

Daniel Brito dos Santos

**GridLink e PingPOMDP: ferramentas para
explorar Inferência Ativa e agentes
computacionais a partir dos protocolos do
DishBrain**

Brasil

Novembro, 2023

Daniel Brito dos Santos

**GridLink e PingPOMDP: ferramentas para explorar
Inferência Ativa e agentes computacionais a partir dos
protocolos do DishBrain**

Monografia apresentada ao Curso de Graduação em Ciência da Computação da Universidade Estadual do Norte Fluminense Darcy Ribeiro como requisito para obtenção do título de Bacharel em Ciência da Computação, sob orientação da Profª. Drª Annabell Del Real Tamariz.

Universidade Estadual Do Norte Fluminense Darcy Ribeiro - UENF

Orientador: Annabell Del Real Tamariz

Brasil

Novembro, 2023

Esse trabalho é dedicado a todos que precisam saber urgentemente por que

Agradecimentos

À Dóris, minha eterna companheira de aventuras;
Aos meus pais que me deram curiosidade infinita, e amor incondicional para perseguí-la.
À toda minha família gigante e ciumenta que não posso citar nominalmente;
À tio Bebeto e tia Jô que me acolheram com todo carinho nesses quatro anos de dedicação.
Ao Cava que me apresentou a Inferência Ativa.
Aos amigos e colegas que encheram essa jornada de cor, especialmente João Vitor Fernandes Dias e José Lúcio (isso é uma piada sobre o gosto de João por sobrenomes).
Áo Tang, o incansável coordenador do nosso curso que sempre fez tudo o que podia, e mais um pouco.
Á Annabell que me acompanhou desde o primeiro ano, me dando todo apoio e liberdade.

*“Caminante, son tus huellas
el camino y nada más;
Caminante, no hay camino,
se hace camino al andar.
(Antonio Machado)*

Resumo

A Inferência Ativa (AIF) é uma teoria emergente da neurociência que busca elucidar a cognição e o comportamento de organismos com base em princípios fundamentais. Uma literatura crescente e diversa têm demonstrado sua aplicabilidade e potencial explicativo. Um exemplo notável é o sistema *DishBrain*, uma interface biotecnológica inédita que integra neurônios a um ambiente virtual do jogo Pong, por meio de protocolos derivados da AIF. Inspirados por esta inovação, criamos um sistema computacional com o objetivo de investigar se um modelo computacional simples, baseado na AIF, poderia apresentar um desempenho semelhante ao dos neurônios no *DishBrain*, que supostamente operam sob a mesma teoria, em situações de interação análogas. Para isso, desenvolvemos a interface GridLink agentes baseados em AIF e um ambiente Pong, bem como o sistema PingPOMDP para integrar e orquestrar esses componentes de modo a permitir a realização de experimentos de integração entre agentes e ambientes por meio dos protocolos propostos pelo *DishBrain*. Nos testes realizados, o agente demonstrou uma capacidade de aprendizado comparável à observada no *DishBrain* sob condições e parâmetros específicos. Assim, confirmamos nossa hipótese e sugerimos direções para aprofundar esse trabalho em desenvolvimentos futuros.

Palavras-chave: Inferência Ativa, *DishBrain*, Modelagem Computacional

Abstract

Active Inference (AIF) is an emerging theory of neuroscience that seeks to elucidate the cognition and behavior of organisms based on fundamental principles. A growing and diverse literature has demonstrated its applicability and explanatory potential. A notable example is the DishBrain system, an unprecedented biotechnological interface that successfully embeds neurons into a virtual environment of the Pong game, through protocols derived from AIF. Inspired by this innovation, we developed a computational system to investigate whether a simple AIF-based computational model could mimic the performance of neurons in the DishBrain, which are presumed to operate under the same theoretical framework, in analogous interactive situations. To achieve this goal, we created the GridLink interface, AIF-based agents, and a Pong environment, along with the PingPOMDP system to integrate and orchestrate these components. This setup enabled us to conduct experiments on the integration of agents and environments using protocols proposed by the DishBrain. Our tests, conducted under specific conditions and parameters, revealed that some agents exhibited a learning capability comparable to that observed in the DishBrain. Thus, we confirmed our hypothesis and suggested directions to deepen this work in future developments.

Keywords: Active Inference, DishBrain, Computational modeling

Listas de ilustrações

Figura 1 – Ciclo de ação-percepção	25
Figura 2 – Exemplo ilustrativo do chuveiro	31
Figura 3 – Diagrama conceitual dos componentes do PingPOMDP	40
Figura 4 – Tela do jogo Pong, 1972	41
Figura 5 – Ilustração do Pong no DishBrain	42
Figura 6 – Ilustração da nossa versão de Pong	42
Figura 7 – Diagrama de estados do Pong	44
Figura 8 – Neurônios sobre a HD-MEA interagindo com o Pong	45
Figura 9 – Esquema da HD-MEA utilizada no DishBrain	46
Figura 10 – Abstração da MEA por meio da GridLink	48
Figura 11 – Diagrama de sequência da Gridlink	49
Figura 12 – Modelo da GridLink utilizada no PingPOMDP	53
Figura 13 – Protocolos de Feedback da PongGridLink	55
Figura 14 – Representação geral dos principais componentes do PingPOMDP	61
Figura 15 – Modelo Arquitetural do PingPOMDP	61
Figura 16 – Diagrama de fluxo do PingPOMDP interagindo com seus componentes	62
Figura 17 – Diagrama de classes do PingPOMDP.	63
Figura 18 – Ilustração das condições experimentais do DishBrain	68
Figura 19 – Ilustração das condições experimentais das células corticais no DishBrain . .	69
Figura 20 – Total Partidas Longas de agentes AA1 até AA100	74
Figura 21 – Duração Média de Partida ao longo de 120.000 iterações	75
Figura 22 – Duração média de partida dentre agentes com semente 1	76
Figura 23 – Duração média de partida dentre agentes com semente 1	77
Figura 24 – Duração média de partida dentre agentes com semente 2	78
Figura 25 – Duração média de partida dentre agentes com semente 2	79
Figura 26 – Comparação de agentes com e sem feedback	80
Figura 27 – Comparação de agentes com e sem feedback	81
Figura 28 – Resultados das métricas do DishBrain comparando desenhos experimentais	82
Figura 29 – Resultados das métricas do PingPOMDP comparando desenhos experimentais	82

Sumário

1	INTRODUÇÃO	17
1.1	Problema	19
1.2	Hipótese	19
1.3	Objetivos	19
1.4	Justificativa	20
1.5	Escopo e Limitações	20
1.6	Métodos	20
1.7	Visão Geral do Trabalho	21
2	INFERÊNCIA ATIVA E SUA IMPLEMENTAÇÃO COMO POMDP	23
2.1	Inferência Ativa no Contexto de Inteligência Artificial	23
2.2	Introdução à Inferência Ativa	25
2.3	Postulados da Inferência Ativa	25
2.4	Formalização da Inferência Ativa como Processo de Decisão Markovi- ano Parcialmente Observável (POMDP)	27
2.4.1	Conceitos Básicos	28
2.4.2	Modelo Generativo	28
2.4.3	Inferência Bayesiana	28
2.4.4	Distribuição de Reconhecimento	30
2.4.5	Energia Livre Variacional	31
2.4.6	Energia Livre Esperada	32
2.5	Implementação Computacional da AIF	33
2.5.1	Especificação do Modelo Generativo	33
2.5.2	O Modelo pymdp	34
2.6	Trabalhos Relacionados	35
2.6.1	Fundamentos do PingPOMDP	35
2.6.2	Fundamentos da Inferência Ativa	36
2.6.3	Aplicações	36
2.7	Conclusões	37
3	MODELAGEM E IMPLEMENTAÇÃO DO SISTEMA PINGPOMDP	39
3.1	Ambiente Pong	40
3.1.1	Pong Clássico	40
3.1.2	Pong no DishBrain	41
3.1.3	Implementação do Ambiente Pong no PingPOMDP	42

3.1.3.1	Inicialização e Configuração	42
3.1.3.2	Dinâmica do Jogo	43
3.1.3.2.1	Detecção de Colisão	44
3.1.3.3	Lançamento da Bola	44
3.2	Interface GridLink	45
3.2.1	Interface e Protocolos do DishBrain	45
3.2.1.1	Protocolos de feedback	46
3.2.2	Implementação da GridLink	47
3.2.3	Modelagem da GridLink	47
3.2.3.1	Input e Feedback na GridLink	48
3.2.3.2	Estrutura Básica	50
3.2.3.3	Observações e Feedback	50
3.2.3.4	Atuação	51
3.2.3.5	Etapas de Interação	51
3.2.3.6	Funcionalidades Adicionais e Abstrações	52
3.2.4	GridLink no PingPOMDP	52
3.2.4.1	Modelagem da PongGridlink	53
3.2.4.2	Observações e Feedback	53
3.2.4.3	Conversão das ações do agente em ações no ambiente	55
3.3	Agente de Inferência Ativa	55
3.3.1	Estrutura do Agente de Inferência Ativa	56
3.3.2	Espaços de Observação e Ação	56
3.3.3	Modelo Generativo	57
3.3.3.1	Crenças e Preferências no PingPOMDP	57
3.3.4	Atualização de Crenças	59
3.3.5	Ação e Política	59
3.4	Sistema PingPOMDP	60
3.4.1	Estrutura do Código	63
3.4.1.1	PingPOMDP	63
3.4.1.2	PongGridlink	64
3.4.2	ActiveInferenceAgent	64
3.4.2.1	ControlAgent	64
3.4.2.2	Configurações	65
3.4.2.3	ExperimentDB	65
3.5	Desenho de Experimentos	67
3.5.1	Condições Experimentais do DishBrain	67
3.5.2	Configuração dos Agentes no PingPOMDP	69
3.5.3	Métricas do DishBrain	70
3.5.4	Métricas no PingPOMDP	70

3.5.4.1	Parâmetros Experimentais no PingPOMDP	70
3.5.4.2	Parâmetros do Agente	71
3.5.4.3	Parâmetros do Pong	71
3.5.4.4	Parâmetros da GridLink	71
4	RESULTADOS E DISCUSSÕES	73
4.1	Testes preliminares	73
4.1.1	Mapeamento de sementes aleatórias	73
4.1.2	Número de iterações	74
4.2	Resultados dos Experimentos com Agentes	75
4.2.1	Comparação entre agentes com semente aleatória 1	75
4.2.2	Comparação entre Performance dos Agentes com semente aleatória 2 . .	77
4.2.3	Comparação Entre Agentes Com e Sem Feedbak	79
4.2.4	Comparação com o DishBrain	81
4.3	Discussão de resultados	83
4.3.1	Influência da Estrutura do Modelo Generativo no Comportamento do Agente	83
5	CONSIDERAÇÕES FINAIS	85
5.1	Relevância e Contribuições	85
5.2	Limitações	86
5.3	Sugestões para Pesquisas Futuras	86
	REFERÊNCIAS	89

1 Introdução

Desde os filósofos da antiguidade clássica até os neurocientistas computacionais contemporâneos, a investigação sobre os mecanismos subjacentes à aprendizagem e inteligência tem permanecido no centro das indagações mais profundas da humanidade (THAGARD, 2023). À medida em que nos aprofundamos em um período marcado por revoluções tecnológicas e científicas sem precedentes, essa compreensão torna-se ainda mais pungente, não apenas para desvendar os segredos do cérebro humano (WANG et al., 2020), mas também para nos orientar no desenvolvimento de sistemas artificiais capazes de aprender e adaptar-se de maneira similar aos organismos vivos (SAVAGE, 2019).

No panorama atual de pesquisa sobre cognição e aprendizado, a **Inferência Ativa (AIF)** tem surgido como uma teoria promissora. A ideia central da AIF é de que agentes, sejam eles biológicos ou articiais, estão incessantemente empenhados em **minimizar a surpresa** ou imprevisibilidade dos estímulos sensoriais que recebem. De acordo com esse paradigma, organismos buscam criar um modelo interno do mundo que é constantemente atualizado para reduzir a discrepância entre suas previsões e suas verdadeiras observações (FRISTON, 2010). Essa minimização da surpresa não é apenas uma estratégia passiva de processamento de informações, mas um processo ativo que integra ação, cognição e aprendizado. Sob esta ótica, a essência do comportamento inteligente pode ser interpretada como uma propriedade emergente desse contínuo esforço para alinhar percepções com expectativas (TSCHANTZ et al., 2020a; COSTA et al., 2020; PARR; PEZZULO; FRISTON, 2022).

Prova do destaque da AIF é a extensão e diversidade de sua literatura. Segundo (HEINS et al., 2022a; COSTA et al., 2020; SMITH; FRISTON; WHYTE, 2021), a pesquisa acadêmica sobre as aplicações da Inferência Ativa está em franca expansão nos mais diversos campos, dentre eles podemos citar:

- Modelagem de comportamento humano e animal (ADAMS et al., 2021; HOLMES et al., 2021; PARR et al., 2020).
- Neurociência cognitiva, especialmente tomada de decisão sob incerteza (SCHWARTEN-BECK et al., 2015; SMITH et al., 2020; SMITH et al., 2021).
- Modelos computacionais de psicopatologia (MONTAGUE et al., 2012; SMITH et al., 2021) como a modelagem de esquizofrenia e a resposta a antipsicóticos (ADAMS et al., 2022).
- Control Theory, área da engenharia que se ocupa de algoritmos e controladores para sistemas dinâmicos, desde braços robóticos até telecomunicações e foguetes (BAIOUMY et al., 2022; BALTIERI; BUCKLEY, 2019; MILLIDGE et al., 2020).

- Aprendizado de máquina por reforço (FOUNTAS et al., 2020; MILLIDGE et al., 2020; SAJID et al., 2021; TSCHANTZ et al., 2020b; TSCHANTZ et al., 2020c).
- Cognição social (ADAMS et al., 2021; TISON; POIRIER, 2021; WIRKUTTIS; TANI, 2021).

Dentre as aplicações da AIF, Kagan et al. (2022) apresentou uma importante demonstração da Inferência Ativa como referencial teórico ao desenvolver o "**DishBrain**", uma interface biotecnológica inédita que integra neurônios *in vitro* a um ambiente virtual, utilizando uma grade de eletrodos e protocolos de estímulos fundamentados na AIF. Nesse sentido, **comportamentos indesejáveis** receberam **estímulos imprevisíveis**, enquanto **comportamentos desejáveis** foram reforçados por **estímulos previsíveis**. Assim, a expectativa desse protocolo é que ao minimizar a "surpresa", os neurônios **aprendam** a otimizar seus comportamentos.

Em seu experimento, Kagan et al. (2022) empregou uma versão do jogo "Pong" como ambiente virtual dos neurônios, representando a posição da bola como impulsos elétricos, lendo a atividade neural para controlar a raquete do jogo e administrando feedback de acordo com o sucesso ou fracasso no controle dessa raquete para rebater a bola. O resultado desse experimento foi que os neurônios demonstraram a capacidade de adaptar-se com sucesso à interface, alteraram a sua estrutura física, disposição na placa, e principalmente a sua atividade elétrica, para interpretar seus estímulos e organizar sua atividade de modo a controlar adequadamente a raquete, diminuir as perdas, e aumentar a sua pontuação. Os pesquisadores denominaram esta capacidade de resposta auto-regulada e orientada a objetivos como: "inteligência biológica sintética".

Assim esse experimento demonstrou que seu protocolo de estímulos baseado na AIF "significantly shapes the behavior of neural cultures in real time. (KAGAN et al., 2022)". Tanto que os neurônios apresentaram aprendizado notável em apenas cinco minutos de gameplay. Portanto, além de fortalecer a AIF, esse trabalho abre novas possibilidades de exploração de interfaces biotecnológicas.

Inspirado pelo trabalho de Kagan et al. (2022), o presente projeto tem como propósito explorar a seguinte questão: "Se substituirmos os neurônios, que teoricamente operam sob a AIF, por um modelo computacional genérico dessa mesma teoria, como esse modelo se comportaria em relação ao DishBrain?". Para esta investigação, desenvolvemos a "**GridLink**", uma ferramenta que facilita a integração de modelos computacionais a ambientes virtuais, utilizando abstrações da grade de eletrodos e dos protocolos de feedback do DishBrain. A intenção é estabelecer uma estrutura que permita responder nossa pergunta de pesquisa, mas também possa ser utilizada em pesquisas futuras que explorem interações agente-ambiente fundamentadas nos protocolos propostos por Kagan et al. (2022).

Complementando a **GridLink**, implementamos também um ambiente Pong conforme as especificações de Kagan et al. (2022) e um modelo computacional de AIF para interação via

GridLink. Juntos, estes elementos compõem o "**PingPOMDP**", uma tentativa de modelar, de forma **computacional e simplificada**, o sistema bio-sintético **DishBrain**.

Vale ressaltar que, enquanto os neurônios são extremamente eficientes, os modelos de AIF são notoriamente intensivos em termos computacionais ([TSCHANTZ et al., 2020a](#)). Assim, é razoável esperar um **desempenho limitado** do modelo computacional. Entretanto, mensurar essa diferença e estabelecer os fundamentos para comparações futuras tem o potencial de enriquecer este campo de pesquisa em ascensão.

1.1 Problema

A pesquisa de [Kagan et al. \(2022\)](#) abriu uma nova janela para entendermos como os neurônios *in vitro* podem aprender e adaptar-se a um ambiente simulado. No entanto, a literatura atual não oferece uma biblioteca ou sistema que integre um modelo de AIF com uma versão do jogo Pong, que emule a grade de eletrodos e os protocolos de feedback utilizados no DishBrain. Portanto, este projeto visa preencher essa lacuna, desenvolvendo e testando um sistema que permita investigar o potencial de um modelo de AIF para alcançar resultados similares nesse contexto de ambiente.

1.2 Hipótese

A hipótese central deste trabalho é de que um modelo computacional de AIF, quando integrado a um sistema inspirado no DishBrain, exibirá aprendizado e performance comparáveis, embora possivelmente limitados, em um ambiente de Pong semelhante. Esta hipótese será avaliada através da mensuração e comparação de métricas específicas documentados por [Kagan et al. \(2022\)](#) como: duração média de partidas, número de partidas com mais do que três acertos, e número de partidas com zero acertos.

1.3 Objetivos

O propósito principal deste projeto é investigar se um sistema computacional, modelado a partir do DishBrain e operando sob os princípios da AIF, pode manifestar um aprendizado análogo, mesmo que em uma escala possivelmente reduzida. Conforme expresso em nossa hipótese de trabalho.

Os objetivos específicos incluem:

- **Desenvolvimento de Interface:** criar uma interface computacional, denominada GridLink, que permita a comunicação entre um modelo de AIF e uma versão modificada do jogo Pong, emulando as especificações e protocolos do DishBrain.

- **Avaliação de Desempenho:** medir e analisar o aprendizado e desempenho do sistema, utilizando métricas como taxa de acerto, taxa de partidas sem nenhum acerto e número de partidas com três ou mais acertos.
- **Comparação de Resultados:** confrontar os dados obtidos com os resultados apresentados por [Kagan et al. \(2022\)](#) utilizando visualizações gráficas.

1.4 Justificativa

A justificativa para este projeto encontra-se na importância crescente da Inferência Ativa (AIF) como uma teoria para explicar aspectos da cognição e comportamento, tanto em seres biológicos quanto em sistemas artificiais. A inovação do DishBrain, demonstrando que neurônios *in vitro* podem aprender e se adaptar em um ambiente virtual, marca um avanço significativo, indicando que os princípios da AIF podem ser aplicados na criação de sistemas inteligentes e adaptáveis. Este estudo busca ampliar essa área de pesquisa, investigando se um modelo computacional baseado na AIF pode replicar o aprendizado e a adaptabilidade observados nos neurônios *in vitro* no DishBrain. Ao realizar essa investigação, o projeto não apenas testa a aplicabilidade dos protocolos do DishBrain em um contexto computacional, mas também avança na pesquisa ao desenvolver e disponibilizar as ferramentas GridLink e PingPOMDP. Estas ferramentas permitem a integração de agentes computacionais a ambientes virtuais utilizando os protocolos do DishBrain. Essencialmente, este trabalho responde à proposta inovadora de [Kagan et al. \(2022\)](#) de um novo paradigma para induzir aprendizado em sistemas que operam sob a AIF, oferecendo um modelo computacional dessa abordagem.

1.5 Escopo e Limitações

Este projeto se concentra na implementação de uma versão do jogo Pong e de um modelo simplificado de AIF, além de uma biblioteca que permita a interação entre ambos, de acordo com os protocolos especificados no DishBrain. Não se pretende replicar todos os detalhes e nuances de [Kagan et al. \(2022\)](#), nem explorar cenários mais complexos e realísticos.

1.6 Métodos

Este estudo adota uma abordagem metodológica multifacetada, caracterizada por ser uma *pesquisa aplicada* com foco na resolução de um problema prático específico: testar a aplicabilidade dos protocolos do DishBrain em um contexto computacional e desenvolver um modelo computacional que emule o aprendizado observado nos seus neurônios *in vitro*. Utilizamos uma *abordagem quantitativa*, coletando e analisando dados numéricos para avaliar o desempenho do modelo computacional. O método *experimental* é central para nossa pesquisa,

permitindo a manipulação de variáveis e a observação dos efeitos dessas manipulações no comportamento do modelo. Além disso, elementos de *pesquisa exploratória e descritiva* são incorporados para investigar e documentar as características do modelo sob diferentes condições experimentais. Finalmente, a *modelagem computacional* e o *desenvolvimento de software* são os componentes fundamentais que permitiram o desenvolvimento dessa pesquisa.

Assim, enumeramos a seguir os passos adotados para a realização desse trabalho:

1. **Desenvolvimento do ambiente Pong:** implementar uma versão do jogo Pong seguindo as descrições e especificações fornecidas por [Kagan et al. \(2022\)](#). Utilizando a linguagem de programação Python ([ROSSUM; DRAKE, 2009](#)) e assegurando flexibilidade nos parâmetros como velocidade e tamanho da bola e raquete, tamanho da tela, algoritmo de lançamento da bola, etc.
2. **Criação da Interface GridLink:** desenvolver uma interface computacional modelada a partir da grade de eletrodos e protocolos do sistema DishBrain, denominada *GridLink*. Esta interface permitirá comunicação entre o ambiente Pong e o modelo de AIF.
3. **Implementação do agente de AIF:** implementar um modelo de AIF utilizando a biblioteca pymdp ([HEINS et al., 2022a](#)). Esse modelo representa crenças sobre estados e ações de jogo, bem como um algoritmo de inferência que atualiza essas crenças com base nos inputs sensoriais recebidos.
4. **Experimentação e Coleta de dados:** conduzir experimentos variando parâmetros do agente e do ambiente para observar seu impacto no aprendizado e desempenho do sistema. Coletar métricas para validação da hipótese, conforme [seção 1.2](#).
5. **Análise de Dados e Comparaçāo de Resultados:** utilizar estatísticas descritivas como média e desvio padrão para analisar os dados obtidos. Comparar nossos dados com os resultados documentados por [Kagan et al. \(2022\)](#).
6. **Discussāo de Resultados:** avaliar os resultados à luz da hipótese e objetivos do projeto. Contrastar o desempenho do modelo de AIF com as culturas de neurônios do DishBrain, destacando semelhanças, diferenças e possíveis desdobramentos.

1.7 Visão Geral do Trabalho

Neste trabalho, nos inspiramos na pesquisa pioneira de [Kagan et al. \(2022\)](#) e sua demonstração prática da validade da AIF por meio do sistema "DishBrain". Diante da clareza com que os neurônios *in vitro* aprenderam e adaptaram-se em uma interface biotecnológica, propomos investigar **como** um modelo computacional, baseado nos mesmos princípios da AIF, se comportaria se submetido a condições análogas.

O documento está estruturado da seguinte forma:

- **Capítulo 2: Inferência Ativa e sua Implementação como POMDP.** Revisão da literatura sobre a teoria da AIF, seus princípios fundamentais e aplicações em estudos anteriores, com ênfase especial na formalização e modelagem de sistemas biológicos e artificiais. Este capítulo detalha o referencial teórico utilizado neste trabalho.
- **Capítulo 3: Modelagem e implementação do sistema PingPOMDP.** Descrição detalhada dos métodos e modelos utilizados na criação e desenvolvimento do sistema PingPOMDP e seus componentes, abrangendo a implementação do ambiente virtual Pong, a criação da interface GridLink, o desenvolvimento do modelo de AIF, e o desenho do PingPOMDP como plataforma experimental.
- **Capítulo 4: Resultados e Discussão.** Apresentação e interpretação dos resultados experimentais. Os achados são comparados com os dados de [Kagan et al. \(2022\)](#) e discutidos à luz da teoria e referencial teórico apresentado.
- **Capítulo 5: Considerações Finais.** Síntese dos principais achados e conclusões do estudo, destacando as contribuições para os campos de pesquisa que o fundamentam. Aborda também as limitações encontradas e sugere direções para futuras pesquisas, explorando potenciais extensões do trabalho apresentado.

2 Inferência Ativa e sua Implementação como POMDP

A busca pelo entendimento dos mecanismos de aprendizagem e inteligência tem sido uma constante ao longo da história humana. A teoria da Inferência Ativa (AIF), proposta originalmente por Friston, Daunizeau e Kiebel (2009), tem emergido no atual panorama como uma teoria promissora para explicar a cognição e comportamento, sustentando que agentes buscam ativamente minimizar a surpresa de seus estímulos sensoriais (FRISTON, 2010; PARR; PEZZULO; FRISTON, 2022; SMITH; FRISTON; WHYTE, 2021; TSCHANTZ et al., 2020a; COSTA et al., 2020).

O trabalho de Kagan et al. (2022) reforça o impacto e potencial dessa teoria ao estabelecer um protocolo inovador de aprendizado para neurônios *in vitro* inspirado nos princípios da AIF.

