype(dataset)

    for car_dir, sound in mean_aware.items():
        for sound_type, mean_value in sound.items():

            mask = (df.DirecaoCarro == car_dir) & (df.SomCarro ==
                sound_type) & (df.Percepcao <= mean_value)
            df.loc[mask, 'Percepcao_2'] = 'Presente'

            mask = (df.DirecaoCarro == car_dir) & (df.SomCarro ==
                sound_type) & (df.Percepcao > mean_value)
            df.loc[mask, 'Percepcao_2'] = 'Tardia'

    mask = (df.Consciente == 'Inconsciente')
```

```
    df.loc[mask, 'Percepcao_2'] = 'Ausente'

    df.drop('Percepcao', axis=1, inplace=True)
    df.rename(columns={'Percepcao_2': 'Percepcao'}, inplace=True)

def generate_boxplots(df):
    total = []

    color_dict = {'Sim':'green', 'Nao':'black'}
    controls = ['Sim', 'Nao']

    for sound in ('Sim', 'Nao'):
        for direction in ('Direita', 'Esquerda', 'Tras',
            'Frente'):
            df_split = df[(df.DirecaoCarro == direction) &
                (df.SomCarro == sound)]
            total.append(df_split.Percepcao)

    fig, ax = plt.subplots()
    boxplot_dict = ax.boxplot(
        total,
        patch_artist = True,
        widths = 0.25)

    i=0
    for b in boxplot_dict['boxes']:
        lab = ax.get_xticklabels()[i].get_text()
        if i % 2 == 0:
            color = 'green'
        else:
            color = 'black'
        b.set_facecolor(color)
        i += 1


    lgd = ax.legend(['Sim', 'Nao'], loc='center right',
        bbox_to_anchor=(1.3, 0.5),
          title='SomCarro')
    lgd.legendHandles[0].set_color('green')
    lgd.legendHandles[1].set_color('black')
    plt.xticks([1.5,3.5,5.5,7.5], ['Direita', 'Esquerda',
        'Tras', 'Frente'])

def generate_confidence_interval(df):
    for direction in ('Direita', 'Esquerda', 'Tras', 'Frente'):
```

```
        for sound in ('Sim', 'Nao'):
            df_split = df[(df.DirecaoCarro == direction) &
                (df.SomCarro == sound)]

    print(stats.t.interval(0.99, len(df.Percepcao)-1,
        loc=np.mean(df.Percepcao),
        scale=stats.sem(df.Percepcao)))
    print(stats.t.interval(0.95, len(df.Percepcao)-1,
        loc=np.mean(df.Percepcao),
        scale=stats.sem(df.Percepcao)))
    print(stats.t.interval(0.90, len(df.Percepcao)-1,
        loc=np.mean(df.Percepcao),
        scale=stats.sem(df.Percepcao)))

dataset = dataset[dataset.Percepcao <= 26]
generate_awaretime_bins(dataset)
SPLIT = int(len(dataset) * 0.7)
TRAIN = dataset[:SPLIT].copy()
TEST = dataset[SPLIT:].copy()
```

---

agent.py

```
import numpy as np
import random
import json
import os
from environment import AwareEnv
from collections import defaultdict


class Agent(object):
    def __init__(self, actions, q_table):
        self.actions = actions
        self.learning_rate = 0.01
        self.discount_factor = 0.9
        self.epsilon = 0.9
        self.q_table = defaultdict(lambda: [0.0] * len(actions),
            q_table)

    def get_action(self, state):
        if np.random.rand() > self.epsilon:
            action = np.random.choice(self.actions)
        else:
            state_action = self.q_table[state]
            action = self.arg_max(state_action)
```

```
            return action

        def learn(self, state, action, reward, next_state):
            q_1 = self.q_table[state][action]
            q_2 = reward + self.discount_factor *
                max(self.q_table[next_state])
            self.q_table[state][action] += self.learning_rate * (q_2
                - q_1)

        @staticmethod
        def arg_max(state_action):
            max_index_list = []
            max_value = state_action[0]
            for index, value in enumerate(state_action):
                if value > max_value:
                    max_index_list.clear()
                    max_value = value
                    max_index_list.append(index)
                elif value == max_value:
                    max_index_list.append(index)
            return random.choice(max_index_list)

    if __name__ == "__main__":
        q_table = {}

        if os.stat('q_table.json').st_size > 0:
            answer = input('Foi encontrado um arquivo q_table
                contendo uma tabela j preenchida. Gostaria de
                continuar utilizando-na? Y/N: ')

            with open('q_table.json', 'r+') as qtable_json:
                if answer.upper() == 'Y':
                    q_table = json.load(qtable_json)
                else:
                    qtable_json.seek(0)
                    qtable_json.truncate()

        env = AwareEnv()
        agent = Agent(actions=list(range(len(env.actions))),
            q_table=q_table)

        try:
            for _ in range(10):
                state = env.reset()
```

```
            for episode in env.episodes.iterrows():
                env.render()

                index, episode = episode

                adjustment = agent.get_action(str(state))

                next_state, reward =
                    env.step(env.actions[adjustment], episode)
                agent.learn(str(state), adjustment, reward,
                    str(next_state))

                state = str(next_state)
    finally:
        print('\n ######## Rede Bayesiana Final ########')

        for node in env.model.get_cpds():
            print(node)

        with open('q_table.json', 'w') as qtable_json:
            json.dump(agent.q_table, qtable_json, indent=4)
```

---

environment.py

```
import networkx
import numpy as np
from dataset_analysis import TRAIN, TEST
from pgmpy.models import BayesianModel
from pgmpy.inference import BeliefPropagation
from pgmpy.estimators import BayesianEstimator


replacer = {
    'DistracaoApp': {
        'Nao': 0,
        'Sim': 1
    },
    'DirecaoCarro': {
        'Tras': 3,
        'Direita': 0,
        'Esquerda': 1,
        'Frente': 2
    },
    'Consciente': {
        'Consciente': 0,
```

