NOME DO DOCUMENTO

Machine_learning_Report (2).pdf

NÚMERO DE PALAVRAS

6217 Words

NÚMERO DE CARACTERES

34431 Characters

NÚMERO DE PÁGINAS

15 Pages

TAMANHO DO ARQUIVO

612.2KB

DATA DE ENVIO

Jan 15, 2024 6:38 PM GMT

DATA DO RELATÓRIO

Jan 15, 2024 6:39 PM GMT

17% geral de similaridade

O total combinado de todas as correspondências, incluindo fontes sobrepostas, para cada banco de

- 16% Banco de dados da Internet
- Banco de dados do Crossref
- 7% Banco de dados de trabalhos enviados
- 5% Banco de dados de publicações
- Banco de dados de conteúdo publicado no Cross

Relatório de Pesquisa Sobre *Machine Learning*

ANDRÉ PUTOI, Universidade da Beira Interior, Portugal
CARLOTA SANTOS, Universidade Da Beira Interior, Portugal

Machine learning é uma área da Inteligência Artificial, referido ,«mais à ideia à automatização dos modelos analíticos. Ao analisar a informação pode identificar padrões e tomar decisões reduzindo assim a intervenção humana. Com tudo, como futuros cientistas de dados, é necessário saber sua definição técnica e utilidade.

Additional Key Words and Phrases: IA, Machine Learning, Supervised Learning, Learning Decision Trees, Reinforcement Learning, UBI

ACM Reference Format:
André Putoi and Carlota Santos. 2023. Relatório de Pesquisa Sobre *Machine Learning*. , 1 (December 2023), 15 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Contents

Authors' addresses: André Putoi, Universidade da Beira Interior , Covilhã, Portugal; Carlota Santos, Universidade Da Beira Interior, Covilhã, Portugal.

XXXX-XXXX/2023/12-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUÇÃO

Neste relatório iremos abordar sobre e tipos de *Machine Learning*, explicá-los e dar exemplos onde podem ser usados.

Em prólogo da cadeira de Laboratórios de Informática, estamos a fazer um relatório de pesquisa sobre *Machine Learning*, sub-campo da IA, para isso temos de começar pela Inteligência Artificial. Ao longo do tempo [8], pesquisadores procuraram por definir inteligência, entre essas definições, as mais aceitas são:

- Definir inteligência em termos à fidelidade da performance humana.
- Definir inteligência como racionalidade, de forma mais abstrata, podendo ser separado por outras duas formas:
  - Considerar inteligência como propriedade interna do processo de pensamento;
  - Enquanto outros consideram a inteligência como comportamento, e por uma **caracterização externa**.

Em relação à **caracterização externa**, a mesma pode referir-se como opinião publica, dito isto, muitas pessoas acreditam que *machine learning* é igual a Inteligência Artificial, mas existe IAs que não fazem parte de *Machine Learning*.

Neste trabalho iremos dar foco ao *Machine Learning*, mas este é um detalhe importante a realçar.

1.1 Inteligência Artificial vs *Machine Learning*

IA e *Machine Learning* [5], as vezes, são usados em simultâneo, mas como dito antes, ambas têm definições bem distintas.

Inteligência Artificial (IA) é um termo usado no *software* de computadores, onde imita-se o raciocínio humano (forma de pensar) com o objetivo de completar tarefas complexas e aprender a partir delas.

Machine learning (ML) é um dos campos da IA que usa algoritmos treinados com informação, de forma a produzir modelos adaptáveis que conseguem concluir uma variedade de tarefas complexas.

Agora que abordamos a diferença destes temas, podemos passar para a origem do *Machine Learning*.

1.2 História do *Machine Learning*

No *Machine Learning*[6] , varios pesquisadores tiveram um papel importante nas suas origens, iremos descrever um pouco de cada um, e outras evoluções importantes na área.

1.2.1 Arthur Samuel e a Definição de Machine Learning.

Arthur Samuel, um pioneiro no campo da Inteligência Artificial, em 1950, criou o termo "*Machine Learning*" como a propriedade de uma maquina aprender sem terem sido explicitamente programadas.

1.2.2 *O Perceptron de Frank Rosenblatt.*

Em 1958, Frank Rosenblatt, psicólogo e cientista da computação desenvolveu o Perceptron, um modelo de rede neural artificial que possibilitou as máquinas a aprenderem a reconhecer padrões, este foi um dos primeiros marcos no *Machine Learning*.

1.2.3 *O Algoritmo de Aprendizado por Reforço de Richard Bellman.*

Na década de 1950, Richard Bellman propôs o algoritmo de Aprendizado por Reforço, outro importante no *Machine Learning*, este modelo de aprendizado permitiu que as maquinas seriam influenciadas com base num sistema de recompensas e punições, sem a necessidade de uma programação explicita.

1.2.4 *Desenvolvimento do Machine Learning ao longo dos anos.*

Ao longo dos anos, a área do *Machine Learning* foi marcada por avanços marcantes, mas outros avanços também potencializaram a evolução deste campo, entre eles, são:

- Aumento da capacidade de processamento;
- Aumento da disponibilidade de dados;
- Evolução dos algoritmos;
- Integração com outras áreas da Inteligência Artificial.

No entanto, *Machine Learning* é uma área muito promissora e com muito que explorar, mesmo assim, já há muitas descobertas, e é isso que iremos falar.

2 *MACHINE LEARNING*

Numa pesquisa científica, referimos as identidades que tomam decisões em relação à informação que são-lhes atribuidas como **agentes**, e será esse nome iremos usar para referir essas identidades, e quando um agente consegue aprender, sendo uma simples tarefa doméstica ou até mesmo a teoria da relatividade de Eisnten, e caso esse agente seja um computador, o consideramos como *Machine Learning*.