No entanto, para compreender e aplicar a AIF no sentido de atingir os objetivos desse trabalho, é imperativo entender seus fundamentos teóricos e como ela se relaciona com modelos computacionais. Neste capítulo, apresentamos a teoria da AIF e discutimos sua formalização, que se dá através do paradigma POMDP (Processos de Decisão de Markov Parcialmente Observáveis). Esta discussão não apenas aprofundará nossa compreensão sobre a AIF e seu potencial, mas também fornecerá o alicerce teórico para a experimentação proposta neste trabalho.

2.1 Inferência Ativa no Contexto de Inteligência Artificial

A emergência da Inferência Ativa (AIF) como uma teoria promissora na neurociência oferece uma nova perspectiva no campo da inteligência artificial (IA). Enquanto a IA tem sido dominada por abordagens como as Redes Neurais (NNs), a AIF propõe um modelo alternativo baseado em princípios fundamentais de cognição e comportamento (FRISTON; DAUNIZEAU; KIEBEL, 2009; SCHWARTENBECK et al., 2013; FRISTON; SAMOTHRAKIS; MONTAGUE, 2012; FRISTON et al., 2015; FRISTON; KILNER; HARRISON, 2006; FRISTON, 2010; BOGACZ, 2017).

A evolução da IA, especialmente desde o início da década de 2010, tem sido marcada por avanços significativos em aprendizado de máquina, processamento de linguagem natural e visão computacional, impulsionados em grande parte pela aprendizagem profunda e NNs. Este período de transformação, detalhado por Russell, Russell e Norvig (2020), destaca a transição do conhecimento artesanal para algoritmos de aprendizado baseados em dados e recursos computacionais avançados.

O marco do modelo AlexNet de Krizhevsky, Sutskever e Hinton (2012) no desafio ILSVRC

ilustra o potencial das NNs em tarefas complexas de reconhecimento de imagens, reacendendo o interesse em aprendizado profundo (RUSSELL; RUSSELL; NORVIG, 2020; CHAWLA et al., 2023). Desde então, modelos como GPT da OpenAI têm demonstrado capacidades notáveis em compreensão e geração de linguagem natural, ampliando ainda mais os horizontes da IA (FLORIDI; CHIRIATTI, 2020; JOHNSON, 2022).

Dessa forma, ao abordar inteligência artificial, NNs surgem como uma referência natural. Além disso, o DishBrain envolve uma Rede Neural Biológica (BNN), composta por neurônios interconectados, e o trabalho de Kagan et al. (2022) representa um avanço significativo na pesquisa sobre BNNs e o potencial de sistemas bio-sintéticos que as utilizam. Sua grande contribuição foi propor e demonstrar **protocolos** capazes de integrar neurônios *in vitro* (BNNs) em um ambiente virtual.

Estes protocolos baseiam-se no conceito de Inferência Ativa, que difere das NNs tradicionais. Enquanto a Inferência Ativa é primordialmente uma teoria normativa fornecendo princípios para formalizar como organismos processam informações e interagem com o ambiente, enfatizando a minimização da surpresa como um princípio central (Da Costa et al., 2020), as NNs representam uma arquitetura computacional específica utilizada em aprendizado de máquina. Nas NNs, os valores que representam a ativação de neurônios são interligados por pesos que simulam sinapses para gerar uma saída. O aprendizado ocorre através do ajuste desses pesos, otimizando a saída da rede para atingir um objetivo específico, como, por exemplo, realizar a tarefa de classificar uma imagem ou continuar uma sentença textual (Russell e Norvig, 2020).

Esses protocolos são fundamentados pela **Inferência Ativa**, que se difere das NNs e outras abordagens tradicionais da IA. Enquanto a Inferência Ativa é primordialmente uma teoria normativa fornecendo princípios para formalizar como organismos processam informações e interagem com o ambiente, enfatizando a minimização da surpresa como um princípio central (COSTA et al., 2020), as NNs são uma arquitetura computacional específica utilizada em aprendizado de máquina. Nas NNs, os valores que representam a ativação de neurônios são interligados por pesos que simulam sinapses para gerar uma saída. O aprendizado ocorre através do ajuste desses pesos, otimizando a saída da rede para atingir um objetivo específico, como, por exemplo, realizar a tarefa de classificar uma imagem ou continuar uma sentença textual (RUSSELL; RUSSELL; NORVIG, 2020).

Dessa forma, embora existam indícios de que as NNs possam realizar uma forma de AIF (ISOMURA; SHIMAZAKI; FRISTON, 2022), este trabalho foca especificamente na AIF como um arcabouço teórico para entender e modelar cognição e aprendizado no contexto do DishBrain. A relação entre AIF e NNs, embora intrigante, está além do escopo deste trabalho e representa uma área promissora para pesquisas futuras.

Portanto, a seção seguinte, [seção 2.2](#), aprofundará na teoria da AIF, estabelecendo a base para nossa investigação e experimentação.

2.2 Introdução à Inferência Ativa

Todo organismo precisa interagir com seu ambiente. Isso ocorre por meio de um ciclo contínuo de ação-percepção no qual o organismo **observa** o seu ambiente e **age** de modo a modificá-lo, conforme ilustrado na Figura 1 (PARR; PEZZULO; FRISTON, 2022).

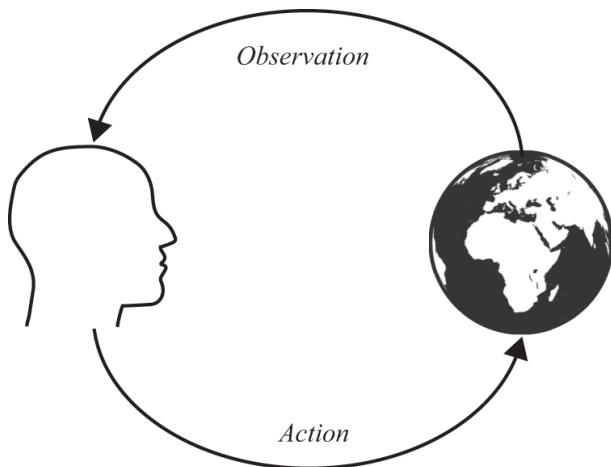


Figura 1 – Ciclo de ação-percepção

Fonte: Parr, Pezzulo e Friston (2022)

Esse ambiente, entretanto, apresenta desafios formidáveis a continuidade de sua existência. Alimentação, hidratação, proteção de predadores, manutenção da integridade física e homeostase são apenas alguns exemplos. Além disso, cada ação gera uma cadeia recursiva de consequências, de modo que existe um vasto campo de ações possíveis no qual apenas um pequeno conjunto de ações são benéficas ao organismo (SMITH; FRISTON; WHYTE, 2021; TSCHANTZ et al., 2020a).

Desse modo, a única forma de um organismo permanecer existindo é se ele adquirir alguma forma de **controle adaptativo** de seu ciclo de ação-percepção. Nesse sentido, a AIF é um **princípio normativo** que propõem postulados para solucionar esse problema de controle adaptativo. Ou seja, AIF propõem postulados sobre **o que** organismos devem fazer para continuar existindo, e oferece explicações do **porquê** tais postulados o ajudariam nesse objetivo. Além disso, a inferência ativa também oferece uma **teoria de processo** para explicar mecanisticamente **como** organismos podem implementar os postulados normativos (PARR; PEZZULO; FRISTON, 2022; COSTA et al., 2020).

2.3 Postulados da Inferência Ativa

A ideia fundamental da AIF é que mesmo com as suas variadas manifestações, os elementos centrais do comportamento, cognição, e adaptação de qualquer organismo podem ser compreendidos e explicados por uma única teoria coesa construída a partir de princípios

fundamentais. Segundo a AIF, todos esses elementos são propriedades emergentes de um processo de **inferência** no qual agentes buscam **ativamente** minimizar a "surpresa"(erro de predição) de cada nova percepção sensorial, tanto por meio de ações quanto por meio da atualização de suas crenças. (PARR; PEZZULO; FRISTON, 2022; COSTA et al., 2020).

Para tanto, a AIF propõem os seguintes postulados:

- Agentes mantém um **modelo probabilístico** de mundo representando crenças probabilísticas sobre a dinâmica de estados do seu ambiente, e como tais estados elicitam diferentes percepções sensoriais no agente. Esse modelo é chamado de **generativo** porque permite ao agente gerar previsões quanto às sensações esperadas (COSTA et al., 2020; TSCHANTZ et al., 2020a; SMITH; FRISTON; WHYTE, 2021).
- Todas as facetas de comportamento e cognição de qualquer organismo seguem um único imperativo: **minimizar a surpresa** das suas observações sensoriais. Surpresa nesse contexto se refere ao erro de predição, a discrepância entre a observação obtida e a observação esperada segundo o modelo generativo do organismo (COSTA et al., 2020; PARR; PEZZULO; FRISTON, 2022; SMITH; FRISTON; WHYTE, 2021).
- A surpresa não pode ser minimizada diretamente nem passivamente, apenas **atualizando as crenças** para se conformarem às observações ou **agindo** para conformar o ambiente às crenças (PARR; PEZZULO; FRISTON, 2022; COSTA et al., 2020; SMITH; FRISTON; WHYTE, 2021). Dessa forma, agentes **minimizam a surpresa** por meio da otimização de duas funções objetivo complementares (COSTA et al., 2020):
 - **Energia livre variacional (VFE)**, que mede a adequação entre um modelo interno de mundo e as suas observações sensoriais.
 - **Energia livre esperada (EFE)**, que classifica possíveis cursos de ação em relação a preferências do agente.
- Fundamentalmente, um modelo de AIF prediz **o que observará** caso aja de uma forma ou de outra. Assim, esses modelos escolhem ações que esperam **gerar observações** de maior preferência, ou que trarão maior quantidade de novas informações. Portanto, as **preferências** e objetivos do agente são representadas como **priors** no modelo generativo do agente. Isto é, **observações preferidas são assumidas como mais prováveis**. Esse viés de otimismo torna resultados não preferidos como literal e tecnicamente “surpreendentes”, ou seja, desviantes dos **priors**. (COSTA et al., 2020; PARR; PEZZULO; FRISTON, 2022; HEINS et al., 2022a; SMITH; BADCOCK; FRISTON, 2021; SMITH; FRISTON; WHYTE, 2021).

Desse modo, se eu estou com fome é mais provável que eu procure comida do que fique parado, precisamente porque estar com fome é inconsistente com minhas preferências

(ou seja, estatisticamente surpreendente), e eu espero que comer produza sentimentos de saciedade (minimizando a surpresa) (SMITH; BADCOCK; FRISTON, 2021; SMITH; FRISTON; WHYTE, 2021). Em outras palavras, uma preferência é simplesmente algo que um agente (acredita que) provavelmente buscará (COSTA et al., 2020).

- AIF pode, desse modo, ser compreendida como a **minimização da surpresa** por meio da **percepção e ação** com o objetivo de realizar suas preferências e portanto minimizar a surpresa que esperam encontrar no futuro (COSTA et al., 2020).

Desse modo, podemos formular o **ciclo de percepção-ação** da seguinte forma:

1. Agentes **percebem o mundo** por meio da minimização da energia livre variacional (**VFE**), garantindo que seu modelo é consistente com suas observações passadas.
2. Agentes **agem** minimizando a energia livre esperada (**EFE**), para tornar as **observações futuras** consistentes com o seu modelo.
3. Essa conceituação de comportamento pode ser resumidamente descrita como "**auto-evidenciamento**"(COSTA et al., 2020).

Assim, assumindo as premissas apresentadas, é possível descrever o comportamento de organismos viáveis como performando AIF. Resta apenas o desafio de determinar os processos computacionais e fisiológicos que a implementam (COSTA et al., 2020).

2.4 Formalização da Inferência Ativa como Processo de Decisão Markoviano Parcialmente Observável (POMDP)

Conforme explorado na [seção 2.3](#), a AIF fundamenta-se na minimização da **Energia Livre Variacional (VFE)** para percepção e aprendizagem, e da **Energia Livre Esperada (EFE)** para seleção de ações, planejamento e tomada de decisões. A EFE avalia a VFE de diversas ações com base em resultados futuros projetados.

Neste contexto, a **AIF** é estruturada como um Processo de Decisão de Markov Parcialmente Observável (**POMDP**), um **modelo gerativo** específico amplamente reconhecido na literatura. Um POMDP descreve **crenças** sobre estados abstratos do mundo, suas mudanças esperadas ao longo do tempo, e como as ações são selecionadas para buscar resultados preferenciais, com base nessas crenças. Esta classe de modelos adota a "propriedade de Markov", significando que para um agente, as crenças sobre o estado atual do mundo são tudo o que importa ao decidir quais ações tomar, assumindo que todo conhecimento sobre estados passados está implicitamente contido nas crenças sobre o estado atual (TSCHANTZ et al., 2020a; SMITH; FRISTON; WHYTE, 2021; COSTA et al., 2020; HEINS et al., 2022a).

O agente utiliza seu modelo, em conjunto com as crenças sobre o estado atual, para selecionar ações, fazendo previsões sobre possíveis estados futuros. Ser "parcialmente observável"

vel" implica que o agente pode ter incertezas sobre seu estado no mundo. Nesse caso, os estados são denominados "ocultos". O agente deve inferir a probabilidade de estar em um ou outro estado oculto com base em observações (ou seja, entrada sensorial) e usar essas informações para selecionar ações. Na AIF, essas tarefas são resolvidas por meio de uma forma de inferência aproximada conhecida como inferência variacional, que consiste em converter a soma ou integral intratável, necessária para a inversão do modelo, em um problema de otimização resolvível de maneira computacionalmente eficiente (TSCHANTZ et al., 2020a; SMITH; FRISTON; WHYTE, 2021; COSTA et al., 2020; HEINS et al., 2022a).

Nesta seção, detalharemos esses conceitos fundamentais, incluindo o modelo gerativo, inferência Bayesiana, Distribuição de Reconhecimento e as estratégias para minimizar a Energia Livre Esperada.

2.4.1 Conceitos Básicos

- **Estado do Ambiente:** a cada instante t , o **verdadeiro estado** do ambiente $\hat{s}_t \in \mathbb{R}^{d_s}$ evolui de acordo com a dinâmica de transição estocástica $\hat{s}_t \sim P(\hat{s}_t | \hat{s}_{t-1}, a_{t-1})$ onde $a \in \mathbb{R}^{d_a}$ denota as **ações** do agente.
- **Observações:** agentes não têm acesso ao verdadeiro **estado do ambiente** s_t , apenas observações $o_t \in \mathbb{R}^{d_o}$ geradas a partir do estado real do ambiente: $o_t \sim p(o_t | \hat{s}_t)$.
- **Crenças dos Agentes.** Portanto, a partir das observações, os agentes operam com **crenças** $s_t \in \mathbb{R}^{d_s}$ sobre o verdadeiro estado do ambiente \hat{s}_t . Essas crenças são chamadas de "**estados ocultos**" pois dizem respeito a variáveis que o agente não pode observar diretamente mas acredita que existam.

2.4.2 Modelo Generativo

A partir desses conceitos básicos, a AIF propõe que agentes implementam e atualizam um **modelo gerativo** do seu ambiente, com crenças probabilísticas sobre a dinâmica de estados do seu ambiente, e como tais estados elicitam diferentes percepções sensoriais no agente. Esse modelo é uma função de distribuição de probabilidade conjunta representada por $p(\tilde{o}, \tilde{s}, \pi, \theta)$, que especifica como **estados ocultos** causam consequências sensoriais (**observações**). A função representa a probabilidade de observações o , estados ocultos s , política π e parâmetros θ . Nessa notação, o til denota uma sequência de variáveis no tempo, π denota uma política (sequência de ações) $\pi = \{a_0, \dots, a_T\}$ e $\theta \in \Theta$ denota os parâmetros do modelo gerativo.

2.4.3 Inferência Bayesiana

A partir dessa formalização, a cada nova observação, um agente pode determinar em qual estado o seu ambiente se encontra, e qual seria a melhor ação a ser tomada. Isto é, determinar

o estado oculto e sua ação correspondente, de acordo com determinada observação. Portanto, bastaria "inverter" o seu modelo generativo calculando a posterior: $p(\tilde{s}, \pi, \theta | \tilde{o})$.

Nesse sentido, a **inferência Bayesiana** representa a maneira ótima de inferir tais crenças *a posteriori* dentro de um modelo generativo. Essa inferência é um procedimento estatístico que descreve como atualizar as crenças (entendidas como distribuições de probabilidade) ao fazer novas observações (ou seja, receber novas entradas sensoriais), com base nas regras da probabilidade. Especificamente, as crenças são atualizadas à luz de novas observações usando o **Teorema de Bayes**, que pode ser escrito da seguinte forma:

$$p(s|o, m) = \frac{p(o|s, m) \cdot p(s|m)}{p(o|m)} \quad (2.1)$$

Onde:

- $p(s|m)$ indica a probabilidade de diferentes estados possíveis s sob um modelo do mundo m , conhecida como crença *a priori*, ou prévia antes de fazer uma nova observação o .
- $p(o|s, m)$ é o termo de *likelihood* (verossimilhança), que codifica a probabilidade dentro de um modelo m de que uma observação particular o seria feita caso o estado s fosse o estado verdadeiro.
- $p(o|m)$ é a *evidência do modelo* e indica quão consistente uma observação é com um modelo do mundo em geral.
- Finalmente, $p(s|o, m)$ é a crença *a posteriori*, que codifica qual deveria ser a nova crença após fazer uma nova observação.

Em essência, a regra de Bayes descreve como atualizar de forma ótima as crenças à luz de novos dados. Para chegar a uma nova crença (sua *posterior*), você deve: (1) considerar o que você acreditava previamente (sua *prior*); (2) combiná-la com o que você acredita sobre a consistência de uma nova observação com diferentes estados possíveis (sua *likelihood*); e (3) considerar a consistência geral dessa observação com seu modelo.

Porém, o Teorema de Bayes é **computacionalmente intratável** para distribuições que não sejam as mais simples. Principalmente devido à necessidade de calcular o termo $p(o|m)$ (evidência do modelo), que exige a soma das probabilidades de observações para **todos os estados possíveis**. À medida que o número de dimensões (e possíveis valores) dos estados aumenta, o número de termos que devem ser somados cresce exponencialmente. Dada essa complexidade, técnicas de aproximação, como a Inferência Variacional, são empregadas (SMITH; FRISTON; WHYTE, 2021).

2.4.4 Distribuição de Reconhecimento

Nesse contexto, a Inferência Variacional é utilizada para aproximar a distribuição *a posteriori* verdadeira com uma distribuição mais simples e tratável. Essa distribuição, é denominada "**distribuição de reconhecimento**" e é caracterizada por $q(\hat{s}, \pi, \theta)$. Esta distribuição representa as crenças do agente sobre estados ocultos \hat{s} , políticas π e parâmetros do modelo $\theta \in \Theta$ (TSCHANTZ et al., 2020a).

Dessa forma, além do modelo generativo, os agentes mantêm uma distribuição de reconhecimento que reflete suas crenças atuais sobre os estados ocultos do mundo, dadas suas observações. A AIF, portanto, não apenas oferece uma solução normativa para o problema de controle adaptativo, mas também uma solução variacional para aproximar a inferência bayesiana dos agentes (PARR; PEZZULO; FRISTON, 2022).

Um exemplo ilustrativo seria um agente em um chuveiro elétrico, conforme ilustrado na Figura 2. Como estamos tratando sobre ambientes parcialmente observáveis, o estado real do chuveiro é inacessível ao agente. Porém, o agente acredita (corretamente) que existem três estados possíveis do seu ambiente (os três estados ocultos: Inverno, Desliga e Verão). Nesse caso, o **modelo generativo** do agente representa crenças sobre como os estados do chuveiro levam a diferentes temperaturas da água, como por exemplo:

- estado oculto: **Inverno** -> a sensação: **Quente** | p=0.8
- estado oculto: **Inverno** -> a sensação: **Gelado** | p=0.1
- estado oculto: **Inverno** -> a sensação: **Morno** | p=0.1
- estado oculto: **Desliga** -> a sensação: **Quente** | p=0.1
- ...
- estado oculto: **Verão** -> a sensação: **Morno** | p=0.8

A **distribuição de reconhecimento**, por outro lado, apresenta crenças sobre os estados possíveis a partir das observações, como por exemplo, ao sentir água gelada, qual a probabilidade do estado oculto estar na configuração "desligado".

- sensação: Quente -> estado oculto: Inverno | p=0.6
- sensação: Quente -> estado oculto: Desliga | p=0.2
- sensação: Quente -> estado oculto: Versão | p=0.2
- ...
- sensação: Morno -> estado oculto: Verão | p=0.6

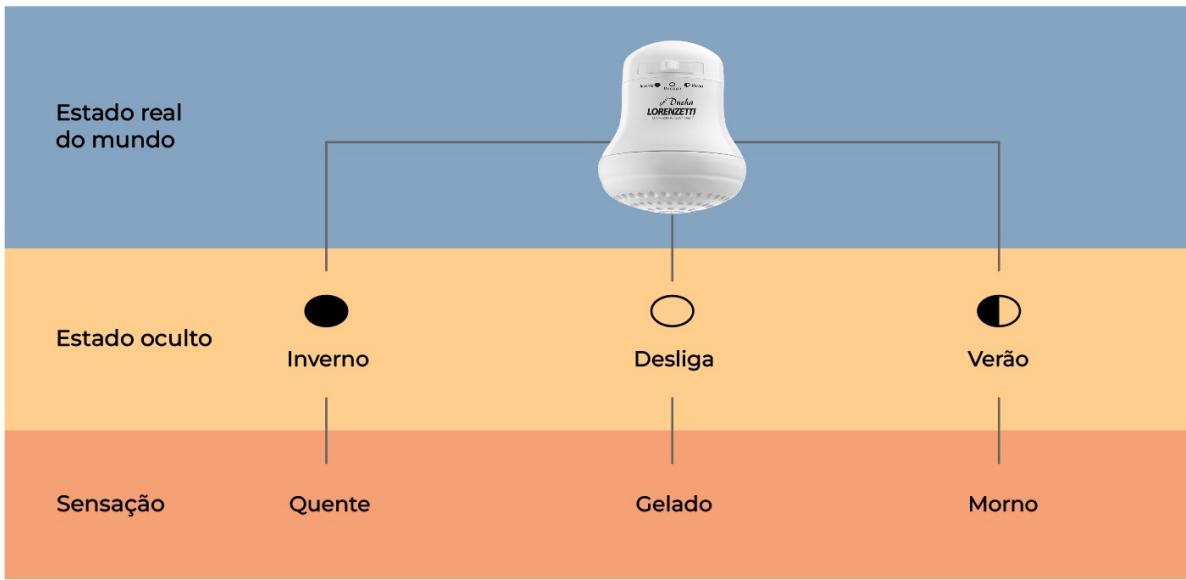


Figura 2 – Exemplo ilustrativo do chuveiro

Fonte: própria

2.4.5 Energia Livre Variacional

Conforme novas observações são coletadas, os agentes atualizam os parâmetros de sua distribuição de reconhecimento para minimizar a Energia Livre Variacional (VFE), denotada por \mathcal{F} (COSTA et al., 2020; TSCHANTZ et al., 2020a):

$$\mathcal{F}(\hat{o}) = \mathbb{E}_{q(\hat{s}, \pi, \theta)} [\ln q(\hat{s}, \pi, \theta) - \ln p(\tilde{o}, \tilde{s}, \pi, \theta)] \quad (2.2)$$

Onde:

- \hat{o} representa as observações sensoriais recebidas pelo agente.
- $q(\hat{s}, \pi, \theta)$ é a distribuição de reconhecimento, uma distribuição probabilística que representa as crenças do agente sobre os estados ocultos \hat{s} , políticas π e parâmetros θ .
- $\mathbb{E}_{q(\hat{s}, \pi, \theta)}$ denota o valor esperado sob a distribuição de reconhecimento.
- $\ln q(\hat{s}, \pi, \theta)$ e $\ln p(\tilde{o}, \tilde{s}, \pi, \theta)$ são os logaritmos naturais das probabilidades atribuídas pela distribuição de reconhecimento e pelo modelo gerativo, respectivamente.
- $p(\tilde{o}, \tilde{s}, \pi, \theta)$ é o modelo gerativo, que expressa a probabilidade dos dados observados \tilde{o} e dos estados ocultos \tilde{s} dados as políticas π e parâmetros θ .

A VFE pode ser interpretada como uma medida de surpresa: ela quantifica o quanto inesperadas são as observações dadas o modelo atual do agente. Ao minimizar a VFE, o agente procura a melhor explicação para suas observações, ajustando suas crenças (distribuição de reconhecimento) para melhor se alinhar com os dados recebidos. Este processo é fundamental para o aprendizado e adaptação do agente em um ambiente dinâmico.

Dessa forma, minimizar \mathcal{F} faz com que a distribuição de reconhecimento $q(\hat{s}, \pi, \theta)$ converja para uma aproximação da distribuição *a posteriori* intratável $p(\tilde{s}, \pi, \theta | \tilde{o})$, realizando assim a inferência Bayesiana aproximada. Desse modo, energia livre atua como uma *proxy* para quanto surpreendente, ou improvável cada observação é, de acordo com o modelo do agente, visto que ela é justamente a medida dessa discrepância.

2.4.6 Energia Livre Esperada

Para minimizar efetivamente a surpresa de suas observações, o agente deve influenciar essas observações por meio de ações.

