Inteligência Artificial

Prof. Doutor Felipe Grando

felipegrando@uffs.edu.br

Aprendizagem de máquina

Melhorando o desempenho da máquina e seus algoritmos através de observações autônomas do ambiente

(RUSSELL; NORVIG, 2022. cap. 19-22)

Sistemas especialistas

- Esses sistemas dependiam exclusivamente do conhecimento de especialistas humanos sobre o domínio do ambiente para desempenharem suas operações através de regras-condições pré-estabelecidas programaticamente
  - Ainda são largamente utilizados como alternativas simplificadas para a resolução de problemas computacionais diversos onde já são conhecidas boas e eficientes soluções algorítmicas
- Predominantes durante as décadas de 70 e 80, gradualmente foram sendo substituídos por técnicas de aprendizado de máquina, que exigiam máquinas com maiores recursos computacionais mas que possibilitavam a resolução de instâncias e problemas mais complexos sem intervenção humana direta ou necessidade de especialistas no domínio
  - Vale ressaltar que especialistas de um domínio, quando disponíveis, podem e continuam auxiliando as técnicas de aprendizado, tornando-as mais eficientes e menos custosas computacionalmente

Aprendizado

- Qualquer componente de um sistema ou algoritmo pode ser melhorado através do aprendizado de máquina sem a necessidade de conhecimento prévio por parte do agente inteligente
- O aprendizado de máquina pode ser dividido em três grandes grupos quanto as técnicas utilizadas e tipos de problemas resolvidos
  - Aprendizado supervisionado: o agente usa dados rotulados para criar um modelo que mapeia novas entradas em saídas. Usado em problemas de classificação e regressão.
  - Aprendizado não-supervisionado: o agente cria um modelo que mapeia relacionamentos entre os dados sem qualquer rótulo previamente estabelecido. Usado em problemas de agrupamento (*clustering*), redução de dimensionalidade e geração de novos dados.
  - Aprendizado por reforço: o agente cria um modelo do ambiente a partir de uma série de recompensas e punições recebidas (observadas) em ambientes simulados. Usado em problemas de controle onde o agente interage diretamente com o ambiente do problema

Aprendizado não supervisionado

Observa exemplos de entradas e cria relacionamentos entre eles

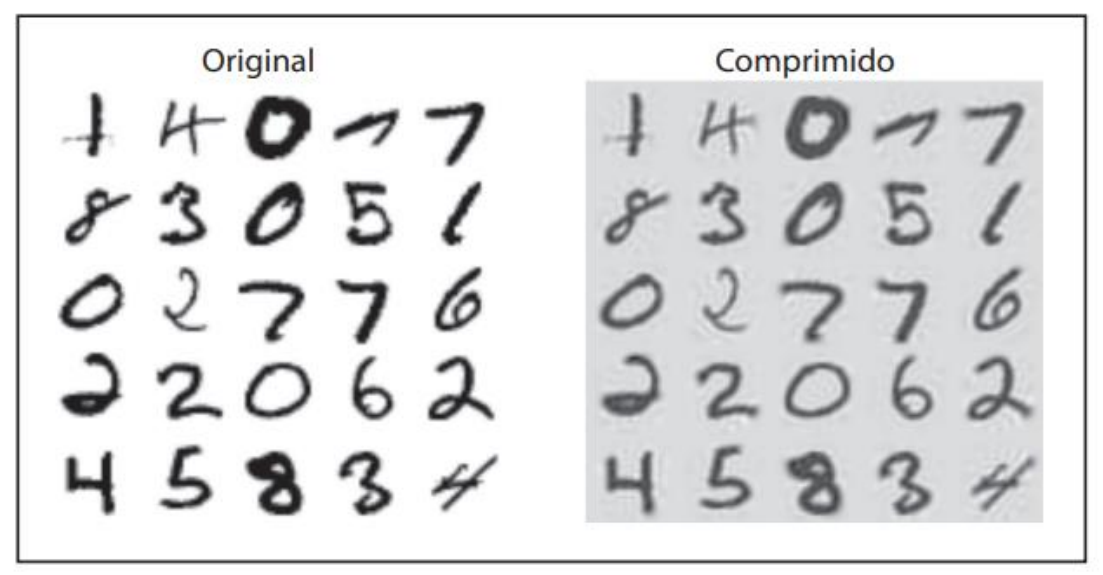
(RUSSELL; NORVIG, 2022. cap. 19-21)

PCA (Análise de Componentes Principais)

- Vantagens
  - Computacionalmente simples e eficiente
  - Fundamentado em conceitos de álgebra linear (vetores e transformações)
- Desvantagens
  - Usa como estimador apenas a variância dos dados em cada dimensão vetorial e considera que todos os atributos são independentes entre si
- Treinamento
  - Computa a variância representada por cada dimensão (atributo) e realiza uma projeção vetorial do conjunto em um número menor de dimensões ao eliminar as dimensões que representam menor variância

PCA – exemplo: redução de dimensionalidade

- Ao aplicar o PCA em um conjunto multidimensional como o da imagem ao lado é possível reduzir consideravelmente o tamanho do conjunto de entrada
- A imagem da direita representa a entrada original que foi comprimida, usando PCA mantendo-se 95% da variância original (5% de perda de informação), e depois descomprimido para a visualização
- É possível verificar que os dígitos ainda são visíveis mesmo que a entrada tenha cerca de 20% do tamanho original