```
        'Inconsciente': 1
    },
    'SomCarro': {
        'Nao': 0,
        'Sim': 1
    },
    'Percepcao': {
        'Presente': 1,
        'Tardia': 2,
        'Ausente': 0
    }
}

approximate = {
    'DistracaoApp': {
        'Nao': 0.00177777778,
        'Sim': -0.00177777778
    },
    'DirecaoCarro': {
        'Tras': -0.00111111111,
        'Direita': 0.00166666667,
        'Esquerda': 0.0022222222,
        'Frente': -0.0088888889
    },
    'SomCarro': {
        'Nao': 0.00155555556,
        'Sim': -0.00155555556
    },
    'Percepcao': {
        'Presente': -0.00622222222,
        'Tardia': -0.00377777778,
        'Ausente': 0.00997777778
    }
}

inverse_replacer = {
    outer_k: {
        inner_v: inner_k for inner_k, inner_v in outer_v.items()
    }
    for outer_k, outer_v in replacer.items()
}


class AwareEnv(object):
    def __init__(self):
```

```
        self.actions = [
            0.0, 0.01, 0.02, 0.03, 0.04, 0.05,
            -0.01, -0.02, -0.03, -0.04, -0.05
        ]
        self.model = BayesianModel([
            ('Consciente', 'DistracaoApp'),
            ('Consciente', 'DirecaoCarro'),
            ('Consciente', 'SomCarro'),
            ('Consciente', 'Percepcao')
        ])
        self.episodes = TEST.copy().drop('Consciente', axis=1)

    def reset(self):
        self.model = BayesianModel([
            ('Consciente', 'DistracaoApp'),
            ('Consciente', 'DirecaoCarro'),
            ('Consciente', 'SomCarro'),
            ('Consciente', 'Percepcao')
        ])
        self.model.fit(TRAIN, estimator=BayesianEstimator)
        aware = [node for node in self.model.get_cpds() if
             node.variable == 'Consciente'].pop()
        self.state = [np.round(aware.values, 2)]

        self.cpds = self._tabular_cpds_to_dict(self.model)

        for node in self.model.get_cpds():
            print(node)

        return self.state

    def render(self):
        aware = [node for node in self.model.get_cpds() if
             node.variable == 'Consciente'].pop()
        self.state = np.round(aware.values, 2)

        self.cpds = self._tabular_cpds_to_dict(self.model)

    def _tabular_cpds_to_dict(self, model):
        return {
            node.variable: {
                state: value for state, value in
                    zip(node.state_names[node.variable],
                    node.values)
            }
```

```
            for node in model.get_cpds()
        }

    def _get_cpd_values(self, node_values):
        cpds = []

        for state, param in node_values.items():
            if type(param) == dict:
                cpds.append(list(param.values()))
            else:
                cpds.append(param)

        return np.array(cpds)

    def step(self, adjustment, episode):
        print('######## Ajustes ########')
        print(adjustment)
        print('######## Episdio atual ########')
        print(episode)

        bp = BeliefPropagation(self.model)
        replaced_episode = {k: replacer[k][v] for k, v in
            episode.iteritems()}

        upper_bound = self.state[0] + adjustment
        lower_bound = self.state[1] - adjustment

        if not (upper_bound > 1 or upper_bound < 0):
            state_aware = [upper_bound, lower_bound]

            cpds = self._tabular_cpds_to_dict(self.model)
            adjustments = self.fit_probabilities(cpds, adjustment)
            for node in self.model.get_cpds():
                if node.variable != 'Consciente':
                    node.values =
                        self._get_cpd_values(adjustments[node.variable])
                    node.normalize()
                else:
                    node.values = np.array(state_aware)

            for node in self.model.get_cpds():
                print(node)
        else:
            state_aware = [self.state]
```

```
        print('######## Consciente ########')
        bp = BeliefPropagation(self.model)
        print(bp.query(['Consciente'],
            evidence=replaced_episode)['Consciente'])

        reward = float(input('Recompensa entre -1 e 1: '))
        next_state = []
        next_state.append(np.round(state_aware, 2))
        next_state.extend(list(replaced_episode.values()))

        return next_state, reward

    def fit_probabilities(self, cpds, adjustment):
        del cpds['Consciente']

        adjusted_probabilities = {}
        position = int(adjustment < 0)

        for state, param in cpds.items():
            params = list(param.keys())
            param_values = list(param.values())

            new_param_values = []
            npt = np.transpose(param_values)

            for cpd_list, param in zip(npt, params):
                fitting = approximate[state][param] * (adjustment
                    * 100)

                values = []
                for cpd in cpd_list:
                    fit = cpd + fitting

                    if fit < 0:
                        fit = 0
                    elif fit > 1:
                        fit = 1

                    values.append(fit)

                new_param_values.append(self.normalize(values))

            npt = np.transpose(new_param_values)
            adjusted_probabilities[state] = {}
```

```
            for i, param in enumerate(params):
                adjusted_probabilities[state][param] =
                    np.array(npt[i])

