



INTELIGÊNCIA ARTIFICIAL

Unidade 04 - Métodos
de Aprendizagem

Marco Antonio Sartori



Fonte: unsplash

INTELIGÊNCIA ARTIFICIAL

Unidade 04 - Métodos
de Aprendizagem

Marco Antonio Sartori

Copyright Centro Universitário Fundação Assis Gurgacz, Núcleo de Educação a Distância.

Todos os direitos são reservados. É proibida a reprodução, distribuição, comercialização ou cessão sem autorização dos detentores dos direitos. Esse livro foi publicado no Ambiente Virtual de Aprendizagem do Centro Universitário Fundação Assis Gurgacz, para leitura exclusiva online pelos alunos matriculados no Ensino a Distância. Os leitores poderão imprimir as páginas para leitura pessoal. Os direitos dessa obra não foram cedidos.

Autor: **Marco Antonio Sartori**

Título: **Inteligência Artificial**

Direção geral: Cleber Fagundes Ramos

Gerente de produção: Vanessa Aparecida Anderle Zaiatz

Design Instrucional: Anderson Julio de Souza

Projeto gráfico: Bruno Vinícius dos Reis

Diagramação: Júlia Tonin Bordin

1ª Edição

fag.edu.br/ead


Av. das Torres, 500, Bloco 03

Fone: (045) 3321-3422


Bairro Santo Inácio | CEP: 85.806-095

Cascavel | Paraná | Brasil

SUMÁRIO



Apresentação da Disciplina.....	10
Ponto de Partida.....	11
Roteiro do Conhecimento.....	12
Unidade 04 - Métodos de Aprendizagem	14
4.1 - Introdução aos Métodos de Aprendizagem	15
4.1.1 - Aprendizagem Supervisionada.....	18
4.1.1.1 - Classificação	21
4.1.1.2 - Regressão.....	22
4.1.2 - Aprendizagem em Árvores de Decisão.....	24
4.2 - Aprendizagem por Reforço	27
4.2.1 - Aprendizagem por Reforço Passiva	30



4.2.2 - Aprendizagem por Reforço Ativa.....	33
4.2.3 - Busca de Políticas	34
4.2.4 - Aplicações da Aprendizagem por Reforço	35
4.3 - Algoritmos Genéticos	39
4.3.1 - Evolução de Regras	41
4.3.2 - Agrupamento	44
4.4 - Lógica Difusa.....	51
4.4.1 - Wang-Mendel.....	55
O Que Aprendemos.....	58
Leitura Complementar	59
Considerações Finais.....	61
Referências	62

APRESENTAÇÃO DA DISCIPLINA

Caro (a) estudante, seja muito bem-vindo (a) a mais uma etapa do seu processo de formação profissional! É com muito prazer que apresento o e-book da disciplina de Inteligência Artificial! No decorrer deste e-book você irá conhecer as principais características, conceitos e aplicações da Inteligência Artificial. O principal objetivo deste material é permitir que você obtenha os conhecimentos básicos necessários para que você possa não apenas conhecer os principais conceitos da IA, mas também ter uma ideia de que formas poderá aplicá-la em sua carreira enquanto profissional da área de tecnologia. O mercado tem utilizado cada vez mais a IA, nos mais diversos segmentos e para as mais diversas finalidades. Por isso é tão importante que você conheça o tema e busque aprofundar os seus conhecimentos sobre ele.

Ao longo da disciplina, trataremos dos seguintes assuntos:

Unidade 01 – Introdução à Inteligência Artificial.

Unidade 02 – Métodos de resolução de problemas.

Unidade 03 – Redes neurais artificiais.

Unidade 04 – Métodos de aprendizagem.

PONTO DE PARTIDA

Caro (a) aluno (a), chegamos a nossa última unidade. Na unidade 3, falamos sobre as Redes Neurais, as quais possuem a capacidade de aprender com os dados e com os problemas que precisam resolver. A partir de agora, falaremos com mais detalhes sobre as formas de aprendizagem e sobre como elas podem ser utilizadas para resolver os mais diversos tipos de problemas.

Ao final desta unidade, você deve apresentar os seguintes aprendizados:

- Conhecimento sobre os principais conceitos relacionados ao aprendizado de máquina;
- Conhecimento sobre as principais formas de aprendizado de máquina;
- Conhecimento sobre as principais aplicações do aprendizado de máquina;
- Conhecimento sobre os principais algoritmos que possuem capacidade de aprendizado.

ROTEIRO DO CONHECIMENTO

No decorrer desta unidade, o nosso foco será o aprendizado de máquina, as suas aplicações e os principais algoritmos que implementam esse recurso. A seguir, você pode acompanhar todas as seções que serão trabalhadas ao longo da unidade:

- 4.1 - Introdução aos métodos de aprendizagem.
- 4.2 - Aprendizagem por reforço.
- 4.3 - Algoritmos Genéticos.
- 4.4 - Lógica Difusa.

UNIDADE 04 - MÉTODOS DE APRENDIZAGEM

4.1 - INTRODUÇÃO AOS MÉTODOS DE APRENDIZAGEM

Quando falamos em aprendizado de máquina, estamos nos referindo à capacidade que agentes possuem de aprender com os dados e com os ambientes nos quais estão inseridos. Nesse sentido, poderíamos pensar que, se esses agentes são programados, as soluções poderiam ser programadas também, correto? Na verdade, não. Os problemas são bastante dinâmicos, assim como são dinâmicos os dados envolvidos nesses problemas.

Essa combinação de fatores, aliada às próprias limitações dos programadores, faz com a programação de soluções se torne algo inviável. Desta forma, tornar os agentes mais inteligentes e dar a eles a capacidade de aprender, torna essa tarefa, de certa forma, mais fácil.

Uma das principais aplicações do aprendizado de máquina (Machine Learning) é a busca padrões ou regularidades entre os dados. As técnicas utilizadas pelo Machine Learning envolvem induções de regras, árvores de decisão, modelos conexionistas e o aprendizado baseado em instâncias (SILVA; PERES; BOSCAROLI, 2016).

O aprendizado pode acontecer sob vários aspectos. As formas pelas quais ele ocorre dependem de quatro fatores (RUSSELL; NORVIG, 2013):

- Quais os componentes do agente que devem ser melhorados;
- O conhecimento prévio que o agente possui;
- Qual a representação é utilizada para os dados e para o componente que precisa ser melhorado;
- Qual feedback está disponível para aprendizagem.

Em relação aos **componentes do agente**, há seis deles que podem ser alterados durante o processo de aprendizagem (RUSSELL; NORVIG, 2013):

- 1 - O mapeamento direto de condições no estado atual, para cada ação possível de ser executada;
- 2 - O meio para deduzir propriedades relevantes a partir de uma sequência de percepções sobre o mundo ou ambiente;
- 3 - Informações sobre o modo como o mundo evolui e sobre os resultados que as ações possíveis para o agente podem causar;
- 4 - Informações sobre o que se espera de cada possível estado do mundo;
- 5 - Informações sobre os valores de cada ação, indicando o resultado esperado de cada uma delas;
- 6 - Metas que descrevem classes de estados cuja realização permite maximizar a utilidade do agente.

Russell e Norvig (2013) citam um exemplo bastante interessante e que facilita a compressão de como o mecanismo de aprendizagem poderia ser aplicado a um agente que precisa aprender a dirigir um táxi:

“Toda vez que o instrutor gritar “freie”, o agente poderá aprender uma condição-ação sobre quando frear (componente 1); o agente também sabe toda vez que o instrutor não grita. Ao ver muitas imagens que lhe mostram ônibus, o agente pode aprender a reconhecê-los (2). Experimentando ações e observando os resultados – por exemplo, freando bruscamente em uma estrada molhada –, ele poderá aprender os efeitos de suas ações (3). Depois, se não receber nenhuma gorjeta de passageiros que foram sacudidos durante o percurso, poderá aprender um comportamento útil de suas funções de utilidade global (4)” (RUSSELL; NORVIG, 2013).

Em relação à representação do **conhecimento prévio** que o agente possui, há várias formas de garantir essa representação, de acordo com a solução que se pretende utilizar. Alguns exemplos são as sentenças lógicas proposicionais e de primeira ordem, utilizadas para os componentes de um agente lógico, as redes bayesianas para os componentes de um agente de decisão que toma decisões baseado em inferências, dentre outros. Geralmente os algoritmos de aprendizagem mais eficazes utilizam a combinação de todas essas formas de representação (RUSSELL; NORVIG, 2013).

Sob uma outra perspectiva, é possível afirmar que o aprendizado pode ser representado por meio de uma função ou regra, definida a partir de pares de entrada e saída. Esse tipo de aprendizagem é chamado de **aprendizagem indutiva**. É possível, entretanto, trabalharmos com uma abordagem de aprendizagem **analítica** ou **dedutiva**, na qual há uma regra conhecida a partir da qual deriva uma outra regra, nova e com maior grau de utilidade. Nessa última abordagem, temos resultados melhores devido a um processamento mais eficiente (RUSSELL; NORVIG, 2013).

Outro fator importante para que um algoritmo possa aprender está ligado ao componente do feedback. Dentro do contexto do aprendizado de máquina temos três tipos de feedbacks, os quais determinam os três principais tipos de aprendizagem (RUSSELL; NORVIG, 2013):

Aprendizagem não supervisionada: Nesse tipo de aprendizagem, o agente aprende padrões na entrada, ainda que não exista nenhum feedback explícito. Uma das tarefas mais comuns quando tratamos desse tipo de aprendizagem é o agrupamento. O agrupamento consiste em encontrar grupos a partir de um conjunto de dados de entrada que possam ser úteis. A partir daí dados futuros podem ser agrupados de acordo com os critérios utilizados para os grupos iniciais.

Aprendizagem por reforço: Nesse tipo de aprendizagem o agente aprende a partir de reforços. Esses reforços podem ser punições ou recompensas. Um exemplo de como isso pode acontecer são os jogos, como o de Xadrez. Nesse caso, uma vitória ou perda serve como base para o agente, para que ele possa determinar quais foram as ações anteriores que culminaram na vitória ou na perda.

Aprendizagem supervisionada: Nesse tipo de aprendizagem o agente observa exemplos de pares de dados de entrada e saída, e aprende uma função que faz o mapeamento da entrada para a saída.

4.1.1 - APRENDIZAGEM SUPERVISIONADA

Para facilitar a compreensão de como o aprendizado supervisionado funciona, vamos retomar ao exemplo anterior, no qual o agente busca aprender a dirigir um táxi, aqui representado pela figura 4.1.

	Mapeamento direto de condições no estado atual para ações	Dedução de propriedades relevantes do mundo a partir de uma sequência de percepções	Informações sobre como o mundo evolui e sobre os resultados de ações possíveis que o agente pode executar
Entradas	Percepções	Imagens de uma câmera	A teoria da frenagem é uma função de estados e ações de frenagem até a distância de parada
Saídas	Instrutor que diz: "freie" ou "vire à esquerda"	Instrutor diz: "isso é um ônibus"	Percepção do agente

Figura 4.1 - Exemplo de feedbacks com base no exemplo do agente que está aprendendo a dirigir um táxi
Fonte: Adaptado de Russell e Norvig (2013)

É necessário, entretanto, considerar que nem sempre temos todas as informações de que precisamos para garantir as melhores soluções para os problemas. Nesses casos, pode ser utilizada a aprendizagem semissupervisionada. Nesse tipo de aprendizagem, são dados alguns poucos exemplos rotulados e, a partir deles, deve ser feito o que pudermos com uma grande coleção de dados não rotulados. O detalhe é que mesmo os rótulos, propriamente ditos, podem não representar a verdade que aparentam ser.

Para tornar a compreensão mais clara, Russell e Norvig (2013) citam um outro exemplo bem legal:

“Imagine que você deseja construir um sistema para descobrir a idade de uma pessoa a partir de uma foto. Você reúne alguns exemplos rotulados tirando fotos das pessoas e perguntando a idade. Isso é aprendizagem supervisionada. Mas, na realidade, algumas pessoas mentiram sua idade. Não é só que haja ruído aleatório nos dados, mas as imprecisões são sistemáticas, e descobri-las é um problema de aprendizagem não supervisionada, envolvendo imagens, idades autorrelatadas e idades (desconhecidas) verdadeiras. Assim, tanto ruído como falta de rótulos criam um continuum entre aprendizagem supervisionada e não supervisionada “.

Assim como outras abordagens de aprendizagem, a aprendizagem supervisionada é representada por uma função em que, dado um conjunto de dados para treinamento, contendo N pares de exemplo de entrada e saída $(x_1, y_1), (x_2, y_2), \dots (x_n, y_n)$, onde cada y_i foi gerado por uma função desconhecida $y = f(x)$ deve-se descobrir uma função h que se aproxime da função verdadeira f .

Nesse contexto, x e y podem possuir qualquer valor, sendo ele numéricos ou não. Como a função h é um objetivo, ela é chamada de hipótese. A aprendizagem, portanto, consiste em uma busca em um espaço de hipóteses possíveis, por aquela que terá um bom desempenho, mesmo quando passar a atuar sobre exemplos que vão além do conjunto de dados de treinamento.

