1 - Breve Introdução à Inteligência Artificial

Carlos Ramos

csr@isep.ipp.pt

Unidade Curricular: Algoritmia Avançada

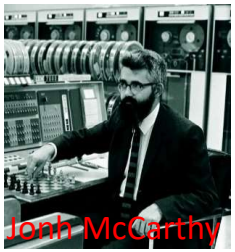
Curso: Licenciatura em Engenharia Informática (ISEP)

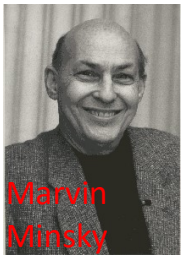
IA – afinal não é tão nova!

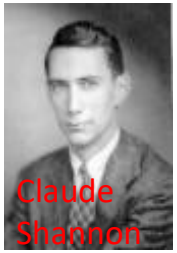
- O termo IA foi introduzido por John McCarthy em **1956** no Dartmouth Summer Research Project on Artificial Intelligence (EUA)

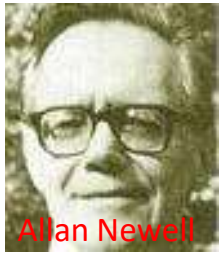
- Participantes: Marvin Minsky; John McCarthy; Claude Shannon; Trenchard More; Nat Rochester; Oliver Selfridge; Julian Bigelow; W. Ross Ashby; W.S. McCulloch; Abraham Robinson, Tom Etter; John Nash; David Sayre; Arthur Samuel; Kenneth R. Shoulders; Alex Bernstein; Herbert Simon; Allen Newell

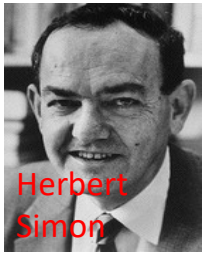
- O trabalho de Warren McCulloch e Walter Pitts, de **1943**, sobre **neurónios artificias** é o primeiro associado à IA

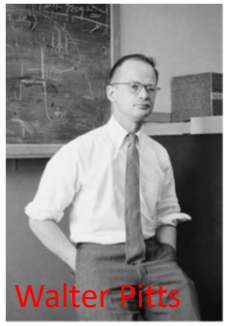


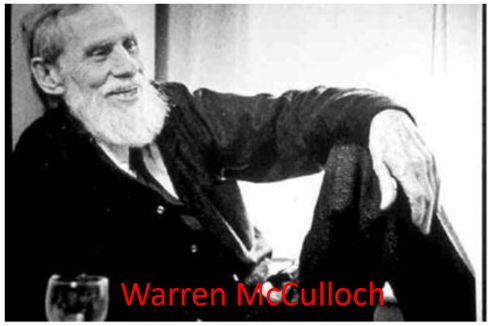






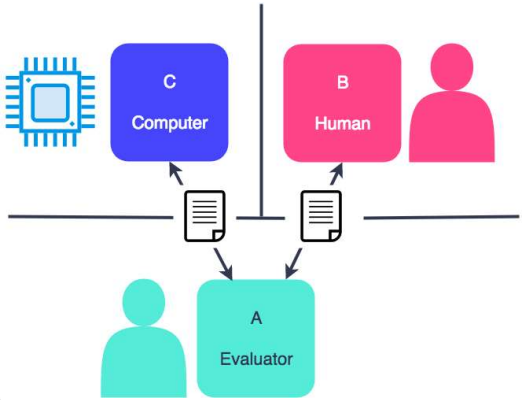






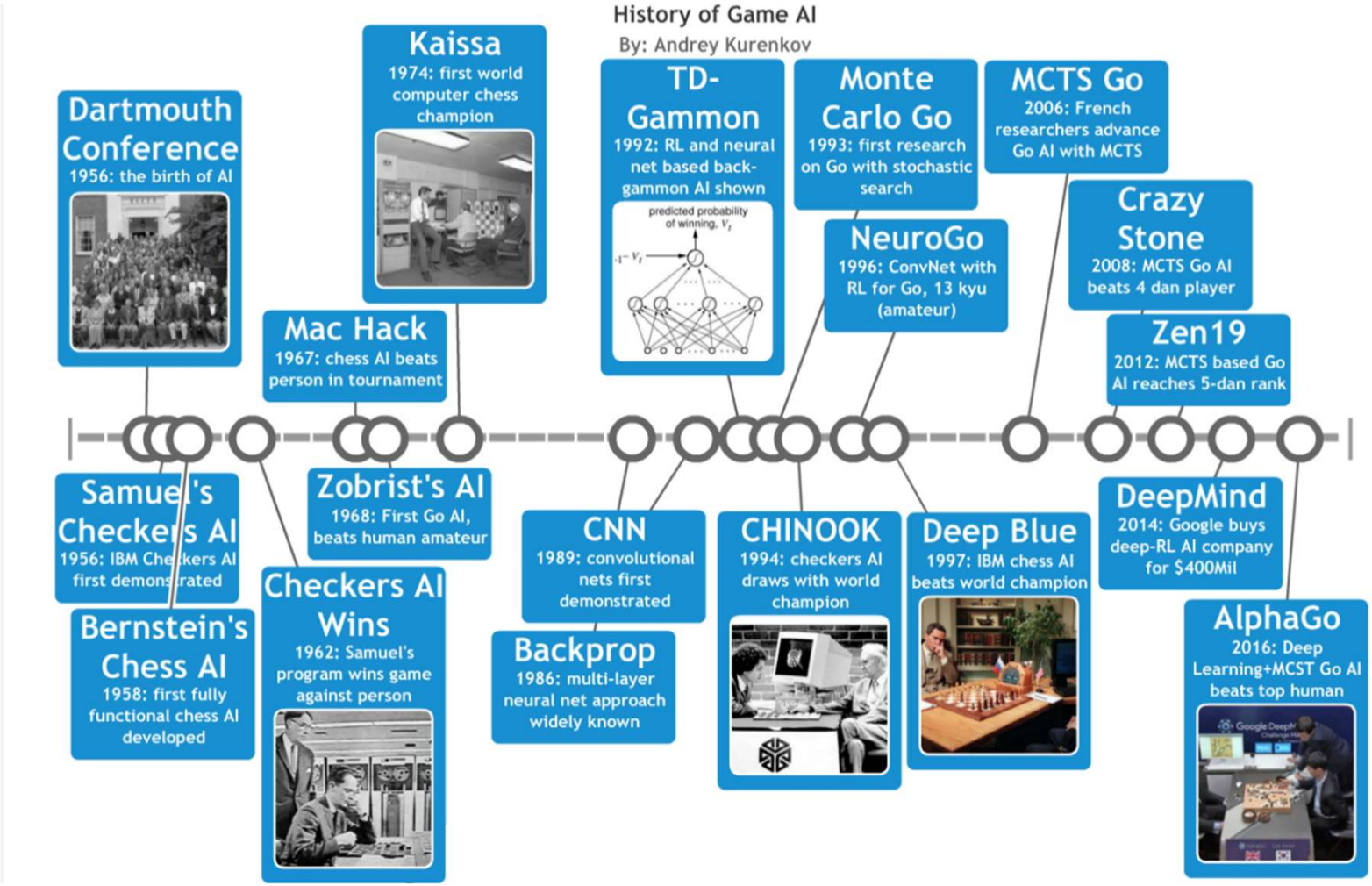
Teste de Turing

Alan Turing [1912-1954], génio da Computação
Criou a Máquina de Turing
Descriptou o Código Enigma usado pelos Nazis durante a 2ª Gerra Mundial (retratado no filme "O Jogo da Imitação")



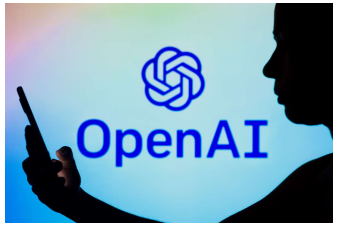
O Teste de Turing é fácil de enunciar mas difícil de resolver (ainda não o foi)
Há um ser humano que através de um teclado coloca questões cujas respostas são visualizadas num ecran (A), podendo vir de 2 lados: de um sistema computacional (C) ou de um ser humano (B). Se ao fim de algum tempo o ser humano (A) conseguir identificar pelas respostas de que lado está o ser humano que responde (B) e de que lado está o Sistema Computacional (C) então dizemos que o sistema computacional C não passou no Teste de Turing

IA - há muitos anos a tentar superar o ser humano!



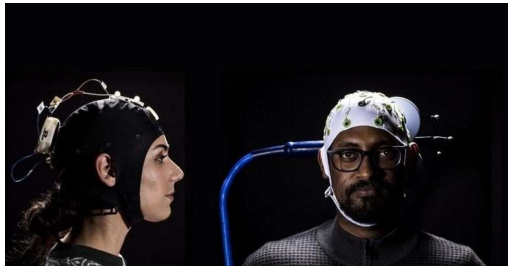
Other Challenges:
- Poker
- StarCraft II
- Dota 2
- Hanabi

IA – ajudando ou ultrapassando o ser humano ?



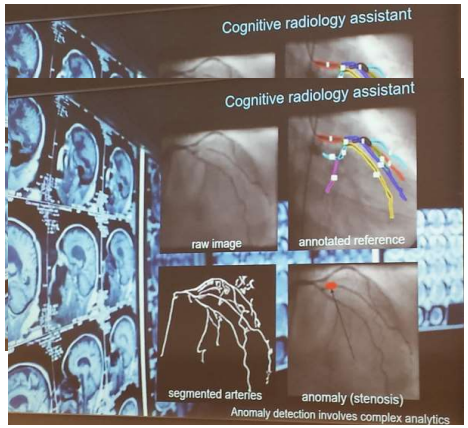
Com base em Large Language Models e Transformers o ChatGPT da OpenAI é lançado em finais de 2022 causa uma verdadeira revolução – a Generative AI é a nova realidade

Prevê-se que já em 2024 o melhor tradutor será automático (IA) e que em 2049 sistemas de IA escreverão os melhores contos e novelas



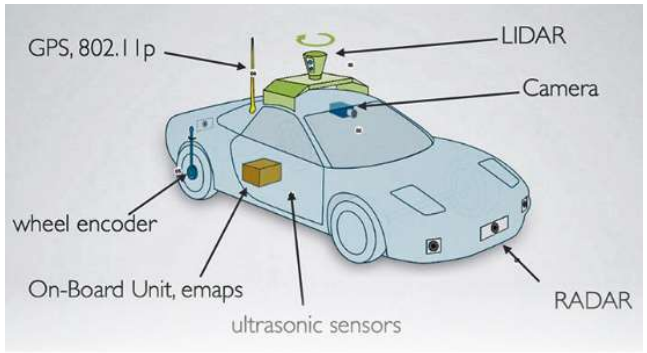
Convertendo Ondas Cerebrais em texto

Alphastar da DeepMind Atinge o Grandmaster level no Starcraft II



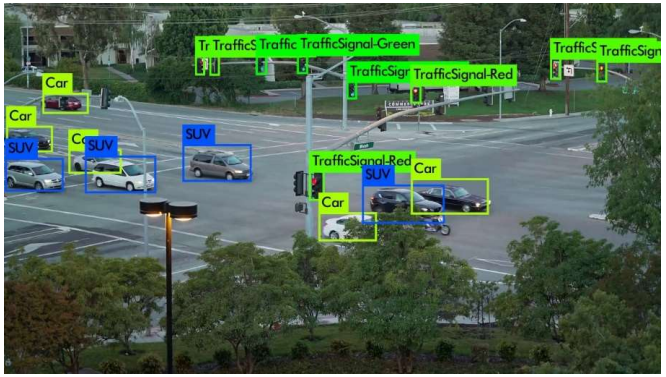
IBM Cognitive Radiology Assistant

SoundSpaces e Semantic MapNet Sistemas da Facebook de Embodied AI

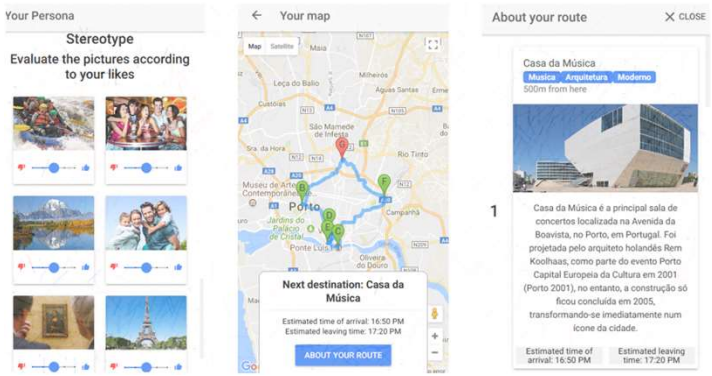


Veículos Autónomos

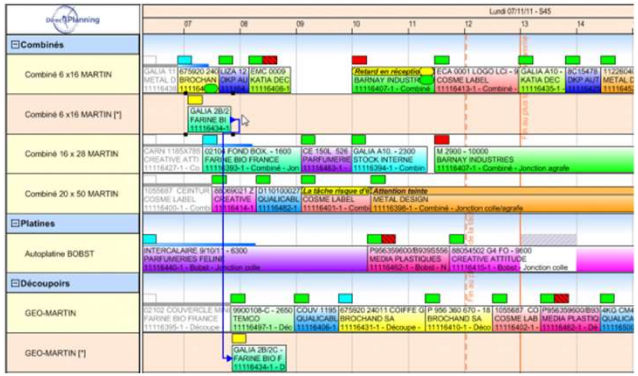
A IA está em todo o lado!