- Original – 784 atributos
- Comprimido – 150 atributos

K-médias

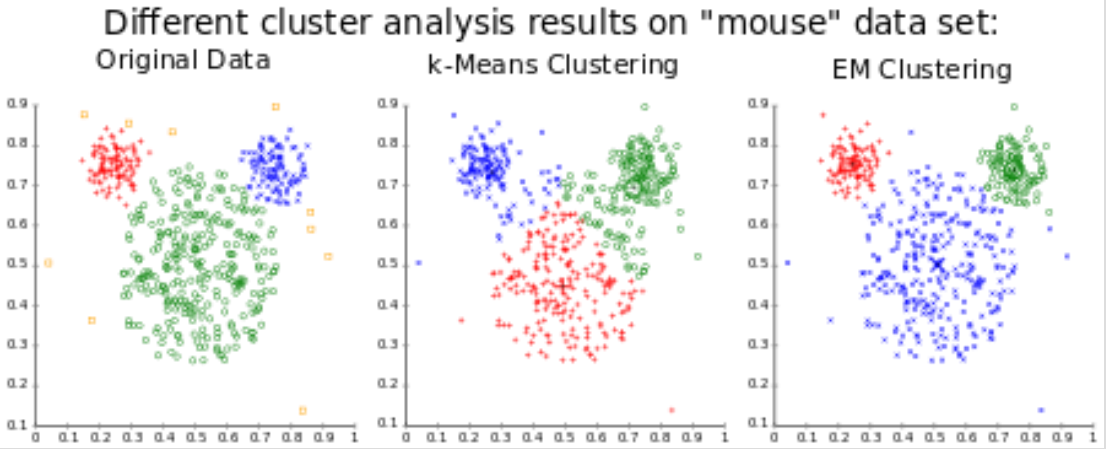
- Vantagens
  - Computacionalmente simples e eficiente
  - Fundamentado em estimadores estatísticos (média)
- Desvantagens
  - Cria apenas grupos de formato circular pois usa apenas a média como estimador
  - Dificuldade na estimativa do parâmetro k (quantidade de grupos)
- Treinamento
  - Computa interativamente k centroides a partir da média de valor dos elementos do grupo a qual está representando. Os elementos são redistribuídos entre os k centroides a cada iteração, usando o critério de proximidade, até que a convergência ocorra (centroides não mudam de valor/posição após uma iteração)

Algoritmo EM (Expectation Maximization)

- Vantagens
  - Mais flexível pois cria grupos de formato elíptico
  - Mais robusto a valores atípicos (outliers)
  - Fundamentado em estimadores estatísticos (média e variância)
- Desvantagens
  - Custoso computacionalmente, especialmente em problemas de alta dimensionalidade (quantidade de atributos de entrada)
  - Assume que os dados de entrada seguem uma distribuição Normal (Gaussiana)
- Treinamento
  - Computa interativamente k centroides a partir da média e da variância de valor dos elementos do grupo a qual está representando. Os elementos são redistribuídos entre os k centroides a cada iteração, usando critério probabilísticos de uma distribuição Normal de dados, até que a convergência ocorra (centroides não mudam de valor/posição após uma iteração)

K-médias e Algoritmo EM – exemplos

- A primeira imagem ao lado compara o agrupamento (k=3) realizado pelo k-médias e pelo algoritmo EM
- A segunda imagem compara a quantização de cores resultante de um algoritmo k-médias para uma mesma foto com diferentes valores para k



K-médias – passos

- Definir o valor de k, esse valor determinará a quantidade de grupos resultante
- Iniciar com k centroides aleatórios
  - Indica-se copiar valores de amostras aleatórias presentes no conjunto de treinamento
- Computar os grupos de cada amostra do conjunto de treinamento considerando o centroide mais próximo
  - Comumente utiliza-se a fórmula da distância Euclidiana *d* entre dois pontos *p* e *q* com *n* atributos (dimensão da entrada)

$$d(p,q) = \sqrt{\sum_{i}^{n}(p_i - q_i)^2}$$

- Computar a média de cada grupo, individualmente por atributo, e usar os valores computados como os novos valores dos centroides
- Repetir os dois últimos passos acima até que os centroides mantenham o mesmo valor

Exercício 1 – k-médias

- Construa um modelo de agrupamento usando o k-médias, com k=3, para o conjunto de entradas abaixo

A1	A2	i = 1	i = 2	i = 3
10	7			
5	5			
7	20			
12	1			
15	2			
16	7			
20	15			
2	10			
5	6			
9	22			

i	Centroide 1	Centroide 2	Centroide 3
1			
2			
3			

Exercício 2 – k-médias

- Construa um modelo de agrupamento usando o k-médias, com k=2, para o conjunto de entradas abaixo

Clima	Temperatura	Humidade	Vento
Chuvoso	Frio	Alta	Intenso
Ensolarado	Agradável	Baixa	Fraco
Nublado	Calor	Normal	Fraco
Nublado	Agradável	Normal	Fraco
Ensolarado	Calor	Normal	Fraco
Nublado	Frio	Normal	Intenso
Chuvoso	Calor	Alta	Intenso
Chuvoso	Agradável	Alta	Intenso
Ensolarado	Agradável	Baixa	Intenso
Nublado	Frio	Baixa	Fraco

Aprendizado supervisionado

Observa exemplos de pares de entradas e saídas e cria um mapeamento entre elas

(RUSSELL; NORVIG, 2022. cap. 19-21)

Regressão linear

- Vantagens
  - Computacionalmente simples, robusto e fácil de interpretar
  - Fundamentado em conceitos de cálculo (funções e derivadas)
- Desvantagens
  - Seu uso é restrito a dados linearmente dependentes
- Treinamento
  - Usa um conjunto de dados de treinamento rotulado (X, y) para aprender os pesos (W) adequados através da técnica de gradiente descendente

Regressão linear

- O objetivo de uma regressão é aprender um modelo que representa um conjunto de entradas.
- Uma regressão é linear quando o modelo gerado pode ser representado através de uma função de primeiro grau, na forma