        return adjusted_probabilities

    def normalize(self, lst):
        s = sum(lst)
        return list(map(lambda x: float(x)/s, lst))
```

ANEXO A – Artigo

Aplicação de aprendizagem por reforço para um modelo bayesiano de consciência situacional

Rodolfo Lottin Pereira[1]

[1]Departamento de Informática e Estatística - Universidade Federal de Santa Catarina (UFSC)
Campus Universitário – Florianópolis – SC - Brasil

rodolfolottin1@gmail.com

Abstract. *Bayesian networks are graphical models to reason and represent a knowledge about an uncertain environment. The huge amount of accidents that occur with pedestrians traveling near urban roads is associated with the excessive use of mobile devices, as it causes a lack of awareness of the environment. With this in mind, the Road Awareness project proposes a situational awareness model of smartphone users near urban roads, to build up with the decrease in the incidence of these events. To contribute to the project, an algorithm of reinforcement learning was implemented and applied in the parameter learning of the Bayesian network.*

Resumo. *Redes Bayesianas são modelos gráficos para raciocinar e representar um conhecimento sobre um meio incerto. A enorme quantidade de acidentes que acontecem com pedestres trafegando próximos de vias urbanas está associada ao uso excessivo de dispositivos móveis, por acarretar na falta de consciência em relação ao meio. Tendo isso em mente, o projeto ``Road Awareness'' propõe um modelo de consciência situacional de usuários de smartphones próximo a vias urbanas, a fim de contruibuir com a diminuição da incidência desses acontecimentos. Com o objetivo de contribuir com o projeto, foi implementado e aplicado um algoritmo de aprendizagem por reforço na aprendizagem dos parâmetros da rede Bayesiana.*

1. Introdução

Devido ao surgimento e desenvolvimento de algoritmos de aprendizado prático, houve um crescente interesse de pesquisadores na abordagem conexionista e assim nas redes neurais artificiais. Proporcionando a origem de uma área chamada de aprendizagem de máquina (AM), criada com o objetivo de resolver problemas de natureza prática, por meio de métodos e modelos da estatística e teoria da probabilidade (MITCHELL, 1997).

Um dos modelos de AM é o supervisionado. Nele existe uma espécie de professor a fim de ajudar o sistema no desenvolvimento da função responsável pela classificação. É manuseado um conjunto de dados categorizados, com a finalidade de treinar o classificador e então utilizá-lo com dados que ainda não foram processados. Portanto, de acordo com o que já foi visto tenta-se prever os que se seguem.

Em situações onde não se possui bases rotuladas, é necessário uma abordagem diferente, conhecida como não-supervisionada. Embora o sistema não tenha um

professor para guiá-lo, é responsabilidade do próprio algoritmo de aprendizado avaliar seus conceitos (LUGER, 2013), onde seu objetivo é explorar os dados e encontrar estruturas padronizadas ou relações nele. Por conseguinte, é possível agrupar os elementos com características similares e assim tratá-los de formas diferentes.

Diferindo-se das abordagens citadas acima, existe a aprendizagem por reforço (AR), sendo comumente utilizada com agentes que devem aprender sobre seu comportamento por interações de tentativa e erro (LUGER, 2013). É apresentado a ele a recompensa pela escolha feita e o estado avançado, porém não é dito qual seria a melhor opção de acordo com seus interesses a longo prazo (KAELBLING; LITTMAN; MOORE, 1996).

Redes Bayesianas (RB) foram criadas para permitir uma eficiente representação sob a incerteza de um determinado conhecimento. Essa abordagem propicia uma aprendizagem por experiência, e combina alguns conceitos da inteligência artificial (IA) clássica e das redes neurais. Uma RB é um grafo dirigido em que cada nodo tem uma informação probabilística quantitativa correspondente. A topologia dessa rede (o conjunto de nodos e arestas) indica as relações condicionais que existem no domínio. Uma vez que a estrutura tiver sido estabelecida, é necessário especificar a distribuição de probabilidade condicional de cada variável, de acordo com seus parentes (PEARL, 2014).

É de comum acordo que a elaboração da estrutura de uma rede e seus parâmetros requer a necessidade de um especialista e uma análise criteriosa do domínio. Com o crescimento das pesquisas na área de IA meios de aplicar técnicas de aprendizagem em RB foram e têm sido desenvolvidos. Sendo assim, este trabalho de conclusão de curso atuará na análise, desenvolvimento e experimentação de uma técnica de aprendizagem por reforço com o propósito de aprender os parâmetros da rede.

2. Motivação

Esse trabalho contribui em parte do projeto de pesquisa *Computational Model of Situational Awareness for users of Smartphones*. O projeto propõe o desenvolvimento de um modelo bayesiano de atenção situacional de usuários que trafegam próximo a rodovias e áreas urbanas com dispositivos móveis. Portanto, de acordo com dados provindos de análise e experimentação do projeto, será aplicada uma técnica com o intuito de alimentar essa rede previamente definida. Esse modelo poderá ser utilizado para ajudar a reduzir a incidência de acidentes com pedestres por conta da distração provocada por smartphones. Colaborando de tal forma com o projeto citado.

Objetivando contribuir com o desenvolvimento de um modelo computacional a aplicação de reforço se dará de forma a proporcionar ajustes no modelo gerado. Essa melhoria é fundamental para que se haja um modelo mais fidedigno e utilizável de acordo com as situações que lhe são apresentadas.

3. Fundamentação Teórica

3.1 Aprendizagem por reforço