Agora qual seria a razão de que queremos que uma maquina aprenda, em vez de simplesmente criar um código diretamente? Para isso, podemos referir duas razões principais. Primeiramente, é impossível antecipar todas as situações futuras, por exemplo, um robô designado para resolver labirintos tem que aprender o formato de para cada labirinto que entra. A segunda razão é que as vezes nem sabemos qual programa seria a solução para um certo problema, melhor exemplo seria reconhecimento facial, onde nós conseguimos reconhecer nossos familiares pela cara, mas fazemos isso inconscientemente, por isso, nem os melhores programadores sabem como programar um computador para fazer essa tarefa, com exceção de algoritmos de *Machine Learning*.

O *Machine Learning* [7], é um dos grandes polos da Inteligência Artificial, por isso é impossível falar do tema como um todo, de tal forma que foi dividido em outras grandes áreas para facilitar sua pesquisa, mas seguindo os objetivos impostos pela cadeira de Laboratórios de Informática, iremos trabalhar em relação das seguintes áreas:

- *Learning from Examples* (Aprendizagem por exemplos)
- *Reinforcement Learning* (Aprendizagem reforçada)

3 *LEARNING FROM EXAMPLES*

Resumidamente, descreve-se quando um agente consegue melhorar seu comportamento por um estudo detalhado com acontecimentos do passado e previsões futuras. Neste capitulo, iremos discutir vários modelos, pela ordem do seguinte índice:

- Formas de Aprendizagem
- *Supervised Learning*
- *Learning Decision Trees*

3.1 Formas de Aprendizagem

Um agente pode sempre ser melhorado com *Machine Learning*, essas melhorias, e as técnicas utilizadas para as mesmas, dependem dos seguintes fatores:

- Qual componente a ser melhorado;
- Qual conhecimento antecedente o agente possui que influencia o *modelo* que o constitui.
- Quais são os dados e qual é o *feedback* nesses dados disponível.

Neste capitulo assumimos que conhecimento que o agente possui antecipadamente é muito pouco, como começa-se do zero. Começando por recorrentes observações precisas para a criação de uma regra geral, este processo é chamado de **indução**; por observações que o sol nasceu todos os dias no passado, nós induzimos que no dia seguinte o sol vai nascer. Sendo diferente da **Dedução**, onde se premissas tiverem corretas, a sua conclusão será sempre correta, ao contrario da indução, onde esta pode ser errada.

Neste capitulo, iremos nos concentrar em problemas onde o input é uma **representação fatorada**, sendo um vetor de valores atribuídos, onde o input pode ser qualquer estrutura de dados. Quando o output é um valor de um conjunto finito de valores (como feliz/triste/zangado/aborrecido), este problema de aprendizagem é chamado de **classificação** e quando é um numero (podendo ser uma previsão de temperatura, medido como um inteiro ou um numero real), o problema de aprendizagem tem o nome de **regressão**.

Em consideração do ***feedback***, estes podem acompanhar os inputs, e isso indica os três principais tipos de aprendizagem (onde cada um será explicado posteriormente ao longo do relatório):

- ***Supervised Learning***
- ***Unsupervised Learning***
- ***Reinforcement Learning***

3.2 *Supervised Learning*

Supervised Learning [2] (Aprendizagem supervisionada) é o método de aprendizagem onde um agente é treinado com um conjunto de informação, cada input é rotulado a um output, assim, estes modelos começam a aprender padrões e relações entre cada input e cada output rotulado, assim podendo prever resultados precisos com informação recente. Nesta subárea do *Machine Learning*, o objetivo é tentar obter um sentido numa pergunta especifica, onde o torna muito bom nos problemas de classificação e de regressão, como descobrir qual tema uma noticia pertença ou qual a quantidade de vendas numa data futura, e em relação a empresas, estes modelos são usados para a deteção de anomalias ou fraudes, classificação de imagens, avaliação de riscos e filtragem de spam. O contrario de *Supervised Learning* é considerado ***Unsupervised Learning*** (Aprendizagem Não Supervisionada), é quando uma maquina é treinada com input's não rotulados a output's.

3.2.1 Como Funciona o Supervised Learning.

No *Supervised Learning*, durante a fase de treinamento, é fornecido ao sistema dados rotulados, onde é indicado qual output está relacionado ao seu especifico input. Após esta fase, o respetivo modelo é apresentado informação teste, onde o seu objetivo é medir o quão preciso a performance do algoritmo quando trabalha com dados não rotulados.

No geral, ao implementar *Supervised Learning*, segue-se os seguinte passos:

(1) Determinar o tipo de dados a ser usados para o treino defenido;
(2) Coletar a informação rotulada para treinamento;
(3) Dividir os dados coletados em três grupos:
   - Dados para treinamento.
   - Dados-teste.
   - Dados para validação.
(4) Determinar um algoritmo para usar no modelo de *Machine Learning*;
(5) Dar ***Run*** ao algoritmo com os dados definidos para o treino;
(6) Avaliar a performance do modelo, se prevê os corretos output's, então o mesmo é preciso.

3.2.2 Algoritmos de Supervised Learning.

Vários algoritmos e técnicas computacionais são usados nos processos do *Supervised Learning*. A seguinte lista estará presente breves explicações de alguns métodos de aprendizagem mais usados, onde são calculados usando programas com *R* ou *Python*:

Naive bayes é uma abordagem de classificação que adota o princípio da independência condicional de classe do Teorema de Bayes. Isso significa que a presença de uma característica não impacta a presença de outra na probabilidade de um resultado dado, e cada preditor tem um efeito igual nesse resultado. Existem três tipos de classificadores Naive Bayes: Multinomial Naive Bayes, Bernoulli Naive Bayes e Gaussian Naive Bayes. Essa técnica é principalmente utilizada em classificação de texto, identificação de spam e sistemas de recomendação.