Quando a saída do valor de y é um conjunto finito de valores, tais como ensolarado, nublado ou chuvoso, o problema de aprendizagem é chamado de **classificação**. Quando y for um número, como uma temperatura, por exemplo, o problema de aprendizagem é chamado de **regressão**.

4.1.1.1 - CLASSIFICAÇÃO

PARE E PENSE NISSO

A evolução dos aplicativos e os inúmeros dispositivos conectados à internet têm contribuído para a geração de volumes cada vez maiores de dados. Esses dados combinados àqueles gerados pelas operações das empresas, além daqueles capturados por meio de mídias sociais, geram uma gigantesca massa de informações que, quando bem utilizada, pode trazer inúmeros benefícios para as empresas. Uma das formas de se conseguir extrair valor a partir de grandes volumes de dados é a mineração de dados. Mas o que isso tem a ver com métodos de aprendizagem? Bem... a classificação, o agrupamento e a regressão são algumas tarefas importantes realizadas pela mineração de dados, e que utilizam como base os conceitos de Machine Learning. Essas tarefas permitem que padrões sejam descobertos ou que previsões sejam geradas a partir do processamento e análise dos dados. Interessante, não é mesmo?

A classificação implementa o conceito de aprendizado supervisionado, que é um tipo de aprendizado indutivo. Nela, os atributos de um conjunto de dados são separados em dois grupos. Um desses grupos contém o chamado atributo-alvo, para o qual se deseja fazer uma predição de valor. Esse atributo é sempre categórico, ou seja, composto por categorias ou classes. No outro grupo ficam os atributos previsores ou de predição.

Conforme apresentado na figura 4.2, esta tarefa busca por uma função que permita associar corretamente cada registro de de um conjunto de dados a um único rótulo de categoria, também chamado de classe. A partir de sua identificação, esta função pode ser aplicada a novos registros de forma que possa prever as classes em que tais registros se enquadram (GOLDSCHMIDT; PASSOS; BEZERRA, 2015).

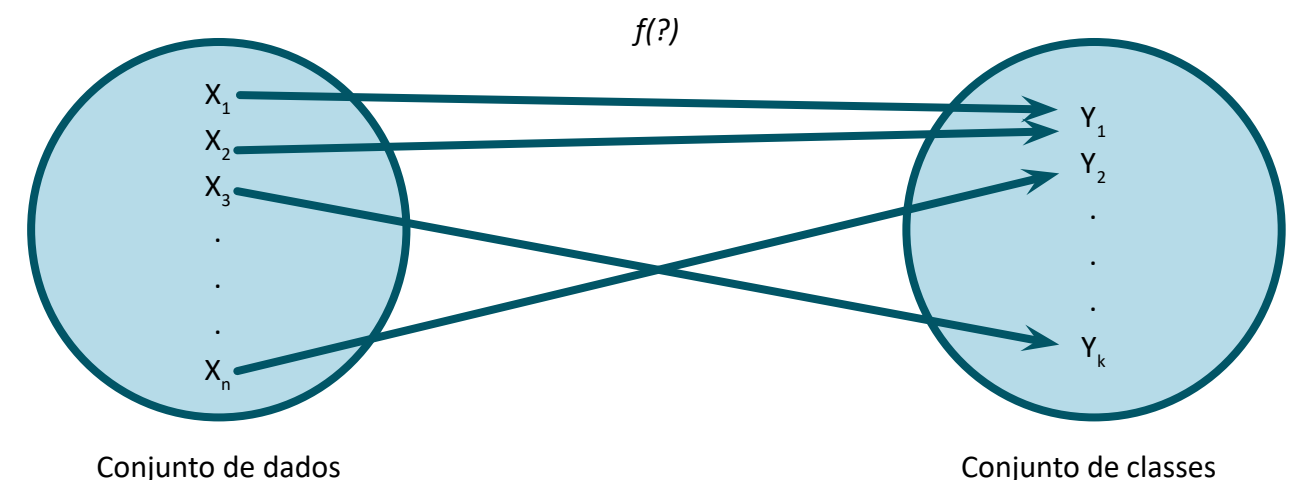


Figura 4.2 - Associações entre registros de dados e classes
Fonte: Goldschmidt, Passos e Bezerra (2015)

4.1.1.2 - REGRESSÃO

O objetivo da tarefa de regressão é buscar por funções, lineares ou não, que mapeiem registros de um conjunto de dados em valores reais. Ela é semelhante à classificação, porém fica restrita a conjuntos de atributos contínuos. Algumas de suas aplicações são, por exemplo, o cálculo da biomassa presente em uma floresta, estimativa de probabilidade de sobrevivência de um paciente, predição de riscos de investimentos, definição do limite do cartão de crédito, dentre outras (GOLDSCHMIDT; PASSOS; BEZERRA, 2015).

A **regressão linear** é a forma mais simples de regressão, na qual a função a ser abstraída a partir dos dados é uma função linear. O número de variáveis ou atributos varia de acordo com o problema. Na situação mais simples, a regressão linear, chamada de regressão linear bivariada, possui uma variável, chamada de variável dependente, e uma variável, chamada variável independente. O objetivo aqui é, a partir dos dados existentes, definir valores adequados para os parâmetros e da função linear representada pela equação 4.1.

$$Y = \alpha + \beta X$$

Equação 4.1 - Função linear
Fonte: Goldschmidt, Passos e Bezerra (2015)

A **regressão linear múltipla** é uma extensão da regressão linear bivariada, envolvendo mais de uma variável independente. Neste tipo de regressão, a variável dependente deve ser modelada como função linear de um vetor de características multidimensional, conforme a Equação 4.2. O método dos mínimos quadrados também pode ser estendido para obter os coeficientes.

$$Y = \alpha + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_k X_k$$

Equação 4.2
Fonte: Goldschmidt, Passos e Bezerra (2015)

Existem muitos problemas em que os dados não apresentam dependência linear entre si. Nestes casos, podem ser aplicadas técnicas de **regressão não linear**. Um exemplo desse tipo de técnica é a regressão polinomial que consiste em adicionar termos polinomiais ao modelo linear. Desta forma, ao aplicarmos transformações às variáveis, um modelo não linear pode ser convertido em um modelo linear, que pode, então, ser resolvido pelo modelo dos mínimos quadrados. A regressão pode ser utilizada em modelos como o Estatístico e o de Redes Neurais (GOLDSCHMIDT; PASSOS; BEZERRA, 2015).

4.1.2 - APRENDIZAGEM EM ÁRVORES DE DECISÃO

A indução de árvores de decisão é uma das formas mais simples de aprendizagem de máquina. As árvores podem representar todas as funções booleanas. Sua heurística de ganho de informações fornece um método eficiente que busca encontrar uma árvore de decisão simples e consistente.

Quando à representação, as árvores representam uma função que recebe como entrada um vetor de valores de atributos e retorna uma decisão, ou seja, um valor único. Os valores de entrada podem ser discretos ou contíguos. Para que a árvore possa chegar a um resultado, ela deve executar um conjunto de testes. Desta forma, cada nó interno da árvore representa um teste para um valor de um dos atributos de entrada, enquanto os nós externos são classificados com os valores possíveis daqueles atributos. Cada nó folha da árvore especifica o valor que deverá ser retornado pela função (RUSSEL; NORVING, 2004).

Como exemplo, vamos considerar uma árvore para decidir se iremos ou não esperar por uma mesa em um restaurante. No exemplo, o objetivo é aprender uma definição para o *predicado de objetivo* *VaiEsperar*. Os atributos que serão utilizados como parte da entrada são os seguintes:

- Alternativa: Se há um restaurante alternativo apropriado por perto;
- Bar: Se o restaurante tem uma área de bar confortável onde se possa esperar;
- Sex/Sáb: Verdadeiro às sextas e sábados;
- Faminto: Se estamos com fome;
- Clientes: Quantas pessoas estão no restaurante (valores: alguns, nenhum, cheio);
- Preço: As faixas de preço do restaurante (\$, \$\$, \$\$\$);

- Chovendo: Se está chovendo do lado de fora;
- Reserva: Se fizemos uma reserva;
- Tipo: O tipo do restaurante (francês, italiano, tailandês, de hambúrguer);
- EsperaEstimada: A espera estimada pelo gerente (0-10 minutos, 10-30, 30-60, >60).

A Figura 4.3 apresenta a estrutura da árvore de decisão para definir se vamos ou não esperar pela mesa.

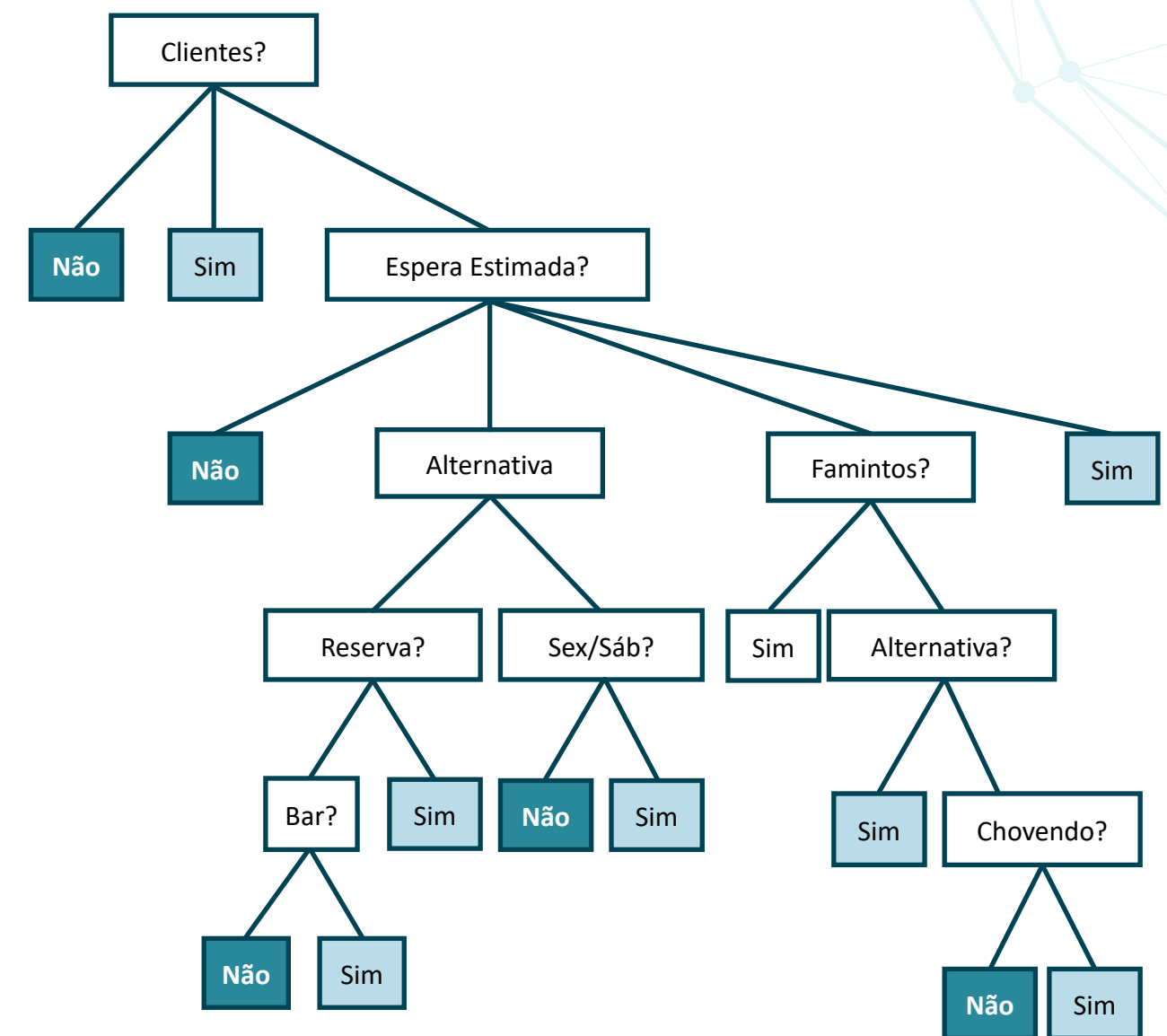


Figura 4.3 - Árvore de decisão para definir se vamos ou não esperar por uma mesa
Fonte: Russell e Norvig (2013)

4.2 - APRENDIZAGEM POR REFORÇO

No fim das contas, a árvore deverá possibilitar que conheçamos o caminho desde os atributos de entrada até a solução desejada. Este resultado poderá ser utilizado posteriormente para solucionar outros problemas semelhantes, ainda que os dados sejam outros (RUSSEL; NORVING, 2004).

Independentemente da abordagem utilizada, os algoritmos de aprendizado têm seu desempenho medido a partir de sua curva de aprendizagem. Essa curva mostra a exatidão da previsão quando submetido ao conjunto de dados de teste como uma função do tamanho do conjunto de treinamento. Para escolher entre as abordagens e definir qual é a melhor, pode ser utilizada a técnica da validação cruzada. Essa técnica permite que seja selecionado o modelo que irá fornecer as melhores generalizações.

Nos tópicos a seguir, falaremos em detalhes sobre outras formas de aprendizagem, além dos principais algoritmos que implementam as técnicas de Machine Learning.