"A tarefa do agente é encontrar uma política de controle**,** mapeando estados a ações, que maximizem alguma medida de reforço a longo termo" (KAELBLING; LITTMAN; MOORE, 1996, p. 239, traduzido pelo autor)**.** Como dito, a tarefa do

agente é encontrar uma política, um mapeamento de estados para ações que sejam melhores em determinadas situações, não só de acordo com o valor numérico de recompensa imediato, mas também com o valor de longa distância.

Segundo Sutton e Barto (1998), a política de controle define o comportamento de um agente em um determinado momento. A cada passo, o agente implementa um mapeamento dos possíveis estados para probabilidades de selecionar cada possível ação. Sendo considerada a principal característica desta abordagem, é dito que ela, por sozinha, é suficiente para determinar o comportamento de uma aplicação.

Usualmente utilizado em problemas interativos, onde, na maioria dos casos, não é prático obter exemplos representativos para todas situações e estados do ambiente que o agente deve agir. Desse modo é necessária a habilidade do agente de poder aprender por suas próprias experiências, de acordo com o feedback colhido pelas alterações no local inserido por conta das suas ações (SUTTON; BARTO, 1998).

Uma função de recompensa mapeia cada par estado-ação do ambiente e do agente em um número indicando o quão interessante, de acordo com os objetivos, é a situação daquele estado em específico. As ações do agente determinam não apenas o seu valor de recompensa imediato, mas também seu próximo estado. Dessa forma o agente deve ter a habilidade de aprender por reforço atrasado (*delayed reward*), pois pode levar uma sequencia de ações com valores baixos de recompensa, até obter um estado com um alto valor de reforço. E é exatamente o seu modelo de longo prazo de otimização que determina como o agente lida com os futuros valores de recompensa (KAELBLING; LITTMAN; MOORE, 1996).

Apesar de parecer limitado, na prática a ideia de um sinal de recompensa se provou flexível e amplamente aplicada. Por exemplo, para um robô aprender a andar, cada passo completado para frente poderia receber um sinal positivo. No caso de um robô aprendendo a escapar de uma prisão, a recompensa seria -1 até escapar, onde passaria a ser 1. No mesmo exemplo, porém com outra abordagem, afim do robô aprender a escapar o mais rápido possível, cada passo que o distanciaria mais de sua fuga, receberia um sinal negativo (SUTTON; BARTO, 1998).

3.2 Redes Bayesianas

Uma das alternativas para se ter mais precisão e confiabilidade ao raciocionar sobre um domínio é tentar sumarizá-lo de tal forma atribuindo valores númericos a cada proposição dispostos em uma distribuição (PEARL, 2014). Tal maneira de raciocínio é conhecido como probabilístico e suas aplicações são principalmente em domínios em que as relações de causa e efeito não são captadas tão facilmente. A informação probabilística pode indicar e priorizar causas e possíveis explicações para uma evidência, por exemplo: a evidência de quais sintomas são mais prováveis de levar um paciente a ter uma certa doença (LUGER, 2013).

“O formalismo de RB foi inventado para permitir uma representação eficiente e um raciocínio rigoroso ao lidar com um conhecimento incerto.” (RUSSELL; NORVIG , 2003, p. 26, traduzido pelo autor). Uma RB é um grafo orientado acíclico, em que cada um de seus nós representam evidências ou hipóteses e as arestas que conectam dois nodos representam uma dependência entre eles. Caso não haja dependência entre dois eventos, é dito que a probabilidade de um evento ocorrer não

interfere na de outro. As RB são comumente utilizadas a fim de representar graficamente um conhecimento de um domínio por meio de relações de dependência entre variáveis e, matematicamente, probabilidades a priori e probabilidades condicionais entre variáveis.

3.2 Consciência Situacional

"Consciência situacional é o entendimento do estado do ambiente, assim como os parâmetros relevantes do sistema" (ENDSLEY ,1995a, p. 65, traduzido pelo autor) afirma que a CS passou a ser reconhecida como uma grande utilidade para pilotos de aeronaves em um período próximo da Primeira Guerra Mundial. Tal que, as operações executadas pelo piloto da aeronave devem estar completamente consistente com os objetivos do piloto e as situações apresentadas a maquina. Sem informações como: condições externas, fatores hostis, informações de navegação e sobre outras aeronaves próximas, pilotos não serão aptos a executarem as suas funções e objetivos efetivamente.

4. Desenvolvimento

Essa capítulo trata do desenvolvimento da proposta de aplicação de AR em uma RB. Essa combinação se dará pela integração do conhecimento do especialista ao reforçar - avaliar entre um sinal positivo e negativo - o caminho o qual o processo de aprendizagem está tomando.

4.1 Estudo de caso

Essa seção tem como objetivo descrever o caso de estudo utilizado, o experimento, informações relevantes sobre o conjunto de dados levantado e a RB criada.

Como estudo de caso foi utilizado o experimento realizado pelo projeto *Road Awareness*. O objetivo do experimento foi coletar informações que pudessem ajudar a entender quais os principais fatores que afetam a consciência situacional de um usuário de *smartphones* trafegando próximo a vias urbanas. Para isso, foram realizados experimentos em um laboratório que lograva de uma plataforma de imersão em realidade virtual capaz de simular ambientes urbanos.

Essa plataforma de imersão é chamada de *Octave*, que é um ambiente completamente configurável, de uma experiência de imersão holográfica que projeta visões 3D ao redor do usuário, recria som e toques no próprio ambiente. Proporcionando ao usuário ficar imerso em uma realidade virtual, com a habilidade de ver, ouvir, mover e manipular objetos.