Support vector machines (SVM) são modelos populares do *Supervised Learning* desenvolvidos por Vladimir Vapnik, utilizados tanto para classificação quanto para regressão de dados. No entanto, são geralmente empregadas em problemas de classificação, construindo um hiperplano onde a distância entre dois grupos de dados é maximizada. Esse hiperplano é conhecido como fronteira de decisão, separando os grupos de dados (por exemplo, laranjas vs. maçãs) em lados opostos do plano.

K-nearest neighbor, também conhecido como algoritmo KNN, é um algoritmo não paramétrico que classifica pontos de dados com base em sua proximidade e associação a outros dados disponíveis. Este algoritmo parte do pressuposto de que pontos de dados semelhantes podem ser encontrados próximos uns dos outros. Assim, ele calcula a distância entre os pontos de dados, geralmente usando a distância euclidiana, e em seguida atribui uma categoria com base na categoria mais frequente ou na média. Sua facilidade de uso e baixo tempo de cálculo o tornam um algoritmo preferido pelos cientistas de dados, mas à medida que o conjunto de dados de teste aumenta, o tempo de processamento se alonga, tornando-o menos atraente para tarefas de classificação. O KNN é geralmente usado em motores de recomendação e reconhecimento de imagem.

Random Forest é outro algoritmo flexível do *Supervised Learning* utilizado tanto para fins de classificação quanto de regressão. A "floresta" refere-se a uma coleção de *Learning Decision Trees* não correlacionadas, que são combinadas para reduzir a variância e criar previsões de dados mais precisas.

3.2.3 Unsupervised vs. Supervised vs. Semi-Supervised Learning.

A grande diferença entre ***Supervised Learning*** e ***Unsupervised Learning*** é como o algoritmo é treinado. No *Unsupervised Learning*, os dado de treinamento não estão rotulados, ou seja, nenhum dos input's está rotulado a um output, assim, seu treino pode ser mais livre sem ser restringido a medidas exteriores, uma vantagem é que quando o algoritmo aprende livremente, pode aprender coisas interessantes e inesperadas sobre a informação definida, outra é para resolver situações onde os investigadores não têm ideia do que os dados têm em comum. Mas será possível usar estes dois métodos ao mesmo tempo, ou seja, usar dados rotulados e não rotulados para treinamento de um agente? Sim, a isso chamamos de ***Semi-Supervised Learning***.

No *Semi-Supervised Learning*, só uma parte dos dados é rotulada, este metodo e o *Unsupervised Learning* podem ser mais tentadores a ser utilizados, pois ter uma base de dados rotulada para o uso de *Supervised Learning* pode ser mais cara e mais demorada a obter.

3.2.4 Benefícios e Limitações.

Modelos de *Supervised Learning* tem algumas vantagens em relação ao *Unsupervised Learning*, mas também inclui limitações. Sendo os seus benefícios:

- Sistemas de *Supervised Learning* são capazes de fazer escolhas que nós consigamos perceber porque foram humanos que deram os critérios para suas decisões;
- O critério de performance é otimizada por ter ajuda adicional experiente;
- Consegue fazer tarefas de regressão e classificação;
- Nós controlamos o numero de classes usadas nos dados de treinamento;
- Agentes são capazes de prever outputs com base na experiência passada;
- As classes dos objetos são rotulados com termos precisos.

Agora as limitações do *Supervised Learning* são:

- No caso de um método baseado em recuperação, os sistemas de aprendizado supervisionado enfrentam dificuldades ao lidar com novas informações. Se um sistema com categorias para gatos e cachorros for apresentado a novos dados, como uma zebra, ele teria que erroneamente agrupá-la em uma das categorias existentes. Se o sistema de IA fosse generativo, ou seja, não supervisionado, ele poderia não saber o que é uma zebra, mas seria capaz de reconhecê-la como pertencente a uma categoria separada.
- No *Supervised Learning* geralmente requer grandes quantidades de dados corretamente rotulados para alcançar níveis aceitáveis de desempenho, e tais dados nem sempre estão disponíveis. O aprendizado não supervisionado não enfrenta esse problema e pode trabalhar igualmente bem com dados não rotulados.
- Estes modelos de muito tempo de treinamento antes de poder-mos utilizá-los.

3.2.5 *Exemplo de Supervised Learning.*

Um bom exemplo de um problemas de *Supervised Learning* em detalhe: **O problema de decidir se vale a pena esperar por uma mesa num restaurante.** Para este problema, X será o input, onde é um vetor com 10 valores atribuídos, sendo cada um com seu valor discreto:

(1) Alternativa: Se existe bons restaurantes por perto ou não.
(2) Bar: Se o restaurante tem um bar confortável para esperar ou não.
(3) Sex/Sab: Será verdade se o dia for sexta ou sábado.
(4) Fome: Se estivermos com fome ou não.
(5) Clientes: quantia de pessoas no restaurante (valores são None, Full e Some).
(6) Preço: Alcance do preço ($, $$, $$$)
(7) Chuva: Se está a chover ou não.
(8) Reserva: Se há reserva marcada ou não.
(9) Tipo: Tipo de restaurante (Francês, Thai(Tailandês), Italiano, Hamb (Hamburger))
(10) EstimativaEspera: Estimativa dada pelo dono, 0 a 10, 10 a 30, 30 a 60 ou >60 minutos

Na seguinte tabela estão 12 inputs rotulados a 12 outputs, sendo estes somente 12 exemplos.