No tópico anterior nós falamos sobre alguns conceitos importantes da aprendizagem de máquina, incluindo algumas das principais formas de aprendizagem. A partir de agora falaremos um pouco sobre a aprendizagem por reforço e sobre algumas curiosidades que envolvem a sua aplicação.

A aprendizagem por reforço é uma forma de aprendizagem na qual o agente utiliza feedbacks ou resultados de suas ações para realimentar o algoritmo e, desta forma, aprender o que é bom ou o que é ruim (RUSSELL; NORVIG, 2013). Se considerarmos, por exemplo, o caso de um agente que joga xadrez, ele poderia usar um xeque-mate que conseguiu dar em um adversário como uma forma de aprender que aquilo é algo bom. A partir daí, o agente poderá analisar as ações que o levaram até ali, de modo que possa replicá-las no futuro para ganhar (ou tentar ganhar) outros jogos.

Esses feedbacks recebidos pelos agentes podem ser chamados também de *recompensas* ou de *reforço*. No caso do exemplo do xadrez, a recompensa sempre vem ao final do jogo, entretanto, em outros contextos, para resolução de outros tipos de problemas, essa recompensa pode ser obtida com maior frequência. Em casos como do jogo de pingue-pongue, por exemplo, a recompensa não é obtida apenas no final, mas em cada ponto marcado no jogo. Nesse caso, o agente deve ser programado para considerar partes da entrada como sendo uma recompensa e assimilar cada um dos pontos ganhos ou perdidos (RUSSELL; NORVIG, 2013).

No fim das contas, o aprendizado, de maneira geral, visa maximizar a recompensa total esperada. No caso da aprendizagem por reforço, especificamente, ela busca utilizar as recompensas recebidas para estabelecer uma política ótima para o ambiente. Outro ponto importante nesse caso é o fato de que o agente não possui conhecimento prévio sobre o ambiente ou sobre o resultado esperado, antes de iniciar sua busca por soluções (RUSSELL; NORVIG, 2013).

A aprendizagem por reforço é, em muitos casos, a única alternativa possível para treinar programas de alto nível, como ocorre nos jogos, por exemplo. Nesses casos, o programa recebe a informação de quando ganhou ou perdeu e, a partir daí, busca encontrar uma função que lhe permita calcular as probabilidades de ganhar ou perder, a partir de cada ação possível de ser executada no espaço de estados.

Embora estejamos falando dos exemplos dos jogos, há outros exemplos de formas de aplicação desse tipo de aprendizagem: um deles é um agente que aprende sozinho como pilotar um helicóptero. A partir das recompensas positivas ou negativas, tais como colisões, queda, dentre outras, o agente aprende a voar sozinho (RUSSELL; NORVIG, 2013).

Nesse modelo de aprendizagem, sempre consideramos que o agente possui a capacidade de aprender sozinho, mas há situações em que o agente não conhece muito bem o ambiente ou as ações que pode executar. É o projeto do agente que determina quais são os tipos de informações que devem ser aprendidas e, conseqüentemente, quais são os tipos de problemas que podem ser resolvidos. Nesse contexto, temos três exemplos de projetos de agentes que podem ser aplicados para solucionar problemas (RUSSELL; NORVIG, 2013):

- **Agentes baseados na utilidade:** nesse modelo, o agente aprende uma função utilidade sobre os estados, e a utiliza para selecionar ações que maximizam a utilidade esperada do resultado. Esses agentes possuem também um modelo do ambiente, de forma que possam tomar decisões sabendo para onde as suas ações os levarão;

- **Agentes de aprendizagem Q (Q -Learning):** esses agentes aprendem uma função do tipo ação-valor, ou função Q , responsável por fornecer a utilidade esperada de se adotar uma determinada ação em um determinado estado. Esses agentes possuem a capacidade de comparar as utilidades de suas ações disponíveis, de forma que possam realizar observações antecipadas. O ponto negativo disso, é que suas possibilidades de aprendizagem podem acabar sendo restringidas;

- **Agentes reativos:** esse tipo de agente aprende uma política que faz o mapeamento direto de estados para ações.

Cada um desses tipos de agentes foi projetado baseado em modelos, usando um modelo e uma função de utilidade .

A seguir falaremos um pouco sobre a aprendizagem passiva, na qual a política do agente é fixa e a tarefa de aprendizagem consiste em aprender as utilidades dos estados, e sobre a aprendizagem ativa, na qual o agente também deve aprender o que fazer.

4.2.1 - APRENDIZAGEM POR REFORÇO PASSIVA

Na aprendizagem por reforço passiva, a política do agente é fixa, ou seja, em um determinado estado, chamado s , ele sempre executa a ação a . A meta desse agente é sempre aprender o quanto essa política é boa, ou seja, ele deve aprender uma função de utilidade

Para uma melhor compreensão de como isso funciona, vamos considerar como exemplo o mundo 4 X 3, representado pela figura 4.4. A lógica do mundo é que o agente, começando do estado inicial, deve escolher uma ação a cada passo de tempo. A interação do agente com o mundo (ambiente) termina quando ele alcança um dos estados objetivo, marcados com +1 ou -1. Nesse contexto de mundo, as ações são acima, abaixo, esquerda e direita.

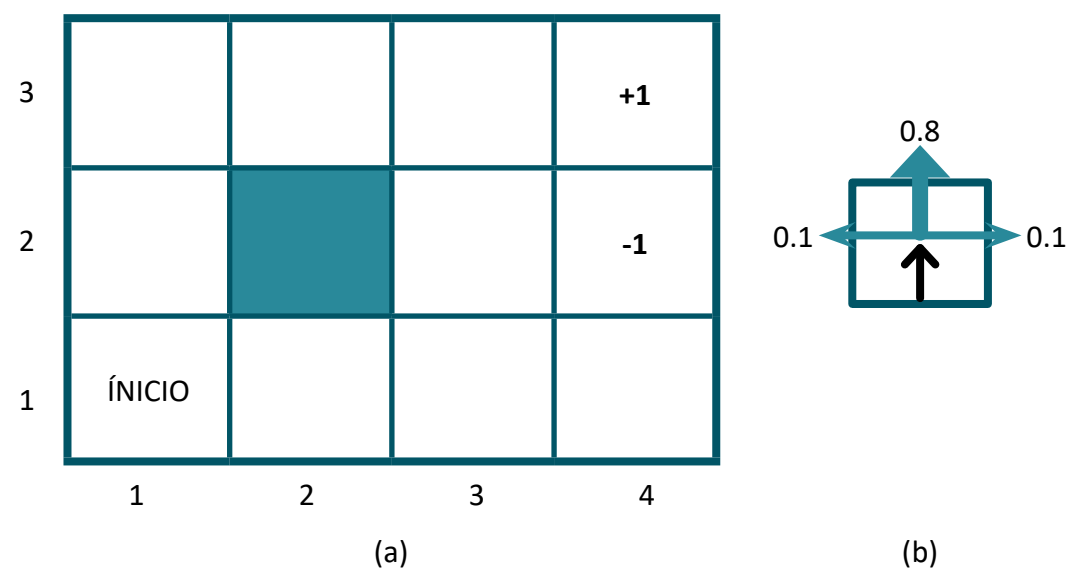


Figura 4.4 - (a) um ambiente simples de 4 X 3 que apresenta ao agente um problema de decisão sequencial. (b) Ilustração do modelo de transição do ambiente
Fonte: Russell e Norvig (2013)

O resultado pretendido para o problema da figura 4.4 ocorre com probabilidade 0,8, mas, com probabilidade de 0,2, o agente se move em um ângulo reto em relação à direção pretendida. Uma colisão com uma parede resulta em nenhum movimento. Os dois estados terminais têm recompensas +1 e -1, respectivamente, e todos os estados têm recompensas -0,04 (RUSSELL; NORVIG, 2013).

Na figura 4.5 temos um exemplo de política para o mundo 4 X 3, além das utilidades correspondentes. Nesse caso, o agente executa um conjunto de experiências no ambiente, sempre utilizando a sua política. Em cada experiência que o agente faz, ele começa no estado (1,1) e experimenta uma sequência de transições de ações, até que consiga alcançar um dos estados terminais (4,2) ou (4,3). As percepções do agente fornecem a ele tanto o estado atual quanto a recompensa recebida nesse estado (RUSSELL; NORVIG, 2013).

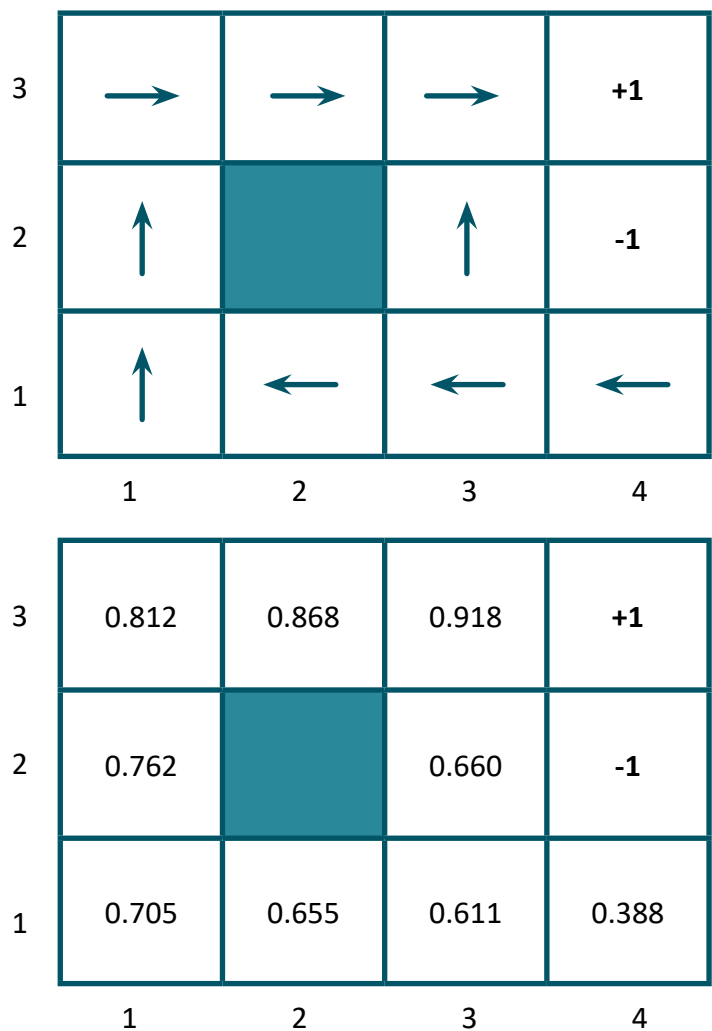


Figura 4.5 - (a) Uma política π para o mundo 4 X 3; essa política é ótima com recompensas iguais a $R(s) = -0,04$ nos estados não terminais e sem desconto. (b) Utilidades dos estados no mundo 4 X 3, dada a política π .
Fonte: Russell e Norvig (2013)

4.2.2 - APRENDIZAGEM POR REFORÇO ATIVA

Cada percepção que o agente tem sobre o estado está vinculada à recompensa recebida. Essa informação é utilizada como base para aprender a utilidade esperada associada a cada estado não terminal. A utilidade é o resultado da soma esperada de recompensas (descontadas) obtidas se a política é seguida. A equação utilizada nesse cálculo é apresentada na figura 4.6, onde: r é a recompensa para o estado, (uma variável aleatória) é o estado alcançado no tempo t quando é executada a política π . Está incluso na equação também um fator de desconto γ , cujo valor é $\gamma \in [0, 1]$ (RUSSELL; NORVIG, 2013).

$$U^\pi(s) = E \left[\sum_{t=0}^{\infty} \gamma^t R(S_t) \right]$$

Figura 4.6 - Equação do cálculo de utilidade
Fonte: Russell e Norvig (2013)

Conforme vimos, o agente de aprendizagem por reforço passivo tem uma política fixa que determina como esse agente deve se comportar. Quando falamos em agentes de aprendizagem ativa, estamos nos referindo a agentes que possuem a capacidade de decidir quais ações deverão ser executadas (RUSSELL; NORVIG, 2013). Como exemplo, vamos considerar um agente chamado de agente de programação dinâmica adaptativa, considerando também que ele possui a liberdade de decidir quais ações executar.

Os agentes de programação dinâmica adaptativa (PDA) levam vantagem das restrições que existem entre as utilidades dos estados, aprendendo o modelo de transição que os conecta, e resolvendo o processo de decisão utilizando um modelo de programação dinâmica. Se considerarmos um agente passivo, isso seria o equivalente a inserir o modelo de transição aprendido e as recompensas observadas, em uma equação para calcular as utilidades dos estados (RUSSELL; NORVIG, 2013).

Para começar, o agente precisa aprender um modelo completo e que contenha todas as probabilidades de resultados para todas as ações. Se fosse um agente passivo, ele aprenderia apenas o modelo para a política fixa. Após o agente ter aprendido o modelo, ele precisa aprender as utilidades definidas como ótimas. Essa definição é feita com base na equação representada pela figura 4.7.

$$U(s) = R(s) + \gamma \max_a \sum_{s'} P(s' | s, a) U(s')$$

Figura 4.7 - Equação para obtenção da utilidade dos estados
Fonte: Russell e Norvig (2013)

Após a definição da utilidade, o agente precisa determinar o que fazer em cada etapa. A partir dos dados de utilidade ótima obtidos para o modelo aprendido, o agente pode extrair uma ação ótima por meio de uma observação antecipada de um passo. Isso permite que ele maximize a utilidade esperada. Uma alternativa é que o agente utilize uma interação de política. Como a política já estará disponível, ele poderá simplesmente executar a ação que a política ótima recomendar (RUSSELL; NORVIG, 2013).