Ela está localizada na Universidade de Salford - UK e suas utilizações estão em fins acadêmicos, pesquisa médica, prototipação de veículos, recriação segura de ambientes perigosos, entre outros. Um sistema de acústica com 264 alto-falantes dá ao usuário uma experiência quase real, enquanto imerso no ambiente, seja no protótipo de um carro ou na superfície de Marte.

A Figura 1 é uma foto do ambiente sendo simulado. O ambiente contém quatro vias, em quatro direções: a frente, trás, direita e a esquerda do usuário, onde, em apenas uma por vez, surge um carro que irá trafegar até o final do seu sentido oposto. Enquanto

isso, existem pequenos animais, chamados de *lemmings*, que aparecem e atravessam a faixa de pedestres que é possível visualizar na figura. O usuário deve ficar atento a quando um carro aparece no cenário e sinalizar de que ficou consciente a ele clicando em um botão no seu *smartphone*, assim o semáforo da faixa de pedestres fecha e os *lemmings* param de atravessá-la.

Figura 1. Usuário imerso no ambiente da simulação. Fonte: Primeiro experimento realizado no laboratório na Universidade de Salford – UK.

A análise dos fatores que afetam a atenção do usuário não está apenas nas direções dos carros e nos *lemmings*. Foram feitos três tipos de simulações: com um aplicativo fazendo perguntas a fim de tomar a sua atenção e distraí-lo, de cunho matemático em sua maioria, que o usuário deveria responder; com as perguntas e com fones de ouvido; e simulações em que o objetivo era apenas apertar o botão de consciência em relação ao carro.

Além das diferentes simulações, os carros que eram simulados no ambiente poderiam vir sem ou com som. Os carros sem som vem a ser útil a fim de simular o efeito da diminuição de sons emitidos por carros elétricos nos dias de hoje e o quanto a percepção do usuário é afetada.

Assim como o projeto *Road Awareness* foi utilizado como estudo de caso para este trabalho, outros trabalhos utilizaram-no em suas pesquisas. Como por exemplo: o desenvolvimento de uma arquitetura para atuação de agentes em ambientes de simulação virtual, que teve como proposta a criação de um modelo de interação entre o agente e o ambiente, objetivando se tornar o mais próximo do visto em um mundo real; integração de AM no desenvolvimento de uma arquitetura baseada em agentes BDI, a fim de realizar o processamento de percepções por meio da união entre os componentes pró-ativo e reativo.

Neste trabalho, foi utilizado como estudo de caso o conhecimento adquirido pelo experimento, assim como o conjunto de dados, possibilitando a criação de um modelo de consciência situacional, uma RB. Posteriormente propiciando a aplicação de AR no modelo gerado como forma de aproximá-lo do mundo real.

4.2 O experimento

O experimento coletou dados sobre o comportamento de 20 usuários nos 3 tipos de simulações. O experimento teve como resultado um conjunto de dados com informações como: identificação do objeto (*lemming*, carro), assim como o tempo em que foi adicionado na simulação, sua posição, direção, tipo (som ligado ou desligado para os carros), qual era o tipo de simulação que estava ocorrendo, perguntas feitas para o usuário e suas respectivas respostas, assim como informações relativas a sua percepção e atenção relacionada aos carros.

A Tabela 1 mostra um exemplo de parte do conjunto e quais informações estão contidas. Esse conjunto de dados é uma abstração das informações que foram coletadas durante o experimento para que mais se adequem ao modelo desejado e foi gerada em um esforço colaborativo do grupo envolvido com a pesquisa. O novo conjunto contém 962 registros, dos quais 40 foram removidos por não estarem completos.

Tabela 1. Exemplo de parte do conjunto de dados utilizado.

Id	Consciente	Simulacao	SomCarro	DirecaoCarro	Tempo
6	Sim	Fones	Não	Esquerda	12.055s
6	Não	Fones	Sim	Esquerda	16.260s
6	Sim	Perguntas	Não	Atrás	1.383s
…	…	…	…	…	...

Para cada um dos carros adicionados ao evento foram consideradas informações como: identificação do usuário, o tipo de simulação que estava ocorrendo, se o som do carro estava ligado ou desligado, a direção do carro, quanto tempo o usuário levou para sinalizar de que percebeu o carro no ambiente e se, a partir desse tempo, o usuário estaria consciente ou não em relação ao carro foram obtidas e utilizadas para preencher o conjunto de dados utilizado.

Apesar de não estar refletido nos dados, há de se considerar que existem três casos de um usuário ter ficado consciente em relação a aproximação de um carro, que são: o fato dele realmente ter ficado consciente e apertado o botão do aplicativo em tempo suficiente, o fato dele não ter apertado o botão em momento algum desde a criação e destruição do carro no ambiente virtual e o fato dele ter apertado o botão porém o carro já ter passado de um limite relativo a zona de segurança.

Além disso, embora todos valores desse tenham o mesmo significado: o tempo desde que o carro apareceu no ambiente até o usuário percebê-lo ou não e sinalizar no seu *smartphone*, foi preciso lidar com alguns fatores técnicos que impossibilitaram definir, para as diferentes direções e opções de som do carro, faixas iguais para a

discretização do atributo Tempo. Assim como problemas relacionados ao ambiente e a sua sincronização com o aplicativo.

Um problema identificado foi no tempo de criação dos carros pelo Unity, um motor de jogo que contém um ferramental necessário que possibilita e facilita o desenvolvimento de jogos. O software possibilita a adição de trilhas sonoras, adição de física a objetos, funções gráficas e outras ações.. Um carro que continha som levava um tempo a mais para ser processado pelo motor de jogo até aparecer no ambiente e isso gerou um atraso em alguns casos para sua criação e registro. Portanto não se pôde considerar que valores maiores que apenas um número em específico corresponderiam a uma classe.