Exp	Alt	Bar	Sex	Fom	Input's client	class Pre	Chu	Res	Tipo	Est	Output WillWait
X1	Sim	Não	Não	Sim	Some	$$$	Não	Sim	Francês	0 a 10	Y1=Sim
X2	Sim	Não	Não	Sim	Full	$	Não	Não	Thai	30 a 60	Y2=Não
X3	Não	Sim	Não	Não	Some	$	Não	Não	Burger	0 a 10	Y3=Sim
X4	Sim	Não	Sim	Sim	Full	$	Sim	Não	Thai	10 a 30	Y4=Sim
X5	Sim	Não	Sim	Não	Full	$$$	Não	Sim	Francês	>60	Y5=Não
X6	Não	Sim	Não	Sim	Some	$$	Sim	Sim	Italiano	0 a 10	Y6=Sim
X7	Não	Sim	Não	Não	None	$	Sim	Não	Burger	0 a 10	Y7=Não
X8	Não	Não	Não	Sim	Some	$$	Sim	Sim	Thai	0 a 10	Y8=Sim
X9	Não	Sim	Sim	Não	Full	$	Sim	Não	Burger	>60	Y9=Não
X10	Sim	Sim	Sim	Sim	Full	$$$	Não	Sim	Italiano	10 a 30	Y10=Não
X11	Não	Não	Não	Não	None	$	Não	Sim	Thai	0 a 10	Y11=Não
X12	Sim	Sim	Sim	Não	Full	$	Não	Não	Burger	30 a 60	Y12=S

Este dados foram tirados no livro [8]. Sabendo que todas combinações possíveis de valores que podem ser atribuídos aos inputs são:

$2^6 \times 3^2 \times 4^2 = 9216$

Mesmo assim, nós fornecemos 12 exemplos dos possíveis, onde os outros 9204 poderiam ser verdadeiro ou falso, não sabemos. Isto é a essência do indução: o agente terá fazer a melhor precisão para os outros 9204 outputs usando como evidencia os 12 exemplos que o fornecemos.

3.3 *Learning Decision Trees*

No *Supervised Learning*, existe um subcampo chamado ***Learning Decision Trees*** [4]. *Learning Decision Trees* é um modelo hierárquico utilizado em suporte à decisão que representa decisões e seus resultados potenciais, incorporando eventos de chance, despesas de recursos e utilidade,

utilizando declarações de controle condicional e por ser não paramétrico é uma boa técnica útil tanto para tarefas de classificação quanto de regressão. Este algoritmos tem a estrutura composta por um node raiz, ramos, nós internos e nodes folha, formando uma estrutura hierárquica semelhante a uma árvore.

3.3.1 Terminologias das Decision Trees.

Antes de começarmos a aprender sobre *Decision Trees*, iremos referir algumas termologias:

- ->***Root Node*:** É o *node* inicial localizado no início da *Decision Tree*, onde está presente toda a população ou onde o *dataset* começa a dividir-se com base nas características e nas condições do algoritmo.
- ->***Decision Nodes*:** *Nodes* resultantes das *nodes* raiz, representam decisões intermediárias ou condições dentro da árvore.
- ->***Leaf Nodes*:** São *nodes* indivisíveis, ou seja, não é possível ramificar além deste *node*. Normalmente indicam um resultado ou uma classificação final, sendo estas também consideradas como *terminal nodes* (*nodes* terminais).
- ->***Sub-Tree*:** Uma subsecção de uma *Decision Tree*, representa uma parte especifica da mesma.
- ->***Pruning*:** Processo de eliminar ou cortar *nodes* específicos para evitar sobre-ajustes ou simplificar o modelo.
- ->***Branch* / *Sub-Tree*:** Uma subsecção da *Decision Tree* que representa um caminho (*path*) especifico de decisões ou resultados de uma árvore, é um caso particular das *sub-trees*.
- ->***Parent* e *Child Node*:** O *Parent Node* é um *node* que é dividido em sub-nodes, onde esses são referidos com *Child Nodes*. *Parent Nodes* representam condições ou uma decisão, enquanto *Child Nodes* podem ser possíveis resultados ou decisões futuras com base na condição.

3.3.2 Premissas das Decision Trees.

Para construir modelos eficazes ao criar *Decision Trees*, é necessário ter estas premissas pois são um bom guia na construção da árvore e isso tem um impacto significativo em sua performance. Aqui estão algumas premissas comuns consideradas ao criar uma *Decision Tree*.

Separações binárias são muito usadas em *Decision Trees*, isto significa que cada *node* pode ser dividido de forma binária, onde esses dados pode ser dividido em duas sob-conjunto com base em único recurso ou condição.

Particionamento Recursivo implica que cada *node* pode ser dividido em *child nodes* até que seja alcançado um critério que pare esse processo, com isto é assumido que informação pode ser dividido em menores dados, para melhor gerência de subconjuntos (*subsets*).

Independência de recursos são os recursos usados na divisão dos *nodes*, apesar que na pratica esses recursos não podem ser considerados independentes, as *Decision Trees* ainda funcionam mesmo quando os recursos estão relacionados.

Homogeneidade é a suposição que os subconjuntos de cada *node* são homogéneos, o que significa que todas as amostras de um *node* são similares o quanto possível, sem ter em conta a variavel alvo, esta premissa ajuda a obter claros limites para as decisões.

Top-Down Greedy Approach é a forma estes modelos são construidos, onde cada divisão é escolhida com objetivo de maximizar o ganho de informação ou minimizar a impureza do *node* a ser dividido. Isto nem sempre resulta em uma árvore totalmente otimizada.