4.2.3 - BUSCA DE POLÍTICAS

A busca de políticas é um método de aprendizagem no qual o objetivo é ajustar uma política enquanto esses ajustes fizerem o seu desempenho melhorar. A partir do momento em que o desempenho não evoluir, os ajustes param. Antes de mais nada, é importante lembrarmos que a política π é uma função que mapeia estados em ações. O que interessa no caso da busca de políticas, são representações parametrizadas de π que possuem muito menos parâmetros que a quantidade de estados existentes no espaço de estados. Como exemplo, π poderia ser representado por uma coleção de funções Q parametrizadas, uma para cada ação, e executar a ação com o valor previsto mais alto (RUSSELL; NORVIG, 2013):

$$\pi(s) = \max_a \hat{Q}_\theta(s, a)$$

Figura 4.8 - Exemplo de função a ser executada para cada ação
Fonte: Russell e Norvig (2013)

A busca de políticas deverá ajustar os parâmetros para melhorar a política. Se a política é representada por funções Q , a busca resulta em um processo que aprende funções Q .

4.2.4 - APLICAÇÕES DA APRENDIZAGEM POR REFORÇO

Quando falamos em aplicações práticas da aprendizagem por reforço, duas das principais áreas em que isso ocorre são a de jogos e a de robótica. No primeiro caso, o modelo de transição é conhecido e o objetivo é o de aprender a função de utilidade. Já no segundo caso, o modelo geralmente é desconhecido.

Inicialmente vamos falar sobre a aplicação em jogos. O primeiro jogo criado utilizando os conceitos de aprendizagem foi um jogo de damas. Para isso, seu criador, Arthur Samuel, usou uma função linear ponderada para a avaliação das posições. O funcionamento do programa de Samuel apresentava diversas semelhanças com modelos que são utilizados atualmente.

Em uma primeira abordagem, ele atualizava os pesos usando a diferença entre o estado atual e o valor de cópia gerado através da previsão do efeito de escolha na árvore de busca. Isso permitia que o programa tivesse uma visão do espaço de estados. Em relação às recompensas, o algoritmo de Samuel não utilizava quaisquer recompensas. Isso quer dizer que os valores dos estados finais ou terminais eram simplesmente ignorados. Qual o problema disso? O objetivo do programa acabava não sendo, necessariamente, o de ganhar. De qualquer forma, Samuel encontrou alternativas que possibilitaram que o algoritmo fosse utilizado e produzisse bons jogos (RUSSELL; NORVIG, 2013).

Outro exemplo no contexto dos jogos foi o algoritmo de gamão (Neurogamão) de Gerry Tesauro. Na sua abordagem, Tesauro utilizou como base movimentos equivalentes a ações de especialistas humanos como base para o aprendizado do algoritmo. Inicialmente, o programa era bastante complexo a nível computacional, entretanto, para os seres humanos, era bastante limitado e entediante.

Um segundo projeto, chamado de TD-Gammon, foi uma tentativa de fazer com que o programa aprendesse a jogar sozinho. Esse aprendizado era possibilitado por um conjunto de recompensas dado ao final de cada jogo. A base para o algoritmo era uma rede neural contendo uma única camada oculta com, aproximadamente, 40 nós. Após uma série de evoluções, incluindo um considerável aumento dos nós da rede neural para 80, além de um treinamento com, aproximadamente, 300 mil partidas, o algoritmo foi capaz de jogar partidas em um nível similar ao dos 3 melhores jogadores humanos a nível mundial (RUSSELL; NORVIG, 2013).

Em relação à aplicação na área de robótica, o problema de balanceamento do carrinho com a vara (pêndulo invertido), é um dos mais conhecidos envolvendo os conceitos de aprendizado, e é apresentado na figura 4.9.

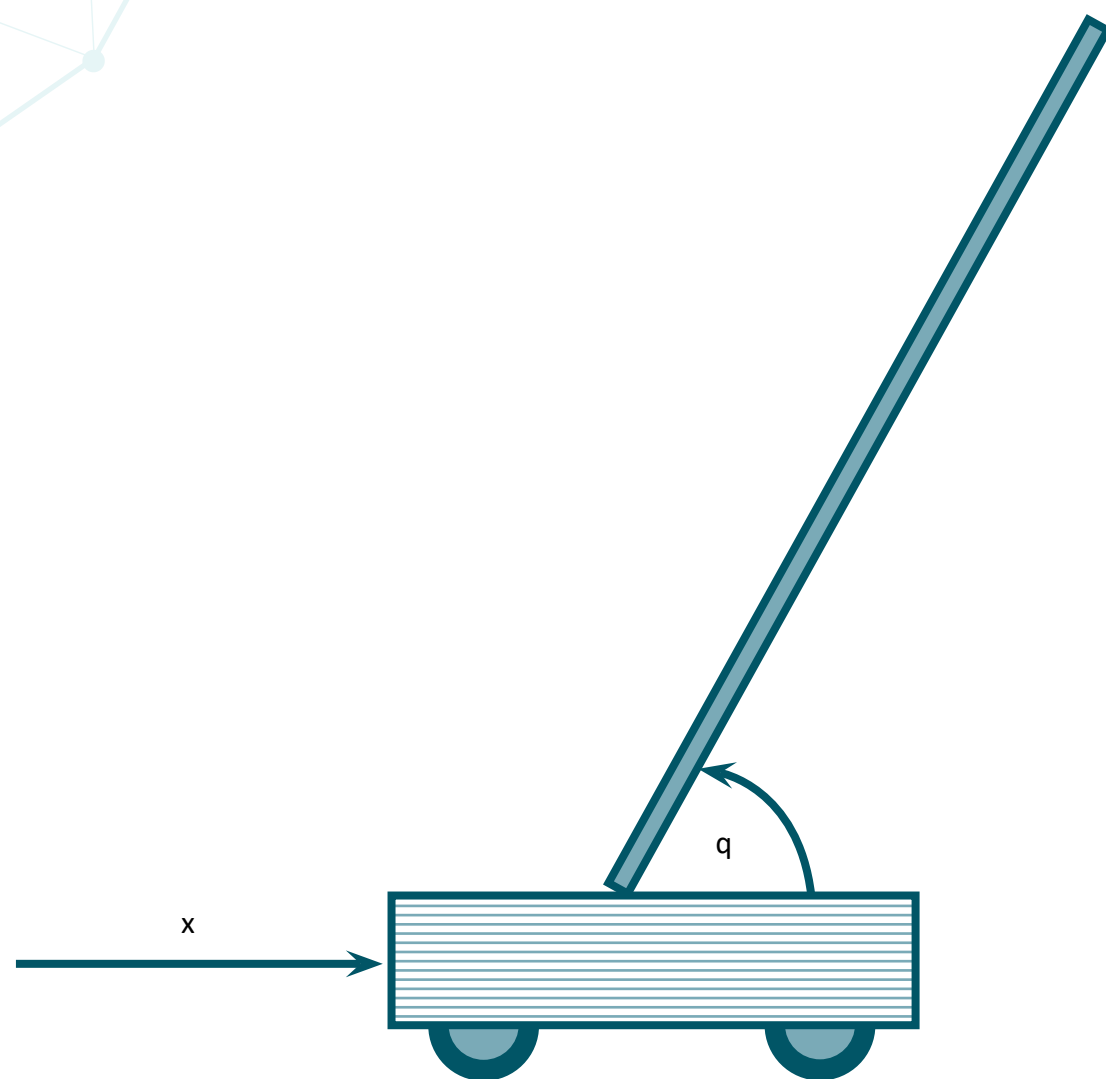


Figura 4.9 - Configuração do problema de equilibrar uma vara comprida em cima de um carrinho em movimento. O carrinho pode ser sacudido para a esquerda ou para a direita por um controlador
Fonte: Russell e Norvig (2013)

A ideia principal desse problema é controlar o carrinho, de modo que possamos garantir que a vara fique o mais próximo possível da vertical, enquanto permanece dentro dos limites da pista do carrinho. Embora pareça um problema simples, milhares de artigos científicos focados no aprendizado de máquina foram publicados sobre ele. Isso ocorreu, em parte, por conta da complexidade das variáveis envolvidas nesse problema (RUSSELL; NORVIG, 2013).

Foi a partir desse problema que os primeiros algoritmos de aprendizagem foram construídos. O primeiro, chamado Boxes, conseguiu equilibrar a vara por mais de 1 hora, depois de apenas 30 tentativas. Uma das principais diferenças entre o Boxes e outros trabalhos anteriores, foi o fato de que ele foi implementado usando materiais reais e não simulações. Durante as tentativas, a vara caía sobre o carrinho ou sobre a borda da pista e os feedbacks negativos foram utilizados para reforçar o aprendizado do algoritmo.

Outro exemplo bastante interessante foi a aplicação da aprendizagem por reforço em um voo de helicóptero (figura 4.10).



Figura 4.10 - Imagens superpostas de lapsos de tempo de um helicóptero autônomo executando uma manobra de círculo com nariz para dentro
Fonte: Russell e Norvig (2013)

4.3 - ALGORITMOS GENÉTICOS

Nesse exemplo, o helicóptero está sob o controle de uma política desenvolvida observando-se os efeitos de várias manipulações de controle no helicóptero real. Em seguida, o algoritmo foi executado no modelo de simulador durante a noite. Uma variedade de controladores foi desenvolvida para manobras diferentes. Em todos os casos, o desempenho do algoritmo excede de longe o de um piloto especialista humano, com a utilização de controle remoto (RUSSELL; NORVIG, 2013).

Devido ao seu potencial para eliminar a codificação manual, feita por programadores, de estratégias de controle, o aprendizado por reforço é uma área bastante ativa dentro do contexto do aprendizado de máquina. Suas aplicações, principalmente na área de robótica, são bastante promissoras e têm potencial para trazer contribuições bastante significativas para a resolução de problemas com IA.

Agora que você já conheceu um pouco melhor mais essa forma de aprendizado de máquina, estamos prontos para falar um pouco mais sobre algoritmos que implementam as técnicas que estamos estudando até o momento. Te espero no próximo tópico!

Os algoritmos genéticos (AG) são baseados em processos relacionados com organismos biológicos. Na natureza, por meio de muitas gerações, populações de uma mesma espécie evoluem de acordo com os princípios da seleção natural e sobrevivência do mais adaptado, inicialmente estudados por Charles Darwin. Os algoritmos são, portanto, baseados na teoria da evolução e na genética. Em geral, os AG são úteis em problemas complexos que envolvem otimização. Geralmente, esses problemas envolvem três componentes: variáveis, restrições e funções-objetivo (GOLDSCHMIDT; PASSOS; BEZERRA, 2015).

As variáveis descrevem os aspectos do problema. Cada possível combinação de valores das variáveis é um elemento contido em um conjunto chamado de espaço de busca (search space). As restrições indicam os valores que as variáveis podem assumir. Por último, as funções-objetivo definem a geografia básica do espaço de busca, e determinam quais técnicas podem ser utilizadas. As técnicas baseadas em modelos heurísticos como os AG não podem garantir que encontraram a solução ótima para o problema, entretanto, essas técnicas podem conseguir soluções muito próximas ou aceitáveis, chamadas também de subótimas.

Na modelagem por AG, cada cromossoma da população representa uma regra e cada gene representa um atributo do banco de dados. Cada regra, por sua vez, pode ser avaliada com relação à sua confiança e à sua abrangência. Quanto maiores forem a confiança e a abrangência de uma regra, melhor é o indivíduo que representa esta regra. Estas medidas podem ser utilizadas para avaliar a quantidade das regras geradas durante a execução do algoritmo genético (GOLDSCHMIDT; PASSOS; BEZERRA, 2015).

4.3.1 - EVOLUÇÃO DE REGRAS

Uma vez definida a modelagem de uma regra por meio de um cromossoma, a próxima definição a ser feita é em relação à forma de realizar o cruzamento genético (crossover) entre um par de cromossomas. A seguir temos um exemplo que ilustra o cruzamento genético entre duas regras, gerando duas novas regras. Consideremos como operador de cruzamento o crossover de um ponto cujo ponto de corte tenha ocorrido na posição 2.