Smart Cities

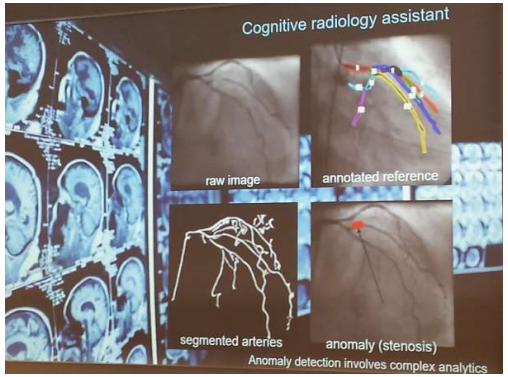


Tourism Recommender Systems



Industria 4.0

Energia



Saúde

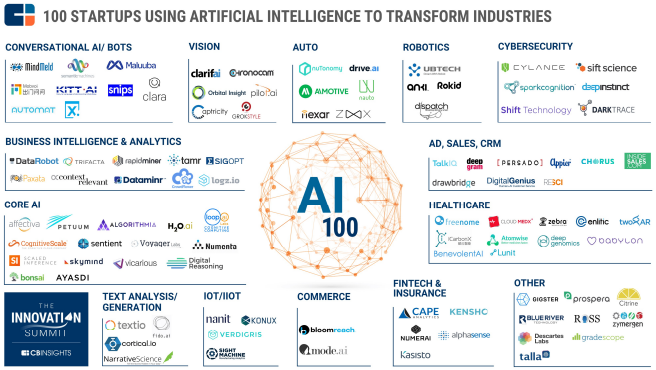
Artes

IA – A tecnologia que valoriza o petróleo do século XXI

Mais de 50 países lançaram Estratégias para a IA

Profissões ligadas à IA
Muito procuradas e bem remuneradas

Emirados Árabes Unidos nomeiam
Omar Bin Sultan Al Olama
Ministro de Estado para a IA



Empresas de ICT colocam IA como 1ª prioridade, grandes empresas apostam na IA, start-ups e spinoffs de IA generalizam-se

MEIA
MESTRADO ENGENHARIA INTELIGÊNCIA ARTIFICIAL

ISEP lança o único Mestrado em Portugal com Inteligência Artificial no nome
Um dos 2 únicos do ISEP a encher logo na 1ª fase

LASI – Laboratório Associado em Inteligência Artificial

LASI Laboratório Associado Sistemas Inteligentes

GECAD unidade de I&D do ISEP da área da IA classificada com Excelente pela FCT

gecad Intelligence & Excellence

PPLWARE.SAPO.PT

ChatGPT passou confortavelmente em exames destinados a futuros médicos - Pplware

CNNPORTUGAL.IOL.PT

Juiz colombiano perguntou ao ChatGPT que decisão deveria tomar - e fez o que a Inteligência Artificial disse

WATCHER.GURU

OpenAI is Training ChatGPT to Eventually Replace Software Engineers

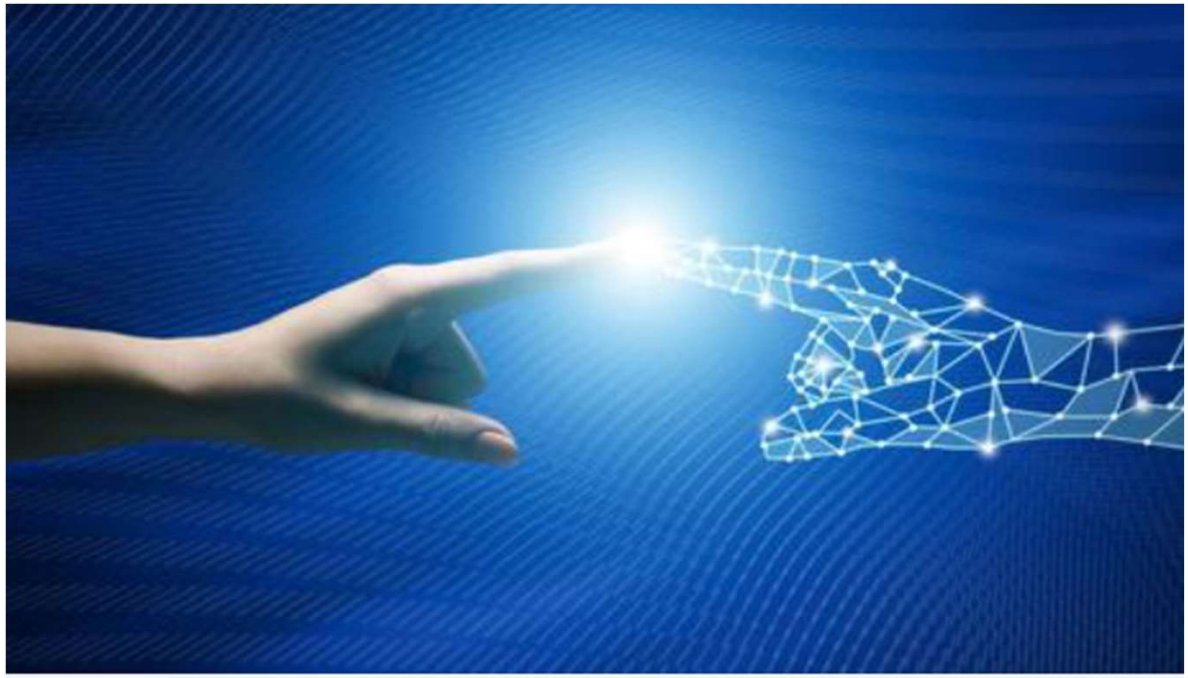
OpenAI has hired hundreds of international contractors to help train ChatGPT to ...

AI / MACHINE LEARNING / TECH LIFE

How Conversational Programming Will Democratize Computing

Will non-developers be able to build software using ChatGPT and large language models? Yes, if they can master "conversational programming".

Apr 22nd, 2023 8:00am by David Eastman



SICNOTICIAS.PT

Há uma profissão a dar que falar, salário pode chegar aos 300 mil euros

THEGUARDIAN.COM

Photographer admits prize-winning image was AI-generated

German artist Boris Eldagsen says entry to Sony world photography awards was ...

Filme curto criado em 24 horas por conjunto de ferramentas de IA
Geração de Imagem: **Dall-E**
Mapa de Profundidade das imagens geradas: **3D Photo Depth**
Geração de Movimento: **Depthy**
Geração de Texto: **ChatGPT**
Geração de Voz: **FakeYou**
Geração de Música: **Soundraw**

Intervenção humana: prompts usados pelas ferramentas

https://www.youtube.com/watch?v=8X706nnsUBI

https://comunidadeculturaearte.com/a-inteligencia-artificial-e-a-arte-combater-o-medo-com-o-medo/

PT.EURONEWS.COM

Inteligência Artificial lê a mente e traduz pensamentos em texto

Sistema de IA consegue traduzir os pensamentos das pessoas em texto, analisan

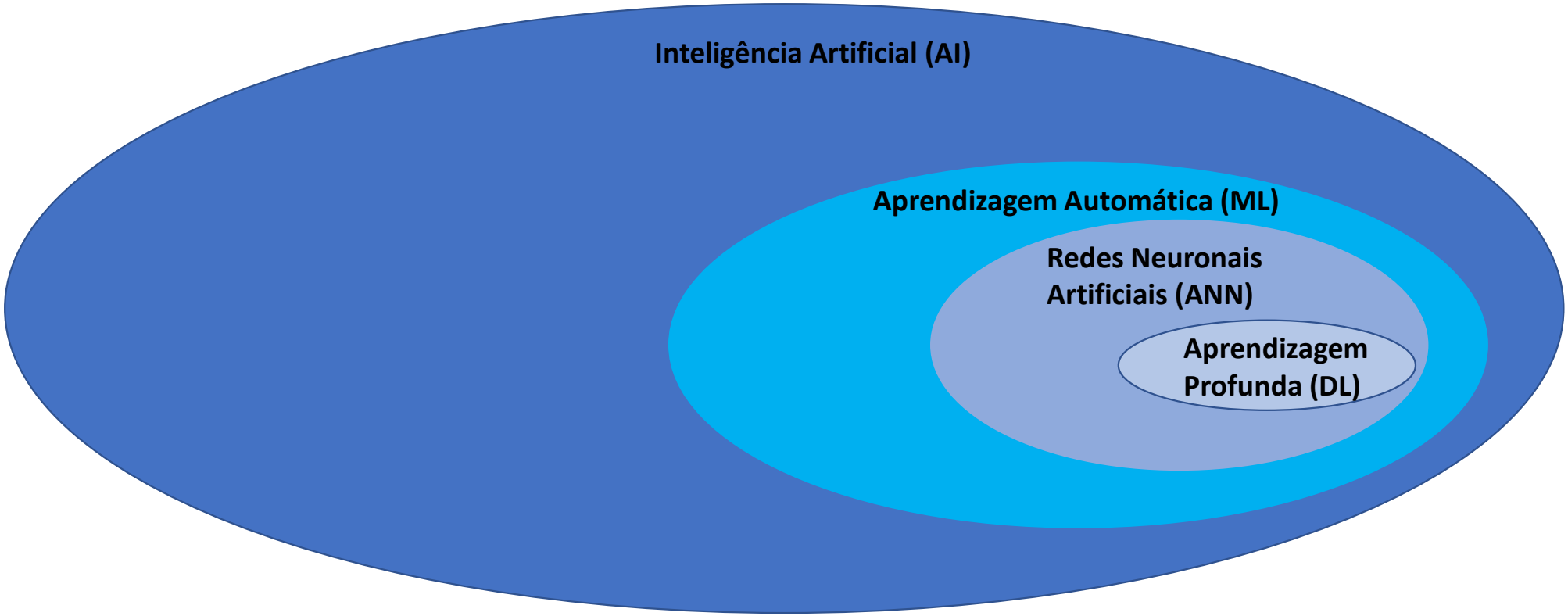
Vamos propor 2 definições para a IA

A **Inteligência Artificial** (**IA**) é a ciência que estuda os fundamentos e ajuda a desenvolver sistemas em domínios diversos com o objetivo de aproximar ou mesmo superar as capacidades de inteligência do ser humano, colaborando também com este na resolução de problemas complexos [Ramos-2020].