Recursos categóricos e numéricos podem ser suportados por *Decision Trees*, no entanto os mesmos podem requerer estratégias de divisão diferentes para cada tipo.

Sobre-ajuste é a tendência de uma *Decision Tree* na descoberta de "impureza" nos dados, onde *Pruning* e uma definição de paragem com o critério apropriado são usados para tratar esta suposição.

Medidas de Impureza são usadas para avaliar a qualidade de uma divisão de classes, onde a escolha do método de medida de impureza (como o *Gini impurity* ou Entropia) pode influenciar a construção da árvore.

Ausência de valores omissos num conjunto de dados é uma tese assumida pelo algoritmo, ou assume que os valores omitidos foram adequadamente tratados por meio de imputação ou por outros métodos.

A Igualdade de importância nos recursos é assumida pelas *Decision Trees* a não ser que a escala ou a ponderação dos recursos sejam aplicadas para enfatizar determinados recursos.

Sem valores discrepantes , pois estes modelos são muito sensíveis a estes valores e valores extremos podem influenciar a sua construção. Um pré-processamento ou métodos robustos para lidar com este tipo de valores de forma eficaz.

Sensibilidade ao tamanho da amostra devem ser balanceadas pois pequenos conjuntos de dados podem gerar sobre-ajuste e grandes conjuntos de dados podem resultar em árvores demasiado complexas.

3.3.3 Entropia de um Decision Tree.

A **Entropia** é a incerteza dentro do conjunto de dados ou por uma medida de uma desordem, explicaremos pelo o seguinte exemplo. Suponhamos que um grupo de doze amigos vão escolher um filme para assistir, havendo duas escolhas possíveis, sendo "**Napoleão**" ou "**O Exorcista**", onde cada um deles terá que dizer a sua escolha. Após cada um fazer sua escolha, para o filme "Napoleão" houve 7 votos e para o filme "O Exorcista" houve 5 votos, qual será o filme que devem escolher? Não será difícil escolher um filme quando o numero de votos em cada filme são praticamente iguais.

É isto que chamamos desordem, onde á um numero igual de votos para cada filme e não conseguimos decidir qual assistir. Seria mais fácil se "Napoleão" tivesse obtido 10 votos e "O Exorcista" tivesse obtido 2 votos, onde poderíamos dizer que como o "Napoleão" teve mais votos então o grupo de amigos irá assistir esse filme. Nas *Decision Trees*, os *outputs* são, na maior parte dos casos, "sim" ou "não". Onde quanto maior for a entropia, mais "impuro" é o *node*. A Formula da Entropia é:

$E(S) = -p_{(+)} \log_2(p_{(+)}) - p_{(-)} \log_2(p_{(-)})$

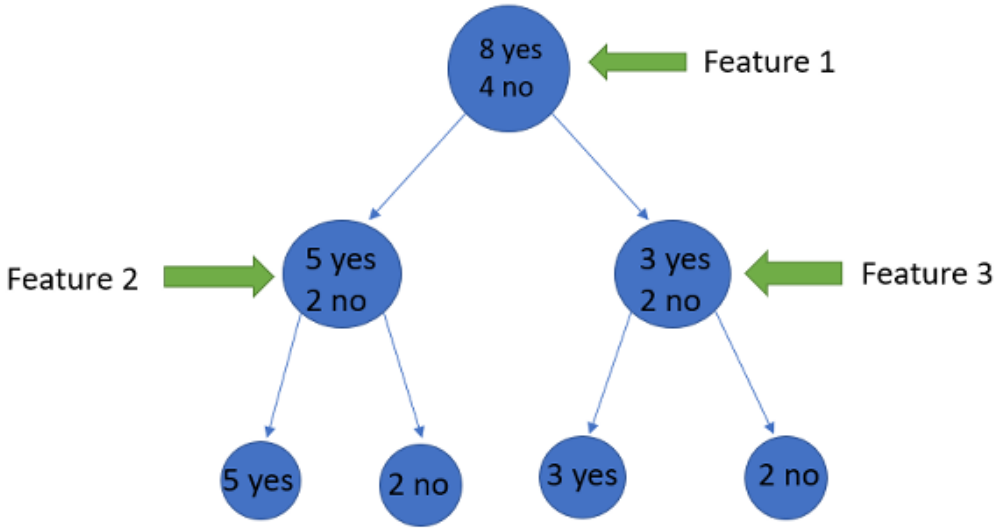
Onde sua simbologia:

- p+ é a probabilidade da classe positiva.
- p- é a probabilidade da classe negativa.
- S é um subconjunto do exemplo em treinamento.

3.3.4 Como as Decision Trees Usam Entropia.

Sabendo a definição de Entropia e sua formula, agora iremos ver como exatamente funciona neste algoritmo. como dito anteriormente, Entropia mede a impureza de um *node*. **Impureza** é grau de aleatoriedade, que indica o quão aleatório nossos dados são. Uma **sub-divisão pura** significa que só teremos dois outputs, sendo ou "sim" ou "não".

Suponhamos que um recurso (***feature 1***) tenha 8 "sim" e 4 "não" inicialmente, onde após primeira divisão, o *node* esquerdo tem 5 "sim" e 2 "não", e o direito tem 3 "sim" e 2 "não". Com isto, concluímos que esta divisão não é pura porque temos algumas classes negativas em cada *node*. Para a transformar em uma *Decision Tree*, vamos calcular a impureza de cada um destes dois *nodes*, e quando for 100% puro, iremos fazer um *leaf node*. Para verificar a impureza do *node* esquerdo (***feature 2***) e do *node* direito (***feature 3***) com a ajuda da formula da Entropia.