Neste contexto, a AIF sugere que agentes escolham *policies* (políticas) para minimizar a energia livre esperada (EFE) \mathcal{G} (COSTA et al., 2020). A EFE de uma determinada *policy* π em um dado momento τ é dada por:

$$\mathcal{G}(\pi, \tau) = \mathbb{E}_{q(o_\tau, s_\tau, \theta | \pi)} [\ln q(s_\tau, \theta | \pi) - \ln p(o_\tau, s_\tau, \theta | \pi)] \quad (2.3)$$

Onde:

- $\mathcal{G}(\pi, \tau)$ representa a EFE associada à política π no momento τ .
- $q(o_\tau, s_\tau, \theta | \pi)$ é a distribuição de reconhecimento condicional, indicando as crenças do agente sobre as observações o_τ , estados ocultos s_τ e parâmetros θ , dado que a política π é seguida.
- $\mathbb{E}_{q(o_\tau, s_\tau, \theta | \pi)}$ denota o valor esperado sob a distribuição de reconhecimento condicional.
- $\ln q(s_\tau, \theta | \pi)$ e $\ln p(o_\tau, s_\tau, \theta | \pi)$ são os logaritmos naturais das probabilidades segundo a distribuição de reconhecimento e o modelo gerativo, respectivamente.
- $p(o_\tau, s_\tau, \theta | \pi)$ é o modelo gerativo condicional, que calcula a probabilidade das observações o_τ e estados ocultos s_τ , dado os parâmetros θ e a política π .

Ao selecionar *policies* que minimizem a EFE, \mathcal{G} , o agente tende a preferir observações prováveis (ou seja, aquelas alinhadas com as *priors* do agente), ao mesmo tempo que busca adquirir informações sobre seu ambiente. Essa abordagem permite que o agente não apenas reaja

às observações, mas também as influencieativamente por meio de suas ações, conduzindo a um processo de aprendizado e adaptação.

Assim, incorporando percepção, planejamento e tomada de decisão, podemos formalizar o ciclo de ação-percepção como (COSTA et al., 2020):

1. Um agente recebe um estímulo,
2. infere as causas latentes desse estímulo,
3. planeja o futuro e,
4. realiza seu curso preferido de ação.

2.5 Implementação Computacional da AIF

Na modelagem da AIF, os agentes são concebidos para manter crenças sobre certos aspectos do ambiente e suas interações. Nesta seção, apresentaremos as principais crenças que os agentes mantêm em implementações de AIF e entenderemos como elas são integradas em um modelo computacional (HEINS et al., 2022a; COSTA et al., 2020; TSCHANTZ et al., 2020a).

2.5.1 Especificação do Modelo Generativo

A representação computacional de um agente de AIF é feita através da especificação do seu modelo generativo $p(\tilde{o}, \tilde{s}, \pi, \theta)$, conforme descrito em subseção 2.4.2. Para definição do modelo generativo em aplicações concretas de inferência ativa, sua distribuição de probabilidade é decomposta em distribuições categóricas representando as probabilidades *a priori* e as funções de verossimilhança do modelo.

- **Distribuições Categóricas:** descrevem variáveis aleatórias discretas com um número finito de categorias, parametrizadas por um vetor ϕ (HEINS et al., 2022a).
- **Função de Verossimilhança (*likelihood*):** distribuições de probabilidade condicional. Descrevem como os dados observados são gerados a partir de um modelo. Em AIF, é usada para atualizar as crenças do agente sobre os estados ocultos com base nas observações (FRISTON, 2010).

A formulação geral do modelo generativo é (HEINS et al., 2022b):

$$P(o_{1:T}, s_{1:T}, \pi, \theta) = P(\theta)P(s_1)P(\pi) \prod_{\tau=2}^T P(s_\tau | s_{\tau-1}, \pi; \theta) \prod_{\tau=1}^T P(o_\tau | s_\tau; \theta) \quad (2.4)$$

Onde:

- **Probabilidade de Observação:** $P(o_\tau|s_\tau)$ representa as crenças do agente sobre como os estados ocultos s geram observações o .
- **Modelo de Transição:** $P(s_\tau|s_{\tau-1}, u_{\tau-1})$ representa as crenças do agente sobre como os estados ocultos em um tempo $\tau - 1$ causam estados ocultos no próximo tempo τ , condicionados a certos estados de controle (ações) $u_{\tau-1}$.
- **Priori sobre Estados Iniciais:** $P(s_1)$ é a crença inicial do agente sobre o estado do ambiente.
- **Priori sobre Observações:** $P(o_{1:\tau})$ especifica os objetivos do agente como uma distribuição desejada sobre observações. Implementação do viés de otimismo que apresentamos na seção 2.3.

2.5.2 O Modelo pymdp

Historicamente, a ferramenta DEM do SPM ([SMITH; FRISTON; WHYTE, 2021](#); [FRISTON; TRUJILLO-BARRETO; DAUNIZEAU, 2008](#)) em MATLAB tem sido o padrão-ouro na simulação de agentes de inferência ativa, fornecendo um conjunto robusto e reproduzível de funções para esta área de estudo. No entanto, a biblioteca `pymdp` emergiu como uma alternativa moderna, abordando algumas das limitações do DEM, como custos de licenciamento e falta de modularidade. A `pymdp` fornece um *framework* e funções auxiliares para implementar modelos e agentes de AIF, inspirada nas rotinas de inferência ativa do DEM, mas com uma estrutura mais modular e flexível, adaptada ao ecossistema Python.

Ela utiliza distribuições categóricas para representar as distribuições do modelo gerativo, seguindo as melhores práticas de modelagem e expandindo a acessibilidade da inferência ativa a uma gama mais ampla de pesquisadores e desenvolvedores ([HEINS et al., 2022a](#)).

Matematicamente, as distribuições categóricas são denotadas como $p(x) = Cat(\phi)$, onde ϕ é um vetor n-dimensional de parâmetros e n é a cardinalidade do espaço amostral de X . Cada componente do vetor ϕ representa a probabilidade de um determinado estado ou categoria.

No `pymdp`, essas distribuições são representadas numericamente como arrays multidimensionais que contêm seus parâmetros. Estes arrays podem ser:

- **Distribuições Marginais Vetoriais:** geralmente atuam como *priors* e são codificadas como vetores 1-D simples.
- **Distribuições Categóricas Condicionais:** representadas na forma de NDarrays (matrizes) que desempenham o papel de funções de verossimilhança no modelo gerativo.

A principal estrutura de dados utilizada para essas implementações é a biblioteca Numpy ([HARRIS et al., 2020](#)) utilizada para manipulações matriciais do ecossistema Python ([ROSSUM; DRAKE, 2009](#)).

Por exemplo, uma distribuição discreta $P(y|x)$ é codificada no `pymdp` como uma matriz de tamanho $N \times M$, onde N é o número de níveis da variável de suporte y e M é o número de níveis da variável condicionada x .

Dentro da notação `pymdp`, a verossimilhança de observação $P(o_\tau|s_\tau)$ é construída como o array **A**. Em modelos simples, **A** será uma matriz $O \times S$, onde O é o número de resultados ou níveis das observações o e S é o número de níveis dos estados ocultos s . A entrada **A**[i,j] codifica a probabilidade de ver a observação i , dado o estado j .

Da mesma forma, a verossimilhança de transição $P(s_\tau|s_{\tau-1}, u_{\tau-1})$ é construída como o array **B**, que é um NDarray $S \times S \times U$, onde U é o número de níveis dos estados de controle.

Outros componentes, como o array **D** que especifica uma *a priori* inicial sobre estados, e o array **C** que representa as "preferências anteriores" para codificar objetivos no modelo gerativo, também são parte integrante da implementação `pymdp`.

Após especificar o modelo gerativo em termos de suas distribuições de verossimilhança e *a priori*, é possível construir um agente de AIF usando o construtor `Agent()`. Os arrays **A** e **B** são obrigatórios, enquanto os arrays **C** e **D** são entradas opcionais para o construtor (os padrões são distribuições uniformes).

2.6 Trabalhos Relacionados

Esta seção apresenta os estudos fundamentais que embasaram nossa pesquisa. Estes trabalhos são cruciais para entender o arcabouço teórico da Inferência Ativa (AIF), suas aplicações práticas, e o impacto potencial que podem ter em variados domínios. A seleção dos trabalhos foi realizada através de uma busca por "Inferência Ativa" em plataformas acadêmicas como *Google Scholar*, *Semantic Scholar*, *Scopus* e *Web of Science*. Posteriormente, realizamos uma escolha ad-hoc das publicações mais representativas. Estas incluem revisões que discutem os fundamentos e o estado atual da AIF, suas diversas aplicações práticas, bem como algumas das publicações originais que introduziram conceitos-chave.

2.6.1 Fundamentos do PingPOMDP

- [Kagan et al. \(2022\). In vitro neurons learn and exhibit sentience when embodied in a simulated game-world.](#) Kagan e colaboradores desenvolveram o DishBrain, um sistema que integra neurônios *in vitro* a um mundo simulado inspirado no jogo arcade "Pong", aplicando conceitos da teoria de AIF. Demonstraram que neurônios podem exibir comportamento adaptativo e orientado a objetivos, auto-organizando sua atividade e

estrutura em resposta a informações sensoriais esparsas fornecidas pelo sistema DishBrain. O PingPOMDP foi diretamente inspirado por este sistema.

- [Heins et al. \(2022a\)](#). **pymdp: A Python library for active inference in discrete state spaces.** Heins e colaboradores apresentam o pymdp, uma biblioteca Python para simular agentes de Inferência Ativa com modelos generativos de espaço de estados discretos. O artigo detalha os princípios teóricos, matemáticos e computacionais por trás da biblioteca, bem como sua motivação, seus recursos e benefícios. Eles também ilustram como a biblioteca pode ser usada para projetar e testar modelos de AIF para diversos domínios e tarefas. Esta biblioteca, juntamente com seu artigo técnico associado, fundamentou a implementação do modelo de agente utilizado neste projeto.

2.6.2 Fundamentos da Inferência Ativa

- [Costa et al. \(2020\)](#). **Active inference on discrete state-spaces: A synthesis.** Da Costa e colaboradores fornecem uma revisão abrangente da teoria e métodos da AIF em espaços de estados discretos, abordando conceitos principais e sua relação com outros frameworks, como aprendizado por reforço, inferência bayesiana e teoria da informação.
- [Parr, Pezzulo e Friston \(2022\)](#). **Active Inference: The Free Energy Principle in Mind, Brain, and Behavior.** Parr e colaboradores oferecem uma visão geral da AIF como uma teoria geral da mente, cérebro e comportamento, explicando seus princípios-chave e implicações para compreender diversos aspectos da cognição e ação.
- [Tschantz et al. \(2020a\)](#). **Scaling Active Inference.** Tschantz e colaboradores investigam a escalabilidade da AIF em diferentes níveis de abstração e complexidade.
- [Smith, Friston e Whyte \(2021\)](#). **A Step-by-Step Tutorial on Active Inference and its Application to Empirical Data.** Smith e colaboradores oferecem um tutorial sobre como aplicar AIF a dados empíricos, guiando o leitor na construção de um modelo de AIF, ajustando-o a dados comportamentais e testando suas previsões.

2.6.3 Aplicações

- [Smith, Badcock e Friston \(2021\)](#). **Recent advances in the application of predictive coding and active inference models within clinical neuroscience.** Smith e colaboradores revisam avanços recentes na aplicação de modelos de codificação preditiva e AIF à neurociência clínica, discutindo como esses modelos podem elucidar mecanismos neurais subjacentes a diversos distúrbios psiquiátricos.
- [Adams et al. \(2022\)](#). **Everything is connected: Inference and attractors in delusions.** Adams e colaboradores utilizam a AIF para explicar a formação e manutenção de delírios,

argumentando que delírios podem ser vistos como crenças anormais que surgem de processos de inferência defeituosos no cérebro.

Os trabalhos selecionados oferecem uma compreensão abrangente dos fundamentos teóricos da AIF e suas aplicações práticas. Eles também servem como referências valiosas para o projeto atual, que visa desenvolver um modelo computacional de AIF interagindo com o jogo Pong.

2.7 Conclusões

Ao longo deste capítulo, exploramos a teoria da AIF e sua formalização como Processos de Decisão de Markov Parcialmente Observáveis (POMDP), proporcionando uma visão abrangente de como os agentes podem interagir e aprender com seu ambiente de maneira adaptativa e eficiente.

A AIF surge como uma teoria unificadora que busca explicar a cognição e o comportamento de agentes, propondo que estes buscam ativamente minimizar a surpresa em suas percepções sensoriais. Esta teoria, fundamentada em princípios matemáticos e computacionais sólidos, oferece uma perspectiva normativa sobre como os agentes devem se comportar para garantir sua continuidade em ambientes dinâmicos.

A implementação da AIF como POMDP permitiu uma compreensão mais profunda dos mecanismos pelos quais os agentes podem realizar inferências sobre seu ambiente e tomar decisões baseadas em suas crenças. Discutimos conceitos como modelo gerativo, inferência Bayesiana, distribuição de reconhecimento, e as funções de minimização da Energia Livre Variacional (VFE) e Energia Livre Esperada (EFE), destacando como esses elementos se integram para formar um ciclo de percepção-ação coeso.

Em conclusão, a AIF e sua formalização como POMDP oferecem uma base teórica e prática promissora para explorar e entender os mecanismos de aprendizagem e tomada de decisão. O aprofundamento e a continuidade das pesquisas nesta área têm o potencial de trazer avanços significativos e inovações para o campo da inteligência artificial e ciências cognitivas.

3 Modelagem e implementação do sistema PingPOMDP

O principal objetivo deste projeto, conforme apresentado na [seção 1.3](#), é avaliar a hipótese de que um modelo computacional inspirado no DishBrain apresentaria resultados comparáveis aos encontrados por [Kagan et al. \(2022\)](#). Especificamente, buscamos investigar se, ao substituir os neurônios, que teoricamente operam segundo os princípios da Inferência Ativa (AIF), por um modelo computacional baseado em AIF, obteríamos desempenhos comparáveis.

Nesse sentido, para atingirmos nosso abjetivo, desenvolvemos:

- Um **ambiente Pong** inspirado na versão utilizada por [Kagan et al. \(2022\)](#) ([seção 3.1](#));
- Um **agente** baseado em **AIF**, que esperamos aprender a jogar Pong ([seção 3.3](#));
- A interface **GridLink** para integrar o agente ao ambiente por meio de protocolos inspirados no DishBrain ([seção 3.2](#));
- E o sistema **PingPOMDP** para orquestrar todos esses componentes e gerenciar os nossos experimentos ([seção 3.4](#)).

A [Figura 3](#) apresenta cada um desses componentes e suas interações. Nela podemos observar o ciclo de percepção-ação([seção 2.2](#)), onde o estado do ambiente Pong é traduzido pela GridLink em uma observação para o agente. Com base nessa observação, o agente formula uma ação que, em seguida, é convertida pela GridLink em uma ação correspondente no ambiente Pong, estabelecendo assim um ciclo contínuo. O sistema PingPOMDP por sua vez, gerencia as configurações dos componentes, o banco de dados, bem como os próprios componentes.

Dessa forma, o modelo inicia com crenças aleatórias, que são atualizadas à medida que interage com o ambiente. Esperamos que ao longo das iterações suas ações tornem-se mais apropriadas de modo a evidenciar aprendizado e adaptabilidade ao ambiente por meio dos protocolos da GridLink. Nesse sentido, o sistema PingPOMDP nos permite realizar experimentos necessários para investigarmos nossa hipótese.

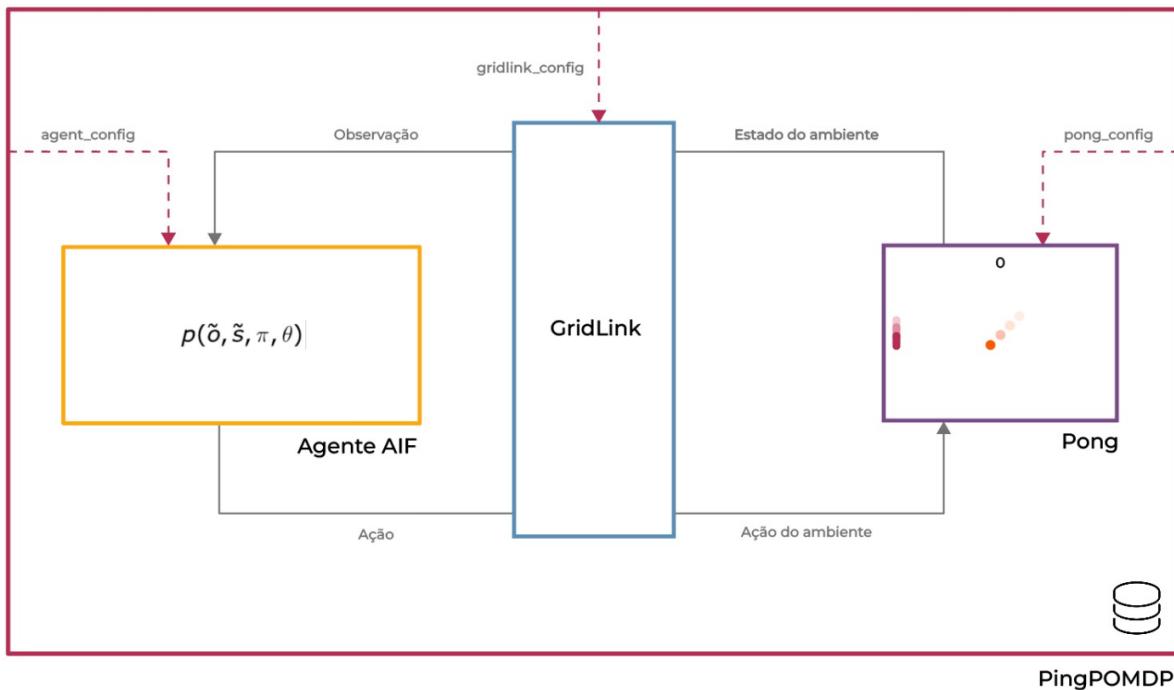


Figura 3 – Diagrama conceitual dos componentes do PingPOMDP

Fonte: Própria

Nesta seção, apresentaremos o sistema PingPOMDP bem como todos os componentes necessários para sua construção, elucidando seus mecanismos e implementações. Ao longo da discussão, também apresentamos os aspectos relevantes do DishBrain conforme necessário para a compreensão.

3.1 Ambiente Pong

Conforme vimos no Capítulo 1, a grande contribuição de Kagan et al. (2022) foi desenvolver um protocolo baseado em AIF que ensinou neurônios *in vitro* a jogar Pong. Dessa forma, nosso objetivo é construir um agente computacional que obtenha desempenho comparável nesse mesmo jogo. Portanto, nessa seção apresentamos o clássico jogo Pong, como ele foi adaptado ao DishBrain, e principalmente como nós o implementamos no PingPOMDP.

3.1.1 Pong Clássico

Lançado pela Atari em 1972, Pong é reconhecido como o primeiro videogame de sucesso comercial, marcando o início da indústria de jogos eletrônicos que, desde então, cresceu exponencialmente (WOLF, 2007; KENT, 2010). Pong é uma abstração bidimensional do desporto Tênis, Conforme ilustrado na Figura 4, o jogo envolve controlar um retângulo vertical

(representando uma raquete) com o objetivo de rebater uma bola, evitando que ela ultrapasse sua raquete e atinja sua parede posterior.



Figura 4 – Tela do jogo Pong, 1972

Fonte: [Kent \(2010\)](#)

3.1.2 Pong no DishBrain

A versão do Pong utilizada no DishBrain é uma reinterpretação do clássico, conforme detalhado por [Kagan et al. \(2022\)](#). Esta adaptação apresenta apenas uma raquete. O desafio principal permanece: acertar a bola com a raquete, evitando que ela alcance sua parede posterior. O jogo começa com a bola sendo lançada do lado direito da tela, seguindo uma direção aleatória. A bola se desloca a uma velocidade constante, ricocheteando nas paredes superior, inferior e direita até que, eventualmente, ultrapasse a raquete, marcando o fim de uma partida de Pong.

O controle do jogador se limita ao movimento vertical da raquete. Cada vez que a raquete consegue rebater a bola, um ponto é adicionado à pontuação do jogador. A cada fim de partida o placar de pontos é reiniciado. A [Figura 5](#) ilustra uma instância do jogo, onde a bola é interceptada pela raquete no canto superior esquerdo.



Figura 5 – Ilustração do Pong no DishBrain

Fonte: Kagan et al. (2022)

3.1.3 Implementação do Ambiente Pong no PingPOMDP

A nossa implementação do ambiente Pong (Figura 6) foi realizada em Python a partir das especificações apresentadas por Kagan et al. (2022) e mencionadas na subseção 3.1.2. A classe principal, denominada Pong, é responsável por gerenciar o estado do jogo, a dinâmica da bola e da raquete, bem como as interações entre eles.

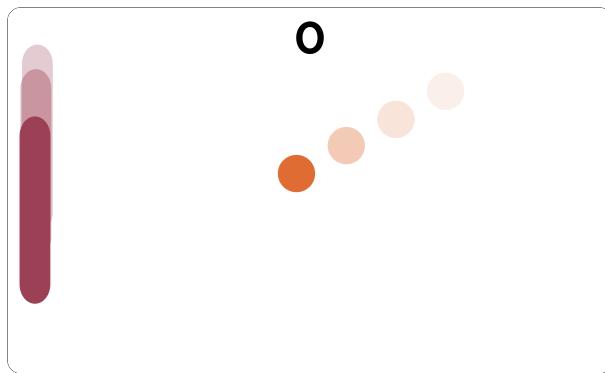


Figura 6 – Ilustração da nossa versão de Pong

Fonte: própria

3.1.3.1 Inicialização e Configuração

Ao instanciar a classe Pong (Código 3.1), são definidas as dimensões da tela, o raio da bola, a velocidade e as dimensões da raquete. Além disso, é possível definir o modo de lançamento da bola (`launch_ball_mode`), que pode ser "fixo" ou "aleatório". O método `reset` é chamado para (re)inicializar o estado do jogo.

Código 3.1 – Classe Pong

```
1 class Pong():
```

```
2     def __init__(self, screen_width=320, screen_height=480,
3                  ball_base_speed=5, ball_radius=30, launch_ball_mode="fix",
4                  paddle_speed=30, paddle_width=15, paddle_x=10,
5                  paddle_over_screen_proportion=0.3):
6         ...
7         self.reset()
```

3.1.3.2 Dinâmica do Jogo

O método `step` (Código 3.2) é responsável por atualizar o estado do jogo a cada iteração. Ele recebe uma ação que determina o movimento da raquete (para cima, para baixo ou parada). A bola é atualizada em sua posição e velocidade, e a raquete é movida de acordo com a ação fornecida, respeitando os limites da tela. Essa dinâmica de estados do Pong é apresentada na Figura 7

Código 3.2 – Método `step`

```
1 def step(self, action):
2     ...
3     return GameState(
4         status,
5         score,
6         ball_x_center,
7         ball_y_center,
8         paddle_y_center)
```

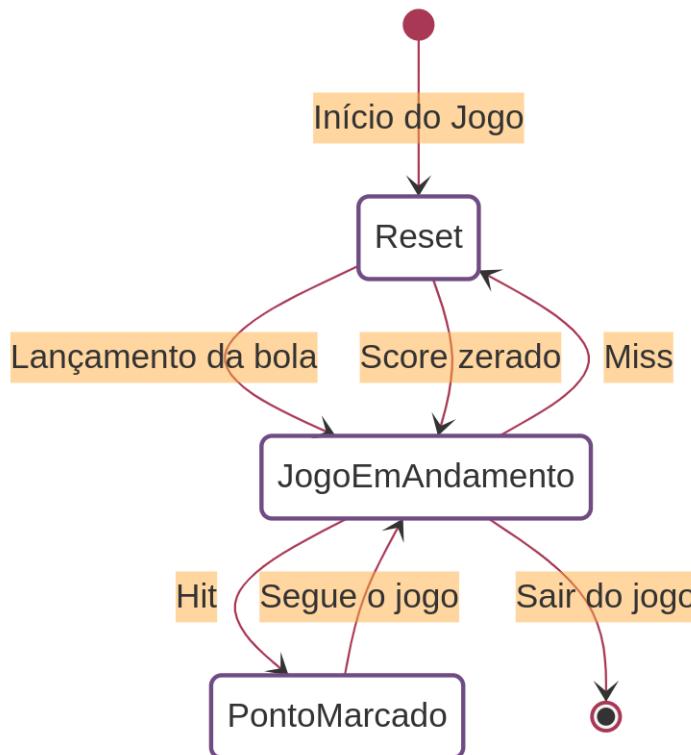


Figura 7 – Diagrama de estados do Pong

Fonte: própria

O estado do jogo a cada instante é composto pelo **status**, **score**, pelas coordenadas **x** e **y** do centro da bola, a velocidade da bola e a coordenada **y** do centro da raquete. Todas essas variáveis são capturadas pela tupla **GameState**, retornada a cada iteração do método **step**. O **status** do jogo é o evento ocorrido no último passo: -1 para fim de jogo, +1 para bola rebatida com sucesso, e 0 para instante sem evento. Enquanto o **score** é a pontuação da partida, ele corresponde ao número de rebatidas consecutivas sem deixar a bola atingir a parede posterior.

3.1.3.2.1 Detecção de Colisão

A detecção de colisão entre a bola e a raquete ou as paredes é gerenciada pelo método **_check_collision**. Se a bola colidir com a raquete, o que chamamos de "*hit*", sua direção horizontal é invertida. Se a bola atingir a parede esquerda, é considerado um "*miss*", e o jogo é reiniciado. Colisões com as paredes superior, inferior e direita fazem a bola refletir.

3.1.3.3 Lançamento da Bola

O método **_launch_ball** é responsável por determinar a posição inicial e a velocidade da bola quando o jogo começa ou é reiniciado. Dependendo do modo de lançamento escolhido, a bola pode ser lançada de uma posição fixa ou de uma posição aleatória.