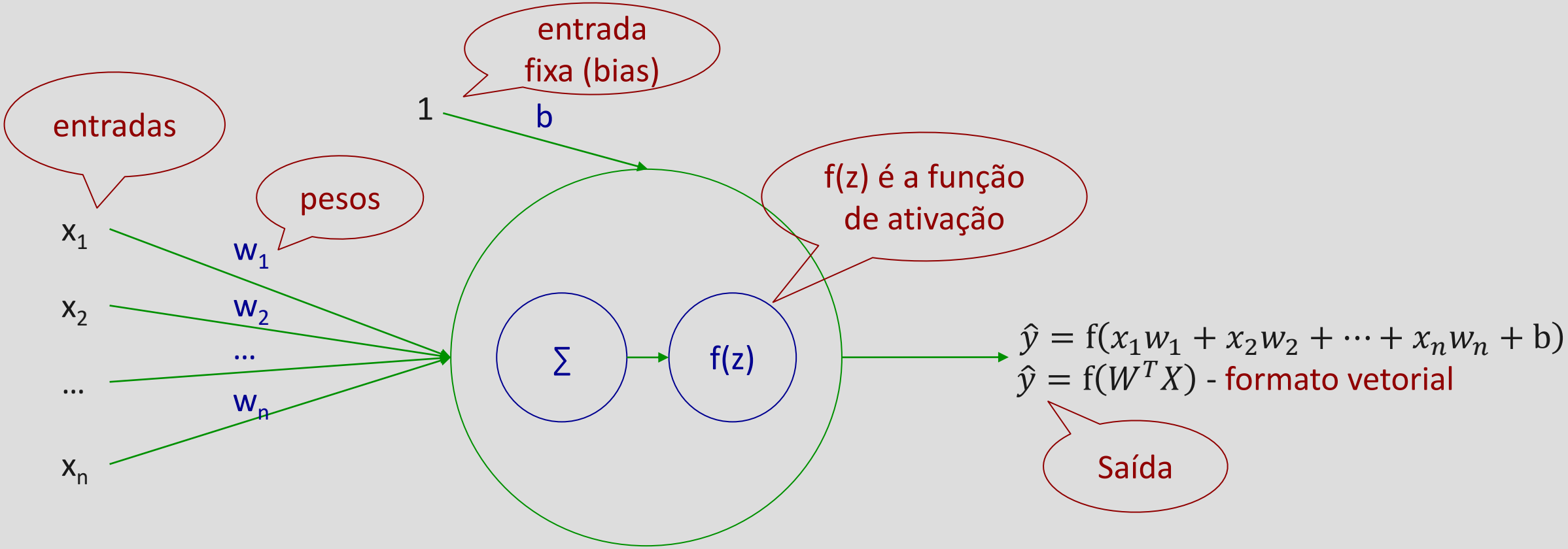
$$\hat{y} = W^T X + b$$

Variável	Descrição
$\hat{y}$	Previsão, $\hat{y} \in \mathbb{R}^N$ Modelo de y
X	Entradas, $X \in \mathbb{R}^{NXd}$
d	Dimensão das entradas (quantidade de atributos)
W^T	Vetor de pesos transposto, $W \in \mathbb{R}^d$ (coeficiente angular)
b	Bias, $b \in \mathbb{R}$ (coeficiente linear ou termo independente)
N	Número de instâncias de entrada

Redes neurais artificiais (RNA)

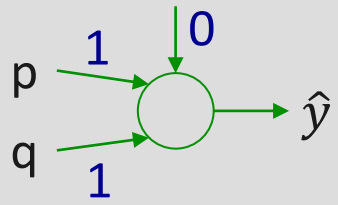
- Vantagens
  - Turing completo, aproximador universal de funções, flexível e gera modelos altamente eficientes em termos de custos computacionais
  - Fundamentado em conceitos de cálculo (funções e derivadas)
- Desvantagens
  - Dificuldade na estimativa dos meta-parâmetros (quantidade de neurônios, tipos, camadas, entre outros), treinamento mais custoso computacionalmente e modelos difíceis de serem interpretados (caixa preta)
- Treinamento
  - Usa um conjunto de dados de treinamento rotulado (X, y) para aprender os pesos (W) adequados através da técnica de gradiente descendente (backpropagation)