A Figura 2 apresenta um gráfico de caixas que representa o tempo em média em que os usuários levaram para apertar o botão do aplicativo para as diferentes direções e em relação ao som dos carros, assim como as suas medidas discrepantes. É possível constatar que as direções da Direita, Esquerda e Frente, especialmente as duas últimas, tiveram vários valores discrepantes. Assim como que o o som do carro afetou diretamente o tempo para resposta do usuário para todas direções, com médias, em um geral, bem diferentes do caso com som. Exceto da direção da Frente. É possível também observar a diferente dispersão dos dados representada pela amplitude de cada uma das caixas.

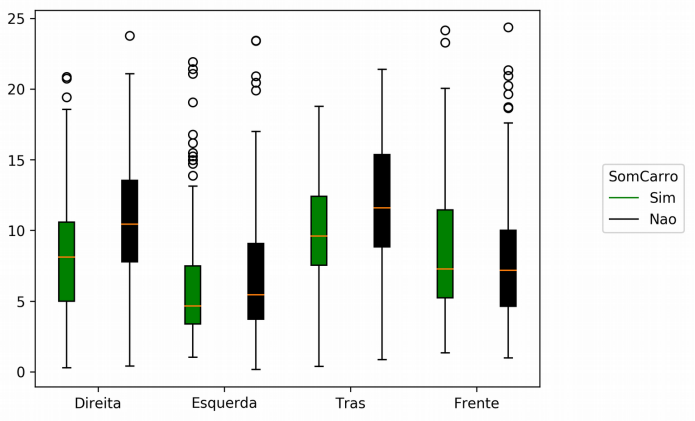

Figura 2. Gráficos de caixa da distribuição do atributo Percepcao para os tipos de som de acordo com a direção do carro.

A Tabela 2 apresenta os exatos valores que foram considerados para a discretização do atributo. Também é possível constatar que carros vindos da esquerda e da direita tem um tempo médio maior para percepção. Isto pode ser explicado pelo nível de oclusão visual do usuário para os carros da direita e ao fato de que, para aumentar o seu campo de visão em relação a via da esquerda, o usuário precisaria se locomover um pouco à frente.

Tabela 2. Média de tempo de Consciente para direções e sons dos Carros.

Som \ Direcao	Trás	Frente	Esquerda	Direita
Sim	6.279s	7.069s	10.635s	8,297s
Não	8.518s	8.160s	11.629s	9.646s

Para a média geral, de todos tipos de carro e direções, foi encontrado o seguinte intervalo de confiança para 95%: [8.508, 9.033], sendo 8.7705 ± 0.2625. Assim, podendo afirmar que se forem construídos intervalos de confiança, nestas condições, para cada um dos dados, 95% conterão estes valores.

A Figura 4 corrobora a ideia da influência do posicionamento do usuário em relação a essas vias, podendo constatar que houveram mais casos de incosciência em relação ao carro para os carros vindos dessas direções.

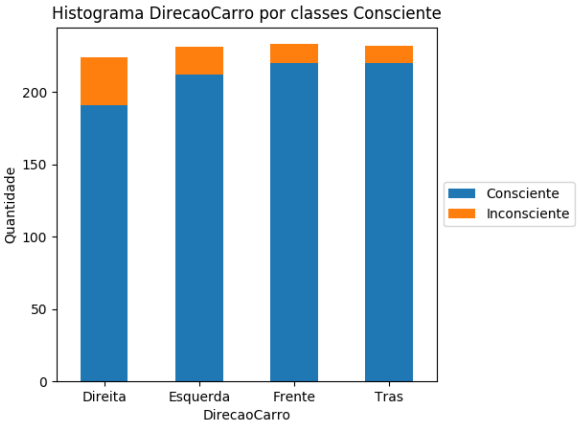

Figura 4. Histograma relação classes Consciente por DirecaoCarro.

Apesar das informações relativas ao tempo médio de carro, considerarem que todos eventos acima destes valores seriam uma classe única resultaria em um erro, uma vez que existem os casos de falsa consciência (casos em que o usuário ficou consciente ao carro porém com atraso). Logo, foi considerado que os registros em que o usuário sinalizou consciência, com tempo de percepção maior que a média da sua direção e som do carro, pertencem a classe de percepção Tardia. Eventos em que o usuário não sinalizou foram categorizados como Ausente e o restante como Presente.

Na Figura 5 é possível visualizar como ficou a distribuição dos dados de acordo com as classes criadas. Existe uma quantidade muito maior de itens pertencente às classes Presente e Tardia em relação a Ausente, isso pode ser explicado pelo fato de que

durante o experimento houveram poucos casos de Inconsciente, como também é possível constatar na própria figura.

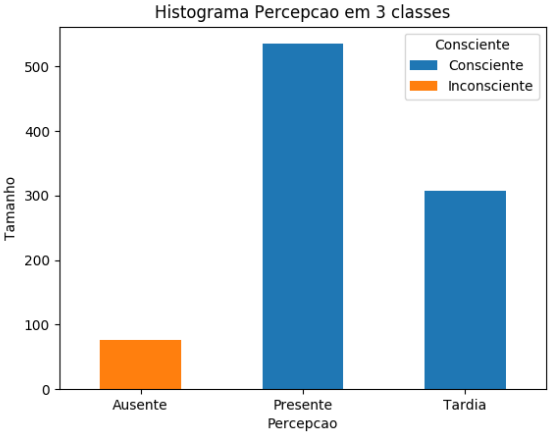

Figura 5. Histograma atributo Percepção após discretização.

No que tange aos outros atributos, houve um processo de discretização apenas no atributo Simulacao, renomeado para DistracaoApp. As simulações com fones de ouvido e que havia perguntas foram agrupadas na mesma classe, a de que há a distração do aplicativo.

4.3 A Rede Bayesiana

Uma vez que RB's são mecanismos eficientes para representar e raciocionar sobre um modelo de incerteza, seu formalismo é facilmente aplicável para este domínio. A definição do modelo partiu do objetivo de se possibilitar analisar o quanto cada um dos fatores influeciam diretamente na consciência do usuário. Para isso, ela foi elaborada de acordo com as situações que lhe são apresentadas durante a simulação.

Com o conjunto de dados gerado e após o processo de discretização, além dos estados de consciência do usuário, obtem-se as situações que lhe foram evidenciadas, como: direção do carro, tempo de percepção, som do carro e o tipo de simulação. Tais situações influenciam diretamente na atenção do usuário em relação ao seu meio.

Portanto, a partir de um esforço colaborativo pelo grupo de pesquisa, que inclui o autor, relacionado ao projeto, foi construído manualmente um modelo de RB, retratado na Figura 6. No modelo construído é possível analisar, por exemplo, o quanto as diferentes direções dos carros influenciam na consciência do usuário, assim como os tipos de simulação e os fatores restantes.

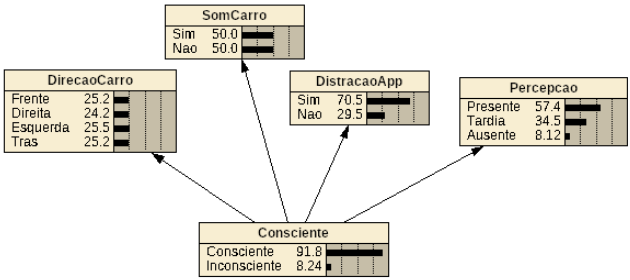

Figura 6. RB original no software Netica.

De acordo com as evidências apresentadas à RB, é possível inferir qual seria o grau de consciência situacional do usuário, como é possível analisar em um exemplo de inferência na Figura 7. O caso simulado, com um carro vindo da direita, sem som e em uma simulação do tipo em que há distração no aplicativo, é o caso que mais interferiu na atenção do usuário em relação ao carro.

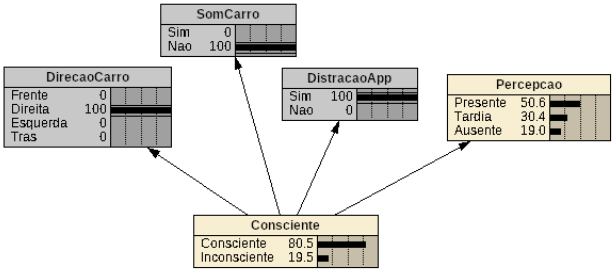

Figura 7. Inferência RB com evidência de DirecaoCarro, SomCarro e DistracaoApp.

Os valores dos parâmetros dos nodos da RB foram obtidos a partir da utilização de um estimador bayesiano, utilizando a biblioteca pgmpy (disponível no repositório: https://github.com/pgmpy/pgmpy), implementada em Python. O conjunto de dados foi separado em 2 conjuntos menores, dos quais: o conjunto maior, com 70% dos dados, foi utilizado para a aprendizagem dos parâmetros; e o menor, com 30%, será utilizado para a aplicação de AR para ajustar os valores da RB.

4.4 O modelo