3.2 Interface GridLink

GridLink é a biblioteca central desse projeto. Ela foi desenvolvida para permitir utilizarmos uma abstração computacional da interface e os protocolos utilizados no DishBrain para integrar agentes computacionais a ambientes virtuais. Ela foi desenvolvida para ser genérica, de modo que no PingPOMDP instanciamos uma classe filha para integrar especificamente o nosso ambiente (seção 3.1) ao nosso agente (seção 3.3).

Portanto, nessa seção apresentarmos mais detalhadamente a interface e protocolos do DishBrain para em seguida apresentarmos a GridLink, e concluirmos apresentando a implementação da GridLink utilizada no PigPOMDP.

3.2.1 Interface e Protocolos do DishBrain

O DishBrain propõe uma integração pioneira entre neurônios e o ambiente virtual do jogo Pong. O fundamento desse sistema repousa sobre um mecanismo de retroalimentação fechada (*closed-loop*) de estimulação e registro eletrofisiológico, utilizando arrays multieletrodos de alta densidade (HD-MEAs). Neste sistema, a placa de eletrodos codifica a **posição relativa da bola** em relação à raquete (representando **estímulos sensoriais**), enquanto decodifica a atividade elétrica (como **controle motor**) de duas regiões arbitrariamente definidas para controlar o **movimento da raquete**. A Figura 8 apresenta uma representação desse sistema.

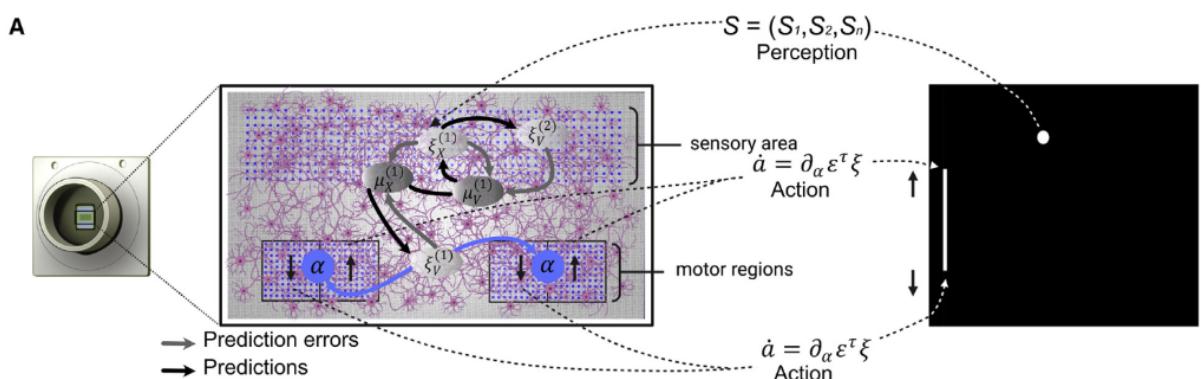


Figura 8 – Neurônios sobre a HD-MEA interagindo com o Pong

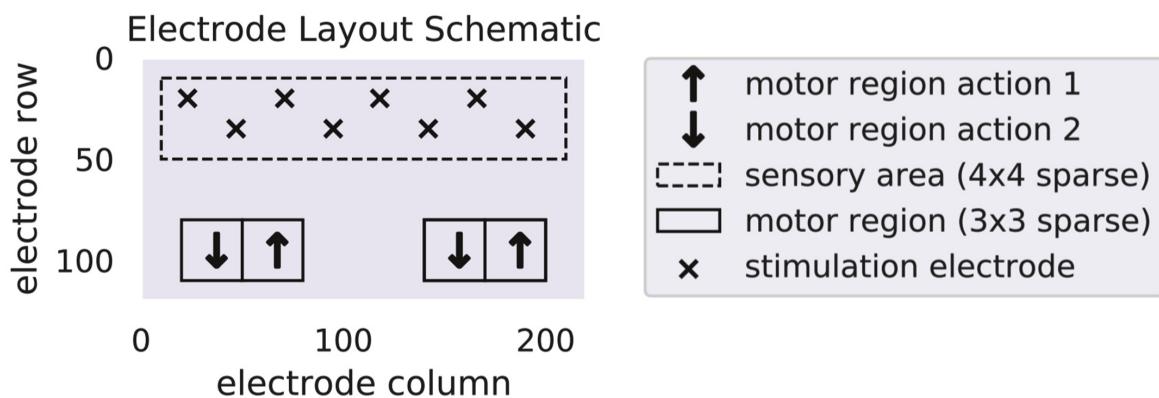
Fonte: Kagan et al. (2022)

Um aspecto fundamental deste sistema é o feedback administrado. Acertos geram **feedback previsível**, enquanto falhas resultam em **feedback imprevisível**. Nesse caso, acertos (*hits*) ocorrem quando a raquete intercepta a bola com sucesso, e falhas quando a raquete "deixa a bola passar" (*misses*) atingindo a parede posterior.

Este design é embasado pela teoria de AIF porque assume a expectativa de que os neurônios adaptem sua atividade interna para evadir estados associados a estímulos externos imprevisíveis,

e buscar estados associados a estímulos previsíveis. A minimização desta imprevisibilidade deve, portanto, manifestar-se na capacidade de controlar adequadamente a raquete dentro do ambiente Pong ([KAGAN et al., 2022](#)).

O esquema da disposição dos eletrodos pode ser visualizado na [Figura 9](#). A HD-MEA foi segmentado em uma "área sensorial" e quatro "áreas motoras". Enquanto a área sensorial codifica a posição relativa da bola (estímulo sensorial) e administra os feedbacks previsíveis e imprevisíveis, as regiões motoras interpretam a atividade neural para comandar a raquete.



[Figura 9 – Esquema da HD-MEA utilizada no DishBrain](#)

Fonte: [Kagan et al. \(2022\)](#)

Detalhamos a seguir as especificações técnicas e os protocolos do DishBrain conforme descrito por [Kagan et al. \(2022\)](#).

- A "Área Sensorial" engloba 626 eletrodos, dos quais apenas 8 são ativamente usados como "eletrodos de estímulo".
- A "Área Motora" é composta por quatro subdivisões: duas do tipo 1 e duas do tipo 2, conforme ilustrado na [Figura 9](#). A atividade elétrica nas regiões é lida para determinar o movimento da raquete: para cima, caso a região 1 seja mais ativa, e para baixo, se a região 2 predominar. A avaliação da atividade elétrica é realizada pela detecção de picos elétricos em cada região durante um intervalo de 10 milissegundos (equivalente a 200 amostras).

3.2.1.1 Protocolos de feedback

A dinâmica do jogo Pong é atualizada em janelas de 10ms. A bola percorre a tela com velocidade constante, sendo rebatida pela raquete e ricocheteando nas bordas até eventualmente ultrapassar a raquete e atingir a parede posterior, o que encerra a partida, como discutido na seção 3.1.

Dessa forma, quando a bola é rebatida com sucesso (*hit*), os neurônios recebem um **feedback previsível**. Contudo, se a bola ultrapassa a raquete e atinge a parede posterior (*miss*), eles são submetidos a um **feedback imprevisível**. Além disso, durante a partida os neurônios recebem **estímulo sensorial** representando a posição relativa da bola.

- O **Feedback Imprevisível** é administrado quando os neurônios "deixam passar" a bola (*miss*). Com duração de 4.000ms, esse estímulo envolve a ativação e desativação aleatória dos eletrodos de estímulo a uma tensão de 150mV e 5Hz. Depois disso, todos os estímulos são interrompidos por um intervalo ajustável.
- O **Feedback Previsível** é acionado quando os neurônios "rebatem" a bola com sucesso (*hit*). Com duração de 100ms, esse protocolo consiste na ativação simultânea dos 8 eletrodos de estímulo, com uma corrente de 75mV e frequência de 100Hz.
- O **Estímulo Sensorial** é administrado durante a partida como a ativação de um único dos oito eletrodos sensoriais de acordo com o alinhamento vertical da raquete com a bola. Isto é, quanto mais abaixo da raquete a bola estiver, mais a esquerda será o eletrodo ativado, quanto mais centralizada a raquete estivar com a bola, o eletrodo mais central será ativado. Esse impulso é caracterizado por um potencial de 75mV, com uma frequência que varia linearmente de 4 a 40 Hz, dependendo da proximidade (horizontal) entre a bola e a raquete.

3.2.2 Implementação da GridLink

Desenhada para ser amplamente aplicável em variados contextos e ambientes, a GridLink apresenta sua estrutura modular e concepção abstrata. Essas características permitem que pesquisadores personalizem a interface para atender às exigências específicas de seus estudos. Essa adaptabilidade é crucial, pois assegura que a GridLink seja adequada como intermediária entre diferentes agentes e ambientes, abarcando as características e particularidades de cada projeto de pesquisa.

3.2.3 Modelagem da GridLink

A GridLink tem como foco principal modelar computacionalmente a interface entre os neurônios *in vitro* e o ambiente Pong no DishBrain. Ela é estruturada em torno de uma matriz que representa a HD-MEA e dos protocolos utilizados por [Kagan et al. \(2022\)](#).

Na GridLink, cada elemento/célula da matriz representa a abstração de um eletrodo ou a uma região composta por vários eletrodos. Cada uma dessas células é binária, ou seja, pode assumir apenas os valores de 0 ou 1, para representar ausência ou presença de atividade elétrica naquele ponto. Nesse sentido, a [Figura 10](#) representa um exemplo de abstração da HD-MEA do DishBrain ([Figura 9](#)) utilizando a GridLink. Ela apresenta as oito células sensoriais com

o símbolo "X" bem como regiões motoras como dois conjuntos de quatro células. As demais células permanecem sempre com o valor 0, representando ausência de atividade elétrica.

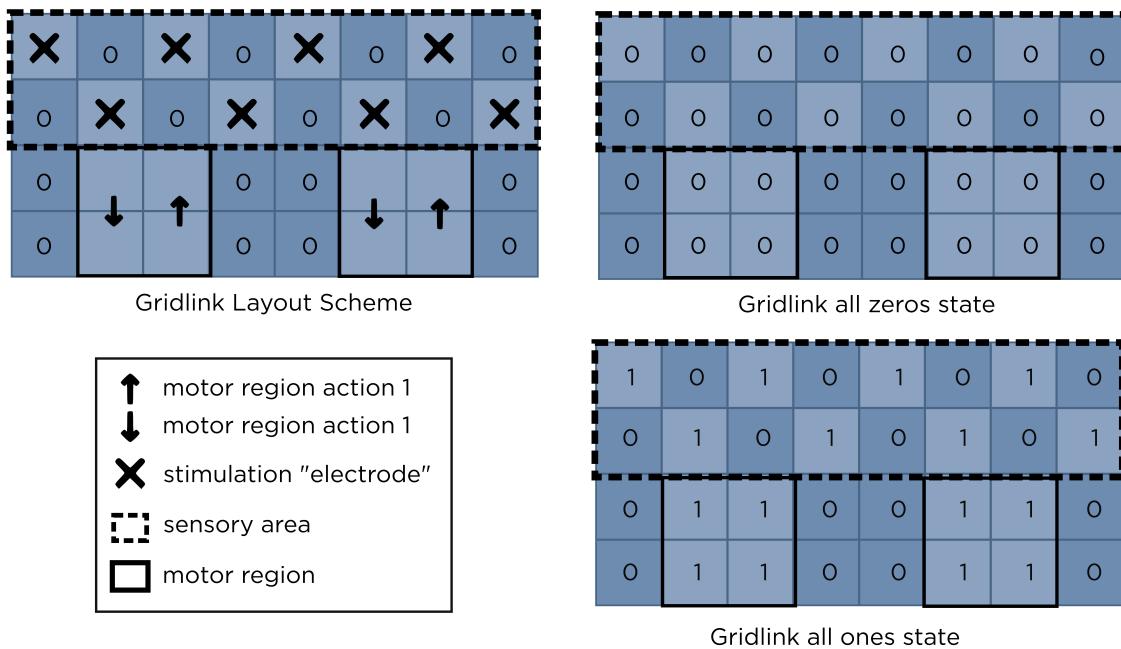


Figura 10 – Abstração da MEA por meio da GridLink

Fonte: própria

3.2.3.1 Input e Feedback na GridLink

Dessa forma, a GridLink administra o *input* sensorial ao modificar os valores das células correspondentes a observação, e enviar essa observação ao agente, conforme os protocolos de estímulo do DishBrain apresentados na [subseção 3.2.1.1](#). Da mesma forma, a GridLink busca a ação do agente, a representa na região motora da matriz, e processa essa representação em uma ação para o ambiente considerando qual das regiões motoras apresenta maior atividade.

Dessa forma, podemos apresentar os diferentes tipos de observação que o agente pode receber nesse contexto:

- Assim como o estímulo sensorial no DishBrain ativa apenas um dos oito eletrodos, a **observação sensorial** da **GridLink** é um vetor do tipo "*one-hot-encoding*", isto é, apenas um dos seus elementos é não-nulo. Analogamente, ele representa o alinhamento entre bola e raquete. Portanto, a observação [1,0,0,0,0,0,0,0] indica que a bola está desalinhada acima da raquete, enquanto [0,0,0,0,1,0,0,0] indica que estão alinhadas.
- O feedback previsível pode ser representado como uma sequência de observações idênticas, representando a sequência de ativação nos oito eletrodos do DishBrain: [1,1,1,1,1,1,1,...],[1,1,1,1,1,1,1,1].

- Da mesma forma, as observações de feedback inprevisível também são sequências de arrays, porém com ativações aleatórias como por exemplo: [10101010],[10011010],...,[11100101].

Assim, a Figura 11 apresenta a sequência de etapas que a GridLink orquestra para efetuar essa integração entre agente (*Agent*) e ambiente (*Env*). Utilizamos um Diagrama de Sequência devido a clareza que ele nos permite apresentar a dinâmica de interação entre participantes. Nesse sentido, a GridLink solicita o estado do ambiente (por meio do método `observe()`) e processa esse estado para produzir a observação sensorial para o agente. Além disso, avalia se o estado do ambiente é desejável ou indesejável e administra os feedbacks correspondentes (previsível e imprevisível). Em seguida, a GridLink solicita uma ação do agente, processa essa ação do agente em uma ação do ambiente e a fornece como parâmetro ao avançar um passo no ambiente (por meio do método `step(action)`).

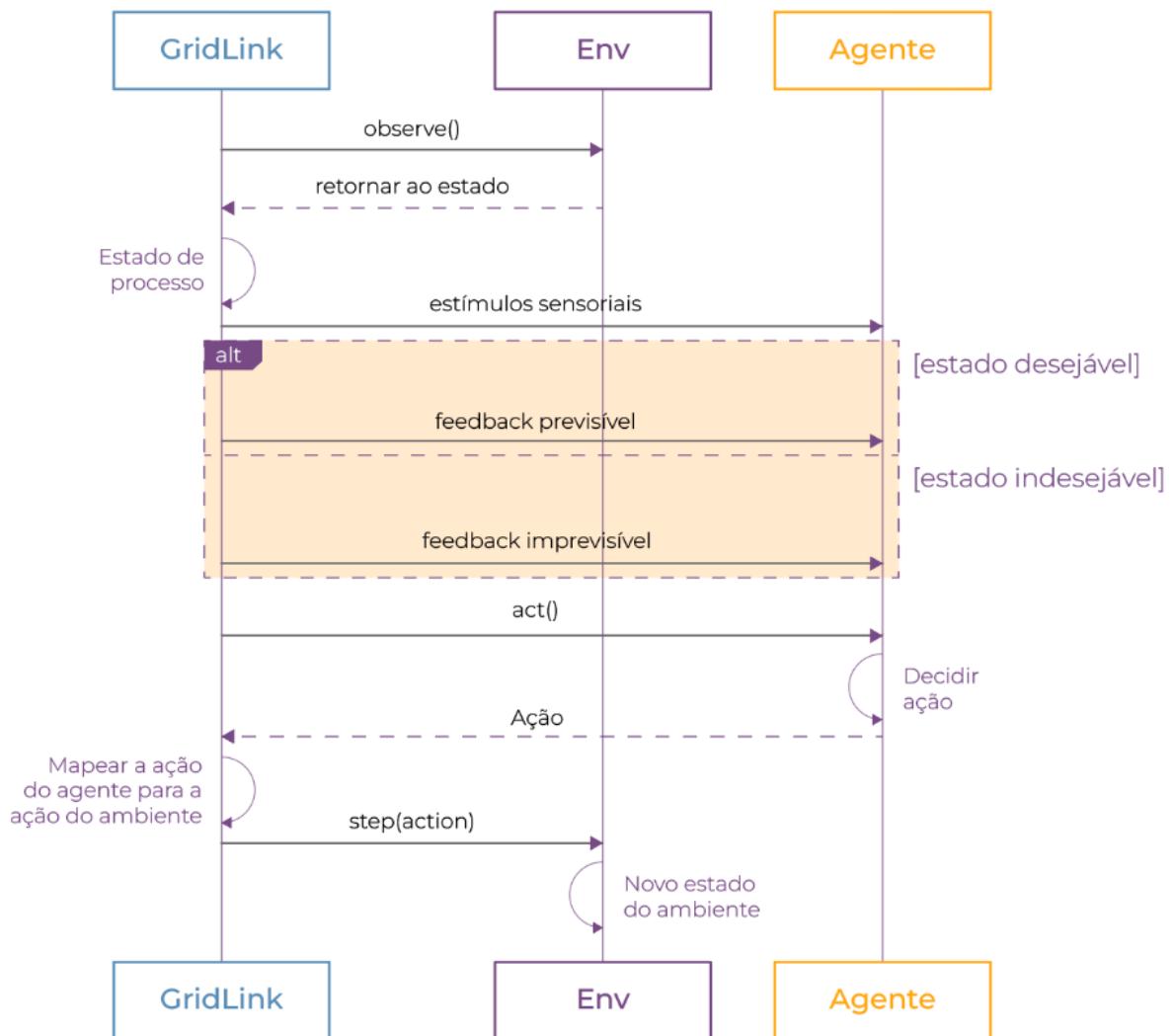


Figura 11 – Diagrama de sequência da Gridlink

Fonte: própria

3.2.3.2 Estrutura Básica

A classe principal, `Gridlink`, gerencia a integração entre o agente e o ambiente. Ela é inicializada (Código 3.3) com um agente, um ambiente e parâmetros opcionais que definem o formato da matriz, as células sensoriais e o modo de observação. Esta classe abstrata serve como uma interface para desenvolvedores adaptarem a `GridLink` a seus agentes e ambientes específicos. O código dessa implementação foi desenvolvido em Python ([ROSSUM; DRAKE, 2009](#)), utilizando a biblioteca NumPy ([HARRIS et al., 2020](#)) para manipulações matriciais e operações de cálculo.

Código 3.3 – Método `init` da classe `GridLink`

```

1 def __init__(self, agent, env,
2     grid_shape = GRID_SHAPE, sensory_cells = SENSORY_CELLS,
3             observation_mode = OBSERVATION_MODE,
4             n_predictable_cycles=N_PREDICTABLE_CYCLES,
5             n_unpredictable_cycles=N_UNPREDICTABLE_CYCLES):
6     ...

```

Dentro da classe, a matriz (`grid_array`) representa a HD-MEA. Com dois parâmetros principais: o formato dessa matriz (`grid_shape`) e a listagem das células sensoriais (`sensory_cells`). Esses parâmetros podem ser estabelecidos de acordo com a aplicação da `GridLink`, por padrão, declaramos as constantes de acordo com uma modelagem simplificada do layout utilizado no DishBrain (Figura 9), uma matriz 4x8 com suas duas fileiras sendo a região sensorial, e dessa região, células alternadas sensoriais (em referência aos eletrodos sensoriais).

```

1 GRID_SHAPE = (4, 8)
2 SENSORY_CELLS = (0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20)

```

3.2.3.3 Observações e Feedback

Considerando a incerteza sobre como os neurônios *in vitro* do DishBrain percebem a HD-MEA e suas ativações elétricas, decidimos implementar na `GridLink` dois modos distintos de observação:

- **whole_grid:** Neste modo, a observação enviada ao agente inclui os valores de todas as células da matriz.
- **sensory_cells:** Nesta modalidade, a observação é restrita às células sensoriais, transmitindo ao agente apenas os valores dessas células específicas.

O método `observe` (Código 3.4) recebe o estado do ambiente e, com base nele, determina se um *feedback* previsível ou imprevisível deve ser aplicado. Isso é realizado pelos métodos `_predictable_feedback` e `_unpredictable_feedback`, respectivamente. Adicionalmente,

a observação do estado do ambiente é representada nas células sensoriais através do método `_sensory_feedback`.

Código 3.4 – Método observe

```

1 def observe(self, env_state):
2     if self._is_env_state_undesirable(env_state):
3         self._unpredictable_feedback()
4
5     elif self._is_env_state_desirable(env_state):
6         self._predictable_feedback()
7
8     self._sensory_feedback(env_state)

```

- **Feedback Sensorial (`_sensory_feedback`):** traduz o estado do ambiente em uma observação, ativando uma célula sensorial específica.
- **Feedback Previsível (`_predictable_feedback`):** em resposta a um estado desejável, uma sequência de n ativações previsíveis ocorre nas células sensoriais, sendo n dada pelo parâmetro `n_predictable_cycles`.
- **Feedback Imprevisto (`_unpredictable_feedback`):** em resposta a um estado indesejável, uma sequência de `n_unpredictable_cycles` ativações aleatórias ocorre nas células sensoriais.

3.2.3.4 Atuação

Para a atuação do agente no ambiente, a GridLink se utiliza do método `act` (Código 3.5).

Código 3.5 – Método act

```

1 def act(self):
2     self.agent_action = self.agent.act()
3     self.env_action = self._map_agent_action_to_env(self.agent_action)
4     return self.env_action

```

Este método solicita uma ação do agente e a traduz para uma ação no ambiente por meio da função `_map_agent_action_to_env`. Essa função é abstrata e deve ser implementada de acordo com o agente e ambiente específicos.

3.2.3.5 Etapas de Interação

O ciclo principal de interação é realizado pelo método `step`. Em cada etapa, este método:

1. Solicita uma ação do agente;

2. Traduz essa ação para uma ação do ambiente;
3. Atualiza o estado do ambiente com base na ação;
4. Gera uma observação do novo estado e fornece feedback ao agente.

3.2.3.6 Funcionalidades Adicionais e Abstrações

Várias funções auxiliares são fornecidas para gerenciar as células da matriz, desde a leitura e gravação de células específicas até a conversão de índices unidimensionais para bidimensionais.

Além disso, temos as seguintes funções abstratas que devem ser implementadas especificamente para cada agente e ambiente:

- `_map_agent_action_to_env` define como as ações do agente são mapeadas para ações do ambiente.
- `_is_env_state undesirable` define o que constitui um estado desejável.
- `_is_env_state_desirable` define o que constitui um estado desejável.
- `_active_cell_index` define qual célula sensorial deve ser ativada para representar um determinado estado do ambiente.

3.2.4 GridLink no PingPOMDP

A aplicação específica da GridLink no PingPOMDP é realizada através da subclasse `PongGridlink` ([Código 3.6](#)), que estende a classe `Gridlink`. Esta extensão incorpora particularidades do ambiente PingPOMDP, e adapta a interface padrão da GridLink para integrar adequadamente o agente de AIF com o ambiente Pong.

Código 3.6 – Inicialização da `PongGridlink`

```

1 class PongGridlink(Gridlink):
2     def _is_env_state undesirable(self, env_state):
3         ...
4     def _is_env_state_desirable(self, env_state):
5         ...
6     def _active_cell_index(self, env_state):
7         ...
8     def _map_agent_action_to_env(self, agent_action):
9         ...

```

3.2.4.1 Modelagem da PongGridlink

A configuração da GridLink empregada em nossos experimentos é ilustrada na [Figura 12](#). Esta modelagem foi estrategicamente desenvolvida para ser representativa do layout da GridLink, mantendo a viabilidade computacional. A necessidade dessa abordagem surgiu após um teste preliminar com o modelo da HD-MEA descrito na [subseção 3.2.3](#) resultar em sobrecarga computacional e falha no sistema da máquina utilizada para executar os experimentos.

Dessa forma, reduzimos as células sensoriais de oito para três, e convergimos o controle de ação para uma única célula de modo que caso seu valor seja 1 representa a ação de mover a raquete para cima, e caso seja zero representa a ação de mover a raquete para baixo.

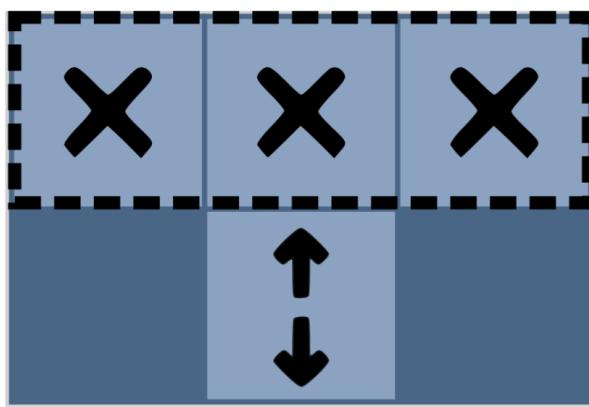


Figura 12 – Modelo da GridLink utilizada no PingPOMDP

Fonte: própria

3.2.4.2 Observações e Feedback

A PongGridlink personaliza o método de observação para interpretar o estado do ambiente Pong e fornecer feedback ao agente. Isso inclui a tradução do estado do jogo em ativações específicas das células sensoriais na matriz representativa da HD-MEA.

O método `_active_cell_index` ([Código 3.7](#)) é um exemplo dessa personalização. Ele calcula o índice da célula ativa com base na posição relativa da bola e da raquete no ambiente do jogo. A posição relativa é determinada subtraindo a posição vertical da bola da posição vertical da raquete. Esta posição relativa é então normalizada para um valor entre 0 e o número total de células sensoriais, representado por `n_sensory_cells`. O resultado é então ajustado para garantir que esteja dentro dos limites válidos, ou seja, entre 0 e `n_sensory_cells - 1`.