Neurônio artificial

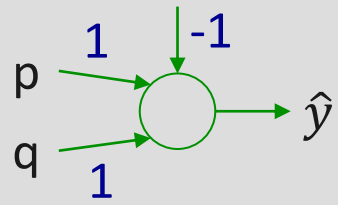


Exercício 4 – RNA

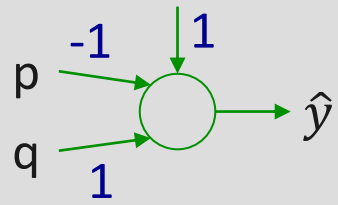
r1



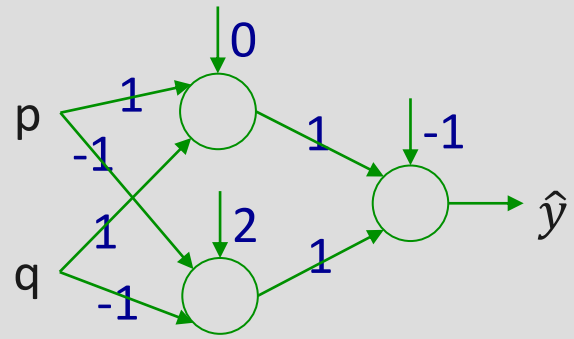
r2



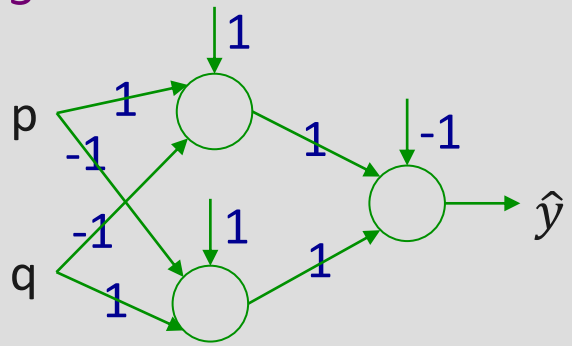
r3



r4



r5

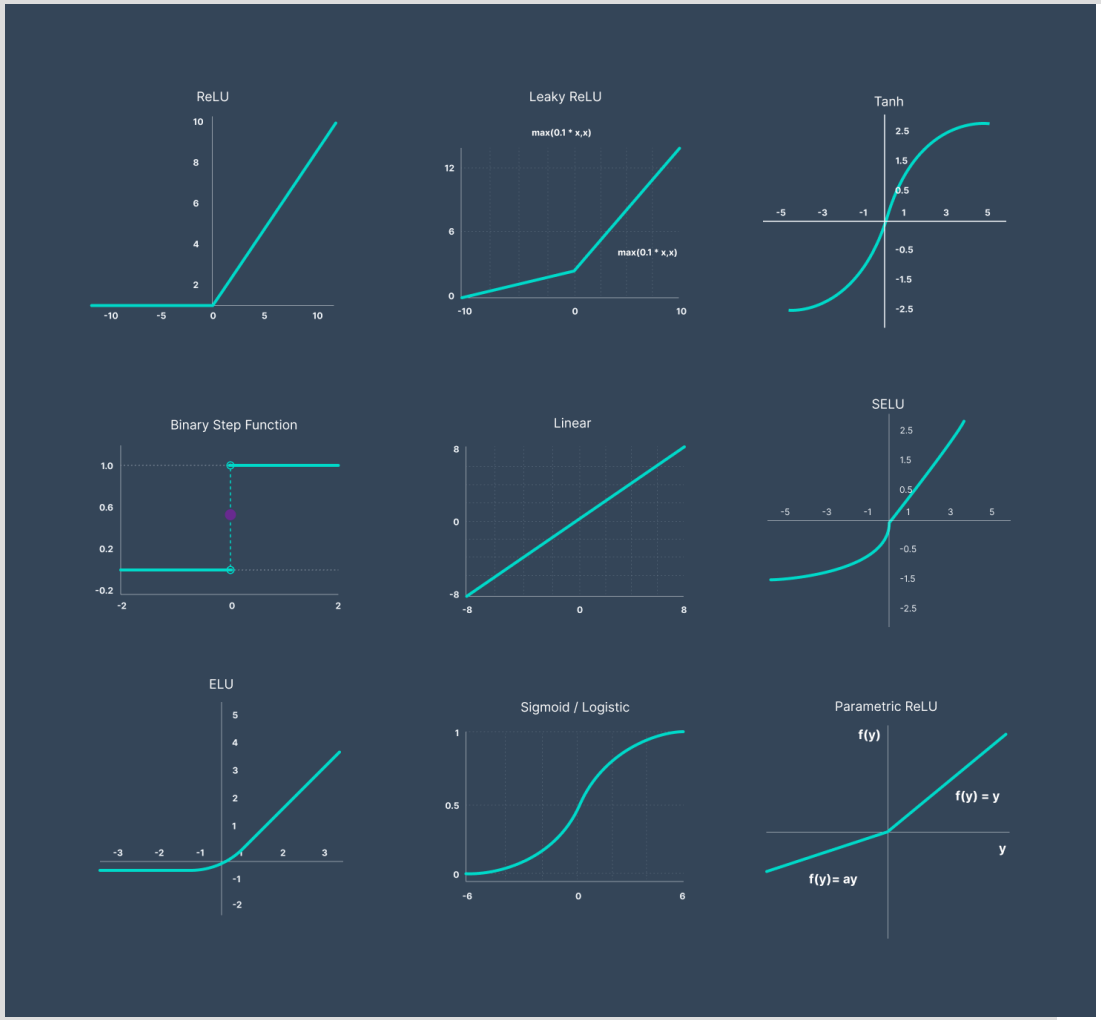


$$f = \begin{cases} 1 \; se \; x > 0 \\ 0 \; se \; x \leq 0 \end{cases}$$

p	q	r1	r2	r3	r4	r5
0	0					
0	1					
1	0					
1	1					

$$r1 = p \quad q$$
$$r2 = p \quad q$$
$$r3 = p \quad q$$
$$r4 = p \quad q$$
$$r5 = p \quad q$$

Funções de ativação



ReLU

$$f(x) = max\,(0, x)$$

Leaky ReLU

$$f(x) = max\,(0.1x, x)$$

Tanh

$$f(x) = \frac{(e^{x} - e^{-x})}{(e^{x} + e^{-x})}$$

Binary step

$$f(x) = \begin{cases} 0 & for\ x < 0 \\ 1 & for\ x \geqslant 0 \end{cases}$$

Linear

$$f(x) = x$$

SELU

$$f(\alpha, x) = \lambda \begin{cases} \alpha(e^{x} - 1) & for\ x < 0 \\ x & for\ x \geqslant 0 \end{cases}$$

ELU

$$\begin{cases} x & for\ x \geqslant 0 \\ \alpha(e^{x} - 1) & for\ x < 0 \end{cases}$$