Os **Sistemas de IA** têm capacidades de interpretação do meio envolvente, através de sensores e de aquisição de dados, informação e conhecimento, são capazes de modelar e analisar problemas complexos, elaborando raciocínio e aprendizagem, de uma forma automática, propondo atuações ou mesmo atuando diretamente sobre o meio envolvente. São capazes de interagir com outros sistemas no âmbito de sistemas multiagente e dialogarem de um modo inteligente com o ser humano. Devem ainda lidar com falhas nos dados e informação, situações não previstas e adaptarem-se ao contexto. Devem ser sistemas confiáveis e benevolentes para o ser humano e sociedade, seguindo princípios morais e éticos [Ramos-2020].

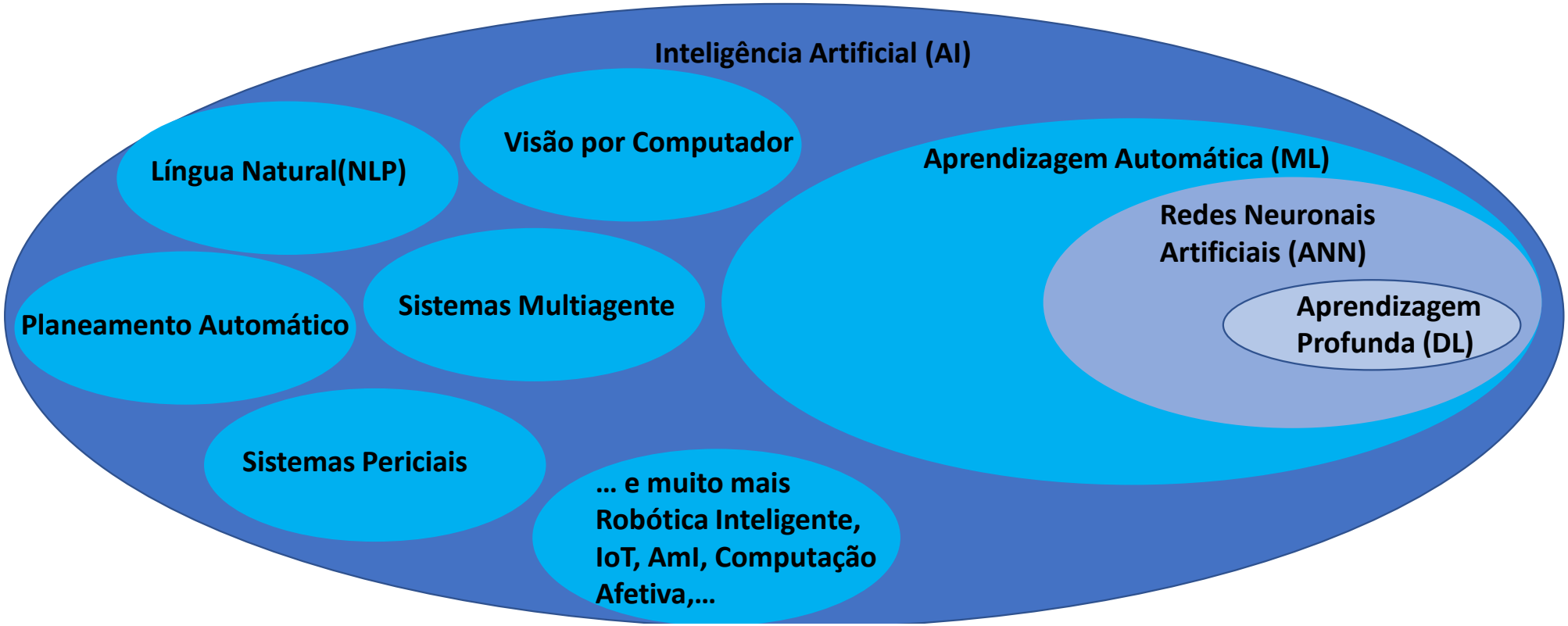
Esclarecendo conceitos

Hoje em dia surgem muitas notícias sobre Inteligência Artificial, Machine Learning (ML), Redes Neuronais Artificiais (ANN) e Deep Learning (DL), mas convém estarmos cientes que estamos a falar de níveis diferentes e incorporados



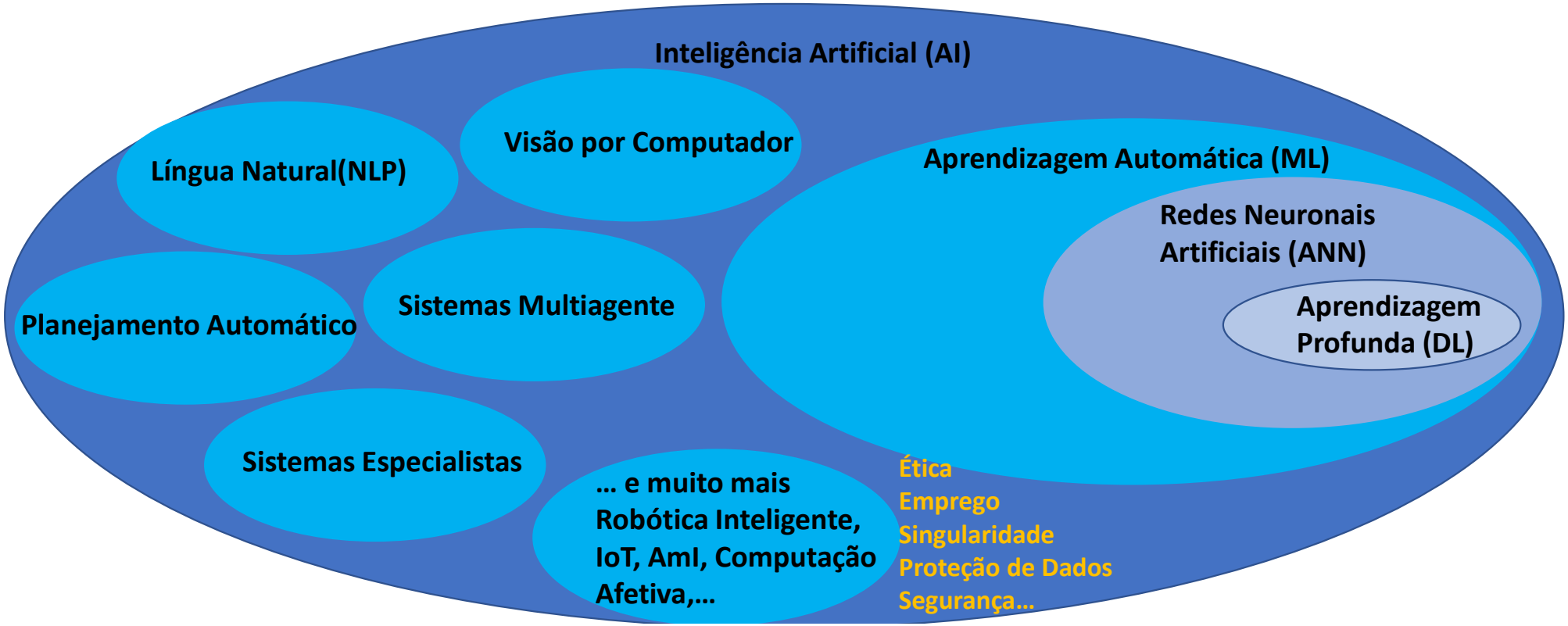
Esclarecendo conceitos

E a Inteligência Artificial não é apenas Machine Learning, apesar do ML ser hoje em dia usado em quase todas as áreas da IA



Esclarecendo conceitos

E novas preocupações aparecem



O ser humano (simplificado)

processa
interpreta
compreende
aprende
modela
raciocina
analisa
resolve
planeia
reage
propõe
argumenta
decide
justifica
cria
emociona-se
segue moral e ética
…

O agente inteligente (simplificado)

processa (processamento de sinais - som, imagem,…)
interpreta (reconhecimento de palavras, objetos)
compreende (língua natural, visão por computador)
aprende (machine learning)
modela (sistema baseado em conhecimento)
raciocina (raciocínio automático)
analisa (teoria de jogos)
resolve (resolução automática de problemas)
planeia (planeamento automático)
reage (abordagem reativa, robots inteligentes)
propõe (agentes inteligentes)
argumenta (argumentação automática)
decide (tomada de decisão)
justifica (XAI - eXplainable AI)
cria (computação criativa, IA generativa)
emociona-se (computação afetiva)
segue moral e ética (computação moral e ética)
…

ouve (adquire sinal áudio por um microfone)

vê (adquire imagem por câmera digital)

fala (emite sinal áudio por altifalante)

e adquire outros dados e informação por sensores que o ser humano não consegue (infravermelhos, ultrasons, laser, BD, Sites, Cloud, etc)

sente por tacto (usando sensor de toque, força, torque)

atua (através de um robot, autómato, agente, site, …)

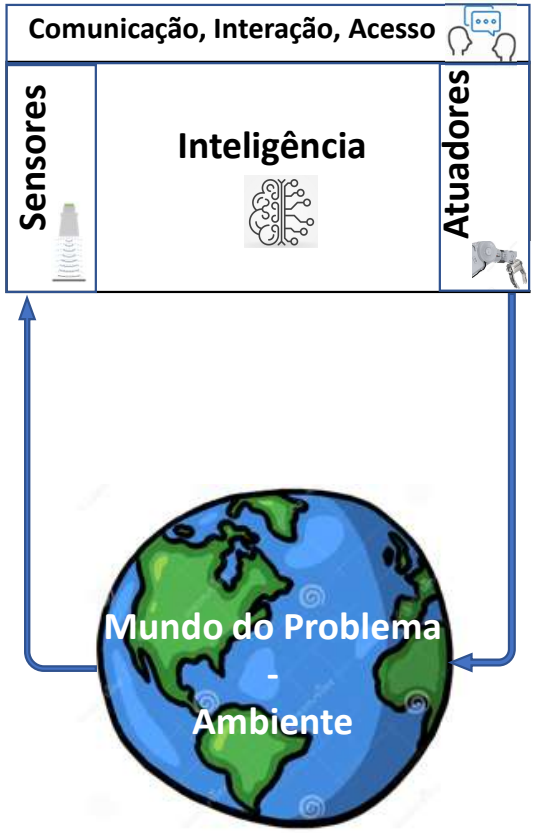
A visão do MEIA

MEIA – Duas metades: o Eu (ME) e a IA

Não vemos a IA como algo que vai substituir o ser humano, mas sim como algo que vai proceder à extensão das capacidades do ser humano em termos cognitivos e no suporte às suas decisões e ações. Assim como podemos usar óculos, binóculos, telescópios, microscópios, imagens térmicas ou por ultrassonografia para ver melhor a níveis diferentes, também podemos usar a IA para obter mais conhecimento, tomar melhores decisões e atuar de uma forma melhor e mais segura no mundo. O profissional munido com este Mestrado terá capacidade para idealizar, especificar, desenvolver, testar, gerir, manter e fazer evoluir sistemas com essas caraterísticas