A aplicação de AR acontecerá no intuito de reforçar a rede e aproximá-la do que se observou e estudou referente a consciência situacional do usuário ao ambiente, tentando aproximar os parâmetros da RB de situações mais reais que, por conta da quantidade de dados, podem não estar retratadas no modelo. Serão feitas simulações com base no conjunto de testes e de acordo com o conhecimento do especialista sobre o domínio, a RB será reforçada. A análise será de acordo com os resultados da rede para os respectivos cenários.

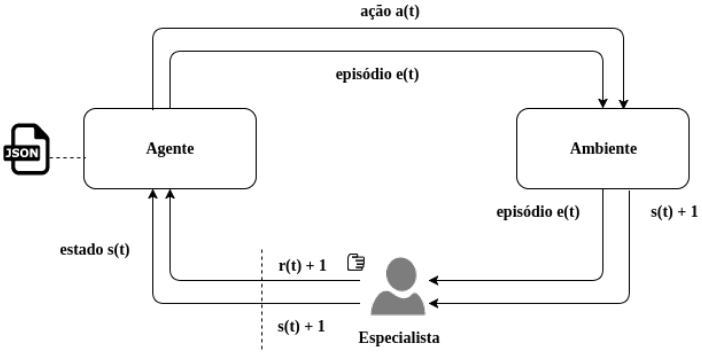

Figura 8. Modelo proposto.

Em uma visão geral o processo de AR na RB será conforme a Figura 8. Nela, o agente propõe ações de ajustes nos valores de probabilidade condicional da RB e envia esta ação ao ambiente juntamente com o episódio da iteração. O ambiente, que contém a RB, recebe o episodio (uma tupla do conjunto de dados que fora separado para testes, e aqui, neste trabalho, estarão sendo utilizados para ajustar os parâmetros da RB) e utiliza a estrutura atual da RB para inferir a probabilidade a posteriori de diagnóstico do nodo Consciente conforme os valores contidos no episódio. Em seguida, utiliza a ação de ajuste proposta pelo agente para alterar os valores de probabilidade condicional do nodo Consciente, propagando os ajustes para os seus nodos filhos e assim construindo uma RB com novos valores de probabilidade condicional e a priori.

A figura do Especialista, apresentada na Figura 8, representa o papel do especialista sobre o domínio da RB. Como escopo deste trabalho, a recompensa relacionada a ação e como o ambiente respondeu a ela, é manualmente inserida pelo especialista. O especialista, conforme o seu conhecimento, analisa a ação e determina o novo valor de probabilidade a posteriori para a RB ajustada e retorna a aplicação um valor numérico entre -1 e 1 simbolizando uma punição ou recompensa.

O agente recebe a recompensa, o novo estado do nodo Consciente da RB, assim como os valores dos atributos do episódio que foram utilizados como evidência para a consulta e armazena essas informações conforme o seu algoritmo de aprendizagem.

4.5 A aprendizagem

Neste trabalho foi utilizado o algoritmo *Q-learning* para implementar a aprendizagem do agente para os estados passíveis de serem atingidos pelo ambiente e as respectivas recompensas, qual o ajuste mais provável para determinada situação. A implementação do agente foi adaptada do repositório *https://github.com/rlcode/reinforcement-learning/*, também escrito em Python, que implementa exemplos dos agentes mais conhecidos na literatura de AR.

Primeiramente é construída a RB e, com o conjunto de dados de treino, aplicado o método de estimação bayesiana. Posteriormente, para cada uma das 10 iterações de um laço, a RB será reiniciada, ou seja, voltará aos valores de probabilidade iniciais. O

fato de voltar a RB ao seu estado original, proporciona que os as ações sugeridas pelo agente passem novamente pelos mesmos processos e, de acordo com o aprendido, sugira ações que maximizem o valor de recompensa em longo termo. Em cada uma dessas iterações, os episódios contidos no conjunto de dados de teste serão aplicados ao modelo proposto. Para cada um desses episódios, o agente fará ajustes de acordo com o que já sabe sobre o ambiente e a recompensa guiará o processo de aprendizagem, numa forma de tentativa e erro.

Para este trabalho foram considerados como estados do ambiente, os valores contidos no epísodio e os parâmetros do nodo Consciente. A sugestão de ajustes pelo algoritmo se dará da seguinte forma: caso não sejam utilizadas informações salvas de outras iterações, o agente irá propor de maneira aleatória a escolha de um ajuste para o estado inicial. A decisão sobre escolher entre explorar novas ações ou escolher entre as ações que já se sabe que são boas, se dá por um processo de verificar um número aleatório gerado e compará-lo com uma taxa, caso seja maior, serão exploradas novas ações, caso contrário se escolherá entre os ajustes que já se conhece terem resultados satisfatórios.

Em se tratando de sua função de aprendizagem, o algoritmo, recebe como entrada o estado apresentado, a sua ação, o valor dado de recompensa e o próximo estado obtido. Nele, seleciona a atual pontuação utilizando como chave para busca o estado anterior e a ação escolhida. Em seguida, soma a recompensa atribuída com um fator de desconto multiplicado pelo valor máximo de pontuação apresentado para as ações escolhidas para o novo estado. É feita uma subtração da última com a primeira variável, multiplicado por um fator de aprendizagem e somado a pontuação que se tem para a respectiva ação no determinado estado.

Diante da dificuldade de se obter o alor exato que é propagado para os outros nodos enquanto se altera a distribuição de probabilidade do nodo Consciente, foi encontrado uma solução aproximada. Os valores de ajustes estabelecidos para cada um dos nodos foram estimados de acordo como o modelo se comportou em ajustes manuais no software Netica. Portanto, os ajustes realizados no nodo Consciente serão propagados para os seus nodos filhos e os seus valores de probabilidade serão somados aos valores obtidos de acordo com o ajuste proposto pelo agente em razão do ajuste de cada nodo, em seguida estes valores serão normalizados.

Os valores dos ajustes podem ser: 0, 1%, 2%, 3%, 4%, 5% tanto positivos quanto negativos. Dessa forma, serão ajustados os valores de probabilidade do nodo conforme os ajustes, não podendo haver ajustes inválidos que ultrapassem a margem de 100% ou 0%. A escolha dessa variedade de ações se deu pela natureza do domínio, uma vez que valores maiores de ajustes poderiam levar a números que não fossem tão fidedignos.

Ao iniciar a aplicação de AR na RB pode se escolher pela opção de utilizar um arquivo JSON (uma maneira compacta de armazenar e transmitir informações) que contém os valores das pontuações de cada ação para os estados do ambiente. Essas informações são salvas sempre que a aplicação é encerrada. Dessa forma é possível continuar o processo de aprendizagem a partir do ponto em que foi encerrada.