Sigmoid / Logistic

$$f(x) = \frac{1}{1 + e^{-x}}$$

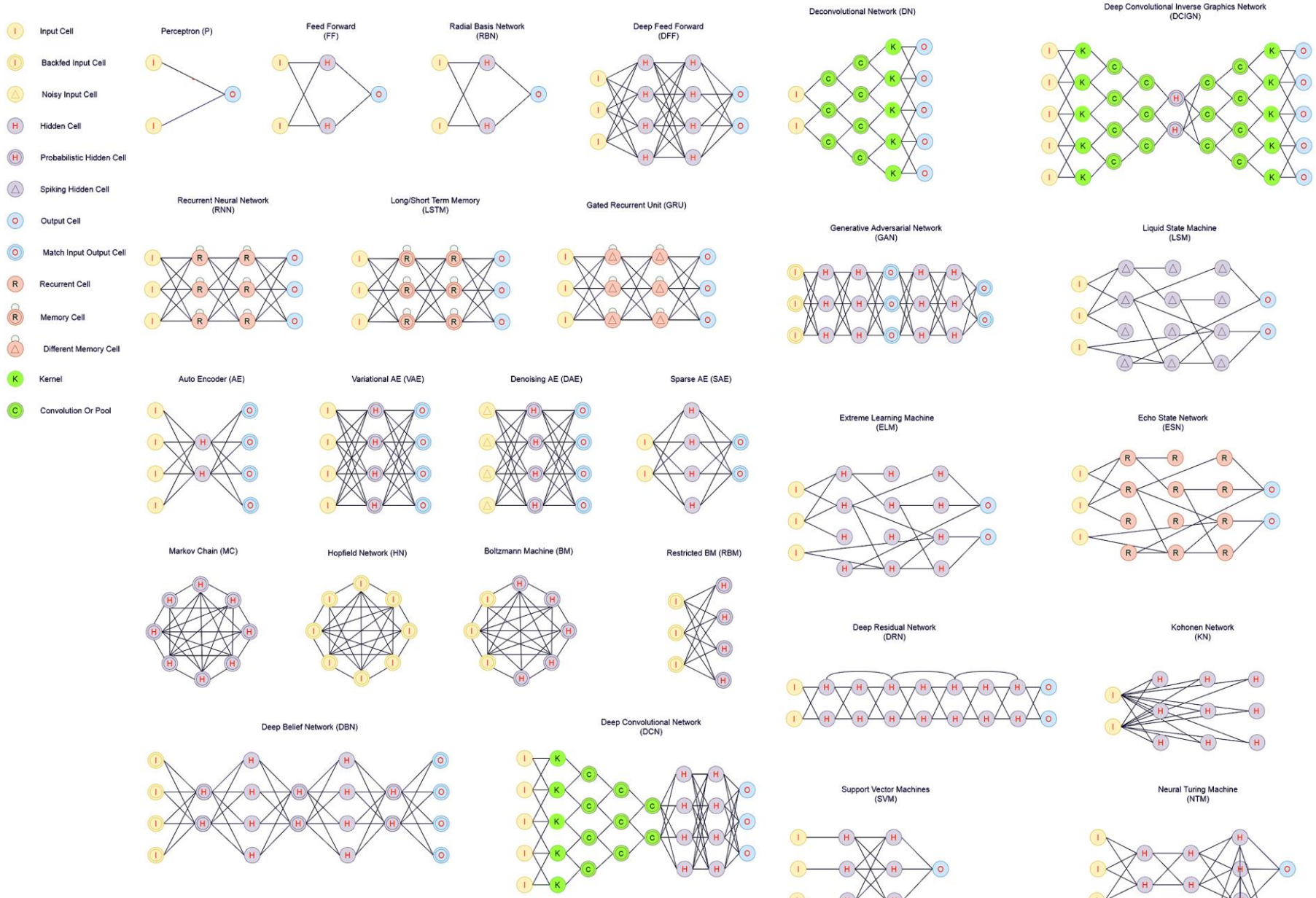
Parametric ReLU

$$f(x) = max\,(ax, x)$$

Softmax

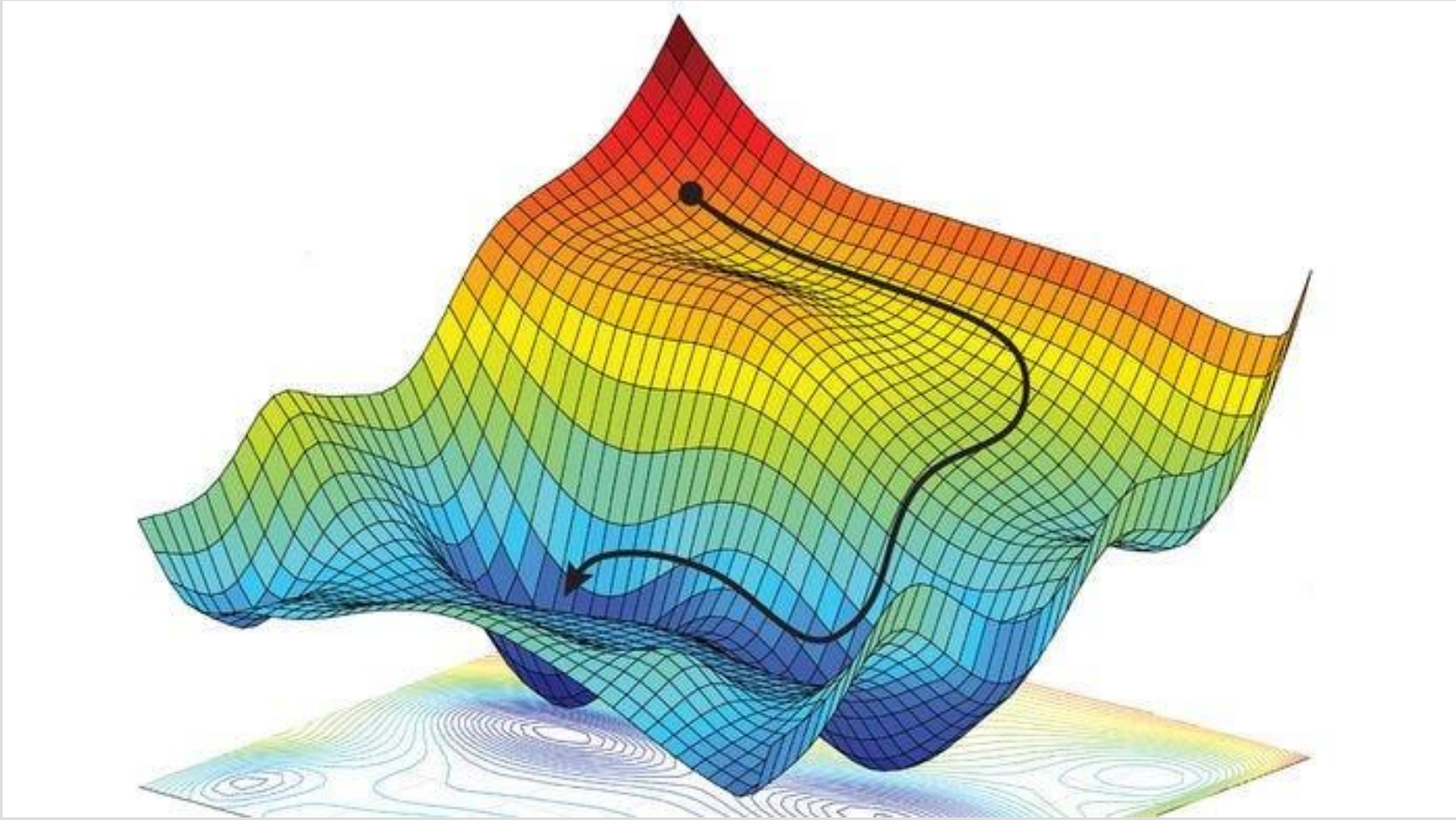
$$\text{softmax}(z_i) = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

V7 Labs



Tipos de topologias de redes neurais

Gradiente Descendente



Gradiente Descendente – Conceito

- Gradiente é variação de uma grandeza em determinada direção
  - A ideia é computarmos o gradiente descendente (direção que minimiza o valor) da função de erro em relação aos pesos da RNA para podermos ajustá-los nessa direção
  - O gradiente de uma função multivariada pode ser computado através de sua derivada parcial, pois a derivada de uma função computa a direção e intensidade de crescimento da função em um determinado ponto
    - Como desejamos o sentido de decrescimento da função de erro (minimização), basta multiplicarmos o resultado das derivadas parciais por -1 para termos o gradiente descendente
  - Quando temos uma função multivariada podemos computar a derivada de segunda ordem e formar a matriz hessiana do gradiente para computar o formato da curvatura da função
    - Esse cálculo custa muito mais poder computacional e dá origem aos métodos de segunda ordem de treinamento de RNAs
    - Os métodos de segunda ordem são mais precisos nos ajustes pois seguem uma direção bidimensional ao invés de direções unidimensionais mas o custo computacional extra necessário para esses ajustes nem sempre compensa