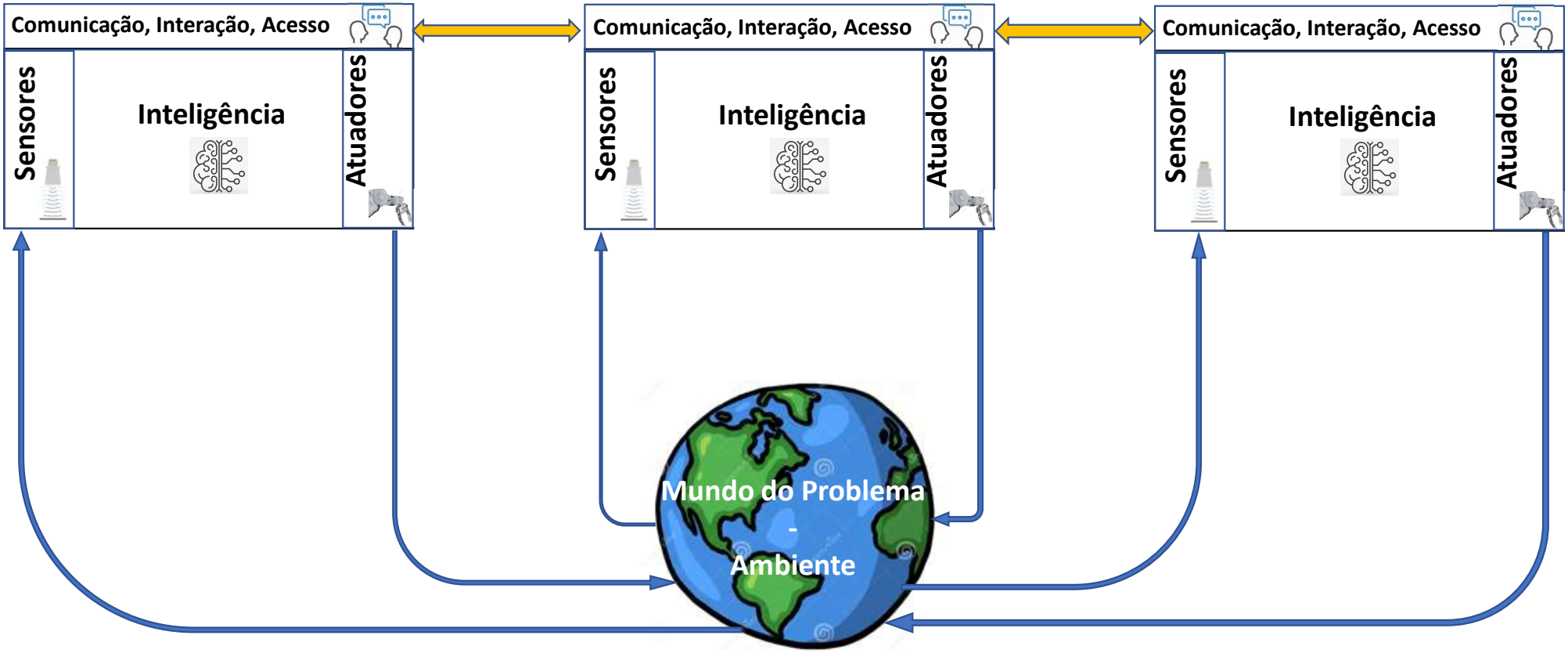
Agente Inteligente

Um Agente Inteligente realiza continuamente 3 funções: perceção das condições dinâmicas do ambiente; ações para afetar as condições do ambiente; raciocínio para interpretar o resultado da perceção, resolver problemas, inferir e determinar as ações a realizar [Hayes-Roth-1995]



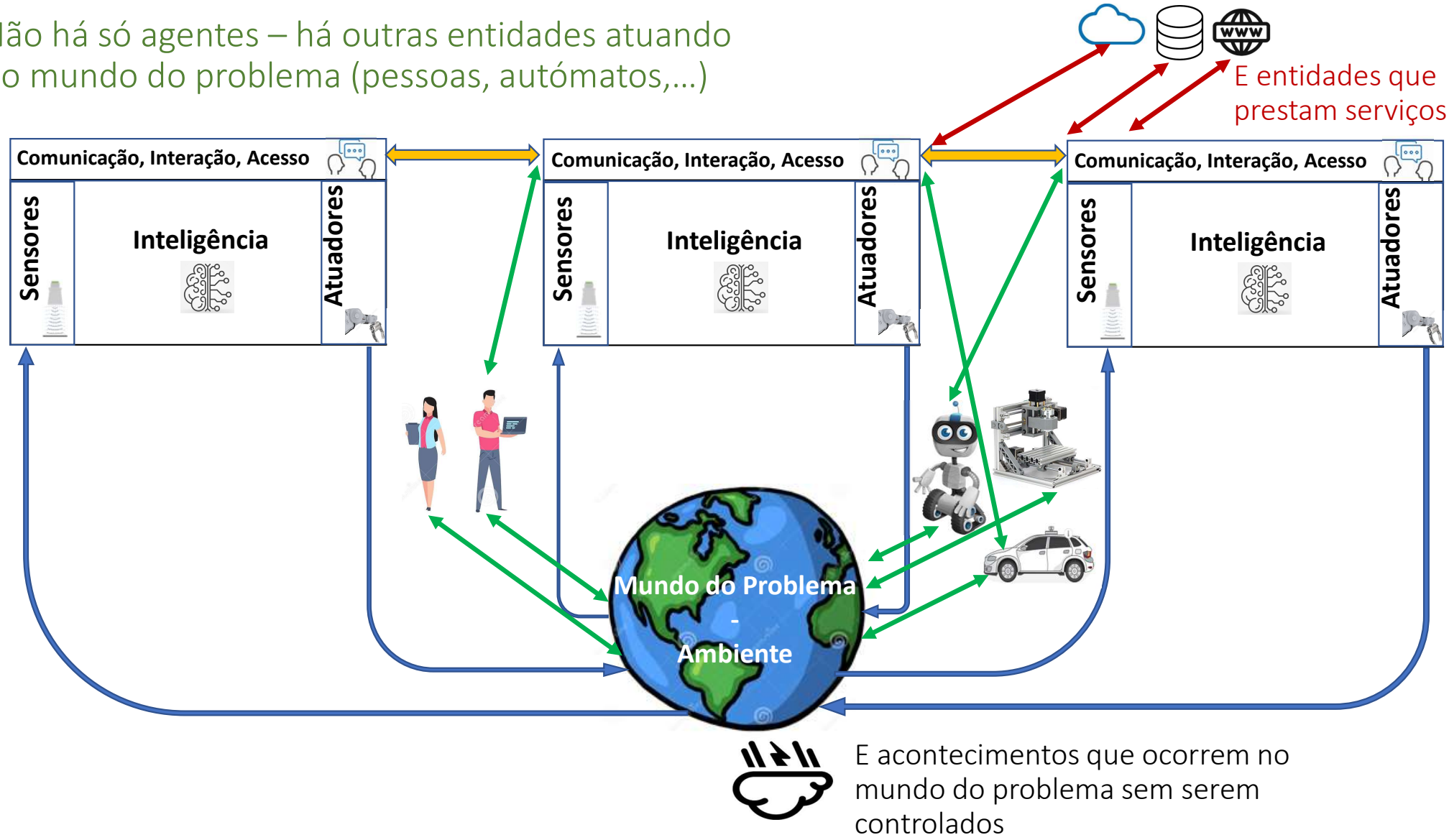
- Sensores: termómetro, sensor de posição, sensor de imagem, sistema laser. Podem ser passivos ou ativos
- Atuadores: robot, máquina CNC, autómato
- Inteligência: Inferência, Raciocínio, Planeamento, Otimização, Compreensão, Aprendizagem, etc
- Comunicação, Interação e Acesso: com outros agentes, humanos, autómatos, bases de dados,web, serviços

Mas o Agente não existe isoladamente – há outros agentes



Sistema Multi-Agente: vários agentes cooperam/competem na resolução de problemas e/ou partilha de recursos

Não há só agentes – há outras entidades atuando no mundo do problema (pessoas, autómatos,...)



O que uma entidade inteligente deve fazer no seu ambiente?

- Interpretar a situação do ambiente
- Comunicar com outras entidades relacionadas com o ambiente
- Representar a informação e conhecimento relacionado com o ambiente
- Modelar e representar as entidades envolvidas no ambiente
- Planear as decisões e ações a tomar
- Aprender com o ambiente e aspetos associados
- Interagir com o ser humano
- Atuar no ambiente

Tipos de Informação, Mundos e Abordagens

Garry Kasparov X Deep Blue (1997)
Informação Completa e Correta
Mundo Estruturado
Abordagem Deliberativa

Veículos Autónomos
Informação Incompleta, Incorreta e Incerta
Mundo Não Estruturado
Abordagem Reativa (mas também com partes deliberativas)

Adaptação ao utilizador

HOJE

Chegamos a casa
Entramos numa divisão
A luz acende automaticamente

Apagamos a luz porque não queremos a luz acesa por uns momentos

AMANHÃ

Chegamos a casa
Entramos numa divisão
A luz acende automaticamente

Apagamos a luz porque não queremos a luz acesa por uns momentos

DEPOIS DE AMANHÃ

Chegamos a casa
Entramos numa divisão
A luz acende automaticamente

Apagamos a luz porque não queremos a luz acesa por uns momentos

Será isto um sistema de iluminação inteligente?

Importa aprender com a interação do utilizador

Compreender a semântica do espaço físico

A habitante da casa tenta aumentar o brilho da televisão, que atinge o máximo
Ela continua a pressionar o botão para aumentar o brilho
O estore fecha reduzindo a iluminação externa e dando a sensação de aumento do brilho

2 objetos (TV e estore) coordenaram as suas atuações porque compreendem a física do espaço onde existem

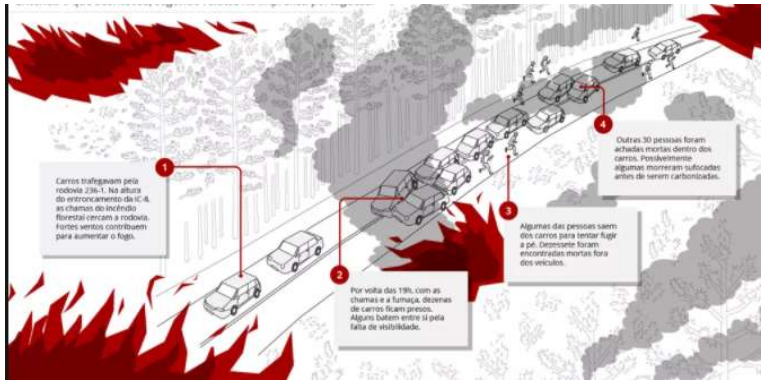
Exemplo baseado no projeto EMBASSI

Adaptação ao Contexto

Estado do tempo

Ajuntamentos de pessoas

Falhas de Energia



Situações de Catástrofe

2 – Lógica

Carlos Ramos
csr@isep.ipp.pt

Unidade Curricular: Algoritmia Avançada
Aulas Teóricas
Curso: Licenciatura em Engenharia Informática (ISEP)

Lógica

- A Lógica pode ser vista como uma forma de representar conhecimento. Por exemplo, na compreensão de texto em Língua Natural é usual transformar frases em representações em lógica
- Em Inteligência Artificial as Lógicas mais usuais são a Lógica Proposicional e a Lógica de Predicados (ou Cálculo de Predicados). Contudo, há outros tipos de Lógica como as Lógicas Modais e as Lógicas Temporais
- O PROLOG é baseado na Lógica de Predicados
- Os símbolos representam conhecimento e os operadores são aplicados aos símbolos para dar origem a raciocínios lógicos

Operador	Símbolos
Conjunção (AND)	$\wedge$, &, $\cap$, .
Disjunção (OR)	$\vee$, $\cup$, +
Negação (NOT)	$\neg$, $\sim$
Implicação	$\rightarrow$, $\supset$
Equivalência	$\equiv$

Lógica Proposicional

- A Lógica Proposicional usa proposições para a representação do conhecimento e raciocínio sobre esse mesmo conhecimento. Por exemplo, poderemos representar através de X a proposição que indica que a bateria está avariada e através de Y a proposição que indica que o carro não arranca:

X = a bateria está avariada

Y = o carro não arranca

A	**C = ¬ A**
F	V
V	F

A	**B**	**A ∧ B**	**A ∨ B**	**A ≡ B**	**A → B**
F	F	F	F	V	V
F	V	F	V	F	V
V	F	F	V	F	F
V	V	V	V	V	V

Propriedades da Lógica Proposicional

Nome da Propriedade	**Equivalência**
Idempotência	$A \rightarrow B \equiv \neg A \vee B$ $A \wedge \neg A \equiv F$ $A \vee \neg A \equiv V$
Propriedade Comutativa	$A \wedge B \equiv B \wedge A$ $A \vee B \equiv B \vee A$
Propriedade Distributiva	$A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$ $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$
Propriedade Associativa	$A \wedge (B \wedge C) \equiv (A \wedge B) \wedge C$ $A \vee (B \vee C) \equiv (A \vee B) \vee C$
Absorção	$A \vee (A \wedge B) \equiv A$ $A \wedge (A \vee B) \equiv A$
DeMorgan	$\neg (A \wedge B) \equiv \neg A \vee \neg B$ $\neg (A \vee B) \equiv \neg A \wedge \neg B$

Lógica de Predicados

- A Lógica de Predicados, baseada no trabalho de Gottfried Frege, vai operar com predicados e argumentos. As frases estabelecem relações entre objetos (pessoas, entidades físicas, conceitos).
- Os predicados têm a ver com as relações enquanto que os argumentos têm a ver com os objetos. Por exemplo, a frase "Portugal é um país europeu" pode ser representada pelo seguinte predicado:

  fica(portugal, europa).
- Na Lógica de Predicados podemos incluir variáveis, representadas por maiúsculas, tal como no PROLOG.
- Adicionalmente, podemos considerar os quantificadores universal (∀ - qualquer) e existencial (∃ - existe)

Frase	**Representação possível em Lógica de Predicados**
Todas as crianças gostam de gelados	∀ C, [criança(C) → gosta(C, gelado)]
Há um oceano que banha Portugal	∃ O, [oceano(O) → banha(O,portugal)]
Algumas aves migram	∃ A, [ave(A) → migra(A)]

Unificação e Princípio da Resolução

- Na Lógica de Predicados a *Unificação* é mais complexa do que na Lógica Proposicional, devido ao facto de termos variáveis
- É então necessário um mecanismo de identificação (matching) ou de unificação que irá tentar instanciar possíveis valores para as variáveis com base nos axiomas existentes
- A *Prova Automática de Teoremas* baseia-se no *Princípio da Resolução*. Segundo esse princípio para provarmos A vamos transformar os axiomas garantidamente verdadeiros na forma clausal, usando os princípios da idempotência, de seguida juntamos ¬A, se por aplicação da regra de resolução chegarmos a uma situação de falha na prova concluímos que ¬A é falso e portanto A é verdadeiro (Prova por Refutação)

Um exemplo

Considere o seguinte texto:

Os gatos gostam de peixe

Os gatos comem tudo do que gostam

teco é um gato

- Exprima as frases em Lógica de Predicados de 1ª ordem.
- Usando a regra de resolução prove por refutação, com base no conhecimento anterior que "*o teco come peixe*"

Resolução

Os gatos gostam de peixe

$\forall_G$ [gato(G) $\rightarrow$ gosta (G, peixe)]

Os gatos comem tudo do que gostam

$\forall_G$ [gosta (G, C) $\rightarrow$ come (G, C)]

teco é um gato

gato (teco)

Provar $\rightarrow$ come (teco, peixe)

Nota: Usamos as maiúsculas G e C por analogia com o PROLOG

Resolução

Provar $\rightarrow$ come (teco, peixe)

Resolução

Estado-0

1. $\neg$ gato(G) $\vee$ gosta (G, peixe)
2. $\neg$ gosta (G, C) $\vee$ come (G, C)
3. gato (teco)
4. $\neg$come (teco, peixe)

Resolução

Provar $\rightarrow$ come (teco, peixe)

Resolução

$\neg$come (teco, peixe) $\neg$ gosta (G, C) $\vee$ come (G, C)

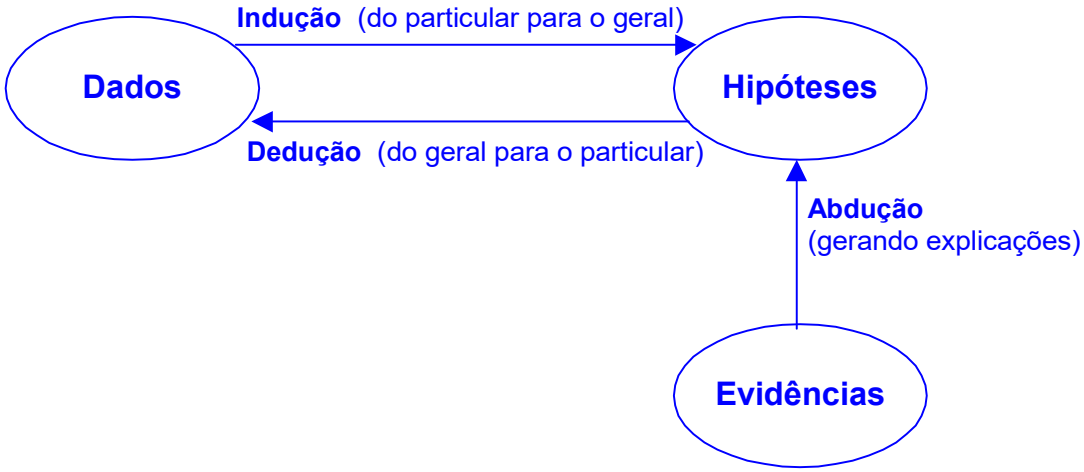
$\neg$ gosta (teco, peixe) $\neg$ gato(G) $\vee$ gosta (G, peixe)

$\neg$ gato (teco) gato (teco)

$\square$

Raciocínio na Lógica de Predicados

- Há vários modos de efectuar raciocínio automático por inferência na Lógica de Predicados. Basicamente há três mecanismos de raciocínio: Dedução, Abdução e Indução



Modus Ponens

•O "modus ponens" é a base do raciocínio baseado em regras. O "modus ponens" pode ser descrito do seguinte modo:

se a declaração p é verdadeira e (p →q) também é verdadeira

então podemos inferir que q também é verdadeira

Vejamos um exemplo do uso do "modus ponens". Para isso vamos considerar a seguinte observação: "se alguém está a nadar então está molhado". Em lógica de predicados esta realidade poderia ser traduzida do seguinte modo:

$\forall$ X, [nada(X) → molhado(X)]

Neste caso, e fazendo a analogia com p e q, vamos considerar que p corresponde a nada(X) e q corresponde a molhado(X). Se soubermos que X está a nadar e sabendo que quem nada se molha então poderemos, pelo "modus ponens", concluir que X está molhado.

Modus Tollens

•Outra regra importante é o "modus tollens". O "modus tollens" pode ser descrito do seguinte modo:

se (p →q) é verdadeira e a declaração q é falsa

então podemos inferir que p é falsa

Usando o mesmo exemplo que se considerou anteriormente, se considerarmos a regra enunciada e concluirmos que X não está molhado então podemos inferir que X não está a nadar.

Raciocínio Dedutivo

•A Dedução é definida como o modo de raciocinar através do qual se parte de um princípio conhecido em direcção a um princípio desconhecido, do geral para o específico, de uma premissa para uma conclusão lógica

Vejamos um exemplo de raciocínio dedutivo traduzido através de uma regra:

SE temos um objecto A maior que outro objecto B

E

o objecto B é maior que o objecto C

ENTÃO

o objecto A também é maior que o objecto C

Esse conhecimento poderia ser traduzido do seguinte modo em Lógica de Predicados:

$\forall A, \forall B, \forall C, [maior(A,B) \wedge maior(B,C) \rightarrow maior(A,C)]$

•A dedução é apontada como o modo correcto de raciocinar, garantindo que se usarmos premissas verdadeiras seremos capazes de chegar a conclusões também verdadeiras.

Raciocínio Abdutivo

•A abdução é um tipo de raciocínio usado geralmente na produção de explicações. Ao contrário da dedução, a abdução não garante que se chegue a conclusões verdadeiras.

•Embora não a compreendamos plenamente, usamos a abdução com frequência. Problemas de diagnóstico são casos usuais onde o raciocínio abdutivo é usado com frequência.

•De um modo simplificado podemos admitir que se sabemos que A →B e que B é verdadeiro então A também será possivelmente verdadeiro. Tal conclusão será sustentada no histórico anterior ou apenas em senso comum.

Para exemplificarmos vamos considerar o conhecimento segundo o qual concluímos que se X nada então X deverá ficar molhado:

$\forall$ X, [nada(X) → molhado(X)].

•Com base nesse conhecimento e se nos aparecer alguém molhado vamos admitir que esse alguém esteve a nadar.

•O senso comum afectará o nosso raciocínio abdutivo, por exemplo, se estiver a chover intensamente atribuiremos a possibilidade de uma pessoa estar molhada ao facto de estar a chover, mas se existir uma piscina nas imediações e se uma pessoa aparece molhada e em fato de banho então atribuiremos uma maior possibilidade ao facto dessa pessoa estar a nadar.

Raciocínio Indutivo

•A Indução é definida como o modo de raciocinar de factos particulares na direcção de uma conclusão geral.

Vejamos um exemplo de raciocínio indutivo:

P(a) é verdadeiro

P(b) é verdadeiro

Então, por indução, concluimos que $\forall$ X, P(X) é verdadeiro.

•O raciocínio indutivo é usado em processos de descoberta de conhecimento

Por exemplo dado um conjunto de consumidores de um determinado tipo de produto, caracterizados pela idade, sexo, estado civil e salário podemos, por indução, concluir que o comprador do produto em causa corresponde aos homens entre 20 e 28 anos, solteiros e com vencimento entre 1000 e 1500 Euros. Isso não garante que qualquer indivíduo que cumpra essas características vá comprar o produto, nem que alguém que não cumpra as características apontadas não possa comprar o produto, apenas dá ideia do cliente tipo, o que pode ser importante para uma campanha de publicidade ou de marketing dirigido.

Programação em Lógica

- O paradigma da Programação em Lógica baseia-se na Lógica de Predicados
- Um programa consiste na descrição de um problema em termos de axiomas lógicos (ao invés da sequência de passos de um programa imperativo)
- Os predicados lógicos especificam os relacionamentos possíveis entre objectos
- A execução de um programa corresponde à dedução controlada de consequentes com base numa teoria lógica, normalmente expressa como um conjunto de cláusulas de Horn (que suportam apenas uma conclusão, $a \wedge b \wedge c \rightarrow d$)

Programação em Lógica

- A principal linguagem de Programação em Lógica é o PROLOG, criada nos anos 70 por Alain Colmareur, na Universidade de Marselha

- A linguagem PROLOG usa o "Princípio da Resolução" para a "Prova de Teoremas", onde a solução de problemas é encontrada diretamente por um conjunto de "Axiomas" (factos) ou indiretamente através da "Inferência Lógica" (usando as regras que representam as implicações)

- A Prova de Teoremas segue a "Pesquisa Primeiro em Profundidade", na tentativa de provar o teorema, a este mecanismo acrescenta-se o "retorno atrás quando ocorre falha" (backtracking)

- As variáveis podem residir num de dois estados: não instanciadas ou instanciadas, quando é encontrada uma solução podem ser exibidas as instanciações possíveis

- Quando algo não está explicitamente definido como um axioma é assumido como sendo falso (Assumpção do Mundo Fechado)

Programação em Lógica

- A Programação em Lógica é passível de várias extensões, nomeadamente para lidar com informação incompleta
- Ao se admitir o Mundo Fechado, dizemos que algo é falso porque não conseguimos provar que é verdadeiro, mas poderia ser apenas desconhecido
- Algumas extensões prevêem dois operadores de negação: a negação forte (¬) e a negação fraca que corresponde ao desconhecido (não)

  Um exemplo:

  ¬comboio(Linha) → atravessar(Linha)

  não comboio(Linha) → atravessar(Linha)

Um exemplo em PROLOG

Vejamos um exemplo, vamos assumir que *a*, *b* e *c* são verdade e que $a \wedge \sim b \rightarrow d$ e $c \vee e \rightarrow f$ também é verdade. Isso em PROLOG seria representado por:

a.

b.

c.

d:- a , \+b.

f:- c ; e.