.

Calculo da Entropia do *feature 2*:

$\Rightarrow -(\frac{5}{7})\log_2(\frac{5}{7}) - (\frac{2}{7})\log_2(\frac{2}{7})$

$\Rightarrow -(0.71^* - 0.49) - (0.28^* - 1,83)$

$\Rightarrow -(-0.34) - (-0.51)$

$\Rightarrow 0.34 + 0.51$

$\Rightarrow 0.85$

De seguida, calculo da Entropia do *feature 3*:

$\Rightarrow -(\frac{3}{5})\log_2(\frac{3}{5}) - (\frac{2}{5})\log_2(\frac{2}{5})$

$\Rightarrow -(0.6^* - 0.73) - (0.4^* - 1,32)$

$\Rightarrow -(-0.438) - (-0.528)$

$\Rightarrow 0.438 + 0.528$

$\Rightarrow 0.966$

Pela representação da *Decision Tree*, o *node* esquerdo tem menor entropia ou maior pureza que o *node* direito por ter mais "sim" do que o outro, respetivamente, então é mais fácil de decidir neste caso.

Sendo o objetivo de uma *Machine Learning* diminuir a incerteza ou impureza do conjunto de dados, ao usar a entropia nós iremos obter a impureza de um *node* em especifico, não saberemos se o *parent node* ou um *node* em particular diminuiu ou não. Para isso, existe uma métrica chamada *"Information gain"*, onde mede a a redução da incerteza dada alguma característica e também é um fator decisivo para qual atributo deve ser selecionado como *decision node* ou *root node*.

3.3.5 3 Tipos da Decision Trees.

Em relação a modelos de *Decision Trees*, os 3 tipos mais usados são:

Classification trees (Árvores de Classificação) rotula, grava, e atribui variáveis a classes discretas, onde também consegue promover a exatidão quanto á certeza da classificação. Este modelo é construido pelo processo conhecido como binary recursive partitioning

Regression Trees (Árvores de Regressão) são *Decision Trees* nas quais as variáveis alvo podem assumir valores contínuos em vez de rótulos de classe nas *leaf nodes*. Estes algoritmos usam critérios de seleção de divisão modificados e critérios de parada.

Reduced Error Pruning Trees (Erros Reduzidos na Poda de Árvores) são modelos onde, a partir pelas *leaf nodes*, cada uma será substituída com a classe mais popular, caso a previsão não seja afetada então a mudança é mantida. Apesar da ideia ser bastante ingénua, estas *Decision Trees* têm a vantagem de sua simplicidade e velocidade.

4 *REINFORCEMENT LEARNING*

Reinforcement Learning é uma área do *Machine Learning* que tem como objetivo ensinar os agentes a tomar decisões com base nas recompensas que ele recebe por causa das suas ações, com propósito de maximizar a quantidade de recompensas no futuro. No *supervised learning*, um agente aprende observando passivamente pares de entrada/saída fornecidos por um "professor". Neste capítulo, veremos como os agentes podem aprender ativamente a partir de sua própria experiência, sem um "professor", considerando seu próprio sucesso ou fracasso final e vamos trabalhar os seguintes temas.

- *Reinforcement Learning* Passiva vs Ativa
- *Reinforcement Learning Hierárquico*
- *Inverse Reinforcement Learning*
- Exemplo de *Reinforcement Learning*

4.1 *Reinforcement Learning* Passiva vs Ativa

No *Reinforcement Learning*, existem dois tipos principais de *Reinforcement Learning* [1]:

***Reinforcement Learning* Ativa** acontece quando o agente tem controlo total das suas ações e explora-as de forma a maximizar a sua recompensa.

***Reinforcement Learning* Passiva** acontece quando o agente não tem poder de decisão, ou seja as ações já estão determinadas por um agente externo como por exemplo por um operador humano ou por um algoritmo pré-programado. Nestas situações o agente só recebe recompensas ou penalidades com base nas suas ações.

4.1.1 Diferença entre Reinforcement Learning Ativa e Reinforcement Learning Passiva.

Reinforcement Learning Ativa e *Reinforcement Learning Passiva* têm as suas diferenças que podemos observar no seguintes tópicos: No *Reinforcement Learning Ativa*

- Começa com um pequeno conjunto de dados rotulados e solicita dados adicionais do usuário.
- O algoritmo interage com o usuário para adquirir dados adicionais.
- Pode continuar a solicitar dados adicionais até que seja atingido um nível satisfatório de precisão.

- Adequado para aplicações onde os dados rotulados são escassos ou caros de adquirir.

No *Reinforcement Learning Passiva*:

- um grande conjunto de dados pré-rotulados para treinar o algoritmo.
- O algoritmo não interage com o usuário.
- Ele não requer a entrada do usuário após a conclusão do treinamento.
- Adequado para aplicações onde um grande conjunto de dados está disponível.

4.2 *Hierarchical Reinforcement Learning*

Hierarchical Reinforcement Learning (aprendizagem reforçada hierárquico) [9] é uma abordagem avançada que tem como propósito lidar com problemas complexos e de longo prazo, permitindo ao agente aprende a tomar decisões em diferentes níveis de abstração, o que facilita uma melhor resolução de tarefas complexas.

4.2.1 Benefícios do Hierarchical Reinforcement Learning.

Hierarchical Reinforcement Learning traz benefícios em relação a abordagens tradicionais por aprendizado por reforço, sendo um dos principais benefícios ser a capacidade de resolver problemas complexos a longo prazo podendo ser resolvidos por métodos convencionais. Além disso *Hierarchical Reinforcement Learning* permite uma melhor organização do conhecimento tornando assim o agente mais eficiente a executar tarefas similares.