Gradiente Descendente – Passo 1/3

- Saída da rede (valor estimado/previsto ou $\hat{y}$)
  - $\hat{y} = f(W^T X + b)$
- Função de ativação (ReLu)
  - $f(z) = \begin{cases} z \; se \; z > 0 \\ 0 \; se \; z \leq 0 \end{cases}$
- Erro Quadrático Médio (EQM ou MSE em inglês)
  - $EQM = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{2} (y_n - \hat{y}_n)^2$

Gradiente Descendente – Passo 2/3

Derivada parcial em relação a w_i

$$\frac{\partial EQM}{\partial w_i} = \frac{\partial}{\partial w_i} EQM = \frac{\partial}{\partial w_i} \frac{1}{N} \sum_{n=1}^{N} \frac{1}{2} (y_n - \hat{y}_n)^2$$

$$\frac{1}{N} \sum_{i=n}^{N} \frac{\partial EQM}{\partial w_i} = \frac{\partial}{\partial w_i} \frac{1}{2} u^2$$

Para $\mathrm{n} = 1$ e w_0

$$\frac{\partial EQM}{\partial w_0} = \frac{\partial}{\partial w_0} \frac{1}{2} u^2 = \frac{1}{2} 2u.u'$$

$$= u.(y_1 - \hat{y}_1)' = u.(0 - 1.\hat{y}_1')$$

$$= -1.u.f(z)' = -u.(f'.z')$$

$$= -u.f'.(W^T X + b)' = -u.f'.(x_0 + 0 + \cdots + 0 + 0)$$

$$= -u.f'.x_0 = -(y_1 - \hat{y}_1).f'.x_0$$

Derivada parcial em relação a b

$$\frac{\partial EQM}{\partial b} = -u.(f'.z')$$

$$= -u.f'.(W^T X + b)' = -u.f'.(0 + \cdots + 0 + 1)$$

$$= -u.f' = -(y_1 - \hat{y}_1).f'$$

Substituições para usar regra da cadeia

$$u = y_n - \hat{y}_n$$

$$\hat{y} = f(W^T X + b)$$

$$z = W^T X + b$$

$$f(z) = \begin{cases} z \; se \; z > 0 \\ 0 \; se \; z \leq 0 \end{cases}$$

Gradiente Descendente – Passo 3/3

- Derivada parcial em relação a w_i
  - $\frac{\partial EQM}{\partial w_i} = -(y_n - \hat{y}_n).f'.x_i$
- Derivada parcial em relação a b
  - $\frac{\partial EQM}{\partial b} = -(y_1 - \hat{y}_1).f'$
- O gradiente descendente
  - $\nabla_{w_i} = -1.\frac{\partial EQM}{\partial w_i} = (y_n - \hat{y}_n).f'.x_i$
  - $\nabla_b = -1.\frac{\partial EQM}{\partial b} = (y_n - \hat{y}_n).f'$

Árvores de decisão (AD)

- Vantagens
  - Robusto e gera modelos facilmente interpretáveis (caixa branca)
  - Fundamentado em técnicas estatísticas e probabilísticas
- Desvantagens
  - Os modelos podem ficar extensos e serem pouco flexíveis (especialmente em problemas de regressão)
- Treinamento
  - Usa um conjunto de dados de treinamento rotulado (X, y) para aprender regras usando a entropia para computar o ganho de informação (escolhe o atributo que representa maior ganho de informação ou que gera os conjuntos mais puros a cada divisão da árvore)

Árvores de decisão

- A entropia mede a incerteza de uma variável aleatória e pode ser usada para definir o grau de incerteza de um conjunto

$$Entropia(S) = -\sum_i p_i \log_2 p_i$$

- O ganho de informação é a redução na entropia dado um atributo

$$Ganho(S, A) = Entropia(S) - \sum_a \frac{N_a}{N} Entropia(S_a)$$

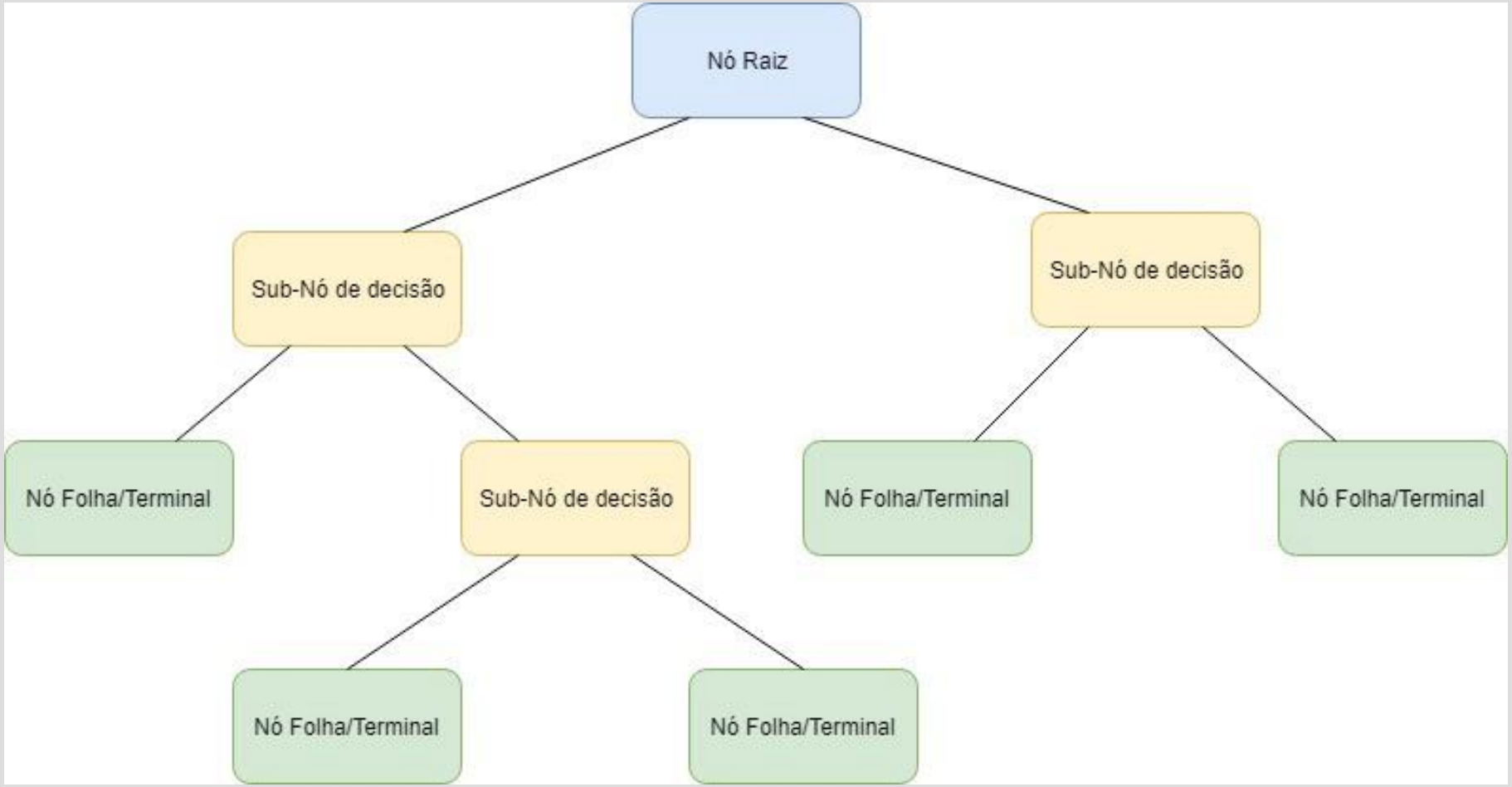
- A pureza de um conjunto pode ser medida com índice de Gini