Em PROLOG a conjunção é a "**,**", a disjunção é o "**;**", a negação é o "**+**" e a implicação é o "**:-**" indicando que o que está à direita implica a conclusão do que está à esquerda

Nota: uma disjunção numa regra (implicação) pode ser substituída por 2 regras, no exemplo *f* poderia ser alternativamente expresso da seguinte forma:

f:- c .

f:- e.

a, *b* e *c* são factos sem argumentos, tal como *d* e *f* são possíveis conclusões de regras, também sem argumentos

Tendo em consideração esse conjunto de axiomas será possível provar *f*, mas não será possível provar *d*

3 – Métodos Básicos de Resolução Automática de Problemas

Carlos Ramos
csr@isep.ipp.pt

Unidade Curricular: Algoritmia Avançada
Aulas Teóricas
Curso: Licenciatura em Engenharia Informática (ISEP)

Método Gerar e Testar

- O método Gerar e Testar é um método “não informado” de pesquisa
  - No Gerar e Testar começa-se por gerar, a partir do estado atual do “mundo”, um novo estado para o “mundo” do problema em questão
  - De seguida testa-se o novo estado gerado, comparando-o com o estado final pretendido, que corresponde à solução
  - Se o estado geral for a solução para o problema então o método termina, caso contrário volta-se a gerar novo estado

- O inconveniente do Gerar e Testar é o facto de poder demorar muito tempo até encontrar a solução

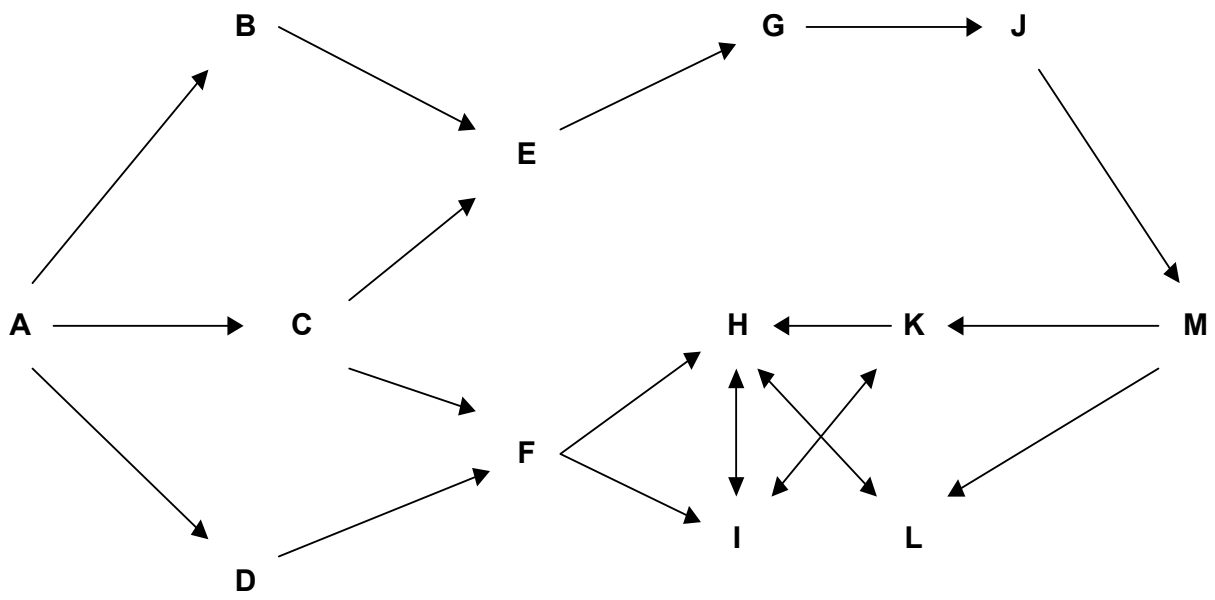
- Trata-se, praticamente, de um método de pesquisa aleatória

Primeiro em Profundidade

- A ideia que está inerente ao método Primeiro em Profundidade é a de tentar avançar de estado para estado até que se encontre a solução
- É um método adequado se as opções tomadas forem na direção correta, mas pode ser inadequado se a direção tomada for desadequada
- O método Primeiro em Profundidade apresenta a vantagem de ter poucos requisitos em termos de memória
- O método é também adequado para problemas que tenham várias soluções pois nesses casos aumenta a possibilidade de se optar por um caminho adequado

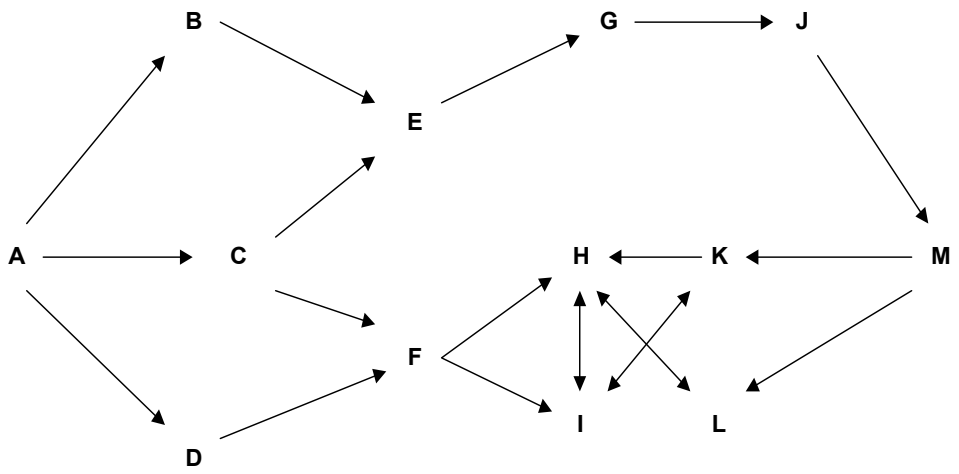
Primeiro em Profundidade

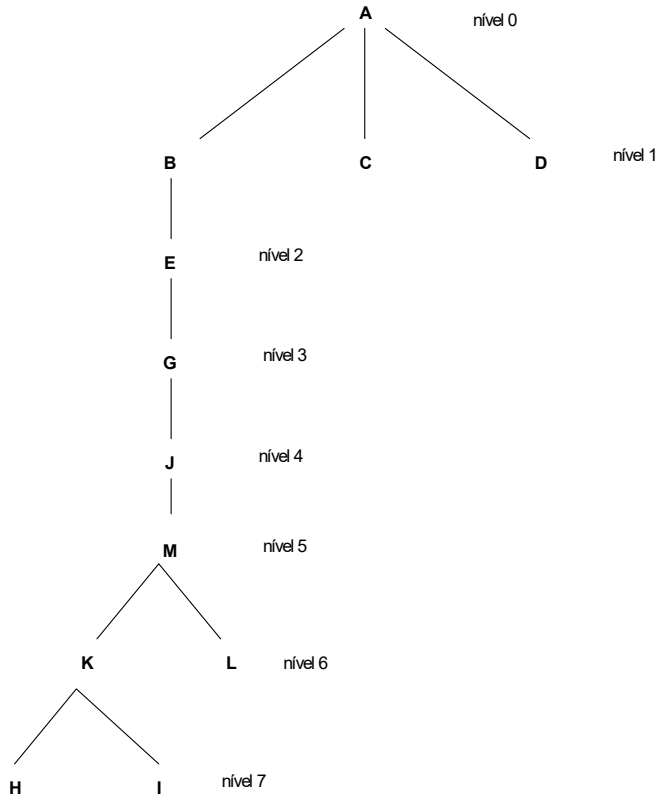
Vejamos um exemplo, considerando o seguinte grafo:



Primeiro em Profundidade

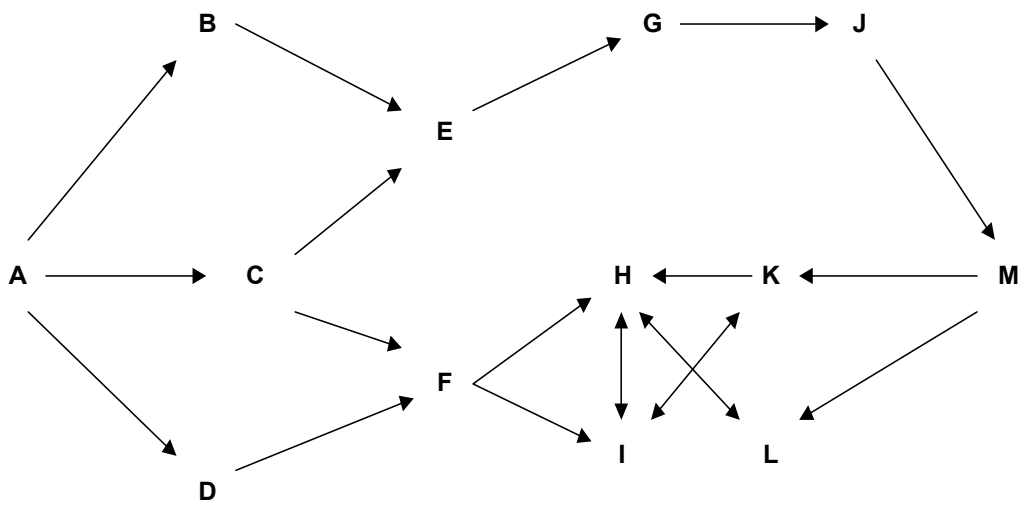
Para ir do nó A até ao nó I,
a árvore de pesquisa gerada poderia ser:





Primeiro em Profundidade

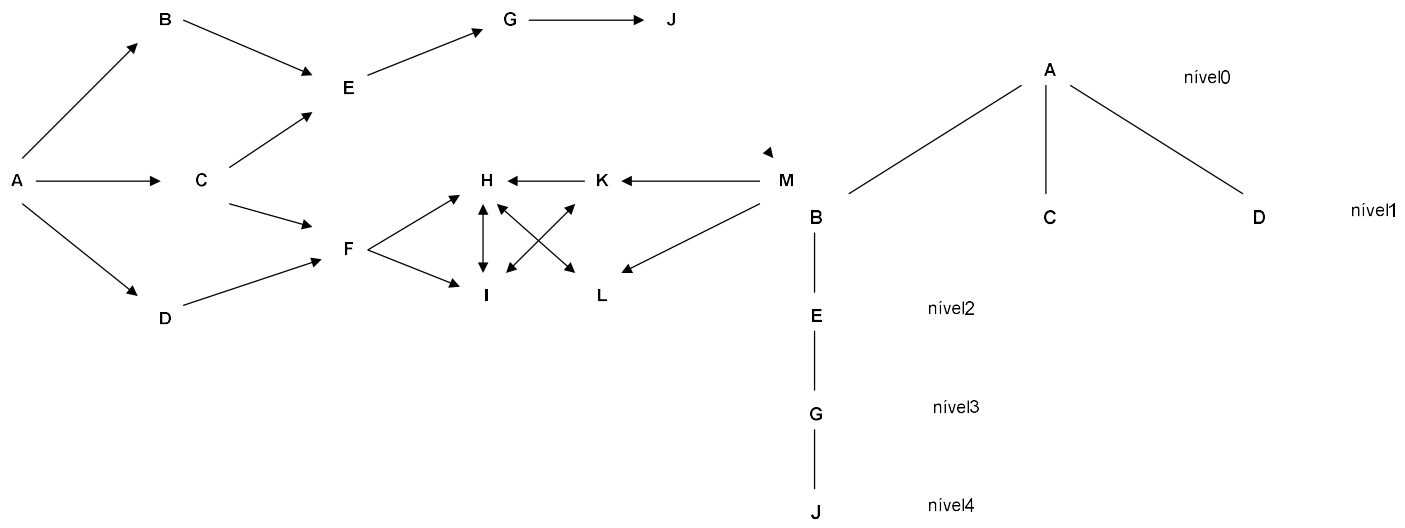
Não há garantia em que o método permita obter a melhor solução, ou a solução ao nível mais próximo da raiz
Note-se que teríamos soluções ao nível 3 como, por exemplo, o caminho A-C-F-I ou A-D-F-I.