Código 3.7 – Método `grid_active_cell_index`

```

1 def _active_cell_index(self, env_state):
2     relative_position = (env_state.paddle_y -
3         env_state.ball_y)
4     normalized_rel_pos = round(relative_position /
5         self.env.screen_height *

```

```

6     self.n_sensory_cells +
7     self.n_sensory_cells / 2)
8
9     return max(0,
10    min(
11      normalized_rel_pos,
12      self.n_sensory_cells - 1))

```

Nesse sentido, como adotamos o padrão de três células sensoriais ($n = 3$), a observação é um array de três elementos binários, portanto, temos três possibilidades de observação sensorial:

- [1, 0, 0] para a bola acima da raquete.
- [0, 1, 0] para a bola alinhada com a raquete.
- [0, 0, 1] para a bola abaixo da raquete.

Além disso, conforme vimos nas funções abstratas da GridLink (subseção 3.2.2), a PongGridlink também implementa critérios específicos para determinar se um estado do ambiente é desejável ou indesejável, de modo a determinar se o feedback previsível ou imprevisível será administrado.

Conforme a implementação da dinâmica do Pong, descrita na subseção 3.1.3.2, o estado do ambiente é considerado indesejável quando o `status` é "`-1`" e desejável quando é "`1`". Essa classificação ocorre ao definir as funções abstratas correspondentes, conforme o Código 3.8.

Código 3.8 – Classificação de estados em desejáveis ou indesejáveis

```

1 def _is_env_state_undesirable(self, env_state):
2     return env_state.status < 0
3
4 def _is_env_state_desirable(self, env_state):
5     return env_state.status >= 1

```

A Figura 13 retrata os protocolos de feedback previsível e imprevisível que a PongGridLink administra como sequências de observações para o agente, conforme apresentamos na subseção 3.2.3.1:

- Feedback Previsível: após um *hit*



- Feedback Imprevisível: após um *miss*

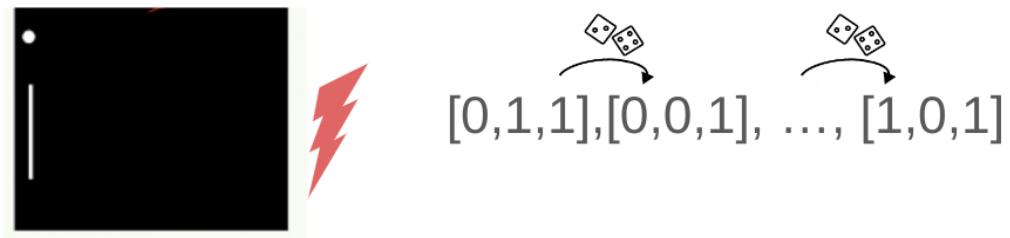


Figura 13 – Protocolos de Feedback da PongGridLink

Fonte: própria

3.2.4.3 Conversão das ações do agente em ações no ambiente

Conforme vimos na subseção 3.2.3.4, o agente age no ambiente mediado pelo método `act` (Código 3.5). Este método obtém uma ação do agente e a converte em uma ação correspondente no ambiente Pong, garantindo que as ações se alinhem com o que é permitido no jogo.

Para realizar essa conversão, a PongGridlink implementa o método abstrato `_map_agent_action_to_env` (Código 3.9). No contexto do jogo, o agente pode escolher entre duas ações: 0 ou 1. Estas correspondem, respectivamente, a mover a raquete para baixo ou para cima no jogo Pong. Assim, a ação **0** do agente é traduzida como **-1** no Pong (mover para baixo) e a ação **1** do agente como **1** no Pong (mover para cima). Optamos por não incluir a ação de manter a raquete estática para manter a fidelidade com o DishBrain.

Código 3.9 – Implementação do método abstrato `_map_agent_action_to_env`

```
1 def _map_agent_action_to_env(self, agent_action):
2     return -1 if agent_action == 0 else 1
```

3.3 Agente de Inferência Ativa

Nesta seção, detalharemos a implementação computacional do agente de AIF. Iniciaremos apresentando a estrutura geral do agente, seguida pela descrição do modelo gerativo que guia

suas crenças e ações. Em seguida, exploraremos os mecanismos de atualização de crenças e seleção de ações, culminando na integração do agente com o ambiente Pong através da interface GridLink.

3.3.1 Estrutura do Agente de Inferência Ativa

A implementação do agente de AIF no projeto PingPOMDP é fundamentada na biblioteca *pymdp* (HEINS et al., 2022a), que oferece uma série de ferramentas para simular agentes equipados com modelos generativos formulados como Processos de Decisão de Markov Parcialmente Observáveis (POMDPs) (subseção 2.5.2). O agente é modelado para manter crenças sobre estados ocultos do ambiente e, com base nessas crenças, selecionar ações que buscam minimizar a divergência entre as observações esperadas e as observações obtidas.

3.3.2 Espaços de Observação e Ação

O agente é inicializado com espaços de observação e ação definidos. No contexto do PingPOMDP, utilizamos um espaço de observação composto por n elementos binários que representam as n células sensoriais do GridLink, conforme apresentamos na subseção 3.2.4.2.

O valor de n é um parâmetro experimental. Em nossos experimentos, conforme vimos em subseção 3.2.4.2, adotamos o número de células sensoriais como 3. Logo, utilizaremos o padrão de $n = 3$. Desse modo, teremos observações sensoriais como: [0,1,0] representando a posição relativa da raquete como alinhada com a bola, [1,0,0] para bola acima da raquete, e [0,0,1] para bola abaixo da raquete. Além disso, como também teremos as observações decorrentes do feedback previsível (sequência de observações [1,1,1]), e as observações imprevisíveis (sequência de triplas binárias aleatórias), nosso espaço de observação é composto por todas as combinações possíveis desses três elementos binários. Portanto, podemos definir o espaço de observação do agente como sendo de 2^n .

O espaço de estados, conforme apresentamos na seção 2.4, se refere ao número de diferentes estados ocultos que o agente acredita que seu ambiente possa assumir. Esse número também é uma **variável experimental**, que normalmente é definida a partir de considerações gerais quanto ao ambiente e experimentações (HEINS et al., 2022a). Utilizamos como padrão o espaço de estados como sendo do mesmo tamanho do espaço de observações do agente.

Finalmente, o espaço de controle (ou de ação), do nosso agente é binário, representando as duas ações possíveis: mover a raquete para cima ou para baixo.

Dessa forma, a inicialização do agente se inicia ao definir os parâmetros dos espaços de observação, estados e ações, conforme o Código 3.10.

Código 3.10 – Inicialização do Agente

```
1 from pymdp.agent import Agent
```

```

2
3 class ActiveInferenceAgent(Agent):
4     def __init__(self, n_obs, n_states, matrices_mode):
5         self.num_obs = [2**n_obs]
6         self.num_states = [2**n_states]
7         self.num_controls = [2]

```

3.3.3 Modelo Generativo

Além dos espaços de observação, estados e ações, o agente é equipado com um modelo generativo que define suas crenças sobre como o mundo funciona. Nesse sentido, conforme vimos na [seção 2.5](#), definimos as matrizes A, B, C e D da seguinte maneira:

- **Probabilidade Sensorial** (matriz **A**): define a probabilidade receber determinada **observação** estando em determinado **estado oculto**.
- **Probabilidade de Transição** (matriz **B**): representa a probabilidade de mudança entre os estados ocultos com base em uma ação específica do agente. Em nosso cenário, essa matriz pode, por exemplo, refletir a crença de que a bola está posicionada acima da raquete. Assim, ao escolher a ação `mover_para_baixo`, o sistema pode prever a transição para o estado `bola_alinhada_com_raquete`. Por outro lado, ao optar por `mover_raquete_para_cima`, o estado previsto seria `bola_ainda_mais_acima_da_raquete`. Com base nessa matriz, o agente pode estrategicamente planejar suas ações para atingir estados favoráveis e evitar estados indesejados.
- **Preferências** (**C array**): representa os estados de observação que o agente deseja alcançar. Nossos agentes, sempre iniciam com preferências uniformes (ou seja, sem preferência, visto que todas as observações são igualmente desejáveis). Esperamos que ao longo das iterações, o agente adquira preferência por manter a raquete centralizada na bola para evitar que ela atinja a parede posterior. O que se manifestaria em preferências por ativações nas células centrais e aversão a ativações nas células das extremidades do vetor observação. Em termos das observações que estamos exemplificando, tal preferencia atribuiria alta probabilidade a observação [0,1,0].

3.3.3.1 Crenças e Preferências no PingPOMDP

Como um agente, em nosso contexto, é definido por seu modelo generativo que representa suas crenças probabilísticas sobre o mundo e sobre como deve agir, podemos dizer que nosso agente é definido pelas matrizes que compõem suas crenças ([HEINS et al., 2022a; COSTA et al., 2020; SMITH; FRISTON; WHYTE, 2021](#)).

Em nossos experimentos todos os agentes serão inicializados com preferências uniformes, ou seja, **C** array com o mesmo valor para todas as observações possíveis ([Código 3.11](#)). Dessa forma, no início de cada experimento, o agente não tem nenhuma preferência sobre o seu ambiente. Testaremos se ao longo das iterações o agente desenvolverá preferências de modo a apresentar comportamento adaptativo mesmo iniciando com preferências uniformes.

Código 3.11 – Definição do C array

```
1 C = utils.obj_array_uniform(self.num_obs)
```

Além disso, os agentes que esperamos aprender a jogar Pong serão inicializados com crenças aleatórias, isto é, preencheremos suas matrizes **A** e **B** com valores distribuídos de forma randômica de acordo com uma semente aleatória definida pela variável **agent_seed**, conforme o [Código 3.12](#). Denominamos esses Agentes de crenças Aleatórias de **AA**. Desse modo, esperamos que ao longo das iterações esses agentes atualizem suas crenças de modo a melhorar o seu desempenho no ambiente. Dessa forma, cada agente desse tipo é definido por sua categoria e por sua semente aleatória, portanto uma teremos uma instância **AA1** como sendo o agente do tipo **AA**, cujas matrizes **A** e **B** foram inicializadas aleatoriamente a partir da semente **1**.

Código 3.12 – Definição das matrizes A,B e C que definem um agente

```
1 from pymdp import utils
2
3 class ActiveInferenceAgent(Agent):
4     def __init__(...):
5         ...
6         # Create the generative model
7         C = utils.obj_array_uniform(
8             self.num_obs)
9         A = utils.random_A_matrix(
10            self.num_obs, self.num_states)
11        B = utils.random_B_matrix(
12            self.num_states, self.num_controls)
```

Também podemos construir agentes cujas matrizes **A** e **B** são uniformes como o array **C** de preferências. Esse é um caso experimental interessante porque reitera os fundamentos apresentados na [seção 2.3](#). Isso porque, é possível esperar que crenças neutras levem a um aprendizado mais consistente, porém, conforme vimos na [seção 2.4](#), o aprendizado em modelos de AIF se dá por meio da minimização da discrepância entre as observações esperadas e obtidas. Crenças uniformes significam que o agente acredita que qualquer ação leva a qualquer estado, que produz qualquer observação. Portanto, crenças uniformes não oferecem subsídios para que o agente as modifique. Desse modo, esperamos que agentes de crenças uniformes sejam "inertes" a observações e feedback. Chamamos esses Agentes com crenças Uniformes de **AU**.

3.3.4 Atualização de Crenças

A atualização de crenças é uma parte integral do ciclo de percepção-ação proposto pela AIF. No contexto da AIF, a percepção é formulada como um problema de inferência no qual agentes tentam inferir os estados ocultos que causam suas observações ([seção 2.3](#)). No nosso código, essa inferência é realizada pelo método `observe` ([Código 3.13](#)).

Quando o agente recebe uma nova observação, a primeira etapa é converter o vetor binário em um índice. O motivo dessa conversão é que a biblioteca pymdp utiliza índices para representar observações discretas ([subseção 2.5.2](#)). Ou seja, cada observação possível corresponde ao seu índice no espaço de observações.

Desse modo, a observação recebida é um *array* binário que representa o estado atual do ambiente Pong, capturado pelas células sensoriais. Esse *array* deve ser convertido em um índice único. Esta conversão é realizada através da combinação dos elementos binários da observação em uma única *string*, que é convertida para um número inteiro. Por exemplo, uma observação representada pela matriz binária [1, 0, 0, 1] seria convertida para a *string* "1001", que corresponde ao número decimal 9.

Código 3.13 – Método `observe` da classe `ActiveInferenceAgent`

```

1 def observe(self, observation):
2     """
3     Update beliefs based on the received observation.
4     """
5     # Convert observation (binary array) to index
6     obs_idx = [int("".join(map(str, observation.astype(int)))), 2]
7
8     # Update beliefs
9     self.infer_states(obs_idx)

```

Em seguida, o índice é passado para o método `infer_states` que atualiza as crenças do agente sobre os estados ocultos utilizando o algoritmo *fixed-point iteration* ([HEINS et al., 2022a](#)). Esse processo é consistente com a ideia de que agentes percebem o mundo minimizando a Energia Livre Variacional (VFE) para tornar seu modelo consistente com suas observações passadas ([seção 2.3](#)).

3.3.5 Ação e Política

Após atualizar suas crenças, o agente precisa decidir qual ação tomar. Este processo de decisão é guiado pela minimização da Energia Livre Esperada (EFE) ([Equação 2.4.6](#)). A ideia central é que, ao minimizar a EFE, o agente está escolhendo ações que ele espera que minimizem a surpresa de observações futuras, tornando-as consistentes com seu modelo ([seção 2.3](#)).

O método `act` (Código 3.14) começa por inferir a política ótima utilizando o método `infer_policies`, (HEINS et al., 2022a). A política ótima aqui é a sequência de ações que o agente acredita que minimizará a EFE no futuro. Uma vez que a política ótima é inferida, o agente seleciona uma ação dela usando o método `sample_action`. Esta abordagem de seleção de ação está alinhada com a formulação da AIF como POMDP, onde agentes operam com base em crenças sobre o estado do ambiente e selecionam ações que maximizam uma função objetivo, neste caso, minimizando a EFE (seção 2.4).

Código 3.14 – Método `act` da classe `ActiveInferenceAgent`

```

1 def act(self):
2     """
3     Select an action based on current beliefs.
4     """
5     # Infer the optimal policy based on current beliefs
6     self.infer_policies()
7
8     # Sample an action from the inferred policy
9     action = self.sample_action()
10    return action

```

3.4 Sistema PingPOMDP

O sistema **PingPOMDP** é a plataforma experimental que desenvolvemos para atingir o objetivo deste trabalho (seção 1.2): avaliar a capacidade de um agente baseado em AIF de adaptar-se e aprender a interagir adequadamente com um ambiente Pong, tal como os nerônios *in vitro* fizeram no DishBrain (KAGAN et al., 2022).

PingPOMDP é composto por três componentes principais que podem ser representados resumidamente na Figura 14:

1. **Ambiente Pong:** o ambiente virtual que simula o clássico jogo Pong (subseção 3.1.3).
2. **Agente de Inferência Ativa:** o modelo computacional baseado na teoria da AIF (seção 3.3), que mantém crenças sobre o estado do ambiente e age com base nessas crenças.
3. **Interface GridLink:** a interface entre o agente e o ambiente Pong subseção 3.2.4. Ela é responsável por traduzir o estado do ambiente em observações para o agente e mapear as ações do agente para movimentos no ambiente, utilizando protocolos análogos aos propostos por Kagan et al. (2022).

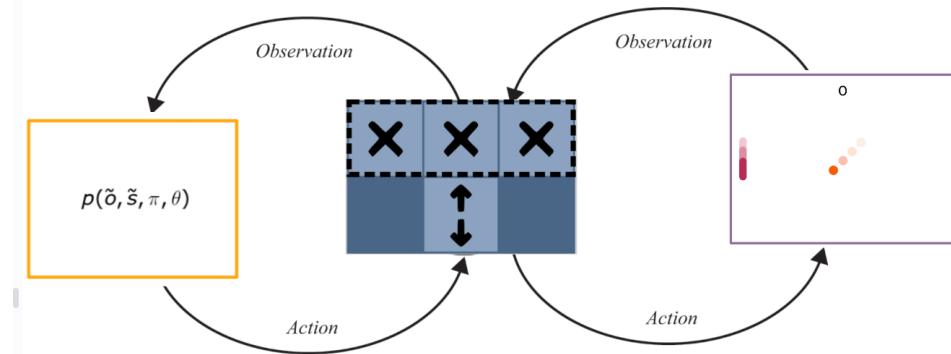


Figura 14 – Representação geral dos principais componentes do PingPOMDP

Fonte: própria

A Figura 15 (reiteração da Figura 3), trás um pouco mais de detalhe sobre a estrutura do PingPOMDP e seus componentes.

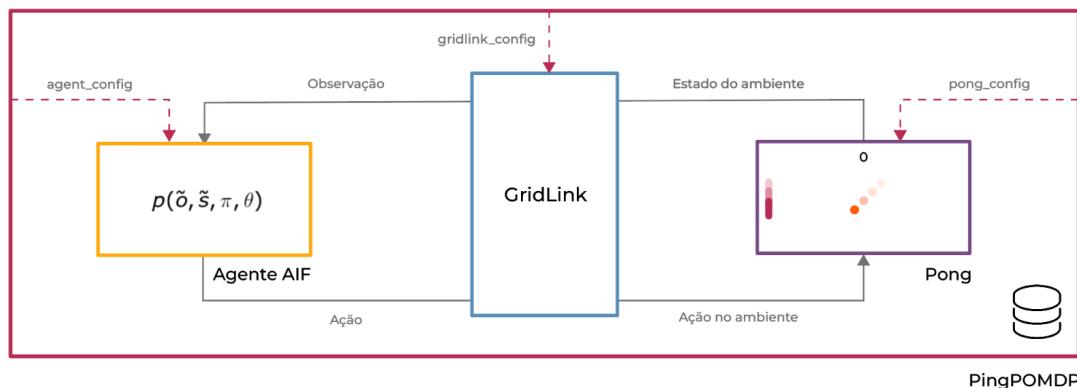


Figura 15 – Modelo Arquitetural do PingPOMDP

Fonte: própria

Dessa forma, o PingPOMDP integra e orquestra esses componentes ao exercer três funções principais:

- **Inicialização e Configuração:** o PingPOMDP inicializa e configura os três componentes principais – o Ambiente Pong, o Agente AIF (ou Controle) e a Interface GridLink. Esta etapa envolve definir configurações específicas, como as sementes aleatórias para o agente e o ambiente, garantindo reproduzibilidade e consistência no experimento.
- **Execução de Passos Experimentais:** uma vez configurado, o PingPOMDP conduz cada etapa do experimento. Isso envolve gerenciar a coleta de dados e provocar cada execução do método `step()` da GridLink, que avança um passo no ambiente e no agente, conforme vimos na Figura 11.
- **Gerenciamento de Experimentos e Dados:** além disso, o sistema PingPOMDP gerencia o armazenamento e a análise dos dados experimentais. Isso inclui registrar cada passo do

experimento, as ações do agente, os estados do jogo e os resultados observados, assegurando a integridade dos dados e facilitando a análise posterior dos resultados.

A figura [Figura 16](#) proporciona uma visão geral da interação e das funções dos componentes dentro do sistema PingPOMDP, enfatizando como eles trabalham em conjunto para alcançar os objetivos do experimento.

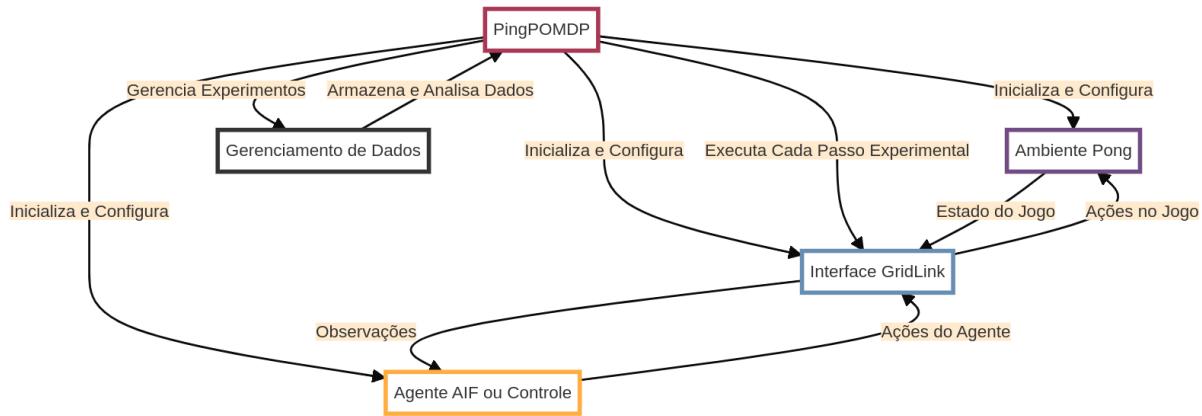


Figura 16 – Diagrama de fluxo do PingPOMDP interagindo com seus componentes

Fonte: própria

3.4.1 Estrutura do Código

O código do sistema PingPOMDP, apresentado na Figura 17 é organizado em várias classes e funções que trabalham em conjunto para realizar o experimento de interação entre o agente e o ambiente.

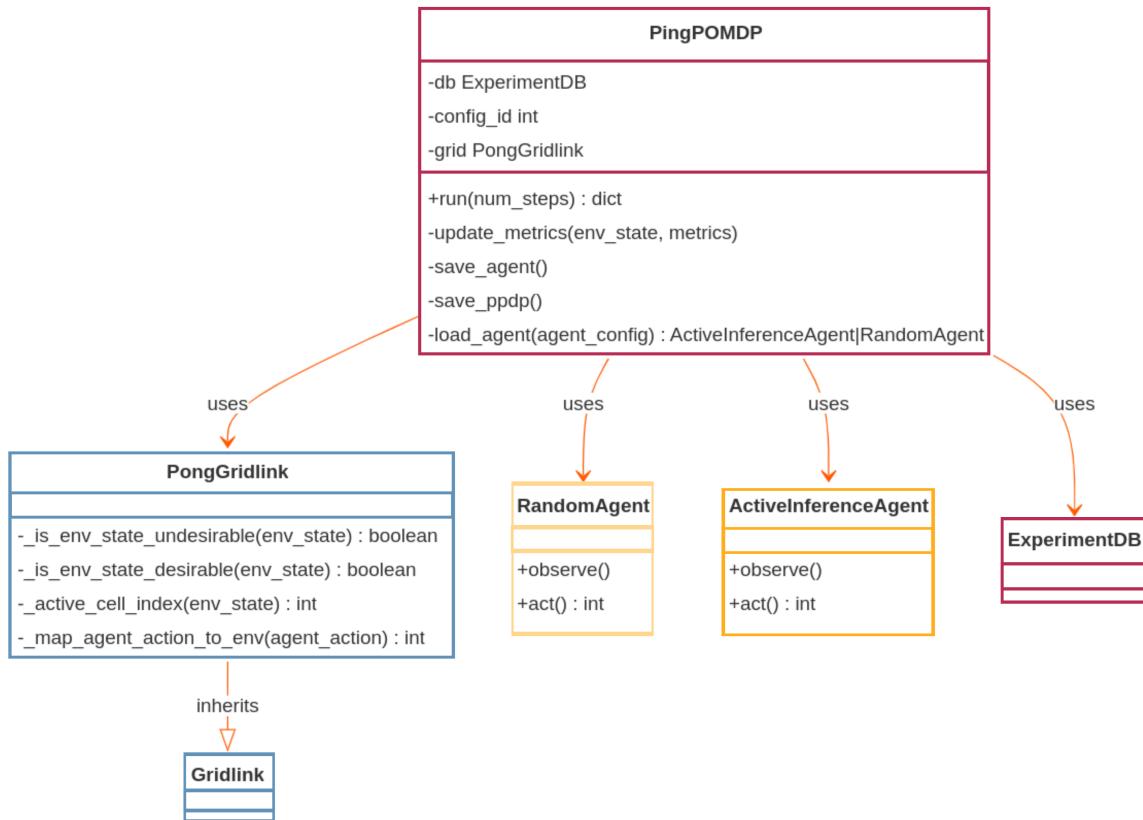


Figura 17 – Diagrama de classes do PingPOMDP.

Fonte: própria

À seguir, detalhamos cada um desses componentes.

3.4.1.1 PingPOMDP

Esta é a classe principal que integra todos os componentes. Ela inicializa o ambiente, o agente e a interface `GridLink` conforme o Código 3.15.

Código 3.15 – Método `init` da classe `GridLink`

```

1 def __init__(self, seed=1, ...):
2 ...
3     self.db = ExperimentDB()
4 ...
5     self.grid = PongGridlink(
6         agent=agent,
  
```

```

7     env=env,
8     **gridlink_config)

```

Além disso, ela contém métodos para:

- Executar o experimento por um número especificado de etapas (método `run`).
- Salvar e carregar o estado do agente (métodos `load_agent` e `save_agent`).

O método `run` (Código 3.16) é o principal método dessa classe. Ele inicializa o registro do experimento no banco de dados e executa cada iteração avançando o experimento por meio da grid (subseção 3.2.4), registrando os dados de cada passo. Após o experimento, o método insere os dados finais no banco de dados e salva o estado do agente.

Código 3.16 – Método `run` da classe `GridLink`

```

1 self.db.start_experiment(...)
2 def run(self, num_steps=10):
3     ...
4     for i in range(num_steps):
5         self.grid.step()
6         ...
7         self.db.insert_step(...)
8     ...
9 self.db.finalize_experiment(...)
10 self.save_agent()

```

3.4.1.2 PongGridlink

A classe `PongGridlink` é uma extensão da classe `Gridlink` e serve como a interface entre o ambiente Pong e o agente de AIF. Conforme vimos na subseção 3.2.4.