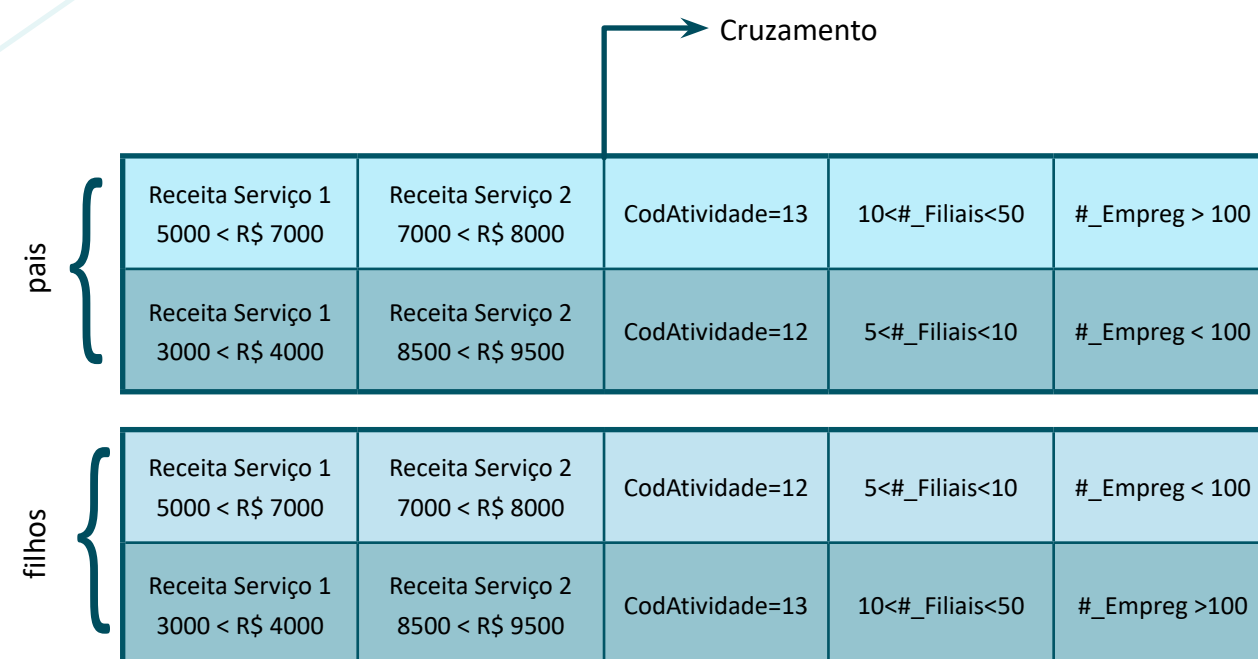


Figura 4.11 - Cruzamento genético entre duas regras para geração de outras regras
Fonte: Fonte: Goldschmidt, Passos e Bezerra (2015)

Os algoritmos genéticos podem ser usados para realizar um processo chamado de evolução de regras, baseado no processo de evolução natural que ocorre na natureza. A estrutura de uma regra pode ser representada conforme exemplo:

$SE C_1 E C_2 E \dots C_n ENTÃO Conclusão$

Os predicados $C_i, i=1, \dots, n$ são condições envolvendo atributos denominados **preditivos**. A conclusão da regra envolve um atributo denominado **objetivo**, que contém um valor fixo não manipulado pelo AG. Os atributos preditivos podem ser classificados quanto à natureza de seu domínio em **atributos contínuos ou atributos categóricos**.

A Figura 4.12 traz uma representação de cromossoma utilizada para codificar regras. Nesta representação, cada gene está associado a um atributo preditivo e compreende dois valores reais: um valor inferior e um valor superior. Para atributos contínuos, esses valores denotam o intervalo em que o respectivo atributo se encontra inserido. Para atributos categóricos, apenas o campo referente ao valor inferior é utilizado.

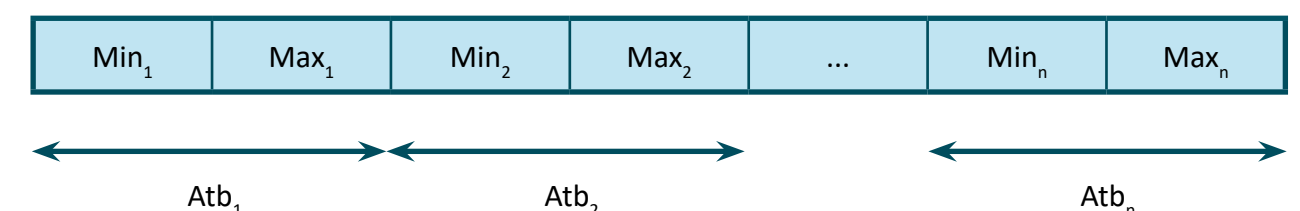


Figura 4.12 - Associações entre registros de dados e classes
Fonte: Goldschmidt, Passos e Bezerra (2015)

Para que possamos compreender melhor, vamos supor um atributo categórico com alfabeto de símbolos possível {v1,v2,...,vk} de cardinalidade k. Nesse exemplo, vamos considerar também o Ruler Evolver, um ambiente utilizado para mineração de dados. No Ruler Evolver, cada valor deste atributo contido em um cromossoma qualquer deve preservar a representação real, adaptando-se apenas a interpretação deste valor para caracterização dos símbolos do atributo categórico efetivamente representados. Tal interpretação compreende a conversão do valor real para uma representação binária onde cada dígito indica a presença (1) ou ausência (0) do símbolo correspondente.

Por exemplo, seja um atributo categórico denominado tipo de residência com alfabeto de cardinalidade 4 {própria, alugada, funcional, parentes}. A figura 4.13 mostra o mapeamento entre os valores inteiros possíveis para o atributo e os símbolos efetivamente representados. Esse mapeamento utiliza o operador lógico de disjunção “ou”.

Alelo	Decodificação	Tipo de Residência
0	0000	Não informada (null)
1	0001	Própria
2	0010	Alugada
3	0011	Própria ou alugada
...
15	1111	Própria ou alugada ou parente ou funcional (don't care)

Figura 4.13 - Mapeamento entre os valores do atributo tipo de residência e os símbolos representados.
Fonte: Goldschmidt, Passos e Bezerra (2015)

Os operadores genéticos disponíveis no Rule Evolver são os seguintes:

a) Crossover de um ponto, crossover de dois pontos, crossover uniforme, crossover de média, mutação simples e mutação don't care. Esses operadores são aplicáveis aos genes referentes a atributos quantitativos;

b) Crossover lógico e mutação lógica. Esses são operadores voltados exclusivamente para aplicação nos genes referentes a atributos categóricos. Conforme apresentado na figura 4.14, o crossover lógico implementa duas operações lógicas bit a bit entre as sequências binárias correspondentes aos atributos categóricos dos cromossomas pais: uma operação “e” e uma operação “ou”. A mutação lógica, exemplificada na figura 4.15, implementa uma operação de inversão dos bits das sequências binárias correspondentes aos atributos categóricos. Conforme apresentado na figura 4.14, o crossover lógico implementa duas operações lógicas bit a bit entre as sequências binárias correspondentes aos atributos categóricos dos cromossomas pais: uma operação “e” e uma operação “ou”.

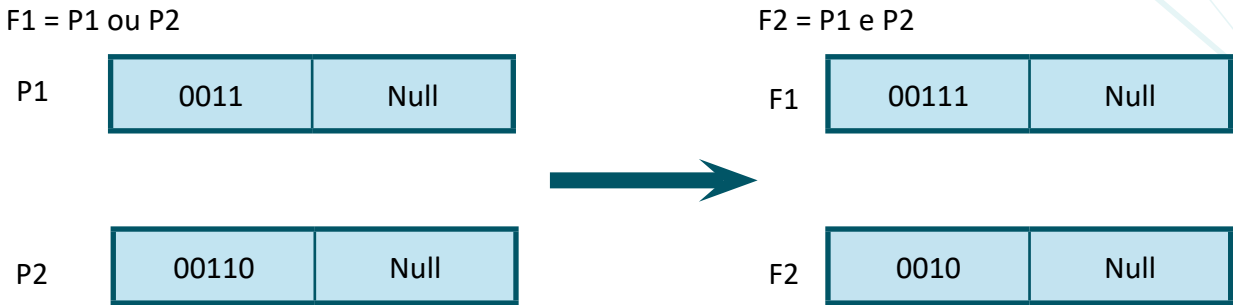


Figura 4.14. Crossover Lógico do Rule Evolver
Fonte: Goldschmidt, Passos e Bezerra (2015)



Figura 4.15 – Mutação Lógica do Rule Evolver
Fonte: Goldschmidt, Passos e Bezerra (2015)

O critério de seleção dos operadores baseia-se na interpolação linear das probabilidades de ocorrência de cada operador, ajustada em função da geração corrente. Tais probabilidades são parametrizadas pelo usuário do sistema conforme a aplicação.

4.3.2 - AGRUPAMENTO

O número de possibilidades com que se pode agrupar n objetos em k grupos é dado pela seguinte expressão:

$$N(n, k) = \frac{1}{k!} \sum_{i=0}^k (-1)^i \binom{k}{i} (k - i)^n$$

Se tomarmos $n = 25$ objetos e fixarmos $k = 5$ grupos, teremos $N(25,5) = 2.436.684.794.110.751$ maneiras de agrupar os 25 objetos em 5 grupos. Para 25 objetos existem mais de 4×10^8 possibilidades de clusters. Se o número de grupos for desconhecido, a complexidade do espaço de busca aumenta consideravelmente. Neste caso, podemos ter os objetos agrupados em $\sum_{i=1}^n N(n, k)$ possibilidades.

Estes exemplos ilustram a complexidade da tarefa de agrupamento e inviabilidade de resolvê-la por enumeração exaustiva. Como você pode perceber, é impraticável para um algoritmo de busca exaustiva encontrar a melhor solução nesse espaço de busca em um período de tempo computacionalmente viável. Nesse contexto, a aplicabilidade de AG acaba sendo a mais adequada (GOLDSCHMIDT; PASSOS; BEZERRA, 2015).

Existem diversas possibilidades de representação de cromossomas para solução de problemas de agrupamento, dentre as quais podemos citar como exemplo a seguinte: Suponha um conjunto de dados composto pelos elementos X_1, X_2, X_3, X_4, X_5 e X_6 . Considere também um agrupamento de tamanho 2 sobre esses dados, da seguinte forma: $\{(X_1, X_3, X_6), (X_2, X_4, X_5)\}$. Desse modo, podemos ter as representações de cromossomas apresentadas na figura 4.16.

A representação por grupamento de número é um cromossoma cujo número de genes é a quantidade de elementos a serem agrupados, e o valor de cada gene representa o cluster ao qual cada elemento pertence.

Já a representação por matriz utiliza uma matriz $n \times k$ para representar o cromossoma, onde cada linha corresponde a um cluster e cada coluna está associada a cada elemento. Cada coluna contém exatamente um “1”, pois cada elemento só pode pertencer a um único cluster.

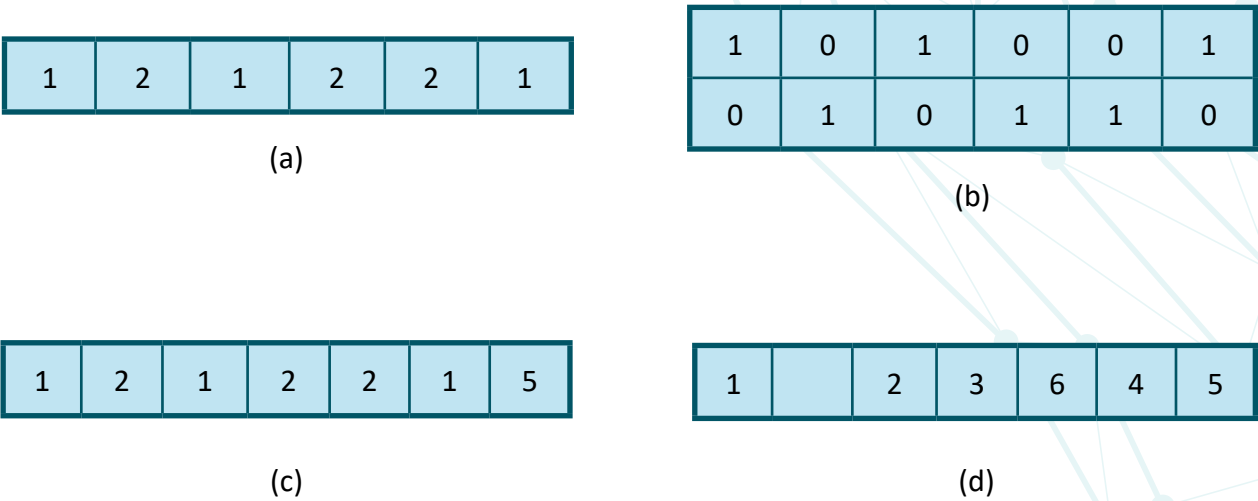


Figura 4.16 - Representações de cromossomas para clusters (x1, x3, x6), (x2, x4, x5): (a) grupamento de números; (b) matriz; (c) permutação com o caractere 7 representado como o separador de dois clusters; (d) permutação greedy. Fonte: Goldschmidt, Passos e Bezerra (2015)

A permutação com separador é a representação de cromossoma que utiliza inteiros $n + 1$ até $n + k + 1$ (ou outros separadores apropriados) para indicar os *clusters* na permutação. Cada gene do cromossoma pode ser o número do elemento, ou um separador, e o tamanho do cromossoma é dado por $n + k + 1$, onde n é o número de elementos e k o número de clusters.

A permutação **greedy** necessita de uma busca local no cromossoma para determinar a qual cluster o elemento pertence. Esta permutação utiliza os k primeiros genes do cromossoma para semear k *clusters*. Os genes restantes do cromossoma, na forma em que eles aparecem na permutação, são adicionados aos clusters como centroide mais próximo.