E se fosse retirada a ligação entre a cidade J e M ?

Primeiro em Profundidade

E se fosse retirada a ligação entre a cidade J e M ?
Uma vez chegados a J não teríamos por onde avançar

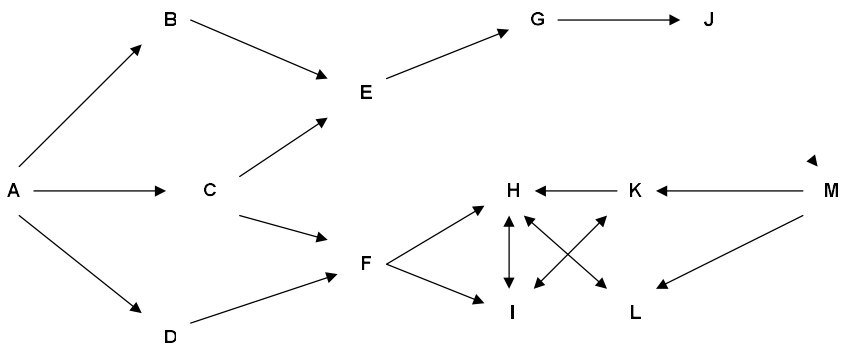


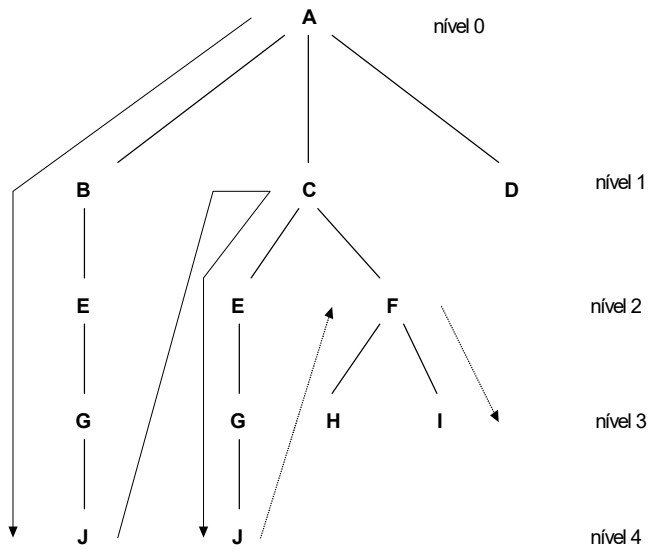
Logo, não seria encontrada nenhuma solução

Primeiro em Profundidade

- A solução é usar o Retrocesso

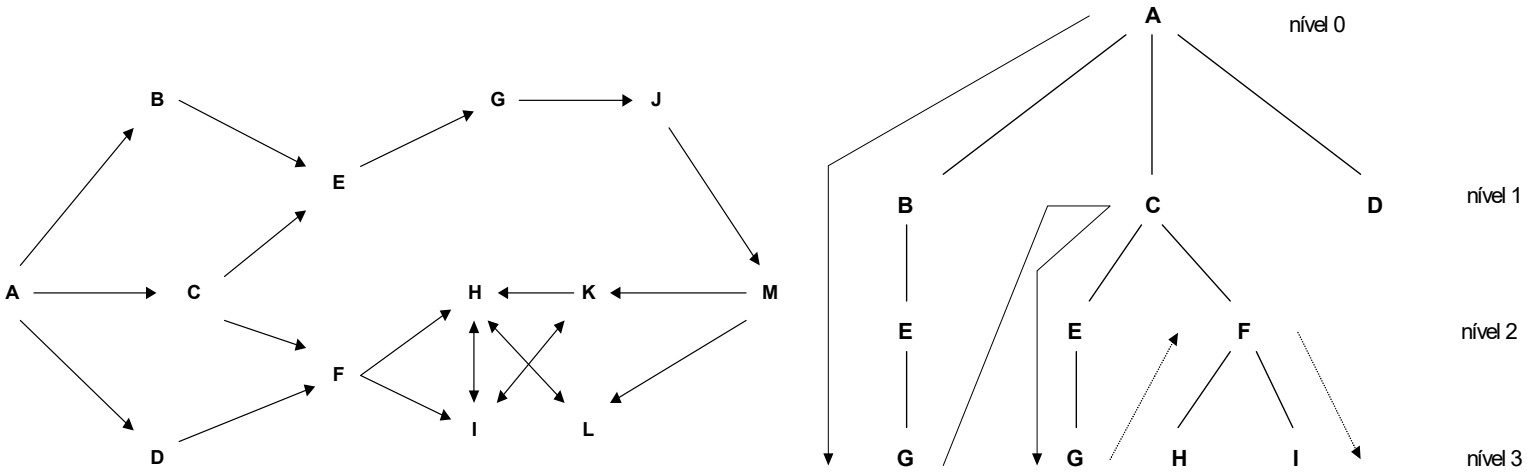
- O método Primeiro em Profundidade com Retrocesso é o método usado internamente na linguagem PROLOG





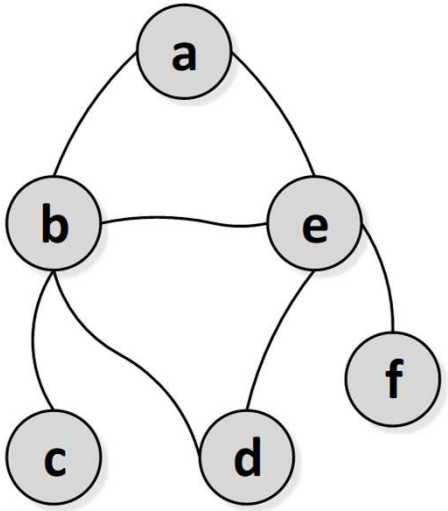
Primeiro em Profundidade

- O problema do Primeiro em Profundidade é avançar por um caminho que até pode estar a afastar-se da solução
- Isso é particularmente comum em grafos de grande dimensão
- Nessas situações podemos optar pelos níveis de corte, veja-se o exemplo seguinte com corte ao 3º nível



Que problema pode trazer o nível de corte? Como resolvê-lo?

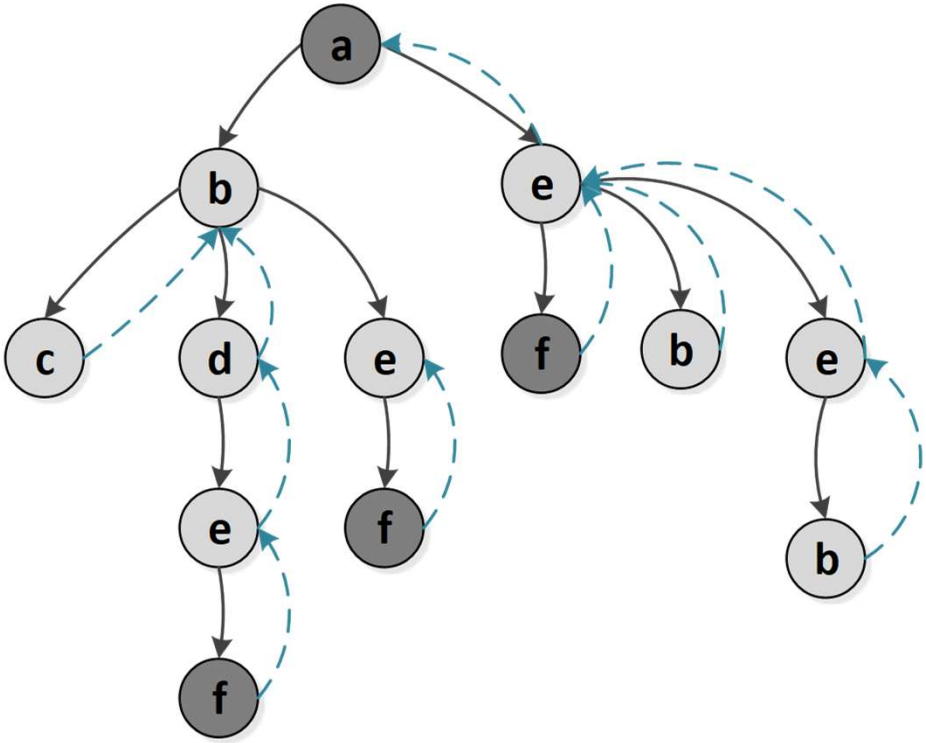
Implementação do Primeiro em Profundidade
Depth First Search (DFS)
Grafo Exemplo



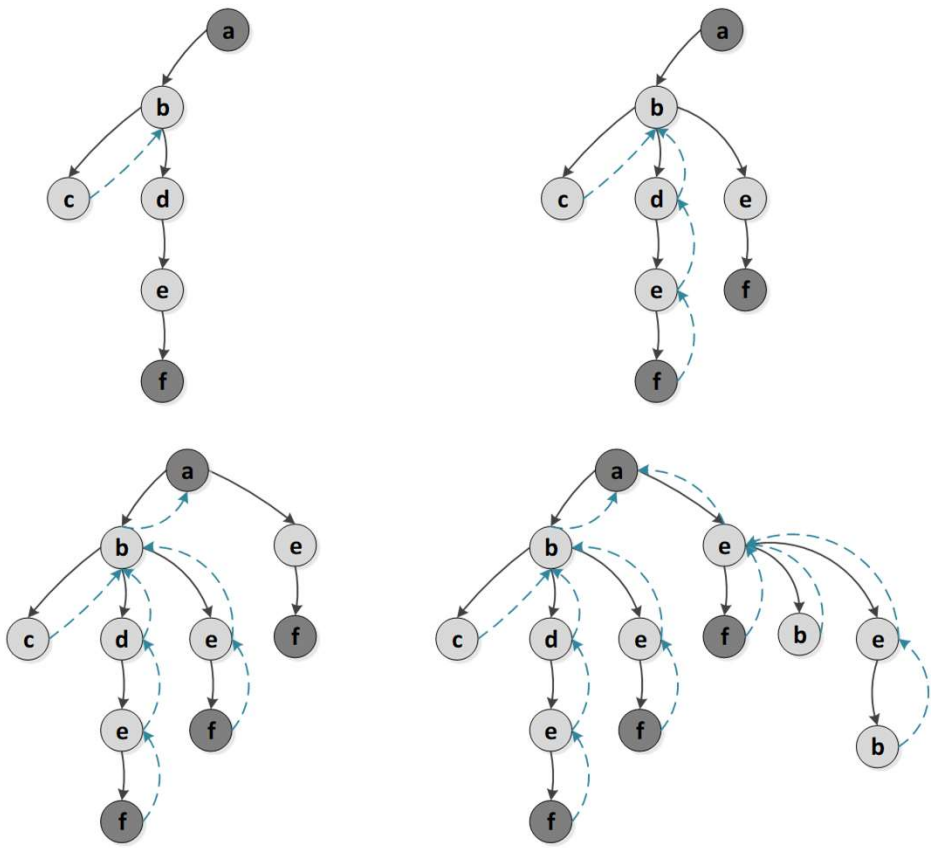

```
%edge(Node1,Node2)
edge(a,b).
edge(a,e).
edge(b,c).
edge(b,d).
edge(b,e).
edge(d,e).
edge(e,f).

% ligacoes bidireccionais
connect(X,Y):-
        edge(X,Y);edge(Y,X).
```