3.4.2 ActiveInferenceAgent

Esta classe define o agente que utiliza AIF para tomar decisões com base em suas crenças sobre o estado do ambiente. Conforme vimos na seção 3.3, o agente atualiza continuamente suas crenças com base nas observações recebidas e decide sobre a ação mais adequada a ser tomada.

3.4.2.1 ControlAgent

Esta classe representa um agente simples que toma decisões de maneira aleatória (Código 3.17). O objetivo desta classe é servir como uma base de comparação, permitindo avaliar o desempenho do `ActiveInferenceAgent` em contraste com um agente que não realiza inferências ativas.

- Método `observe`: recebe uma observação, mas não faz nada com ela, já que o agente é aleatório.
- Método `act`: retorna uma ação aleatória, que pode ser 0 ou 1.

Código 3.17 – Métodos `observe` e `act` da classe `ControlAgent`

```

1 def observe(self, observation):
2     pass
3 def act(self):
4     return np.random.randint(0, 2)

```

3.4.2.2 Configurações

O código também contém várias configurações que definem parâmetros para o ambiente Pong, o agente e a interface GridLink. Estas configurações são passadas como argumentos ao inicializar o sistema PingPOMDP, conforme vemos no [Código 3.18](#).

Código 3.18 – Configurações da classe PingPOMDP

```

1 pong_config = {
2     "screen_width": 320, "screen_height": 480, ...
3 }
4 agent_config = {
5     "n_obs": 3, "n_states": 2, ...
6 }
7 gridlink_config = {
8     "grid_shape": (2, 3), ...
9 }
10 p = PingPOMDP(
11     pong_config=pong_config,
12     agent_config=agent_config,
13     gridlink_config=gridlink_config,
14 )
15 p.run(num_steps=40_000)

```

3.4.2.3 ExperimentDB

A classe `ExperimentDB` é responsável por gerenciar o banco de dados que armazena informações sobre cada experimento realizado.

Dentre os métodos disponíveis nessa classe, vale ressaltar o `get_config_id` ([Código 3.19](#)) que cria uma identificação única no banco para cada **conjunto de parâmetros** que define a configuração de um experimento. Ele é chamado ao criar um objeto da classe

PingPOMDP. Se a configuração for idêntica à de um experimento anterior, o método retorna o ID da configuração existente. Caso contrário, insere a nova configuração e retorna seu ID.

Código 3.19 – Método `get_config_id` da classe `ExperimentDB`

```

1 get_config_id(self,
2   random_seed,
3   pong_config,
4   agent_config,
5   gridlink_config)
```

A classe `ExperimentDB` também possui métodos específicos registrar por meio do método `run` da classe principal PingPOMDP. Estes métodos são responsáveis por inserir, atualizar e gerenciar informações relacionadas aos experimentos realizados:

1. **Método `start_experiment`:** este método é utilizado para iniciar um novo registro de experimento na tabela `Experiment` do banco de dados. Os dados inseridos incluem:

- `config_id`: uma identificação única para a configuração do experimento.
- `start_time`: o horário de início do experimento.
- `notes`: qualquer anotação ou observação relacionada ao experimento, sendo um campo opcional.

Após a inserção, o método retorna o ID do último registro inserido, que representa o ID do experimento.

2. **Método `finalize_experiment`:** este método é utilizado para finalizar um registro de experimento. Ele atualiza um registro específico na tabela `Experiment` com informações sobre o término do experimento. Os dados atualizados incluem:

- `end_time`: horário de término do experimento.
- `steps_taken`: número total de etapas realizadas durante o experimento.
- `results`: resultados obtidos ao final do experimento, armazenados em formato JSON.

O registro a ser atualizado é identificado pelo `experiment_id`, que é fornecido como parâmetro ao método.

3. **Método `insert_step`:** este método insere detalhes de uma etapa específica do experimento na tabela `Steps`. os dados inseridos para cada etapa incluem:

- `experiment_id`: ID do experimento ao qual a etapa pertence.
- `step_num`: número da etapa dentro do experimento.
- `agent_action`: ação realizada pelo agente durante essa etapa.

- *environment_state*: estado do ambiente após a ação do agente, armazenado em formato JSON.

Semelhante ao método `start_experiment`, este método retorna o ID do último registro inserido após a inserção.

3.5 Desenho de Experimentos

Com a implementação do sistema PingPOMDP e seus respectivos componentes finalizada, procedemos à realização de uma série de experimentos. O objetivo central destes experimentos é coletar dados significativos, calcular métricas relevantes e comparar os resultados obtidos. Essa abordagem experimental é essencial para testar nossa hipótese de pesquisa e cumprir com os objetivos estabelecidos neste estudo. A seguir, apresentamos os métodos adotados para a avaliação efetiva da hipótese e a realização dos objetivos propostos.

Inicialmente, descrevemos o desenho dos experimentos realizados com o DishBrain, incluindo suas configurações e as métricas adotadas. Essa descrição serve como base para a apresentação dos métodos utilizados em nossos próprios experimentos, bem como para a compreensão das variáveis consideradas e dos procedimentos de coleta de dados.

3.5.1 Condições Experimentais do DishBrain

Para avaliar os protocolos propostos para integrar as culturas neurais com o ambiente Pong, Kagan et al. (2022) utilizou as seguintes condições experimentais ilustradas pela Figura 18:

- HCC e MCC (Human Cortical Cells e Mouse Cortical Cells): principais objetos de estudo do DishBrain. Representam as culturas de neurônios submetidas aos protocolos propostos. Essa condição está relacionada com o agente **AA** em nossos experimentos.
- IS (*In Silico*): controle aleatório da raquete, equivalente ao nosso **AC**.
- CTL (*Media-Only Controls*): placa de eletrodos (HD-MEA) vazia, sem nenhuma célula. Essa condição busca isolar efeitos do próprio meio. Tentamos emular essa condição na nossa pesquisa utilizando agentes com crenças uniformes (**AU**), visto que esperamos serem inertes as observações.
- RST (*Rest Sessions*): HD-MEA com neurônios, porém sem nenhum estímulo. Essa condição não se aplica a nossa modelagem devido a impossibilidade de fornecer silêncio como observação ao nosso agente.

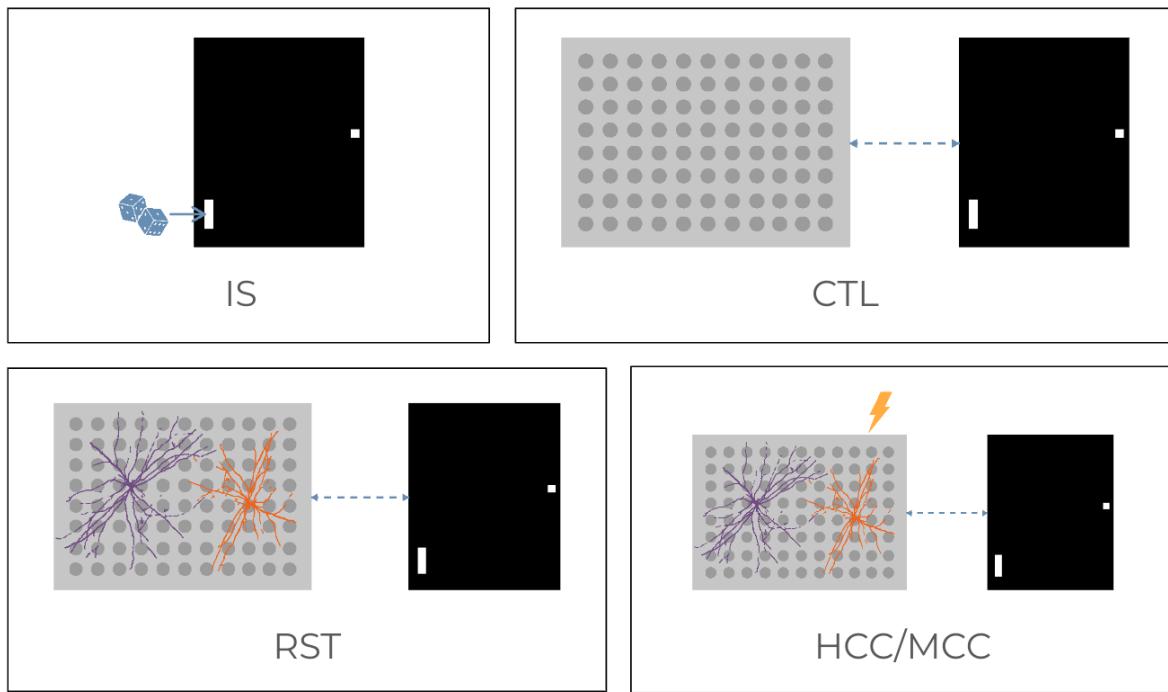


Figura 18 – Ilustração das condições experimentais do DishBrain

Fonte: própria

Além dessas condições experimentais gerais, Kagan et al. (2022) também comparou três condições específicas para investigar os neurônios (HCC e MCC), representadas na Figura 19:

- **STIM (Stimulus)**: protocolo padrão do DishBrain que apresentamos na [subseção 3.2.1.1](#), com estímulo sensorial representando a posição da bola, mais feedback previsível para *hits* e imprevisível para *misses*. Essa é a condição principal a ser investigada.
- **SIL (Silent)**: substitui os feedbacks por silêncio. Mantendo o estímulo sensorial e o relançamento da bola. Assim como na condição experimental RST, não é possível administrar ausência de observação ao nosso agente.
- **NF (No-Feedback)**: considera que os períodos de silêncio podem ainda ser considerados um tipo de feedback, já que os silêncios ocorrem apenas após *hits* ou *misses*. Portanto, nessa condição o estímulo sensorial é mantido constante e ininterrupto durante toda a sessão. Essa condição experimental é testada em nosso experimento ao adotar 0 ciclos de feedback.

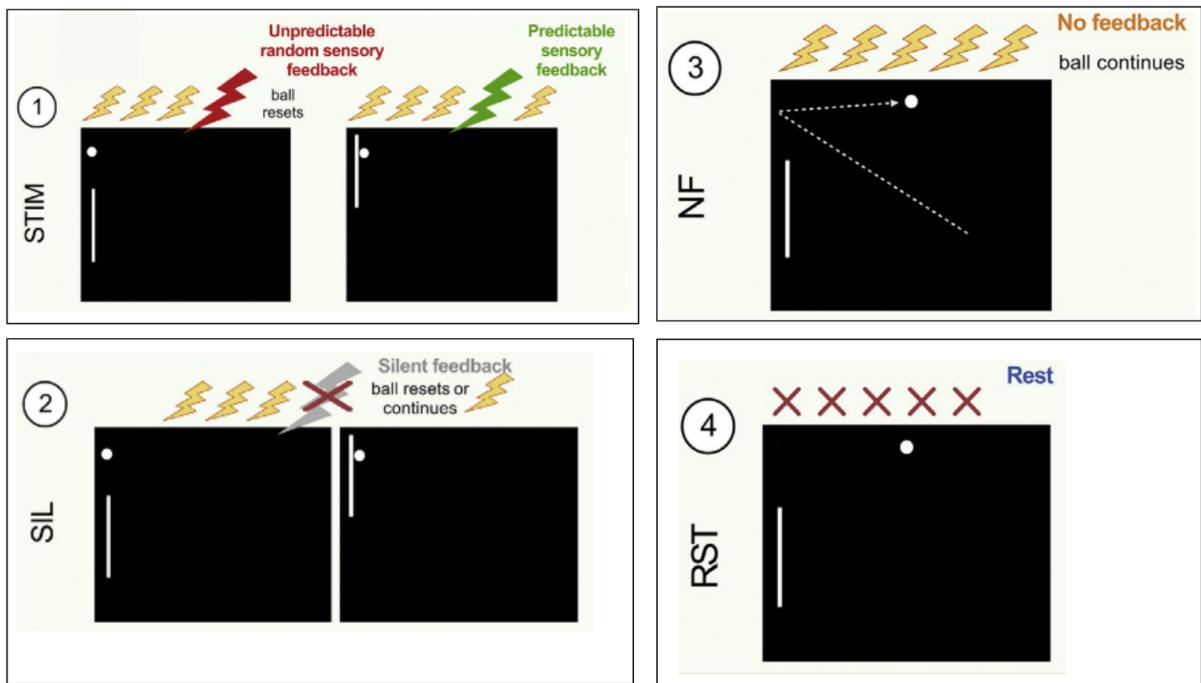


Figura 19 – Ilustração das condições experimentais das células corticais no DishBrain

Fonte: Kagan et al. (2022)

Assim, em nossos experimentos com o PingPOMDP, utilizamos diferentes tipos de agente como referências às condições experimentais mais representativas do DishBrain.

3.5.2 Configuração dos Agentes no PingPOMDP

Conforme vimos na [subseção 3.3.3.1](#), temos dois tipos de agentes de AIF:

- Agentes com crenças Aleatórias (AA).
- Agentes com crenças Uniformes (AU).

Nesse sentido, as condições experimentais dos neurônios (HCC e MCC) no DishBrain são abstraídas como agentes da categoria AA. Com eles testaremos nossas versões da condição **STIM** e **NF**.

Já os Agentes de crenças Uniformes, que esperamos serem "inertes" ([subseção 3.3.3.1](#)) ao aprendizado, serão utilizados como um grupo controle análogo a condição experimental CTL do DishBrain, onde foi testado o efeito da estrutura na ausência das células ativamente aprendendo.

Finalmente, nosso Agente Controle (AC), definido na [subseção 3.4.2.1](#), equivale a condição experimental **IS** do DishBrain, na qual a raquete é controlada aleatoriamente.

3.5.3 Métricas do DishBrain

[Kagan et al. \(2022\)](#) utilizou as seguintes métricas para avaliar o desempenho de cada condição experimental:

- **Average Rally Lenth:** duração média da partida, definida como a razão entre *hits* e *misses*.
- **Long Rallies:** partidas com três ou mais *hits*.
- **Aces:** partidas com nenhum *hit*.

Cada sessão experimental do DishBrain durou 20 minutos e foi dividida nos períodos T1 (primeiros 5min) e T2 (últimos 15min). Cada uma das métricas foi calculada e comparada entre os períodos T1 e T2. A plataforma DishBrain operou em intervalos de 10ms.

3.5.4 Métricas no PingPOMDP

Como nosso sistema opera em passos discretos (isto é, contáveis e não contínuos), cada sessão experimental do DishBrain equivaleria a 120.000 iterações no PingPOMDP, considerando uma sessão com duração de 20min e 10ms como unidade de básica de tempo¹.

Além da conversão de tempo contínuo para iterações discretas, utilizaremos as mesmas métricas de [Kagan et al. \(2022\)](#) para avaliar o desempenho de cada agente:

- **Duração Média de Partida (average_rally_length):** número de *hits* para cada *miss*.
- **Taxa de Partidas Zeradas (adaptação do aces):** número de partidas sem nem um acerto em relação ao total de partidas.
- **Partidas Longas (long_rallies):** número de partidas com três ou mais acertos.

3.5.4.1 Parâmetros Experimentais no PingPOMDP

Conforme vimos na [subseção 3.4.2.2](#), o sistema PingPOMDP foi desenhado para definir um experimento por meio de um conjunto de parâmetros. Esses parâmetros são agrupados entre parâmetros do Agente, do Pong e da GridLink.

Dessa forma, nessa pesquisa, cada conjunto de experimento manteve uma mesma configuração fixa entre os experimentos, variando apenas o parâmetro a ser estudado a partir daquele conjunto de experimentos.

¹ $20\text{min} \cdot 60 = 1.200\text{s}; 1.200\text{s} / 10\text{ms} = 120.000$ iterações

3.5.4.2 Parâmetros do Agente

O principal parâmetro dos agentes é a sua categoria: AA, AU e AC. Além disso, um agente é definido por sua semente aleatória (`agent_seed`), seu espaço de observação (`n_obs`) e espaço de estados (`n_states`). Adotamos o padrão observações de tamanho 3, que geram um espaço de estados de tamanho 8 (2^3), e espaço de estados equivalente.

3.5.4.3 Parâmetros do Pong

O ambiente Pong foi desenvolvido ([subseção 3.1.3](#)) para ser facilmente configurável, desde o tamanho e velocidade da bola e raquete até as dimensões da tela. Dentre os parâmetros vale ressaltar que definimos a altura da raquete para ser proporcional a 10% da altura da tela, e o lançamento da bola como aleatório.

No contexto de nossa pesquisa o único parâmetro que alteramos entre os experimentos foi a semente aleatória que define o vetor de lançamento da bola.

3.5.4.4 Parâmetros da GridLink

A GridLink permite a personalização de diversos parâmetros, como vimos na [seção 3.2](#). Em nossos experimentos, adotamos a grid de formato (2,3), com a primeira linha de células sensoriais, sendo enviada como observação ao agente.

A partir das definições apresentadas na [subseção 3.2.1.1](#), definimos o número padrão de ciclos dos feedbacks previsível e imprevisível:

- Feedback Imprevisível: 4 segundos de estímulo a 5Hz: $4s \times 5Hz = 20$ estímulos
- Feedback Previsível: 100 milisegundos de estímulo a 100Hz: $0.1s \times 100Hz = 10$ estímulos

As sementes aleatórias do agente e do ambiente também são definidas nas configurações da GridLink. Assim, o [Código 3.20](#) apresenta o dicionário de configurações da GridLink no contexto do PingPOMDP.

Código 3.20 – Dicionário de configurações da GridLink

```

1 gridlink_config = {
2     "grid_shape": (2, 3),
3     "sensory_cells": (0, 1, 2),
4     "observation_mode": "sensory_cells",
5     "n_predictable_cycles": 10,
6     "n_unpredictable_cycles": 20,
7     'agent_seed': agent_seed,
8     'env_seed': env_seed
9 }
```


4 Resultados e Discussões

Neste capítulo, apresentamos e discutimos os resultados obtidos com o sistema Ping-POMDP, interpretando-os no contexto da hipótese central deste projeto e comparando-os com os achados da literatura.

4.1 Testes preliminares

Inicialmente, realizamos uma série de experimentos exploratórios para desenvolver uma intuição sobre os protocolos finais. Estes testes preliminares incluíram a análise do comportamento de agentes do tipo AA com sementes aleatórias variando de 1 a 100 e a determinação do número de iterações necessárias para obter resultados representativos. Com base nesses testes, decidimos utilizar as sementes aleatórias 1 e 2 para inicializar os agentes e definimos que um experimento padrão consistiria em 40.000 iterações.

4.1.1 Mapeamento de sementes aleatórias

Realizamos experimentos para mapear o desempenho de diferentes agentes do tipo AA ao longo de 12.000 iterações, com a semente de ambiente fixada em 3 e variando a semente do agente de 1 a 100 (denominados AA1, AA2, ..., AA100). A métrica de **Total de Partidas Longas** (número de partidas com pelo menos três hits seguidos) foi calculada para cada experimento e representada no mapa de calor da [Figura 20](#). O eixo vertical representa cada agente com semente aleatória entre 1 e 100, o eixo horizontal representa o número total de **Partidas Longas** que o agente apresentou.

Os resultados indicaram que a maioria das sementes aleatórias resultou em agentes incapazes de alcançar partidas longas, mas algumas sementes geraram agentes com desempenho notável, como as sementes 2 e 4. Esse achado já sugere que pelo menos alguns agentes baseados em AIF podem exibir desempenho comparável aos neurônios *in vitro* do DishBrain.

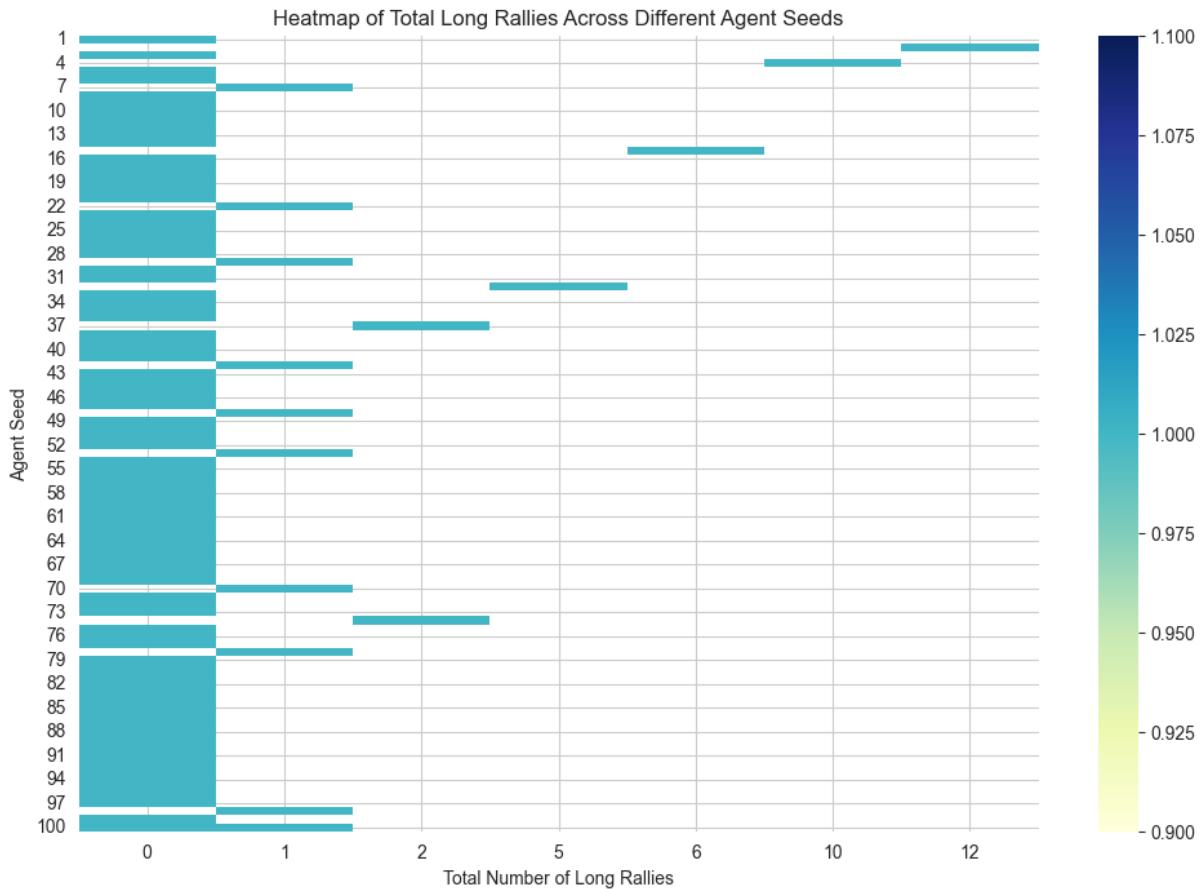


Figura 20 – Total Partidas Longas de agentes AA1 até AA100

Fonte: própria

Dessa forma, selecionamos duas sementes representativas desses dois casos, a semente 2 (agente AA2) para representar o grupo de agentes que demonstraram aprendizado, e a semente 1 (agente AA1) para representar o grupo dos agentes que não conseguiram nem uma partida longa (pelo menos uma sequência de três *hits*).

4.1.2 Número de iterações

Com as duas sementes representativas selecionadas, investigamos a possibilidade de reduzir o número de iterações sem comprometer a robustez dos experimentos. Observamos na Figura 21 que o desempenho dos agentes tende a se estabilizar por volta das 10.000 iterações. Assim, optamos por utilizar quatro vezes esse número, definindo 40.000 como o padrão de iterações de um experimento, o que corresponde a $\frac{1}{5}$ da duração proporcional de uma sessão experimental do DishBrain, conforme calculamos na subseção 3.5.4.

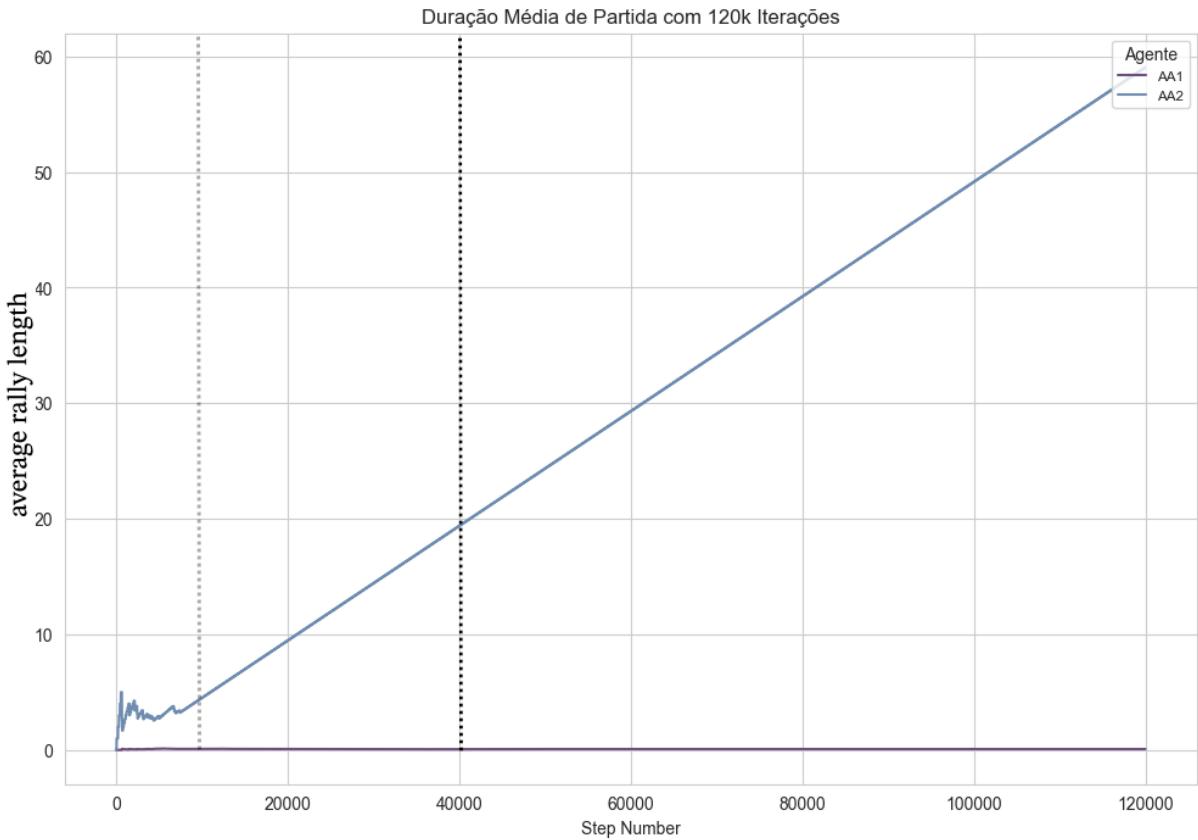


Figura 21 – Duração Média de Partida ao longo de 120.000 iterações

Fonte: própria

4.2 Resultados dos Experimentos com Agentes

A partir dos testes preliminares, conduzimos experimentos com os três tipos de agentes - AC, AU e AA - utilizando duas sementes aleatórias distintas (1 e 2). Esses agentes foram testados em cinco ambientes Pong, cada um gerado com uma semente aleatória ambiente diferente. Avaliamos cada agente nas cinco variações de ambiente com foco principal na métrica de Duração Média de Partidas, além do número de partidas longas e partidas zeradas.