Para o problema da clusterização, com k grupos, qualquer cromossoma que não represente os k grupos é uma representação inválida. Dessa forma:

- A representação por grupamento de número que não inclui todos os números de clusters como valores para os genes é inválida;
- A representação do cromossoma por matriz que possui uma linha contendo apenas 0 é inválida;

- A representação do cromossoma por permutação com separador, com separadores adjacentes, ou com separadores no primeiro ou no último gene, é inválida; e
- A representação do cromossoma por permutação greedy que contenha valores repetidos também é uma representação inválida.

O operador de *crossover* é utilizado para transferir material genético de uma geração para outra.

É necessário que o cromossoma gerado pelo processo de crossover seja verificado, a fim de se evitar cromossomas inválidos. Além de cromossomas inválidos, existem também cromossomas redundantes, isto é, cromossomas com codificação diferentes que representam, na verdade, a mesma solução (figura 4.17).

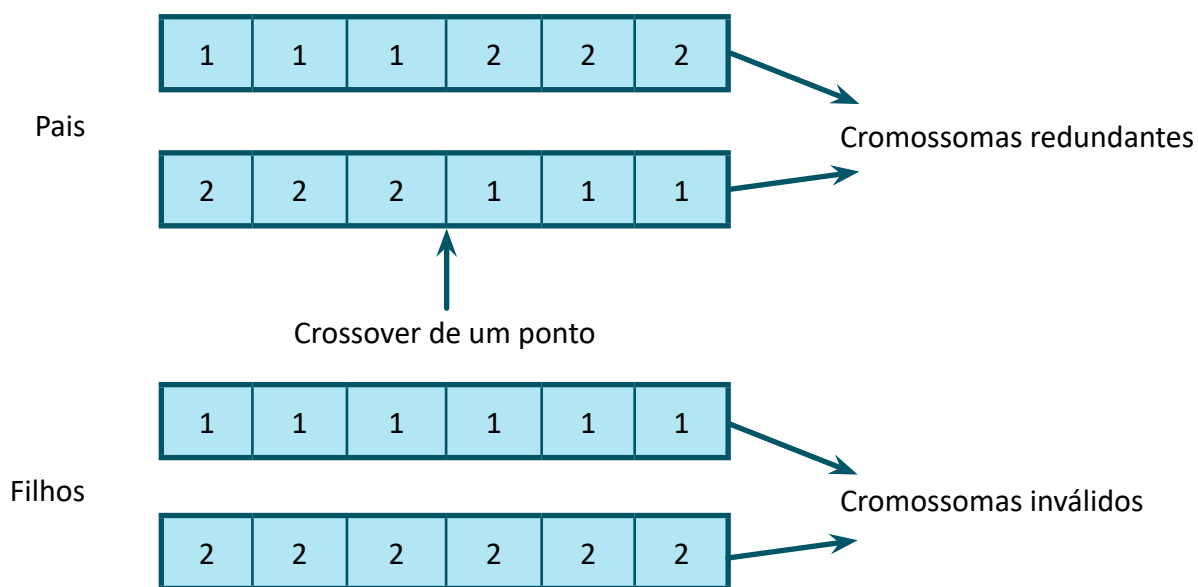


Figura 4.17 – Representação do crossover de cromossomas redundantes gerando cromossomas inválidos. Fonte: Goldschmidt, Passos e Bezerra (2015)

A seguir encontram-se relacionados alguns tipos de crossover que variam em função da representação utilizada no cromossoma:

Representação do cromossoma por agrupamento de número:

Crossover de um ponto e crossover uniforme podem ser utilizados para essa representação de cromossoma. Entretanto, ambos os operadores podem produzir cromossomas inválidos. Para evitar esse tipo de problema, pode-se usar o crossover baseado em ordem.

Representação do cromossoma por matriz: Dois tipos de crossover podem ser utilizados para essa representação, sendo eles: *crossover assexual* de um ponto e *crossover sexual* de dois pontos. Ambos os crossovers podem gerar soluções inválidas, isto é, cromossomas com linhas somente de elementos 0. A figura 4.18 e a figura 4.19 mostram o crossover assexual de um ponto e o crossover sexual de dois pontos, respectivamente.

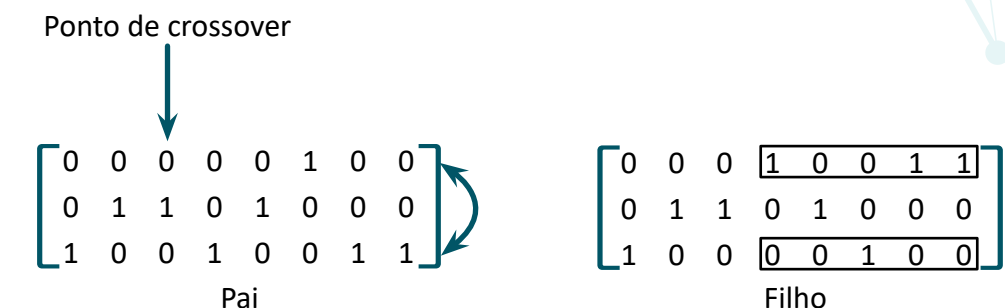


Figura 4.18 – Crossover assexual de 1 ponto
Fonte: Goldschmidt, Passos e Bezerra (2015)

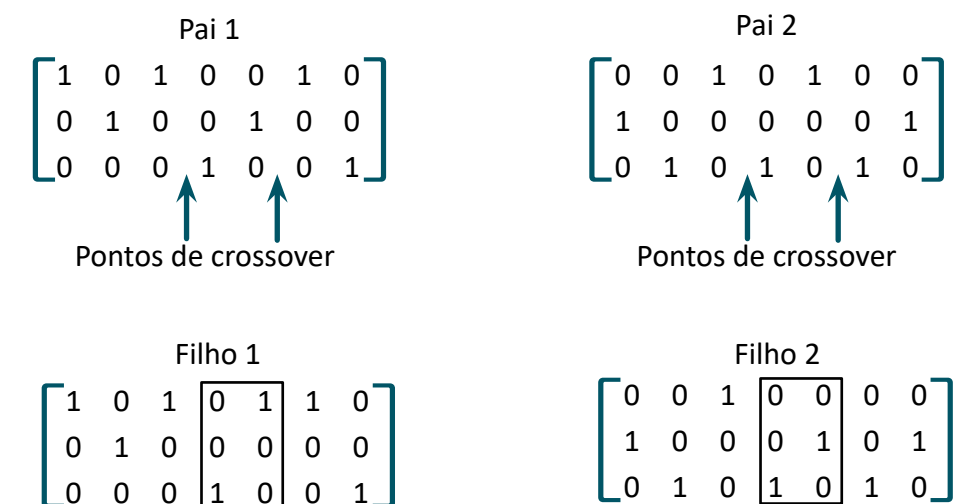


Figura 4.19 – Crossover sexual de 2 pontos
Fonte: Goldschmidt, Passos e Bezerra (2015)

Representação do cromossoma por permutação com separador:

Para esse tipo de representação são recomendados dois tipos de crossover, sendo eles: *crossover baseado em ordem* e *crossover de mapeamento parcial* (figura 4.20).

Representação do cromossoma por permutação greedy: Os operadores de crossover também são: o crossover baseado em mapeamento parcial e crossover baseado em ordem.

A mutação introduz novo material genético na população. Na tarefa de clusterização, a mutação corresponde a mover objetos de um cluster para outro cluster.

Representação do cromossoma por agrupamento de número: A mutação utilizada para esta representação consiste em inverter cada bit do cromossoma com a probabilidade igual à taxa de mutação.

Representação do cromossoma por matriz: Neste caso, o processo de mutação ocorre da seguinte maneira: Uma coluna da matriz sofre a mutação, ou seja, um elemento da coluna da matriz é escolhido aleatoriamente e é colocado o valor “1” para esse elemento. Todos os outros elementos da coluna recebem o valor “0”. Se o elemento escolhido já era de valor “1” o operador não faz efeito. A Figura 4.21 ilustra o operador de mutação.

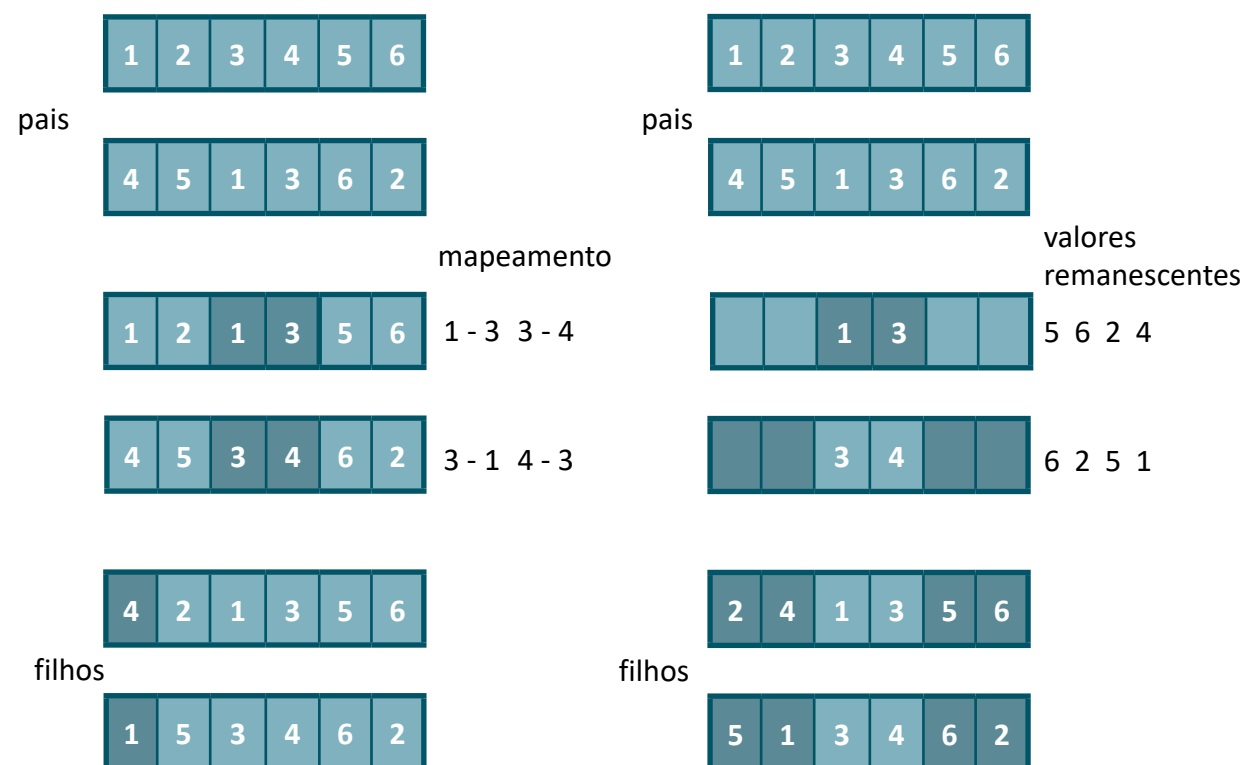


Figura 4.20 - Operadores de crossover para restauração de cromossoma por permutação: (a) Crossover baseado em mapeamento parcial; (b) Crossover baseado em ordem.
Fonte: Goldschmidt, Passos e Bezerra (2015)

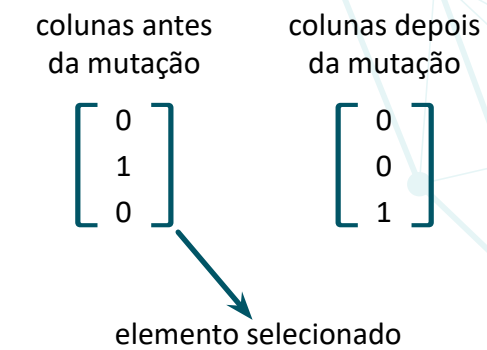


Figura 4.21 - Operador de mutação para representação do cromossoma por matriz
Fonte: Goldschmidt, Passos e Bezerra (2015)

Representação do cromossoma por permutação com separador:

A mutação utilizada nessa representação seleciona aleatoriamente dois elementos e realiza a troca de posição deles. Para assegurar que não serão gerados cromossomas inválidos, não se pode trocar de posição o separador de grupos.

Representação do cromossoma por permutação de greedy:

A mutação utilizada nesse caso é a mesma utilizada no caso da representação por permutação com separador.

As funções de avaliação que são comumente utilizadas para problemas envolvendo a tarefa de classificação são:

- Minimização de traço (\mathbb{W});
- Minimização do determinante de (\mathbb{W});
- Maximização do traço ($\mathbb{B}\mathbb{W}^{-1}$).

$$W = \sum_{i=1}^k \sum_{j=1}^{n_i} (x_{ij} - \bar{x}_i) (x_{ij} - \bar{x}_i) \quad W = B = \sum_{i=1}^k n_i (\bar{x}_i - \bar{x}) (\bar{x}_i - \bar{x})'$$

4.4 - LÓGICA DIFUSA

A lógica difusa é uma técnica que permite construir sistemas que lidem com informações imprecisas ou subjetivas. Diferente da lógica clássica, a lógica difusa (também conhecida como lógica fuzzy ou lógica nebulosa) oferece flexibilidade na definição e na avaliação de conceitos.