4.2.2 Aplicações do Hierarchical Reinforcement Learning.

O *Hierarchical Reinforcement Learning* tem sido aplicado em várias áreas entre as quais robótica, controlo de tráfego aéreo, jogos, entre outros. A robótica é um exemplo da *hierarchial reinforcement learning*, pois é um aprendizado hierárquico que pode ser utilizado para ensinar um robô a desempenhar funções mais complexas, como montar um objeto enquanto que nos jogos pode ser usado para treinar agentes virtuais a tomarem decisões estratégicas em ambientes complexos.

4.2.3 Algoritmos de Hierarchical Reinforcement Learning.

No *Hierarchical Reinforcement Learning* existem diversos algoritmos que podem ser usados, cada um com características e aplicações especificas. Alguns dos algoritmos mais conhecidos procuram encontrar uma hierarquia de ações que permita ao agente tomar decisões eficientes em diferentes niveis de abstração.

4.2.4 Desafios do Hierarchical Reinforcement Learning.

Apesar de dos beneficios que o *Hierarchical Reinforcement Learning* tem também apresenta alguns desafios , sendo os principais defenir a hierarquização que pode ser uma tarefa complexa e ser exigido conhecimento especializado, além disso o o *Hierarchical Reinforcement Learning* pode exigir um grande volume de dados e tempo para aprender o que pode ser um obstáculo ara alguma aplicações.

4.2.5 *Comparação entre o Hierarchical Reinforcement Learning e o Reinforcement Learning tradicional.*

O *Hierarchical Reinforcement Learning* diferencia-se do *Reinforcement Learning tradicional* pela abordagem hierárquica. Enquanto o Reinforcement Learning tradicional procura aprender ações individuais para resolver problemas, o aprendizado hierárquico visa aprender uma sequência organizada de ações em diferentes níveis de abstração. Isso possibilita ao agente lidar com problemas complexos de maneira mais eficiente e com maior capacidade de generalização. Em vez de apenas focar em ações diretas, essa abordagem permite ao agente aprender estruturas hierárquicas de comportamento, o que pode melhorar sua eficácia ao lidar com diversas situações, o que pode ser um obstáculo para alguma aplicações.

4.2.6 *Limitações do Hierarchical Reinforcement Learning.*

Apesar das vantagens, o *Hierarchical Reinforcement Learning* também tem suas limitações. Uma delas é a demanda por uma definição manual da hierarquia de ações, o que pode ser complexo e exigir conhecimento especializado. Além disso, qualquer equívoco na definição dessa hierarquia pode afetar significativamente o desempenho do sistema, levando a resultados sub-ótimos.

4.2.7 *Avanços recentes Hierarchical Reinforcement Learning.*

Recentemente, tem havido significativos avanços no *Hierarchical Reinforcement Learning*. Novos algoritmos mais eficazes e métodos de treinamento mais velozes estão sendo desenvolvidos, ampliando a aplicação desse tipo de aprendizado em diversas áreas. Além disso, a pesquisa está explorando técnicas de transferência de conhecimento, permitindo que um agente aprenda tarefas hierárquicas de forma mais rápida e eficiente, aproveitando conhecimentos adquiridos anteriormente.

4.3 *Inverse Reinforcement Learning*

No *Inverse Reinforcement Learning* [3] é usado quando não sabemos quais ações devemos recompensar, ou seja, serve para aprender para quais ações devem ser recompensadas observando o comportamento do agente. Isto é necessário pois em varias situações reais, especificar a ação que deve ser recompensada pode ser um grande desafio.

Este desafio é abordado por esta sub-área, trabalhando de trás para frente, onde dado o comportamento observado de um agente especialista, o objetivo é inferir a função de recompensa subjacente que provavelmente motivou esse comportamento, assim podendo envolver a moldagem do processo de tomada de decisão do especialista e estimar a estrutura de recompensa que se alinha às suas ações. Processo geral em *Inverse Reinforcement Learning* inclui:

(1) **Observação do Comportamento de Especialistas:** Colete dados sobre como um especialista executa uma tarefa ou toma decisões em um determinado ambiente.
(2) **Modelando o Comportamento do Especialista:** Desenvolva um modelo que represente o processo de tomada de decisão do especialista. Isto poderia envolver técnicas como *Markov Decision Processes* (MDPs) ou outras estruturas de *Reinforcement Learning*.

(3) **Inferência da função de recompensa:** Use algoritmos de otimização ou aprendizagem para inferir a função de recompensa que explica o comportamento observado do especialista. A função de recompensa inferida representa as preferências ou objetivos subjacentes do especialista.
(4) **Aplicação nos Novos Agentes:** Aplique a função de recompensa aprendida para orientar o comportamento de outros agentes em tarefas semelhantes.

Inverse Reinforcement Learning têm aplicações em áreas onde é difícil definir explicitamente uma função de recompensa, como carros autónomos, robótica e interação humano-robô, permitindo às maquinas aprenderam com demonstrações humana, imitandos nossos comportamentos em tarefas complexas.

4.4 Aplicações do *Reinforcement Learning*

O *Reinforcement Learning* é um ramo do *Machine Learning* onde um agente aprende a tomar decisões através da interação com um ambiente, visando maximizar uma recompensa ao longo do tempo. Este são alguns dos exemplos onde podemos observar o *Reinforcement Learning* na sociedade:

4.4.1 Robótica.

Em robótica, o *Reinforcement Learning* é usado para treinar robôs a realizarem tarefas complexas, como manipulação de objetos, navegação em ambientes desconhecidos ou aprendizagem de movimentos.