4.2.1 Comparação entre agentes com semente aleatória 1

O gráfico representado na Figura 22 é um lineplot comparando os três agentes criados com `agent_seed = 1`. Cada um desses experimentos foi repetido alternando a `env_seed` do ambiente Pong entre 1 e 5. Assim, cada linha do gráfico representa a média de duração das partidas para cada agente, com uma área sombreada indicando o desvio padrão. Observamos que o agente AC teve a performance mais baixa, seguido de perto pelo AU. O agente AA demonstrou uma performance superior, com uma média claramente mais alta. Contudo, a variação significativa observada indica uma disparidade nos resultados dos cinco experimentos. Além disso, vale

ressaltar que todas as médias estão no intervalo de 0 a 0,15, sugerindo que as diferenças entre as modalidades dos agentes, nessas condições específicas, são pouco significativas.

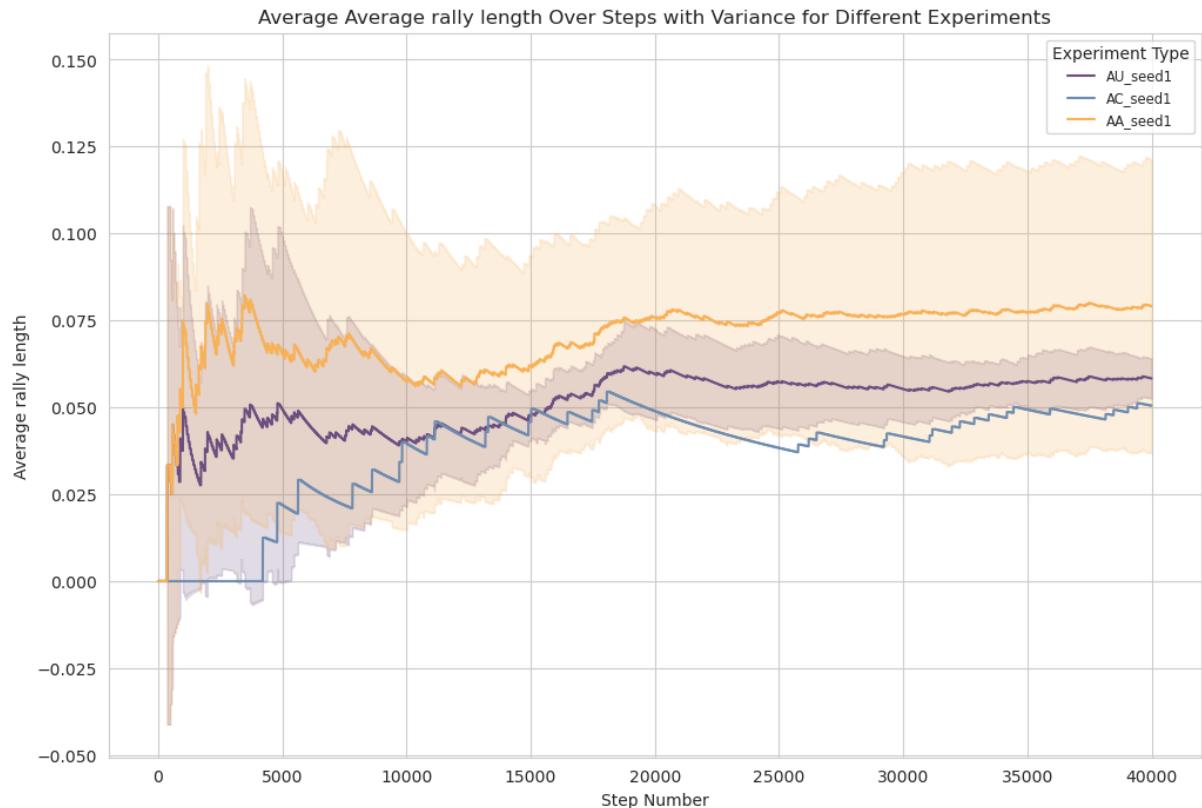


Figura 22 – Duração média de partida dentre agentes com semente 1

Fonte: própria

O próximo gráfico, representado na Figura 23 apresenta uma versão boxplot do mesmo conjunto de dados. Aqui, percebe-se uma menor variabilidade nos resultados dos agentes em T2, com as performances se estabilizando em um patamar similar. Mesmo com a pouca diferenciação, o agente AC mostrou o pior desempenho tanto em T1 quanto em T2, mas com a maior variação entre os períodos. O agente AA, por outro lado, se destacou com desempenho quase alcançando 0,2. Esses dados reforçam a ideia de que, dada a proximidade dos valores com zero, qualquer diferença entre as três modalidades de agentes nessas circunstâncias tem baixa significância.

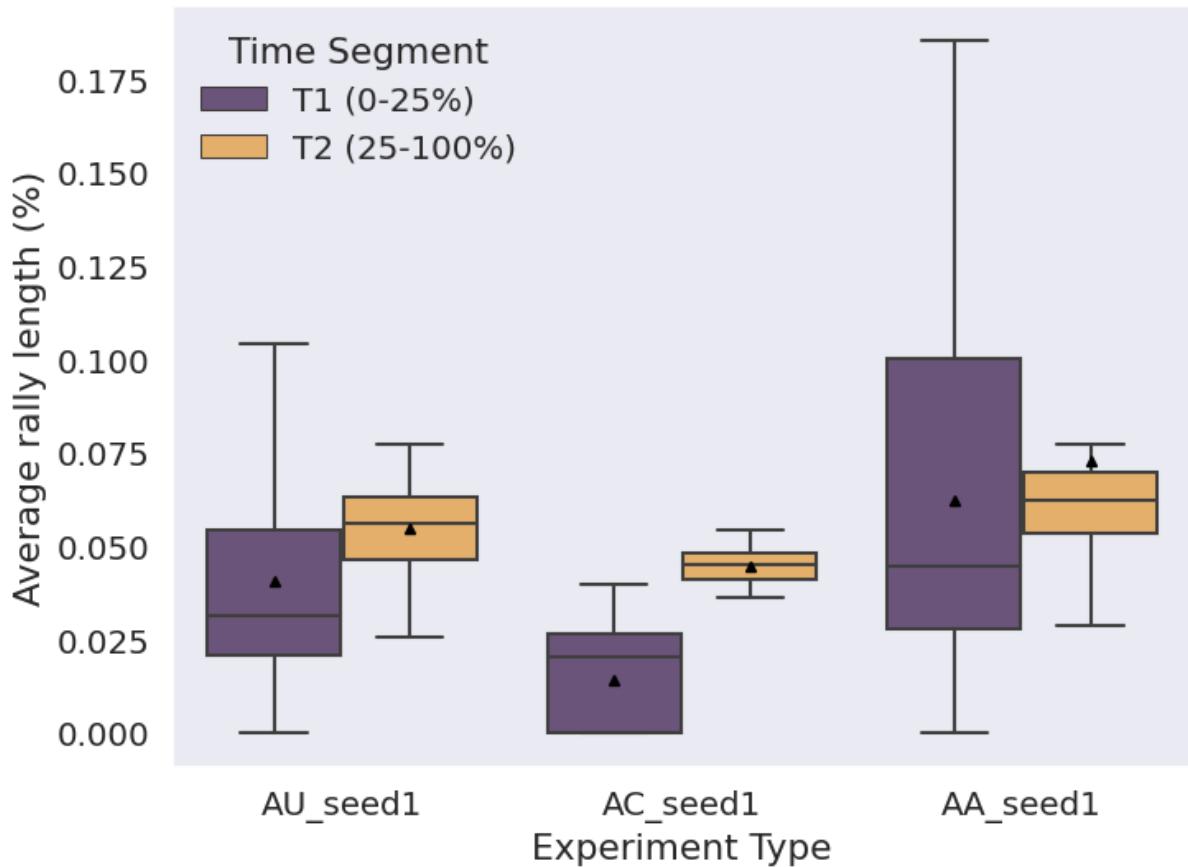


Figura 23 – Duração média de partida dentre agentes com semente 1

Fonte: própria

4.2.2 Comparação entre Performance dos Agentes com semente aleatória 2

A Figura 24 ilustra os experimentos realizados com a semente aleatória 2. Nota-se um aprendizado notável por parte do agente AA, que se inicia em torno da 5.000^a iteração e exibe uma trajetória ascendente, alcançando quase 30 *hits* em média, com picos de 50 *hits* em certos ambientes. Essas métrica provavelmente seguiria aumentando, devido a natureza linearmente positiva da linha que observamos no gráfico. Os agentes AC e AU, por outro lado, se assemelham a linhas horizontais próximas ao eixo y=0, sugerindo um aprendizado substancial do agente AA em comparação.

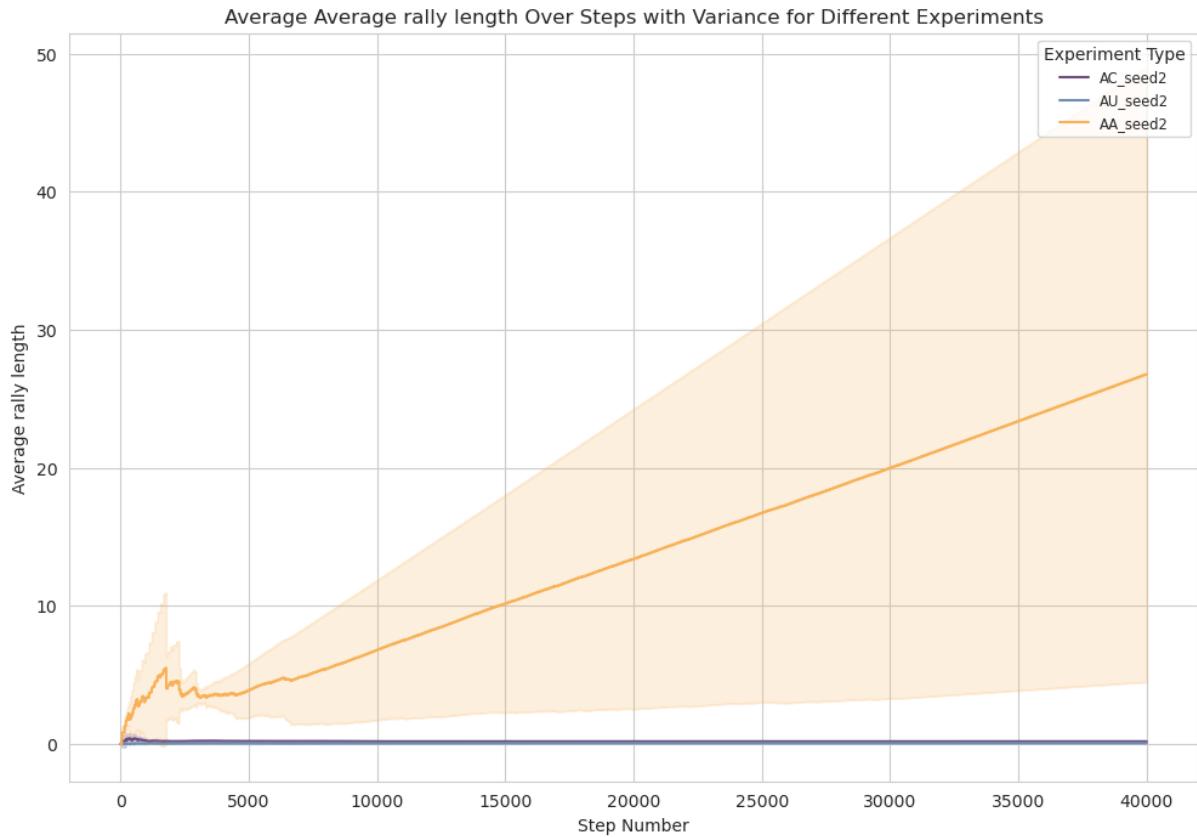


Figura 24 – Duração média de partida dentre agentes com semente 2

Fonte: própria

Similarmente, a Figura 25 exibe esses resultados em formato de *boxplot*. Observa-se que o desempenho do agente AA se sobressai, com uma clara progressão entre T1 e T2. O intervalo interquartil se mostra quase separado, com uma média significativamente superior. AC e AU se assemelham a linhas linhas no gráfico devido a escala necessária para apresentar o desempenho do AA. Esses resultados corroboram com a observação de aprendizagem na Figura 24.

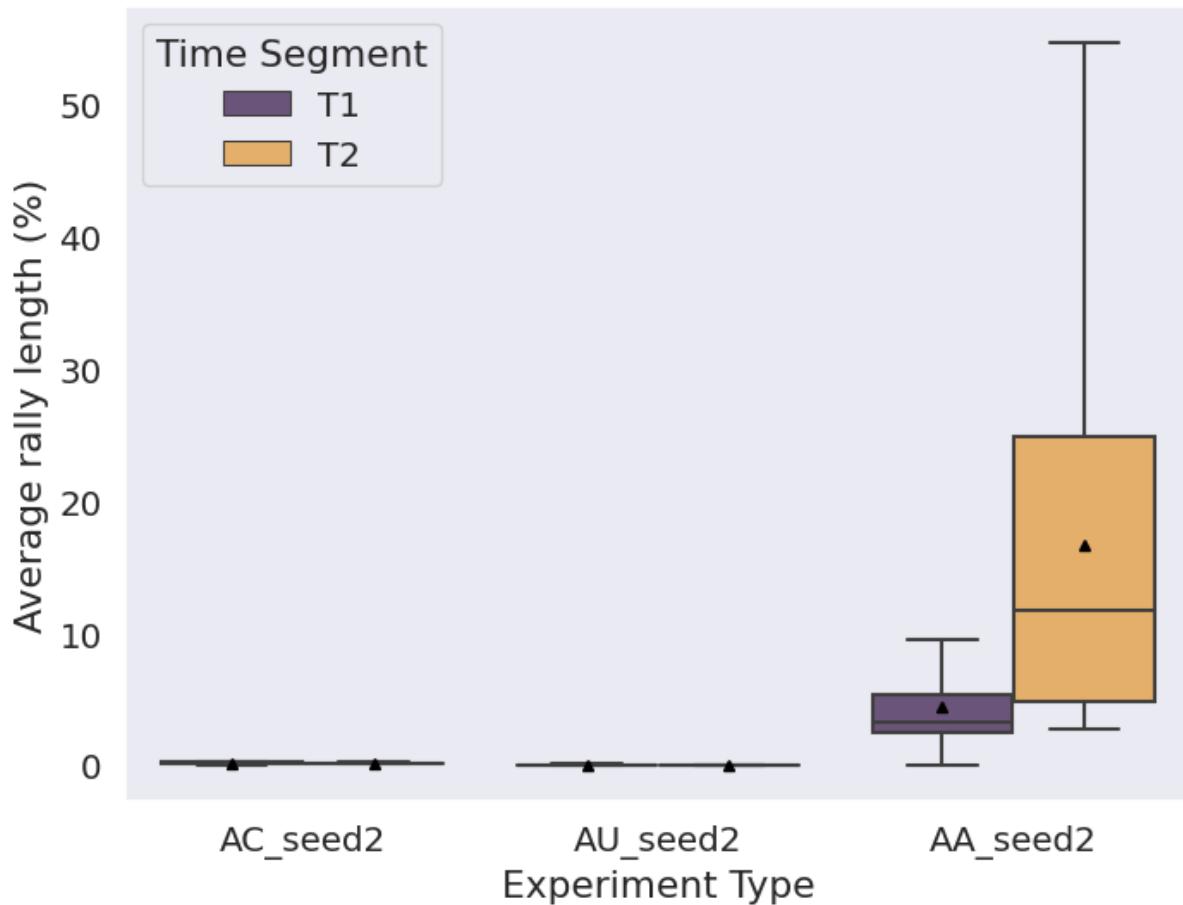


Figura 25 – Duração média de partida dentre agentes com semente 2

Fonte: própria

4.2.3 Comparação Entre Agentes Com e Sem Feedbak

A Figura 26 apresenta provavelmente o resultado mais importante dessa pesquisa. Nela podemos observar o mesmo agente no qual observamos desempenho expressivo, sendo contrastando com uma mesma versão dele, em ambiente idêntico, porém, sem receber os feedbacks previsíveis e imprevisíveis propostos nos protocolos que estamos investigando. Assim como Kagan et al. (2022) compararamos um agente idêntico, que recebia apenas os estímulos sensoriais da localização da boa, sem receber feedback previsível e imprevisível.

Desse modo, observamos uma diferença expressiva na performance do agente com ausência de feedback. A duração média das partidas desse agente não chegou a 10, e observamos uma linha consideravelmente horizontal, e de baixa variabilidade entre experimentos. Indicando que sem esse feedback houve pouco estímulo a aprendizagem desse agente. Sugerindo que o protocolo de feedback proposto no DishBrain e modelado nessa pesquisa, é uma estrutura promissora.

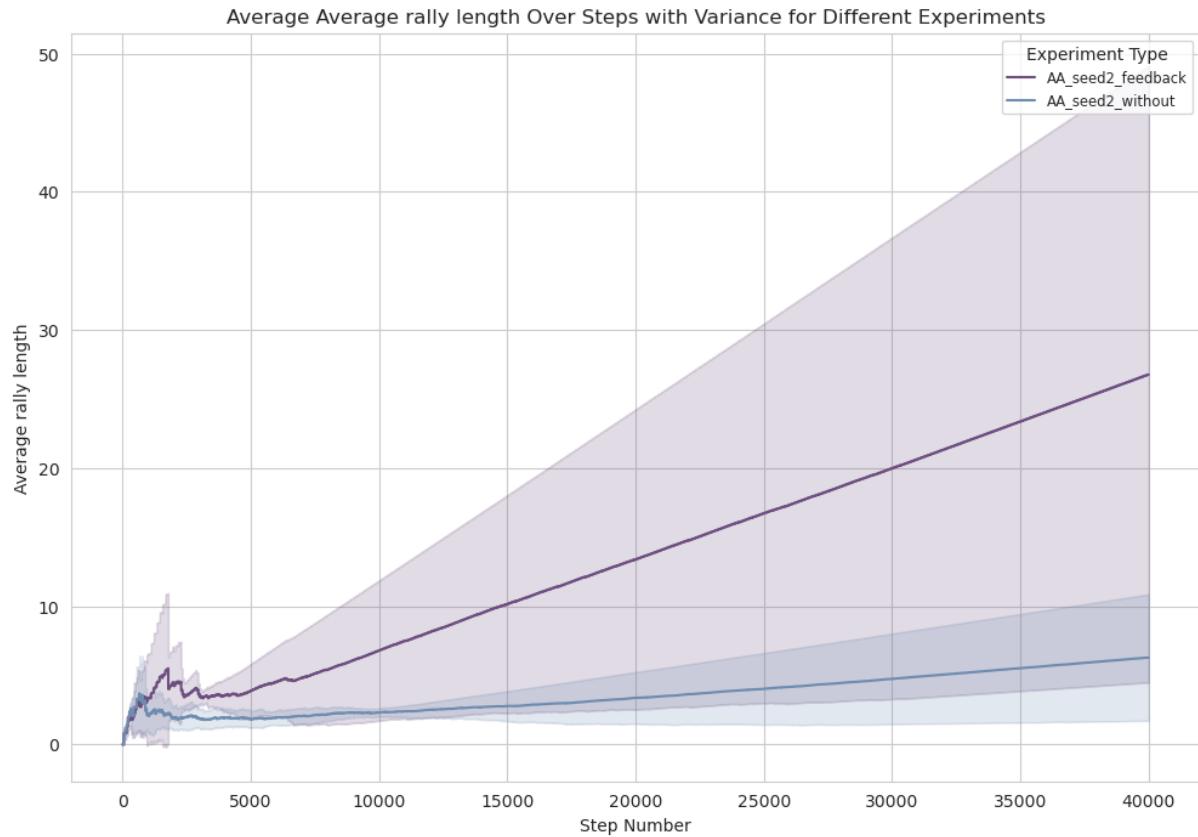


Figura 26 – Comparaçāo de agentes com e sem feedback

Fonte: própria

A Figura 27 apresenta esse efeito de forma ainda mais clara. Ela evidencia como a distribuição com feedback apresenta maior dispersão, o que sugere comportamento exploratório, além do evidente progresso em T2.

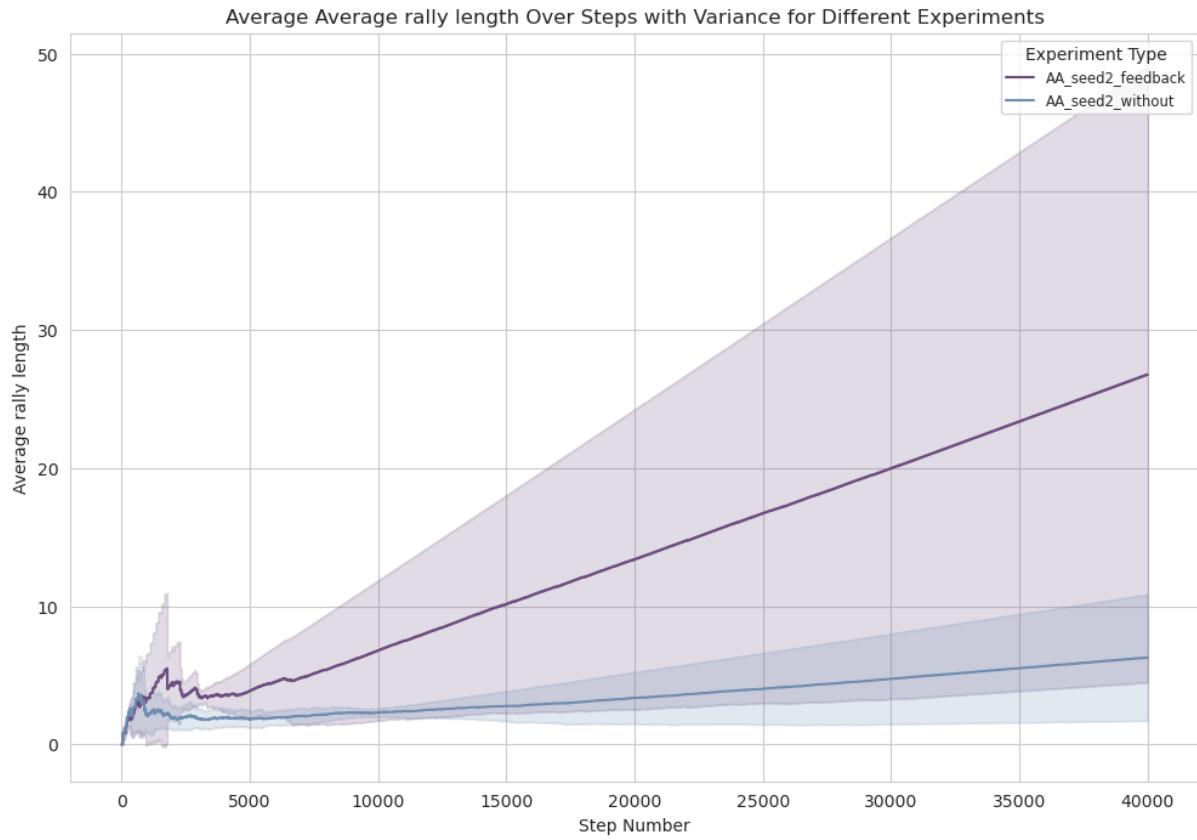


Figura 27 – Comparação de agentes com e sem feedback

Fonte: própria

4.2.4 Comparação com o DishBrain

Na Figura 28, apresentamos os principais resultados encontrados por Kagan et al. (2022). Nela observamos um *boxplot* das três métricas computadas entre os diferentes experimentos das cinco categorias de "agentes" (CTL, IS, RST, MCC, e HCC), conforme apresentamos na subseção 3.5.1. Já na Figura 6 vemos as mesmas métricas computadas nas condições experimentais do **PingPOMDP**, conforme discutido na subseção 3.5.2.