$$Gini(S_a) = 1 - \sum_i {p_i}^2$$

$$Gini(S, A) = \sum_a \frac{N_a}{N} Gini(S_a)$$

Variável	Descrição
S	Conjunto de variáveis
S_a	Conjunto de variáveis com o atributo a
i	Identificador da classe (grupo)
p_i	Probabilidade de uma amostra do conjunto S ser da classe i
A	Atributo de uma classe
a	Identifica um valor (manifestação de um atributo A
N	Número de instâncias do conjunto S
N_a	Número de instâncias do conjunto S com o valor a

Protótipo de uma árvore de decisão



AD – Técnica da entropia e ganho

- Computa-se o ganho de cada atributo e escolhe-se o atributo com o maior ganho
  - O processo se repete a cada divisão da árvore de decisão

$$Entropia(S) = -\sum_i p_i \log_2 p_i = -\left(\frac{2}{6}\log_2\frac{2}{6} + \frac{4}{6}\log_2\frac{4}{6}\right) = 0.92$$

$$Entropia(Alto) = -\left(\frac{2}{3}\log_2\frac{2}{3} + \frac{1}{3}\log_2\frac{1}{3}\right) = 0.91$$

$$Entropia(Normal) = -\left(\frac{0}{1}\log_2\frac{0}{1} + \frac{1}{1}\log_2\frac{1}{1}\right) = 0$$

$$Entropia(Baixo) = -\left(\frac{0}{2}\log_2\frac{0}{2} + \frac{2}{2}\log_2\frac{2}{2}\right) = 0$$

$$Ganho(S, A1) = Entropia(S) - \sum_a \frac{N_a}{N} Entropia(S_a) = 0.92 - \left(\frac{3}{6}\times 0.91 + \frac{1}{6}\times 0 + \frac{2}{6}\times 0\right) = 0.46$$

A1	A2	Classe
Alto	Muito	1
Baixo	Pouco	2
Baixo	Muito	2
Normal	Muito	2
Alto	Pouco	1
Alto	Muito	2

AD – Técnica do índice de Gini

- Computa-se o índice Gini de cada atributo e escolhe-se o atributo com o menor índice (maior pureza)
  - O processo se repete a cada divisão da árvore de decisão

$$Gini(Alto) = 1 - \sum_i {p_i}^2 = 1 - \left(\frac{2}{3}\right)^2 - \left(\frac{1}{3}\right)^2 = 0{,}44$$

$$Gini(Normal) = 1 - \left(\frac{0}{1}\right)^2 - \left(\frac{1}{1}\right)^2 = 0$$

$$Gini(Baixo) = 1 - \left(\frac{0}{2}\right)^2 - \left(\frac{2}{2}\right)^2 = 0$$

$$Gini(S, A1) = \sum_a \frac{N_a}{N} Gini(S_a) = \frac{3}{6} \times 0{,}44 + \frac{1}{6} \times 0 + \frac{2}{6} \times 0 = 0{,}22$$

A1	A2	Classe
Alto	Muito	1
Baixo	Pouco	2
Baixo	Muito	2
Normal	Muito	2
Alto	Pouco	1
Alto	Muito	2

Exercício 5 – AD

- Construa 2 árvores de decisão, uma usando o método de seleção de entropia/ganho e a outra usando o índice de Gini, para o conjunto de entradas abaixo

Clima	Temperatura	Humidade	Vento	Futebol
Chuvoso	Frio	Alta	Intenso	Não
Ensolarado	Agradável	Baixa	Fraco	Sim
Nublado	Calor	Normal	Fraco	Sim
Nublado	Agradável	Normal	Fraco	Sim
Ensolarado	Calor	Normal	Fraco	Sim
Nublado	Frio	Normal	Intenso	Não
Chuvoso	Calor	Alta	Intenso	Sim
Chuvoso	Agradável	Alta	Intenso	Não
Ensolarado	Agradável	Baixa	Intenso	Sim
Nublado	Frio	Baixa	Fraco	Não