A ideia principal dessa lógica é a de que não há obrigação em relação a responder simplesmente “sim” ou “não” a um problema. A partir de sua aplicação, é possível fazer afirmações do tipo “sim”, “não”, “talvez” ou “quase” (RIGNEL; CHENCI; LUCAS, 2011). Um exemplo de um problema que poderia ser resolvido por meio da lógica difusa, é determinar se uma pessoa está ou não na meia idade, considerando que ela é um intervalo entre os 40 e os 55 anos.

Nesse contexto, aqueles 0 e 1 comuns na lógica clássica (booleana), em vez de serem considerados como verdadeiro (1) ou falso (0), passam a ser limites máximo e mínimo, respectivamente, de um intervalo em que há diversos valores possíveis. Para que um determinado valor seja classificado de acordo com essa lógica, são utilizados graus de pertinência. As associações de pertinência são calculadas por meio de funções de pertinência. Essas funções podem ser triangulares, trapezoides ou sigmoidais, conforme pode ser observado na figura 4.22.

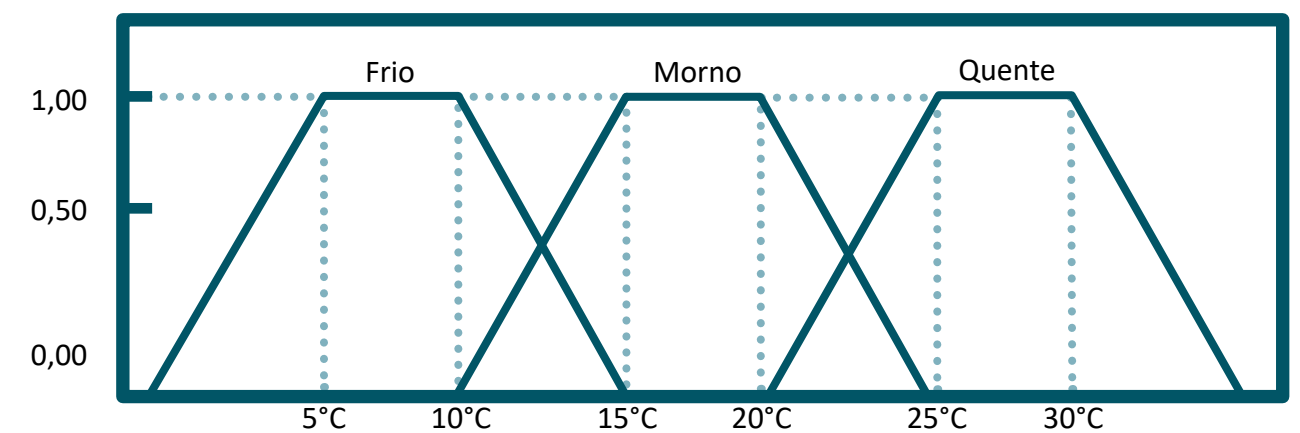


Figura 4.22 - Exemplo de uma função de pertinência
Fonte: Souza (2016)

Nas equações acima temos que:

- k é o número de clusters, n_i é o número de elementos do cluster i ;
- x_{ij} é o j -ésimo objeto do cluster i ;
- $\bar{x}_i = \frac{1}{n_i} \sum_{l=1}^{n_i} x_{il}$ é o centroide do cluster i .
- $\bar{x} = \frac{1}{n} \sum_{l=1}^n x_l$ é média de todos os objetos do conjunto de dados e ; e
- $()'$ significa a matriz transposta.

As matrizes B e W são muito utilizadas em análise discriminante. A minimização do traço (W) é equivalente à minimização da soma do quadrado da distância euclidiana entre o indivíduo e o centroide do seu cluster.

O algoritmo tenta colocar os elementos nos clusters cuja distância do elemento ao cluster seja a menor possível. O processo termina quando não há mais arranjos de elementos que minimizem a função de avaliação.

Como você pode observar na figura 4.22, há conjuntos de valores possíveis para que uma temperatura possa ser classificada como frio, morno ou quente. Quando o grau de pertinência de um determinado conjunto é igual a 1, dizemos que aquele valor pertence por completo àquele conjunto (classe), entretanto, é possível ter valores intermediários, mas que, ainda assim, garantam, por exemplo, pertinência ao conjunto frio, tal como 0,4 ou ao conjunto morno, tal como 0,6 (SOUZA, 2016). É possível ainda, que um determinado valor seja enquadrado em mais de um conjunto, a partir da definição dos graus de pertinência. Um exemplo disso é um dado que pode pertencer, por exemplo, 60% a um determinado conjunto (classe) e 40% a outro. Esse é mais um exemplo que diferencia a lógica difusa da clássica, na qual ou um elemento pertence a um determinado conjunto, ou não pertence (STROSKI, 2017).

Esses conjuntos, conhecidos como conjuntos nebulosos, podem ser entendidos como uma espécie de evolução da teoria clássica de Aristóteles. Segundo essa teoria, cada conjunto é chamado de crisp e os elementos podem ou não pertencer a ele. Quando falamos em conjuntos nebulosos, precisamos considerar o grau de pertinência de um determinado elemento a um determinado conjunto. Dois exemplos desses conjuntos, poderiam ser as pessoas com baixa renda e as pessoas com alta renda. Ao analisarmos esses exemplos, é possível verificar que não há limites que definem quando uma pessoa pertence ou não a qualquer um deles (RIGNEL; CHENCI; LUCAS, 2011).

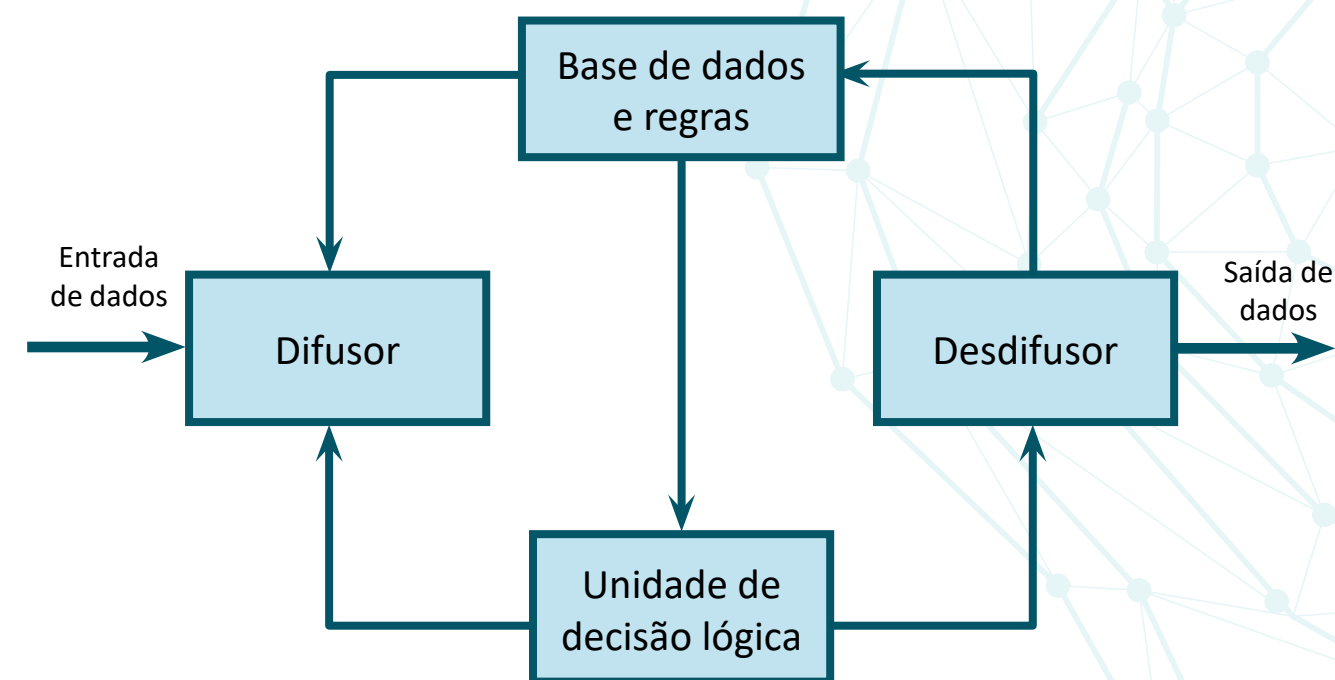


Figura 4.23 - Exemplo de estrutura de um sistema de lógica difusa
Fonte: (STROSKI, 2017)

A partir de um conjunto de funções de pertinência, por meio da utilização de operadores lógicos, é possível construir sistemas de lógica difusa. A figura 4.23 mostra um exemplo de funcionamento de um desses sistemas.

Nesse exemplo, a inferência é realizada na unidade lógica de decisão, tendo como base os dados e regras guardados na base de dados e regras. O difusor é a estrutura responsável por transformar os dados de entrada (binários) em valores de linguagem. A partir daí, esses dados são enquadrados em conjuntos, levando-se em consideração os seus graus de pertinência a esses conjuntos. Ao final, uma estrutura chamada de desdifusor converte os valores difusos em valores binários (STROSKI, 2017).

O resultado desses sistemas podem ser, por exemplo, um índice de qualidade, um percentual de risco de deslizamento, intensidade de uma frenagem, dentre outros.

4.4.1 - WANG-MENDEL

São diversas as aplicações da lógica nebulosa. Na mineração de dados, por exemplo, diversos métodos foram adaptados de forma que pudessem incorporar a flexibilidade proporcionada ela. Dentre esses métodos, podemos citar as versões nebulosas do K-means e do C4.5. Nestas versões, os registros da base de dados podem pertencer a diversos grupos e classes simultaneamente, com diferentes graus de pertinência. A seguir será descrito o algoritmo Wang-Mendel, concebido para a previsão de séries temporais.

Outros exemplos de aplicação são as câmeras com autofoco, a modelagem de sistemas para tomada de decisões e reconhecimento de padrões, computação com palavras em sistemas especialistas e controle de processos com linguagem natural. Sistemas como os de freios ABS e de controle com sensores para equipamento como aparelhos de ar condicionado são exemplos de sistemas do nosso cotidiano e que se utilizam dos conceitos de lógica difusa também (STROSKI, 2017).

O algoritmo Wang-Mendel foi concebido para aplicação em tarefas de previsão de séries temporais utilizando lógica nebulosa. Este método consiste em abstrair regras nebulosas a partir de um conjunto de dados históricos. O algoritmo utiliza esses dados históricos para definir os antecedentes e os consequentes das regras nebulosas. Para uma apresentação formal do método, consideremos $X(k)$, $k = 1, 2, \dots$ uma série temporal, em que $X(k) \in [U^-; U^+]$. O objetivo deste método consiste em abstrair um conjunto de regras nebulosas que permita, a partir de uma janela de n medidas de $X(k)$ ($X(k - n + 1), X(k - n + 2), \dots, X(k)$), determinar os valores de X , t pontos à frente ($X(k + 1), X(k + 2), \dots, X(k + t)$), sendo n e t valores positivos.

Como os valores a serem previstos dependem de n valores passados de X , cada regra nebulosa deverá possuir n antecedentes.

O algoritmo inicialmente divide a faixa de valores possíveis da série $[U^-; U^+]$ em m conjuntos nebulosos de igual comprimento, devendo ser m um valor ímpar. A figura 4.24 ilustra uma série temporal sendo dividida em sete conjuntos nebulosos.

O processo de geração das regras requer que sejam definidos o tamanho da janela n e o horizonte de previsão t . Ambos dependem do domínio da aplicação. Para ilustrar, consideremos o exemplo de uma aplicação hipotética em que a série temporal seja medida de 10 em 10 minutos. Nesse contexto, pode ser utilizada, por exemplo, uma janela de seis pontos, retratando a última hora anterior completa. O horizonte pode ser um ponto, representando a previsão para os dez minutos seguintes.