4.4.2 Sistemas de Recomendação.

Em plataformas de streaming, comércio eletrónico e redes sociais, o *Reinforcement Learning* é empregado para recomendar produtos, músicas, filmes ou conteúdo com base nas interações passadas do usuário.

4.4.3 Finanças.

Na área financeira, é aplicado para modelar o comportamento do mercado, otimização de portfólio e estratégias de negociação.

Estes são apenas alguns exemplos, mas o *Reinforcement Learning* tem uma ampla gama de aplicações em diversas áreas, explorando como os agentes podem aprender a tomar decisões para maximizar recompensas em cenários complexos e dinâmicos.

5 CONCLUSÃO

Concluímos que apesar dos avanços notáveis na área de *Machine Learning*, ainda há desafios cruciais a serem superados. Um dos principais é a compreensão dos resultados gerados por esses algoritmos. Muitas vezes, esses modelos são como "caixas pretas", o que significa que não é fácil entender exatamente como eles chegam a certas conclusões. Isso pode representar um obstáculo significativo em áreas onde transparência e explicabilidade são fundamentais, como na saúde.

No entanto, direções futuras do *Machine Learning* estão voltadas para o aprimoramento das técnicas que possibilitam uma interpretação mais clara dos modelos. Isso inclui o uso de métodos

para tornar os modelos mais explicáveis, éticos e interpretáveis. Além disso, há uma expectativa de expansão significativa do uso do *Machine Learning* em campos como medicina de precisão, internet das coisas, avanços em robótica e também na computação quântica.

REFERENCES

[1] Rachael Ferguson. 2023. *Active and Passive Reinforcement Learning Examples.* medium. Retrieved Novembro 15, 2023 from https://medium.com/@cluelessrae/active-and-passive-reinforcement-learning-examples-a499d2b5fc10

[2] Alexander S. Gillis. 2023. *supervised learning.* TechTarget. Retrieved Outubro 15, 2023 from https://www.techtarget.com/searchenterpriseai/definition/supervised-learning

[3] Alexandre Gonfalonieri. 2018. *Inverse Reinforcement Learning.* medium. Retrieved Novembro 25, 2023 from https://towardsdatascience.com/inverse-reinforcement-learning-6453b7cdc90d

[4] Analytics Vidhya 2023. *Institutional members of the TEX Users Group.* Analytics Vidhya. Retrieved Novembro 15, 2023 from https://www.analyticsvidhya.com/blog/2021/08/decision-tree-algorithm/

[5] Coursera 2020. *Machine Learning: Conceitos, definição e mais.* Coursera. Retrieved Setembro 29, 2023 from https://www.coursera.org/articles/machine-learning-vs-ai

[6] awari 2023. *Machine Learning: Quem criou? – História e desenvolvimento.* awari. Retrieved Outubro 15, 2023 from https://awari.com.br/machine-learning-quem-criou-historia-e-desenvolvimento/

[7] Coursera 2023. *What Is Machine Learning? Definition, Types, and Examples.* Coursera. Retrieved Outubro 15, 2023 from https://www.coursera.org/articles/what-is-machine-learning

[8] Russel Norvig and S Artificial Intelligence. 2002. A modern approach. *Prentice Hall Upper Saddle River, NJ, USA: Rani, M., Nayak, R., & Vyas, OP (2015). An ontology-based adaptive personalized e-learning system, assisted by software agents on cloud storage. Knowledge-Based Systems* 1 (2002), 10–48.

[9] Maicon Ramos. 2023. *O que é Hierarchical Reinforcement Learning (Aprendizado por Reforço Hierárquico)?* glossario. Retrieved Novembro 18, 2023 from https://glossario.maiconramos.com/glossario/o-que-e-hierarchical-reinforcement-learning-aprendizado-por-reforco-hierarquico/

17% geral de similaridade

As principais fontes encontradas nos seguintes bancos de dados:

- 16% Banco de dados da Internet
- Banco de dados do Crossref
- 7% Banco de dados de trabalhos enviados
- 5% Banco de dados de publicações
- Banco de dados de conteúdo publicado no Cross

PRINCIPAIS FONTES

As fontes com o maior número de correspondências no envio. Fontes sobrepostas não serão exibidas.

1	glossario.maiconramos.com Internet	5%
2	ibm.com Internet	4%
3	University of Notre Dame on 2023-12-10 Submitted works	2%
4	export.arxiv.org Internet	2%
5	awari.com.br Internet	1%
6	Devleena Das, Been Kim, Sonia Chernova. "Subgoal-Based Explanation... Crossref	<1%
7	Universidade Aberta on 2023-12-17 Submitted works	<1%
8	Carnegie Mellon University on 2023-11-06 Submitted works	<1%

9	University of Nottingham on 2023-12-13 Submitted works	<1%
10	University of West Florida on 2021-12-08 Submitted works	<1%
11	iis.ipipan.waw.pl Internet	<1%
12	dione.lib.unipi.gr Internet	<1%
13	openaccess.altinbas.edu.tr Internet	<1%
14	jmir.org Internet	<1%
15	Indian Institute of Technology, Madras on 2016-05-05 Submitted works	<1%
16	University of Notre Dame on 2023-09-18 Submitted works	<1%
17	lume.ufrgs.br Internet	<1%
18	University of Glamorgan on 2017-03-07 Submitted works	<1%
19	hal.archives-ouvertes.fr Internet	<1%
20	A. P. Rauter. "Synthetic, Fungicidal Unsaturated-γ-lactones Attached to... Crossref	<1%

21	Technische Universität Dresden on 2022-11-04 Submitted works	<1%
22	California State University, Fresno on 2023-12-11 Submitted works	<1%