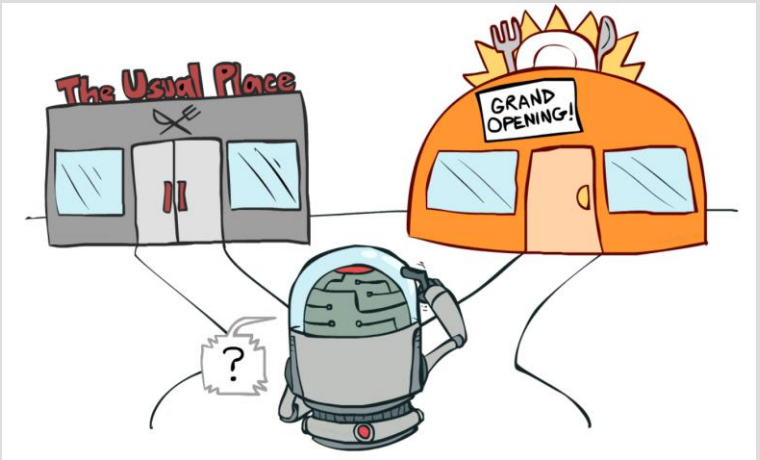
Aprendizado por reforço

Recebe recompensas e punições em ambientes simulados e ajusta seu comportamento de acordo

(RUSSELL; NORVIG, 2022. cap. 19 e 22)

Conceito

- O aprendizado por reforço (*Reinforcement Learning* – RL) foca no aprendizado através de técnicas baseadas em tentativa e erro.
- O treinamento de um agente, em RL, envolve a atualização de um modelo do ambiente, função de utilidade ou política (estratégia) através de recompensas/punições observadas ao interagir com um ambiente simulado de treino.
- Alterna repetidas vezes entre exploração (*exploration*): explora novas possibilidades – e explotação (*exploitation*): reforça e aplica conhecimento já adquirido.



Q-learning – resumo

- Vantagens
  - Algoritmo simples que não precisa de qualquer informação prévia sobre o domínio do problema e não cria um modelo do ambiente (livre de modelo)
  - Fundamentado no conceito de utilidade ou valor da ação
- Desvantagens
  - Pouco eficiente computacionalmente, em especial se as recompensas são muito espaçadas (distantes) no tempo
- Treinamento
  - Alterna entre exploração e explotação em um ambiente simulado repetidas vezes ao mesmo tempo que ajusta constantemente uma função Q (sem hífen ou sem qualidade) que mapeia as recompensas (ou penalidade) acumuladas durante o caminho para cada ação testada

Q-learning – funcionamento

- Aplica uma fórmula baseada em aprendizado temporal que ajusta a função de utilidade conforme as experiência adquiridas e a previsão/expectativa futura
  - $Q(estado, ação) = (1 - \alpha) \times Q(estado, ação) + \alpha \times (recompensa + \gamma \times maxQ(próximoEstado, ações))$
- A taxa de desconto determina a importância que daremos para recompensas futuras (estratégia de longo prazo) frente a uma estratégia mais imediatista
- Ao diminuir a taxa de aprendizado conforme o modelo ganha conhecimento e ao reduzir a taxa de desconto conforme o objetivo final esteja mais próximo torna-se o aprendizado do algoritmo mais eficiente

Variável	Descrição
Q	Função que mapeia o conjunto de estados e suas ações possíveis
α	Taxa de aprendizado, $0 < \alpha \leq 1$
γ	Taxa de desconto, $0 < \gamma \leq 1$
maxQ	Valor máximo da tabela Q em um determinado estado (melhor ação)

Q-learning – passos

- Inicializa-se o mapa (função Q) com zeros para todos os pares de ação/estado possíveis
- Exploração:
  - Selecionar uma ação disponível (aleatoriamente) no estado atual
  - Executar/simular a ação e armazenar a recompensa obtida atualizando o mapa Q
  - Repetir os passos anteriores no novo estado até que o agente atinja o objetivo ou um estado final
  - Inicializar o agente em um estado inicial e repetir todo o processo mais uma vez (até um limite pré-definido de simulações)
- Explotação:
  - Executa os mesmos passos anteriores com a diferença de que as ações são escolhidas a partir do conhecimento já obtido ao invés de aleatoriamente, ou seja, a ação escolhida em cada estado é a ação que maximiza a função Q
- Alterna-se entre as fases exploração e explotação até que o treinamento seja finalizado (agente dominou o problema). Inicialmente a fase de exploração deve ser maior que a de explotação e isso deve ir se invertendo conforme o agente adquire mais conhecimento

Q-learning – exemplo: problema do taxista

- Ambiente de simulação disponível na biblioteca gym (Taxi-v3) do Python
  - https://gymnasium.farama.org/
- O objetivo é transportar passageiros a partir dos pontos {R, G, B, Y} até outro ponto {R, G, B, Y} com o menor custo possível (menor caminho)
- Total de estados (25x5x4 = 500)
  - 25 (grade 5x5) posições para o carro
  - 4 posições para passageiros {R, G, B, Y} +1 para quando este está dentro do carro
  - 4 destinos possíveis {R, G, B, Y}
- Total de ações (6):
  - Sul, norte, leste, oeste, pegar, largar.
- Recompensas:
  - Ação de movimento: -1
  - Ação de pegar ou largar passageiro inválidas: -10
  - Ação de largar um passageiro no destino correto: +20

Bibliotecas em Python

Stable-Baselines3

https://stable-baselines3.readthedocs.io/en/master/

RL_Coach

https://intellabs.github.io/coach/

Tensorforce

https://tensorforce.readthedocs.io/en/latest/

Bibliografia

- BARBETTA, Pedro Alberto; REIS, Marcelo Menezes; BORNIA, Antonio Cezar. **Estatística para cursos de engenharia e informática**. 3. ed. São Paulo: Atlas, 2010. 416 p. ISBN 9788522459940.
- RUSSELL, Stuart J.; NORVIG, Peter. **Inteligência Artificial**: uma abordagem moderna. 4. ed. Rio de Janeiro: GEN LTC, 2022. 1016 p. ISBN 9788595158870.
- SANDERSON, Grant. **Animated Math**: Neural Networks. Estados Unidos da America, 2017. Disponível em: https://www.3blue1brown.com/topics/neural-networks. Acesso em: 26 mar. 2023.
- SANDERSON, Grant. **Animated Math**: Probability. Estados Unidos da America, 2023. Disponível em: https://www.3blue1brown.com/topics/probability. Acesso em: 26 mar. 2023.
- TALBI, El-Ghazali. **Metaheuristics:** from design to implementation. 1. ed. Wiley, 2009. 624 p.