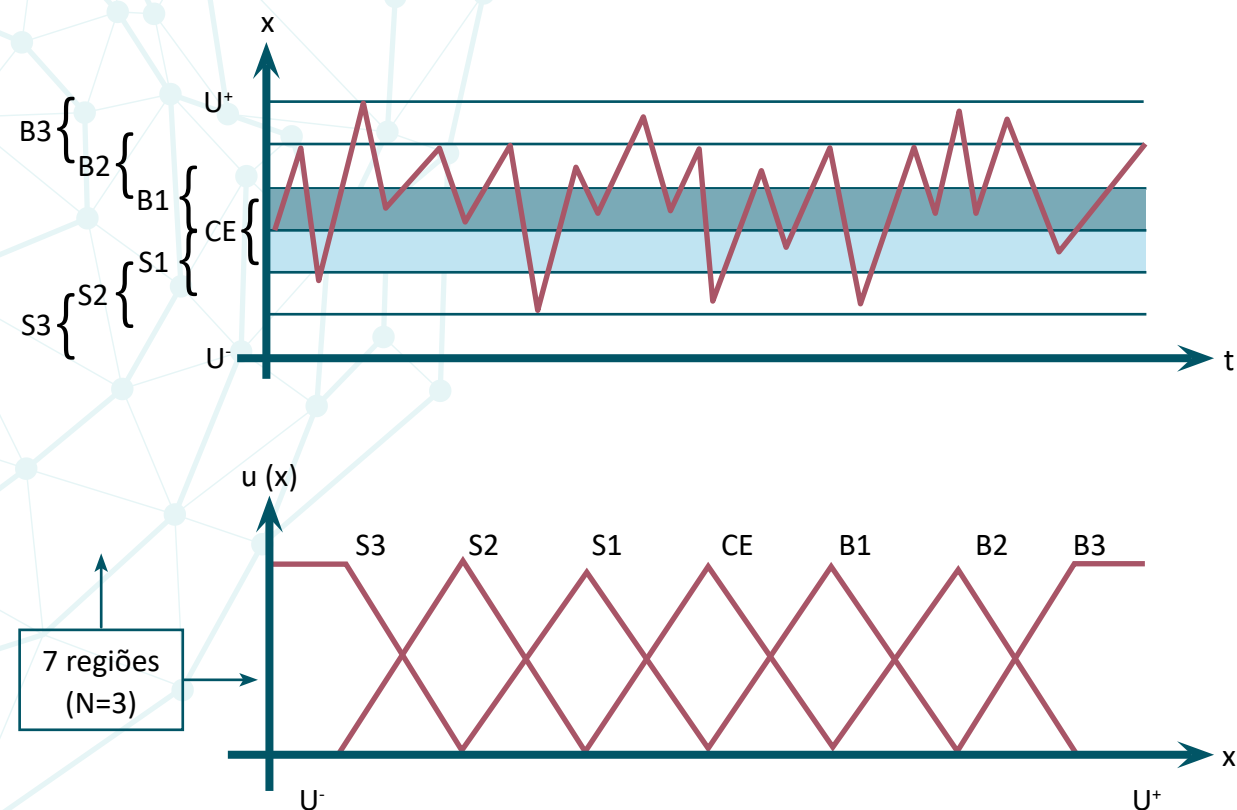


Figura 4.24 - Exemplot de série temporal dividida em 7 conjuntos nebulosos
Fonte: Goldschmidt, Passos e Bezerra (2015)

O processo de geração das regras segue, para cada regra j , os três passos descritos a seguir:

Determinar o grau de pertinência dos elementos de X_j . Considerando o exemplo anterior, teríamos:

Antecedente:

$$X^j(1) = \begin{cases} S3 = 0.8 \\ S2 = 0.2 \end{cases} \quad X^j(2) = \begin{cases} S1 = 0.8 \\ CE = 0.2 \end{cases} \quad X^j(3) = \begin{cases} S1 = 0.95 \\ CE = 0.05 \end{cases}$$

$$X^j(4) = \begin{cases} B1 = 0.9 \\ B2 = 0.1 \end{cases} \quad X^j(5) = \begin{cases} CE = 0.1 \end{cases} \quad X^j(6) = \begin{cases} B2 = 0.9 \\ B3 = 0.1 \end{cases}$$

Consequente:

$$X^j(7) = \begin{cases} S1 = 0.85 \\ CE = 0.15 \end{cases}$$

Atribuir a cada variável o conjunto com maior grau de pertinência:

$$\begin{aligned} X^j(1) &= S3 = 0.8 \\ X^j(2) &= S1 = 0.8 \\ X^j(3) &= S1 = 0.95 \\ X^j(4) &= B1 = 0.9 \\ X^j(5) &= CE = 1.0 \\ X^j(6) &= B2 = 0.9 \\ X^j(7) &= S1 = 0.85 \end{aligned}$$

Gerar uma regra para cada par entrada-saída. Considerando o exemplo anterior, teríamos a seguinte regra:

SE	$X^j(1)$ é	S3
E	$X^j(2)$ é	S1
E	$X^j(3)$ é	S1
E	$X^j(4)$ é	B1
E	$X^j(5)$ é	CE
E	$X^j(6)$ é	B2
ENTÃO	$X^j(7)$ é	S1

As regras obtidas são armazenadas em uma base de conhecimento para serem usadas posteriormente na previsão de valores mediante novas entradas. É importante observar que, em virtude da grande quantidade de dados, é comum obter-se regras conflitantes, ou seja, regras com o mesmo antecedente, mas consequentes distintos.

Para lidar com este problema, para a previsão de um novo valor da série, durante o processamento de cada nova janela associa-se um grau $D(R_i)$ a cada regra, multiplicando-se o grau de pertinência de cada termo do antecedente e do consequente. Utiliza-se então, aquela regra que possui maior grau entre as regras conflitantes.

Muito bem! Chegamos ao final de mais uma unidade, e também ao final do nosso e-book. Durante os quatro tópicos desta unidade, falamos sobre os principais conceitos relacionados ao aprendizado de máquina, suas particularidades, além de algumas aplicações importantes.

Falamos também sobre as principais formas de aprendizagem e sobre as principais características de cada uma. Para demonstrar o quão importante essas formas de aprendizagem são, elencamos também alguns exemplos de sua aplicação, tanto na área de jogos quanto na área de robótica.

Para finalizar, falamos sobre os conceitos de algoritmos genéticos e lógica difusa, além, é claro, de alguns dos algoritmos que implementam esses conceitos. Tanto os algoritmos genéticos quanto a lógica difusa possuem um grande número de aplicações. Algumas delas fazem parte do cotidiano das pessoas e isso acaba fazendo com que acabem passando despercebidas. Mas isso não minimiza sua importância e as grandes contribuições que têm gerado nas mais diversas áreas.

Espero que você tenha gostado do conteúdo e que tenha compreendido a essência do aprendizado de máquina!

Caro (a) aluno (a), no decorrer da nossa disciplina, falamos sobre vários conceitos importantes que envolvem a Inteligência Artificial, bem como sobre muitas de suas aplicações para resolução dos mais diversos tipos de problemas.

Com o a evolução dos recursos computacionais, da internet e dos aplicativos, volumes cada vez maiores de dados têm sido gerados, permitindo que a IA evolua e produza resultados cada vez mais interessantes e importantes para a humanidade.

Para que você possa aprimorar os seus conhecimentos e se aprofundar ainda mais no estudo da Inteligência Artificial, separei alguns materiais para leitura. A partir dessa leitura, além de aprofundar os conhecimentos que você adquiriu até aqui, você também poderá ter uma noção ainda mais clara do que a IA reserva para o futuro e sobre como ela tem sido usada para resolver os mais diversos tipos de problemas:



[Sistema de recomendação Web usando agentes](#)



[Prepare-se para a Revolução: Economia Colaborativa e Inteligência Artificial](#)



[Modelos probabilísticos gráficos aplicados à identificação de doenças](#)



[Deteção de fraudes em movimentações financeiras usando a técnica de sistema imunológico artificial](#)

Ótima leitura!

E assim chegamos ao final da nossa jornada de conhecimento sobre inteligência artificial. Durante as quatro unidades do nosso e-book você pode conhecer um pouco mais sobre o fascinante mundo da IA e sobre as estruturas que a compõem.

Muito daquilo que estudamos aqui tem sido aplicado no mundo inteiro, nas mais diversas áreas, incluindo a de saúde, transportes, educação, robótica, tecnologia da informação, aviação, dentre outras. Seja por meio dos algoritmos genéticos, das redes neurais ou do aprendizado de máquina, soluções que impulsionam negócios, salvam vidas, evitam perdas financeiras ou tornam experiências de compra mais satisfatórias, têm sido desenvolvidas em todo o mundo. Há uma grande possibilidade de que alguma solução baseada em IA esteja sendo criada muito perto de você!

Por isso, é muito importante que você se mantenha sempre ligado (a) nas evoluções tecnológicas e nas oportunidades que essas evoluções trazem para a área da TI em que você gostaria de atuar ou já atua. Estude sempre e busque sempre estar atualizado (a)! O mercado precisa de pessoas capacitadas, proativas e com vontade de crescer e fazer com que as empresas cresçam! O mercado precisa de pessoas empreendedoras!

Espero que você tenha gostado do conteúdo e possa aplicá-lo de forma prática nas suas atividades enquanto profissional de tecnologia da informação! Deixo a você um forte abraço e desejo muito sucesso em sua carreira!

REFERÊNCIAS

CANALTECH. **Inteligência Artificial chega aos sistemas antifraude com aprendizado de máquina.** Disponível em: <<https://bit.ly/2VMYjqL>>. Acesso em: 24 maio. 2019.

CARVALHO, A. P. DE L. F. DE. **Algoritmos Genéticos.**

CHAFFEY, D. 15 **Applications of Artificial Intelligence in Marketing.** Disponível em: <<https://bit.ly/2A0EN3P>>. Acesso em: 26 maio. 2019.

DAVIS, L. D. **Handbook of Genetic Algorithms.** 1a ed. New York: Van Nostrand Reinhold, 1991.

DSA. **Inteligência Artificial e o futuro da detecção de fraudes financeiras.** Disponível em: <<https://bit.ly/2W3gwFa>>. Acesso em: 24 maio. 2019.

FAUSTINO, R. **Como a inteligência artificial está mudando o sistema financeiro.** Disponível em: <<https://glo.bo/2W4Oh9r>>. Acesso em: 26 maio. 2019.

FERRARI, D. **Computação Natural: inspirando-se na natureza para resolver problemas.** Disponível em: <<https://goo.gl/yh9cFp>>. Acesso em: 25 set. 2018.

GOLDSCHMIDT, R; PASSOS, E; BEZERRA, E. **Data Mining: Conceitos, técnicas, algoritmos, orientações e aplicações.** 2. ed. ed. Rio de Janeiro: Elsevier, 2015.

GOMES, D. DA S. M; RODRIGUES, M. DE C. **Introdução à Lógica Fuzzy com Java.** Disponível em: <<https://www.devmedia.com.br/introducao-a-logica-fuzzy-com-java/32444>>. Acesso em: 2 jun. 2019.

LANHELLAS, R. **Redes Neurais Artificiais: Algoritmo Backpropagation.** Disponível em: <<https://bit.ly/2YySHC3>>. Acesso em: 18 maio. 2019.

MAGNUS, T. **Como a Inteligência Artificial pode ajudar no controle financeiro?** Disponível em: <<https://bit.ly/2JF7fN1>>. Acesso em: 26 maio. 2019.

MORENO, Y. **6 aplicações da Inteligência Artificial no Marketing.** Disponível em: <<https://bit.ly/2JENllk>>. Acesso em: 26 maio. 2019.

NOVAES, F. **IA e machine learning no combate às fraudes bancárias.** Disponível em: <<https://bit.ly/2YRejtJ>>. Acesso em: 24 maio. 2019.

PALMIERE, S. E. **Arquiteturas e Topoloias de Redes Neurais Artificiais.** Disponível em: <<https://bit.ly/2VFzTEi>>. Acesso em: 16 maio. 2019.

RIGNEL, D. G. de S; CHENCI, G. P; LUCAS, C. A. **Uma Introdução a Lógica Fuzzy.** Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica, v. 1, n. 1, p. 17–28, 2011.

SAMPAIO, L. M. D; OLIVEIRA, M. J. F. DE; IGNACIO, A. A. V. **Lógica Nebulosa: aplicações e tendências: SPOLM.** Rio de Janeiro - Brasil: [s.n.].

SAS. **Redes Neurais: o que são e qual a sua importância?** Disponível em: <<https://bit.ly/2W2OPwr>>. Acesso em: 24 maio. 2019a.

SAS. **Processamento de Linguagem Natural: O que é e qual sua importância?** Disponível em: <<https://bit.ly/2JG8NX2>>. Acesso em: 25 maio. 2019b.

SILVA, L. A. DA; PERES, S. M.; BOSCARIOLI, C. **Introdução à Mineração de Dados com aplicações em R.** 1. ed. Rio de Janeiro: Elsevier, 2016.

STEFANINI, M. **Medicina do futuro: IA para auxiliar em diagnósticos e aprimorar a relação médico-paciente.** Disponível em: <<https://bit.ly/2Wnnwwd>>. Acesso em: 26 maio. 2018.

REFERÊNCIAS

SOUZA, F. B. de. **O que é Lógica Difusa?** Disponível em: <<https://bit.ly/2Rwktgj>>. Acesso em: 22 jun. 2019.

STROSKI, P. N. **O que é Lógica Fuzzy?** Disponível em: <<https://bit.ly/2RrMxRW>>. Acesso em: 22 jun. 2019.

RODRIGUES, J. **O que é Processamento de Linguagem Natural.** Disponível em: <<https://bit.ly/2prbuzY>>. Acesso em: 25 maio. 2019.

ROMEDER, S. **5 maneiras de otimizar a cadeia de suprimentos usando AI.** Disponível em: <<https://bit.ly/2YKpzb3>>. Acesso em: 25 maio. 2019.

WAKKA, W. **Inteligência artificial é capaz de criar do zero novos compostos para remédios.** Disponível em: <<https://bit.ly/2Qs1e6X>>. Acesso em: 26 maio. 2019.

ZAP. **Humanos para quê? Inteligência Artificial recria a tabela periódica.** Disponível em: <<https://bit.ly/2VQBims>>. Acesso em: 26 maio. 2019.



**CENTRO
UNIVERSITÁRIO**



EAD

CENTRO UNIVERSITÁRIO FAG
EDUCAÇÃO A DISTÂNCIA
www.ead.fag.edu.br