Uma das limitações do trabalho também está no nodo em que o agente propõe as ações de ajustes. Como é apenas no nodo Consciente, limitou-se a apenas analisar como a RB reflete a essas mudanças.

4.6 Testes

O critério utilizado para os valores de recompensas atribuído para cada um dos ajustes propostos pelo agente, foi de acordo com o ajuste proposto para a situação apresentada. Ou seja, no caso de um episódio em que não se há som no carro, sua direção é da esquerda ou da direita e há a distração do aplicativo, ajustes que aumentem o nível de consciência são passíveis de receberem punição, uma vez que estes casos foram os que mais apresentaram atrapalhar o usuário..

Enquanto em casos com percepção presente, carros vindo da frente, com som ligado e não há distração do aplicativo, foram dados valores de recompensas maiores a ajustes pequenos negativos ou que mantivessem o atual estado de consciência, já que para estes casos não há um nível de oclusão significante a ser levado em questão, assim como pelo som do carro despertar a atenção em relação a sua aproximação.

Embora a maioria dos casos houve percepção do usuário em relação ao carro, significando que ele ficou consciente a ele, houveram casos em que a sinalização de consciência pôde ser considerada como precoce. Em tais situações, na ausência ou demora de geração de um carro no cenário, o usuário por muitas vezes precipitou-se em apertar o botão para sinalizar. Estes casos seriam retratados pela RB como uma probabilidade a posteriori próxima de 100%, porém erroneamente. Diante disso, pequenos ajustes em alguns episódios já aproximariam o modelo de valores mais exatos.

As análises feitas durante as iterações também partiram do princípio de não tentar fugir muito dos valores de parâmetros apresentados pela RB após a estimação com o método utilizado, principalmente por tentar manter a RB o mais próximo do que foi apresentado no experimento, apenas tentando ajustá-la às situações mais críticas observadas durante a pesquisa e desenvolvimento do modelo.

4.7 Experimentos

Com o objetivo de testar o quanto o reforço influencia no rumo em que os ajustes são dados as redes, foram criados 2 casos de testes. No primeiro, o qual gerou a RB retratada na Figura 8, foram retornados valores de recompensa aleatórios entre -1, 0 e 1. Para ele foram executadas todas as iterações do algoritmo.

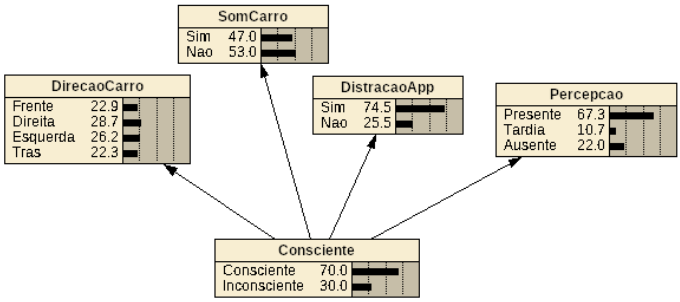

Figura 8. RB ajustada com reforços aleatórios.

Enquanto no segundo, apresentado na Figura 9, foram retornados valores de recompensas mais fidedignos ao domínio. Por conta do fator tempo, para esse segundo, foram executadas apenas três iterações do algoritmo, o que seria no total, aproximadamente, 970 ajustes.

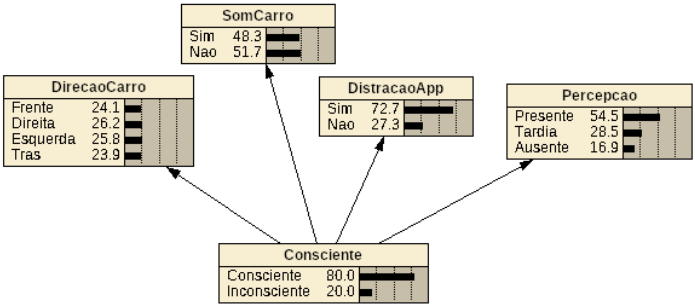

Figura 9. RB ajustada.

Como é um processo supervisionado, o caminho o qual a RB levou durante o segundo teste e o resultado final, estarão, na maioria dos casos, cumprindo o desejado em questão de modificações na RB. Portanto, comparar o quanto os modelos se diferenciam um do outro e atribuir que um está certo ou errado, sem o levantamento desses apontamentos, não seria justo.

Como resultado final, para se ter uma ideia do quanto pequenos ajustes alteram o comportamento da RB em relação a novas inferências, a Figura 10 apresenta a mesma consulta realizada no novo modelo. É possível notar que pequenos ajustes em cada um dos nodos alteram a probabilidade a posteriori para este caso em aproximadamente 21%.

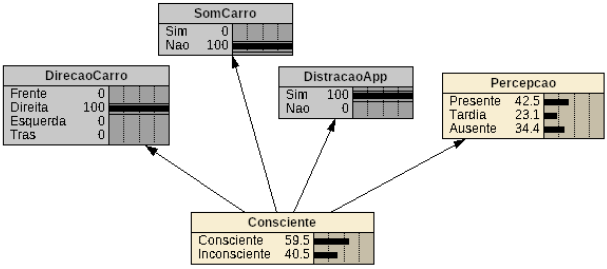

Figura 10. Inferência nova RB com evidência de DirecaoCarro, SomCarro e DistracaoApp.

5. Conclusão

A implementação do modelo bayesiano de atenção situacional, assim como a preparação e discretização dos dados levantados pelo experimento, permitiram a modelagem de uma rede bayesiana capaz de mensurar o nível de consciência e percepção de um usuário de smartphone trafegando próximo a vias urbanas. Tal qual, a análise e diagnóstico dos fatores mais influentes nos níveis de atenção.

As dificuldades enfrentadas durante o desenvolvimento e a integração da AR na aprendizagem bayesiana do modelo, se deram principalmente diante da dificuldade de se modelar as recompensas, ações e estados de um ambiente de RB utilizando os algoritmos mais conhecidos de aprendizagem bayesiana de parâmetros. No entanto, a definição de utilizá-lo como uma segunda etapa proporcionando o refinamento dos parâmetros, trouxe resultados satisfatórios.

Esse trabalho também mostra que a aprendizagem por reforço aplicada a sistemas especialistas cumpre seu papel como maneira de ajustá-los de forma a serem mais congruentes com o estudado sobre o domínio, mesmo na ausência de uma quantidade de dados significativa. Embora a aplicação proposta tenha suas limitações referentes ao local de ajuste e inserção manual de valor de recompensa, os resultados foram considerados satisfatórios para o domínio aplicado.

Referências

ENDSLEY, M. R. Measurement of situation awareness in dynamic systems. *Human factors*, SAGE Publications Sage CA: Los Angeles, CA, v. 37, n. 1, p. 65–84, 1995.

LUGER, G. F. *Inteligência Artificial*. 6. ed. [S.l.]: Pearson Education, 2013.

MITCHELL, T. M. *Machine learning*. [S.l.]: McGraw-Hill Science/Engineering/Math, 1997.

PEARL, J. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. [S.l.]: Morgan Kaufmann, 2014.

RUSSELL, S. J.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. 2. ed. [S.l.]: Pearson Education, 2003.

SUTTON, R. S.; BARTO, A. G. *Reinforcement learning: An introduction*. [S.l.]: MIT press Cambridge, 1998.

KAELBLING, L. P.; LITTMAN, M. L.; MOORE, A. W. Reinforcement learning: A survey. *Journal of artificial intelligence research*, v. 4, p. 237–285, 1996.