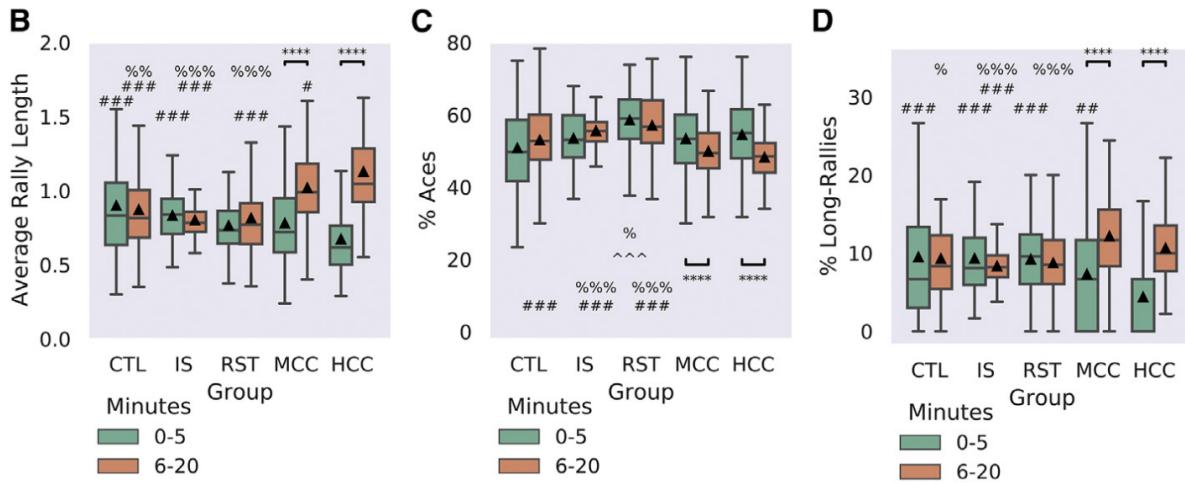


Figura 28 – Resultados das métricas do DishBrain comparando desenhos experimentais

B - Average Rally Length; C - % Aces, e D - % Long-Rallies

Fonte: (KAGAN et al., 2022)

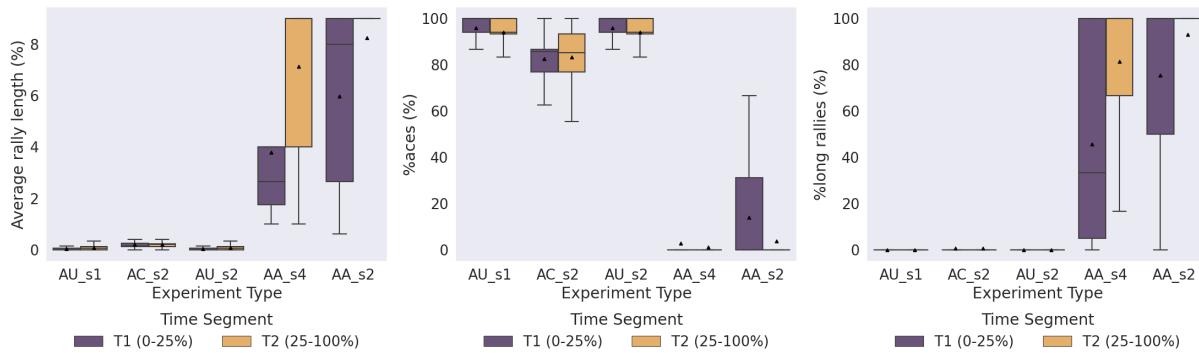


Figura 29 – Resultados das métricas do PingPOMDP comparando desenhos experimentais

Fonte: própria

Desse modo, observamos na Figura 28 que todas as condições experimentais de controlo do DishBrain (IS, CTL e RST) apresentaram distribuições bastante similares. Porém, os neurônios de ratos (MCC) e principalmente os neurônios humanos (HCC) apresentaram distribuições diferentes das condições controle, especialmente na diferença de T1 para T2. Assim, observamos que as células corticais apresentaram melhora de desempenho considerável na duração média de partida, na diminuição de *aces*, e no aumento do número de partidas longas.

Comparando esses resultados aos nossos achados observamos características similares. A partir da Figura 29, vemos que as distribuições dos nossos agentes de controle (AU_S1, AC_s2 e AU_s2) foram bem similares entre si, e bem diferentes das distribuições dos agentes de nossas condições experimentais de interesse (AA_s4 e AA_s2). Em nosso caso a diferença para os grupos controle foi bastante acentuada. Houve clara melhora no desempenho entre T1 e T2.

Também observamos algumas diferenças ao comparar nossos dados aos resultados do Kagan et al. (2022). A principal delas é a escala do efeito observado, tanto nas distribuições de performance nas métricas quanto na diferença de entre T1 e T2. Inclusive em algumas métricas a distribuição foi representada como uma reta, denotando que o agente adquiriu controle perfeito da raquete no período anterior, portanto não houve variações na métrica a partir desse ponto.

É importante ressaltar que embora nossos resultados sejam numericamente superiores aos encontrados por Kagan et al. (2022), nosso sistema é expressivamente mais simples, e abstrai diversas dificuldades que objetos de estudo biológicos impõem. Além disso, conforme apresentamos na [seção 4.1](#), nosso sistema é muito mais sensível a semente aleatória e configurações iniciais.

4.3 Discussão de resultados

Os resultados indicam que, de fato, um agente baseado em AIF, inserido em um ambiente virtual de Pong e submetido a protocolos inspirados no DishBrain, podem apresentar aprendizado. Indicam ainda que esse aprendizado é consideravelmente limitado na ausência dos protocolos de feedback. Desse modo, consideramos que nossa hipótese de trabalho foi confirmada.

4.3.1 Influência da Estrutura do Modelo Generativo no Comportamento do Agente

Nossos resultados confirmam nossa expectativa teórica ([subseção 3.3.3.1](#)) de que agentes equipados com matrizes de crenças uniformes seriam incapazes de aprendizado, indicado que, de fato, tratar todos os estados como igualmente prováveis de levarem a qualquer observação por meio de qualquer ação, torna a sua estrutura não-discriminante. Ou seja, atualizações de crenças e inferência de políticas não são informativas para esses agentes.

Já as matrizes A e B inicializadas aleatoriamente dispõem de variabilidade suficiente para permitir que o agente construa crenças mais precisas, e faça inferências mais adequadas do mundo. Porém, boa parte das matrizes de crenças possíveis também geram agentes incapazes de aprender. De modo que aprofundar nesse contraste e investigar exatamente quais são as crenças e os fatores específicos dessas matrizes que permitem ou impedem o aprendizado é um possível desdobramento dessa pesquisa.

5 Considerações Finais

Neste trabalho, nos aprofundamos na exploração da Inferência Ativa (AIF), uma teoria emergente e promissora para o entendimento da cognição e do comportamento de organismos através de princípios fundamentais. A relevância deste estudo se destaca ao considerarmos o crescente interesse e a aplicabilidade da AIF na literatura recente.

A inspiração deste trabalho foi a abordagem inovadora do DishBrain, que integrou neurônios a um ambiente virtual do jogo Pong. Nossa trabalho propôs um modelo computacional dos protocolos propostos pelo DishBrain.

Nesse sentido, ao desenvolver o sistema PingPOMDP e a interface GridLink, nosso trabalho demonstrou a viabilidade da integração de agentes baseados em AIF em ambiente virtual por meio dos protocolos propostos pelo DishBrain. Além disso, ao estabelecer um paralelo entre os sistemas biológicos e computacionais, esse estudo permite futuras explorações sobre como organismos processam informação e interagem com seus ambientes sob a ótica da AIF, o que pode lançar luz sobre os mecanismos fundamentais que regem o comportamento adaptativo.

5.1 Relevância e Contribuições

Analizando o trabalho de [Kagan et al. \(2022\)](#), que integrou neurônios a um ambiente virtual do jogo Pong, nosso estudo se propôs a avaliar se um modelo computacional baseado em AIF poderia replicar o comportamento adaptativo desses neurônios. Enquanto o DishBrain utilizava um sistema biológico real, nosso modelo utilizou uma abstração matemática das dinâmicas da AIF, proporcionando uma nova perspectiva sobre o experimento realizado em um contexto computacional.

Os resultados do nosso estudo indicam que um agente baseado em AIF, quando inserido em um ambiente virtual de Pong e submetido a protocolos inspirados no DishBrain, é capaz de demonstrar aprendizado. Este achado é notável, pois ressalta a capacidade da AIF de informar a implementação de um agente capaz de aprendizado, em ambiente e condições comparáveis aos neurônios *in vitro* no DishBrain.

Ademais, observamos que esse aprendizado é limitado na ausência dos protocolos de feedback, o que sugere a importância desses protocolos no processo de aprendizagem e adaptação do agente. Assim, confirmamos nossa hipótese [seção 1.2](#) de trabalho, demonstrando que um modelo computacional baseado em AIF pode, de fato, emular aspectos do comportamento adaptativo observado em sistemas biológicos, embora com limitações inerentes ao seu design.

Esta comparação direta com o DishBrain não só valida a aplicabilidade dos protocolos propostos em modelos computacionais, mas também destaca as nuances e limitações que

surgem ao traduzir processos biológicos para um ambiente virtual. Isso reforça a importância de abordagens interdisciplinares na compreensão da cognição e do comportamento, tanto em organismos biológicos quanto em agentes artificiais.

A importância deste estudo é evidenciada pela inovação introduzida por Kagan et al. (2022). Ao evidenciar que neurônios podem aprender de forma significativa por meio de protocolos específicos, o DishBrain pavimentou o caminho para uma compreensão mais aprofundada dos processos neuronais e da Inferência Ativa. O potencial de progresso é amplo, abrangendo desde o entendimento da cognição e comportamento até o desenvolvimento de novos paradigmas computacionais, como redes neurais bio-sintéticas conforme descrito por Kagan et al. (2022).

Nosso trabalho constitui um passo inicial para explorar computacionalmente os protocolos do DishBrain. Nesse sentido, a principal contribuição desse trabalho é o desenvolvimento e disponibilização do sistema **PingPOMDP** e da interface **GridLink**. A GridLink permite utilizar os protocolos do DishBrain para integrar um agente computacional a um ambiente virtual, enquanto o sistema PingPOMDP permite gerenciar experimentos com a GridLink, ambientes e agentes. Ambas podem ser ferramentas valiosas para pesquisadores que desejem avançar na área de pesquisa emergente.

5.2 Limitações

A principal limitação dessa pesquisa é a simplificação inerente ao modelar sistemas biológicos complexos em ambientes computacionais. Enquanto o PingPOMDP busca replicar o comportamento do DishBrain, nossa pesquisa abstraiu as nuances e detalhes específicos do sistema *in vitro*.

Além disso, nossos experimentos computacionais exploraram um subconjunto restrito do espaço de parâmetros possível a partir da nossa implementação. Também não foi possível, devido ao escopo deste trabalho, nos aprofundarmos nos mecanismos internos de aprendizagem dos agentes. Da mesma forma, o escopo teórico desse trabalho foi limitado a AIF devido a sua extensão e complexidade. Seria importante, por exemplo, expandir esse referencial para abranger as outras abordagens dentro da modelagem de agentes computacionais, bem como considerar mais referenciais dentro da área de Inteligência Artificial.

5.3 Sugestões para Pesquisas Futuras

Este estudo representa uma aplicação da AIF em contextos computacionais, abrindo novas direções para pesquisas futuras. A seguir, listamos alguns desses possíveis desdobramentos:

Variar mais parâmetros experimentais como o número de ciclos de feedbacks espaço de estados velocidade da bola, etc Representar frequência e tensão elétrica nos estímulos Analisar os achados frente a literatura clássica de aprendizado de máquina dentre outras

- Analisar detalhadamente as matrizes A e B de crenças dos agentes. O que permitiu que aprendessem e como? Nossa escolha por utilizar observações de tamanho 3 com duas ações possíveis facilitará esse trabalho de interpretabilidade.
- Explorar mais variações nos parâmetros experimentais como o número de ciclos de feedback, o espaço de estados, etc. Também podemos alterar parâmetros do Pong para avaliar se agentes treinados em determinada configuração conseguiram generalizar para outra.
- Desenvolver abstrações mais precisas para tensão e frequência dos "pulsos elétricos" utilizados pelo DishBrain de modo a aumentar a fidelidade dessa representação.
- Levantar a vasta literatura sobre Inteligência Artificial para, contrastar com e expandir sobre, o referencial teórico utilizado nessa pesquisa, que restringiu seu escopo à AIF, conforme a [seção 2.1](#).
- Desacoplar os componentes genéricos dos específicos da classe PingPOMDP. Possivelmente criar uma nova classe, denominada GridXP, que gerencie toda a parte de experimentação. A classe PingPOMDP se tornaria uma subclasse de GridXP, herdando suas funcionalidades e adicionando as configurações e parâmetros específicos do nosso contexto de Pong, inspirado pelo DishBrain. Isso facilitaria a criação de novos experimentos que utilizem a GridLink através da GridXP.

Assim, acreditamos que, por meio de esforços contínuos e colaborativos, podemos abrir novas fronteiras na computação e neurociência, além de avançar substancialmente no entendimento da cognição e do comportamento de todos os organismos, incluindo nós mesmos.

Referências

- ADAMS, R. A. et al. Everything is connected: Inference and attractors in delusions. *Schizophrenia Research*, Elsevier BV, Jul 2021. Disponível em: <<https://doi.org/10.1016/j.schres.2021.07.032>>. Citado 2 vezes nas páginas 17 e 18.
- ADAMS, R. A. et al. Everything is connected: Inference and attractors in delusions. *Schizophrenia Research*, v. 245, p. 5–22, Jul 2022. ISSN 0920-9964. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0920996421003054>>. Citado 2 vezes nas páginas 17 e 36.
- BAIOUMY, M. et al. Towards stochastic fault-tolerant control using precision learning and active inference. In: *Machine Learning and Principles and Practice of Knowledge Discovery in Databases*. Springer International Publishing, 2022. p. 681–691. Disponível em: <https://doi.org/10.1007/978-3-030-93736-2_48>. Citado na página 17.
- BALTIERI, M.; BUCKLEY, C. L. Pid control as a process of active inference with linear generative models. *Entropy*, Multidisciplinary Digital Publishing Institute, v. 21, n. 33, p. 257, Mar 2019. ISSN 1099-4300. Disponível em: <<https://www.mdpi.com/1099-4300/21/3/257>>. Citado na página 17.
- BOGACZ, R. A tutorial on the free-energy framework for modelling perception and learning. *Journal of Mathematical Psychology*, v. 76, p. 198–211, fev. 2017. ISSN 0022-2496. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0022249615000759>>. Citado na página 23.
- CHAWLA, S. et al. Ten years after imagenet: a 360° perspective on artificial intelligence. *Royal Society Open Science*, v. 10, n. 3, p. 221414, 2023. Disponível em: <<https://royalsocietypublishing.org/doi/abs/10.1098/rsos.221414>>. Citado na página 24.
- COSTA, L. D. et al. Active inference on discrete state-spaces: A synthesis. *Journal of Mathematical Psychology*, v. 99, p. 102447, Dec 2020. ISSN 0022-2496. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S0022249620300857>>. Citado 12 vezes nas páginas 17, 23, 24, 25, 26, 27, 28, 31, 32, 33, 36 e 57.
- FLORIDI, L.; CHIRIATTI, M. Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines*, Springer, v. 30, p. 681–694, 2020. Citado na página 24.
- FOUNTAS, Z. et al. Deep active inference agents using monte-carlo methods. In: *Advances in Neural Information Processing Systems*. Advances in Neural Information Processing Systems, 2020. Disponível em: <<https://proceedings.neurips.cc/paper/2020/hash/865dfbde8a344b44095495f3591f7407-Abstract.html>>. Citado na página 18.
- FRISTON, K. The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, Nature Publishing Group, v. 11, n. 22, p. 127–138, Feb 2010. ISSN 1471-0048. Disponível em: <<https://www.nature.com/articles/nrn2787>>. Citado 3 vezes nas páginas 17, 23 e 33.
- FRISTON, K.; KILNER, J.; HARRISON, L. A free energy principle for the brain. *Journal of Physiology-Paris*, v. 100, n. 1, p. 70–87, jul. 2006. ISSN 0928-4257. Disponível em:

<<https://www.sciencedirect.com/science/article/pii/S092842570600060X>>. Citado na página 23.

FRISTON, K. et al. Active inference and epistemic value. *Cognitive Neuroscience*, Routledge, v. 6, n. 4, p. 187–214, out. 2015. ISSN 1758-8928. Disponível em: <<https://doi.org/10.1080/17588928.2015.1020053>>. Citado na página 23.

FRISTON, K.; SAMOTHRAKIS, S.; MONTAGUE, R. Active inference and agency: optimal control without cost functions. *Biological Cybernetics*, v. 106, n. 8, p. 523–541, out. 2012. ISSN 1432-0770. Disponível em: <<https://doi.org/10.1007/s00422-012-0512-8>>. Citado na página 23.

FRISTON, K.; TRUJILLO-BARRETO, N.; DAUNIZEAU, J. Dem: A variational treatment of dynamic systems. *NeuroImage*, v. 41, n. 3, p. 849–885, 2008. ISSN 1053-8119. Disponível em: <<https://www.sciencedirect.com/science/article/pii/S1053811908001894>>. Citado na página 34.

FRISTON, K. J.; DAUNIZEAU, J.; KIEBEL, S. J. Reinforcement learning or active inference? *PLOS ONE*, Public Library of Science, v. 4, n. 7, p. e6421, jul. 2009. ISSN 1932-6203. Disponível em: <<https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0006421>>. Citado na página 23.

HARRIS, C. R. et al. Array programming with NumPy. *Nature*, Springer Science and Business Media LLC, v. 585, n. 7825, p. 357–362, set. 2020. Disponível em: <<https://doi.org/10.1038/s41586-020-2649-2>>. Citado 2 vezes nas páginas 35 e 50.

HEINS, C. et al. pymdp: A python library for active inference in discrete state spaces. *Journal of Open Source Software*, v. 7, n. 73, p. 4098, May 2022. ISSN 2475-9066. ArXiv:2201.03904 [cs, q-bio]. Disponível em: <<http://arxiv.org/abs/2201.03904>>. Citado 12 vezes nas páginas 17, 21, 26, 27, 28, 33, 34, 36, 56, 57, 59 e 60.

HEINS, C. et al. pymdp: A python library for active inference indiscrete state spaces. *Journal of Open Source Software*, v. 7, n. 73, p. 4098, May 2022. ISSN 2475-9066. Disponível em: <<https://joss.theoj.org/papers/10.21105/joss.04098>>. Citado na página 33.

HOLMES, E. et al. Active inference, selective attention, and the cocktail party problem. *Neuroscience & Biobehavioral Reviews*, Elsevier, Sep 2021. Disponível em: <<https://doi.org/10.1016/j.neubiorev.2021.09.038>>. Citado na página 17.

ISOMURA, T.; SHIMAZAKI, H.; FRISTON, K. J. Canonical neural networks perform active inference. *Communications Biology*, v. 5, n. 1, p. 55, 2022. Disponível em: <<https://doi.org/10.1038/s42003-021-02994-2>>. Citado na página 24.

JOHNSON, S. A.i. is mastering language. should we trust what it says? *The New York Times*, Apr 2022. Disponível em: <<https://www.nytimes.com/2022/04/15/magazine/ai-language.html>>. Citado na página 24.

KAGAN, B. J. et al. In vitro neurons learn and exhibit sentience when embodied in a simulated game-world. *Neuron*, p. S0896627322008066, Oct 2022. ISSN 08966273. Disponível em: <<https://linkinghub.elsevier.com/retrieve/pii/S0896627322008066>>. Citado 26 vezes nas páginas 18, 19, 20, 21, 22, 23, 24, 35, 39, 40, 41, 42, 45, 46, 47, 60, 67, 68, 69, 70, 79, 81, 82, 83, 85 e 86.

KENT, S. L. *The Ultimate History of Video Games, Volume 1: From Pong to Pokemon and Beyond... the Story Behind the Craze That Touched Our Lives and Changed the World*. [S.l.]: Crown, 2010. v. 1. Citado 2 vezes nas páginas 40 e 41.

- KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F. et al. (Ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2012. v. 25. Disponível em: <https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf>. Citado na página 23.
- MILLIDGE, B. et al. On the relationship between active inference and control as inference. In: *International Workshop on Active Inference*. Springer International Publishing, 2020. p. 3–11. Disponível em: <https://doi.org/10.1007/978-3-030-64919-7_1>. Citado 2 vezes nas páginas 17 e 18.
- MONTAGUE, P. R. et al. Computational psychiatry. *Trends in Cognitive Sciences*, Elsevier Ltd, Jan 2012. Disponível em: <<https://doi.org/10.1016/j.tics.2011.11.018>>. Citado na página 17.
- PARR, T.; PEZZULO, G.; FRISTON, K. J. *Active Inference: The Free Energy Principle in Mind, Brain, and Behavior*. The MIT Press, 2022. ISBN 9780262369978. Disponível em: <<https://doi.org/10.7551/mitpress/12441.001.0001>>. Citado 6 vezes nas páginas 17, 23, 25, 26, 30 e 36.
- PARR, T. et al. Prefrontal computation as active inference. *Cerebral Cortex*, Oxford University Press, Mar 2020. Disponível em: <<https://doi.org/10.1093/cercor/bhz118>>. Citado na página 17.
- ROSSUM, G. V.; DRAKE, F. L. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009. ISBN 1441412697. Citado 3 vezes nas páginas 21, 35 e 50.
- RUSSELL, S.; RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. Pearson, 2020. (Pearson series in artificial intelligence). ISBN 9780134610993. Disponível em: <<https://books.google.com.br/books?id=koFptAEACAAJ>>. Citado 2 vezes nas páginas 23 e 24.
- SAJID, N. et al. Active inference: Demystified and compared. *Neural Computation*, MIT Press, Mar 2021. Disponível em: <https://doi.org/10.1162/neco_a_01357>. Citado na página 18.
- SAVAGE, N. How ai and neuroscience drive each other forwards. *Nature*, v. 571, p. S15–S17, 2019. Disponível em: <<https://www.nature.com/articles/d41586-019-02212-4>>. Citado na página 17.
- SCHWARTENBECK, P. et al. Exploration, novelty, surprise, and free energy minimization. *Frontiers in Psychology*, v. 4, 2013. ISSN 1664-1078. Disponível em: <<https://www.frontiersin.org/articles/10.3389/fpsyg.2013.00710>>. Citado na página 23.
- SCHWARTENBECK, P. et al. The dopaminergic midbrain encodes the expected certainty about desired outcomes. *Cerebral Cortex*, Oxford University Press, Sep 2015. Disponível em: <<https://doi.org/10.1093/cercor/bhu159>>. Citado na página 17.
- SMITH, R.; BADCOCK, P.; FRISTON, K. J. Recent advances in the application of predictive coding and active inference models within clinical neuroscience. *Psychiatry and Clinical Neurosciences*, v. 75, n. 1, p. 3–13, 2021. ISSN 1440-1819. Disponível em: <<https://onlinelibrary.wiley.com/doi/abs/10.1111/pcn.13138>>. Citado 3 vezes nas páginas 26, 27 e 36.
- SMITH, R.; FRISTON, K.; WHYTE, C. A step-by-step tutorial on active inference and its application to empirical data. PsyArXiv, Jan 2021. Disponível em: <<https://psyarxiv.com/b4jm6/>>. Citado 10 vezes nas páginas 17, 23, 25, 26, 27, 28, 29, 34, 36 e 57.

SMITH, R. et al. Greater decision uncertainty characterizes a transdiagnostic patient sample during approach-avoidance conflict: A computational modelling approach. *Journal of Psychiatry & Neuroscience*, Canadian Medical Association, Jan 2021. Disponível em: <<https://doi.org/10.1503/jpn.200032>>. Citado na página 17.

SMITH, R. et al. Imprecise action selection in substance use disorder: Evidence for active learning impairments when solving the explore-exploit dilemma. *Drug and Alcohol Dependence*, Elsevier, Dec 2020. Disponível em: <<https://doi.org/10.1016/j.drugalcdep.2020.108208>>. Citado na página 17.

THAGARD, P. Cognitive Science. In: ZALTA, E. N.; NODELMAN, U. (Ed.). *The Stanford Encyclopedia of Philosophy*. Spring 2023. [S.l.]: Metaphysics Research Lab, Stanford University, 2023. Citado na página 17.

TISON, R.; POIRIER, P. Communication as socially extended active inference: An ecological approach to communicative behavior. *Ecological Psychology*, Taylor & Francis, Apr 2021. Disponível em: <<https://doi.org/10.1080/10407413.2021.1965480>>. Citado na página 18.

TSCHANTZ, A. et al. Scaling active inference. In: *2020 International Joint Conference on Neural Networks (IJCNN)*. [S.l.: s.n.], 2020. p. 1–8. ISSN 2161-4407. Citado 11 vezes nas páginas 17, 19, 23, 25, 26, 27, 28, 30, 31, 33 e 36.

TSCHANTZ, A. et al. Scaling active inference. In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020. p. 1–8. Disponível em: <<https://doi.org/10.1109/IJCNN48605.2020.9207382>>. Citado na página 18.

TSCHANTZ, A. et al. Reinforcement learning through active inference. In: *Bridging AI and Cognitive Science at the International Conference on Learning Representations*. Bridging AI and Cognitive Science at the International Conference on Learning Representations, 2020. Disponível em: <https://baicsworkshop.github.io/pdf/BAICS_37.pdf>. Citado na página 18.

WANG, X.-J. et al. Computational neuroscience: a frontier of the 21st century. *National Science Review*, v. 7, 09 2020. Citado na página 17.

WIRKUTTIS, N.; TANI, J. Leading or following? dyadic robot imitative interaction using the active inference framework. *IEEE Robotics and Automation Letters*, IEEE, Apr 2021. Disponível em: <<https://doi.org/10.1109/LRA.2021.3090015>>. Citado na página 18.

WOLF, M. J. *The video game explosion: a history from PONG to Playstation and beyond*. [S.l.]: Bloomsbury Publishing USA, 2007. Citado na página 40.