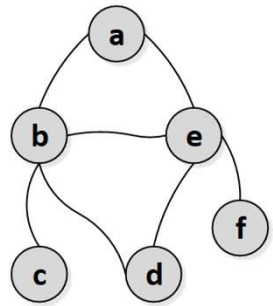
Implementação do Primeiro em Profundidade Depth First Search (DFS) Implementação Prolog / Soluções

```
dfs(Orig,Dest,Cam):-
        dfs2(Orig,Dest,[Orig],Cam).

%condicao final: nodo actual = destino
dfs2(Dest,Dest,LA,Cam):-
        %caminho actual esta invertido
        reverse(LA,Cam).

dfs2(Act,Dest,LA,Cam):-
        %testar ligacao entre ponto
        %actual e um qualquer X
        connect(Act,X),

        %testar nao circularidade p/ nao
        %visitar nodos ja visitados
        \+ member(X,LA),
        %chamada recursiva
        dfs2(X,Dest,[X|LA],Cam).
```



Implementação do Primeiro em Profundidade
Depth First Search (DFS)
Espaço do problema / Soluções






```
%edge(Node1,Node2)
edge(a,b).
edge(a,e).
edge(b,c).
edge(b,d).
edge(b,e).
edge(d,e).
edge(e,f).

% ligacoes bidireccionais
connect(X,Y):-
        edge(X,Y);edge(Y,X).
```

Depth First Search (DFS)
Implementação Prolog / Soluções

```
dfs(Orig,Dest,Cam):-
       dfs(Orig,Dest,[Orig],Cam).

dfs(Dest,Dest,LA,Cam):-
       reverse(LA,Cam).

dfs(Act,Dest,LA,Cam):-
       write(LA),nl,
       connect(Act,X),
       \+ member(X,LA),
       dfs(X,Dest,[X|LA],Cam).
```

```
?- dfs(a,f,Caminho).
[a]
[b,a]
[b,a]
[d,b,a]
[e,d,b,a]
[f,e,d,b,a]
Caminho = [a, b, d, e, f] ;
[b,a]
[e,b,a]
[f,e,b,a]
Caminho = [a, b, e, f] ;
[e,b,a]
[a]
[e,a]
[f,e,a]
Caminho = [a, e, f] ;
[e,a]
[b,e,a]
[b,e,a]
[e,a]
[d,e,a]
[b,d,e,a]
false.
```

Primeiro em Largura

- Segundo este método só irá ser possível efetuar a pesquisa no nível N da árvore se todos os nós do nível N-1 tiverem sido explorados

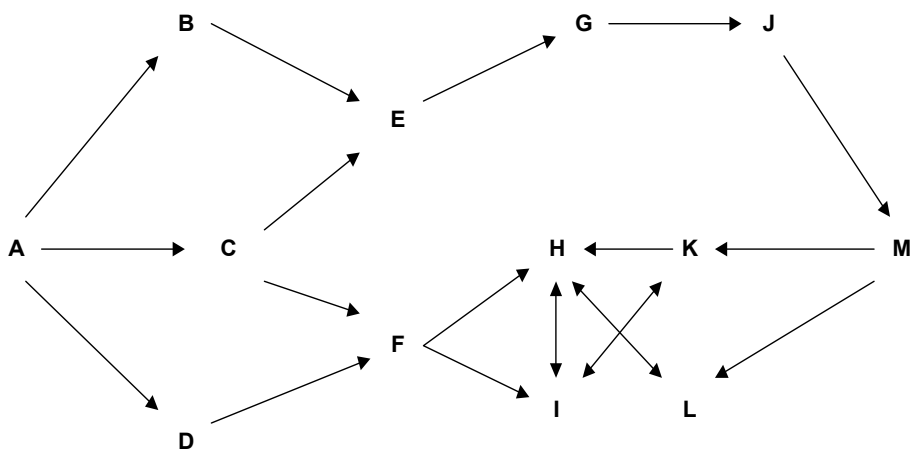
- Pode-se dizer que a árvore é explorada transversalmente, derivando daí o nome do método

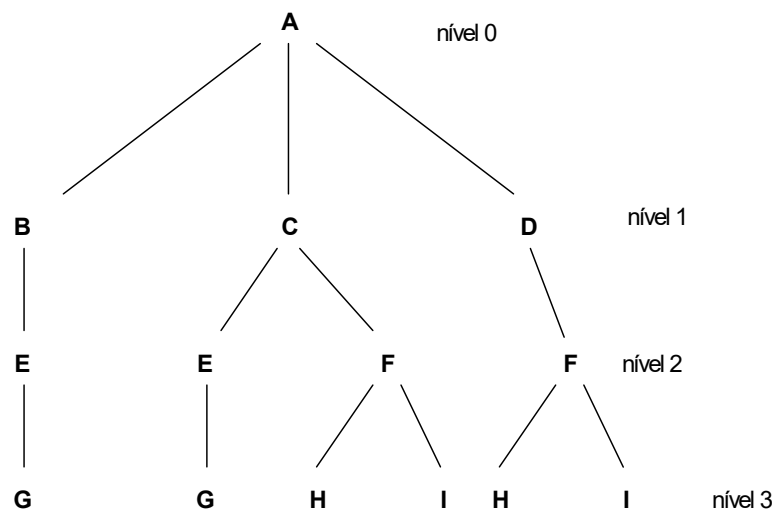
- Garante a obtenção da solução ao nível mais próximo da raiz, o que não quer dizer que seja a melhor solução

- Pode requerer muita memória, espaço e tempo para problemas mais complexos

Primeiro em Largura

•Vejamos como se desenvolve o método para o mesmo exemplo dado anteriormente:



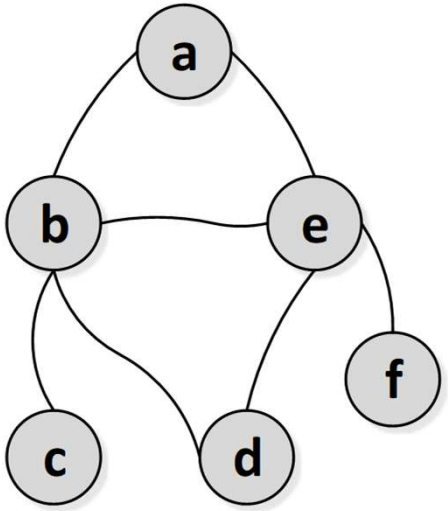


Primeiro em Largura

- A árvore de pesquisa tem o mesmo problema que aparecia no Primeiro em Profundidade, ou seja, há alguma redundância que pode ser evitada, originada pela exploração duplicada dos nós E e F ao nível 2. Isso poderia ser evitado no método. No entanto, deixamos ficar essa redundância para averiguar os efeitos que originam, ou seja a repetição das subárvores que estão debaixo desses nós

- Mas a redundância pode ser pretendida, visto que corresponde a maneiras alternativas de chegar à solução. Por exemplo, ao nível 2 há duas maneiras de chegar a F, uma por C outra por D. Quem aplica o método pode estar interessado em considerar essas duas alternativas
  - Suponha-se, por exemplo, que se pretende ir de A até I, mas parando num posto de gasolina e num restaurante e que foi detetado um restaurante em C e um posto de gasolina em D.
  - A omissão de um dos caminhos de pesquisa pode ser incorreta, pois a decisão entre qual caminho escolher, por C ou por D, dependerá de algo que só seria verificado no processo de pesquisa do caminho de F para I. Por exemplo, se entre F e I houver um posto de gasolina, então o caminho por C é melhor enquanto que se entre F e I houver um restaurante o caminho por D será mais adequado

Implementação do Primeiro em Largura
Breadth First Search (BFS)
Grafo Exemplo

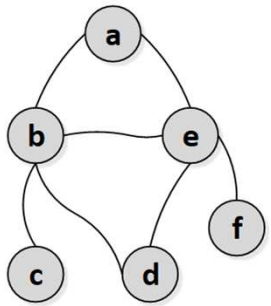


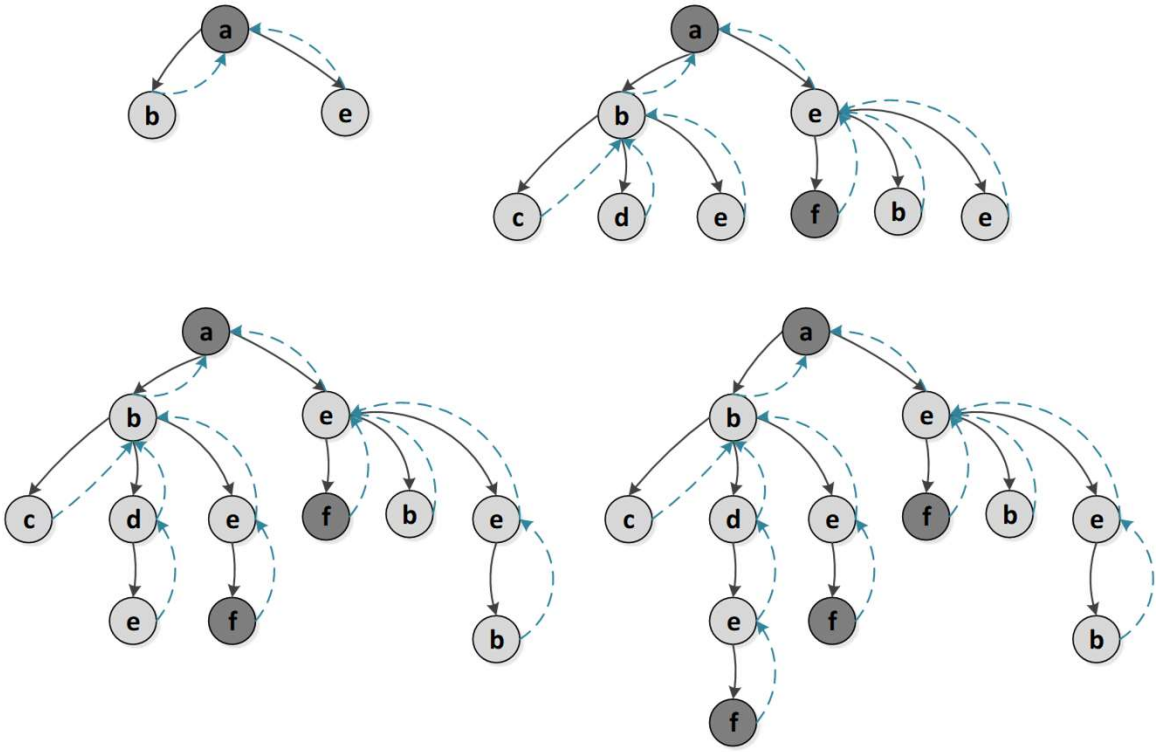

```
%edge(Node1,Node2)
edge(a,b).
edge(a,e).
edge(b,c).
edge(b,d).
edge(b,e).
edge(d,e).
edge(e,f).

% ligacoes bidireccionais
connect(X,Y):-
        edge(X,Y);edge(Y,X).
```

Implementação do Primeiro em Largura Breadth First Search (BFS) Implementação Prolog

```
bfs(Orig,Dest,Cam):-bfs2(Dest,[[Orig]],Cam).

%condicao final: destino = nó à cabeça do caminho actual
bfs2(Dest,[[Dest|T]|_],Cam):-
        %caminho actual está invertido
        reverse([Dest|T],Cam).

bfs2(Dest,[LA|Outros],Cam):-
        LA=[Act|_],
        %calcular todos os nós adjacentes nao visitados e
        %gerar um caminho novo c/ cada nó e caminho actual
        findall([X|LA],
                (Dest\==Act,connect(Act,X),\+ member(X,LA)),
                Novos),
        %novos caminhos são colocados no final da lista
        %p/ posterior exploracao
        append(Outros,Novos,Todos),
        %chamada recursiva
        bfs2(Dest,Todos,Cam).
```

Implementação do Primeiro em Largura
Breadth First Search (BFS)
Espaço do Problema / Soluções






```
%edge(Node1,Node2)
edge(a,b).
edge(a,e).
edge(b,c).
edge(b,d).
edge(b,e).
edge(d,e).
edge(e,f).

% ligacoes bidireccionais
connect(X,Y):-
        edge(X,Y);edge(Y,X).
```

Breadth First Search (BFS)
Implementação Prolog / Soluções

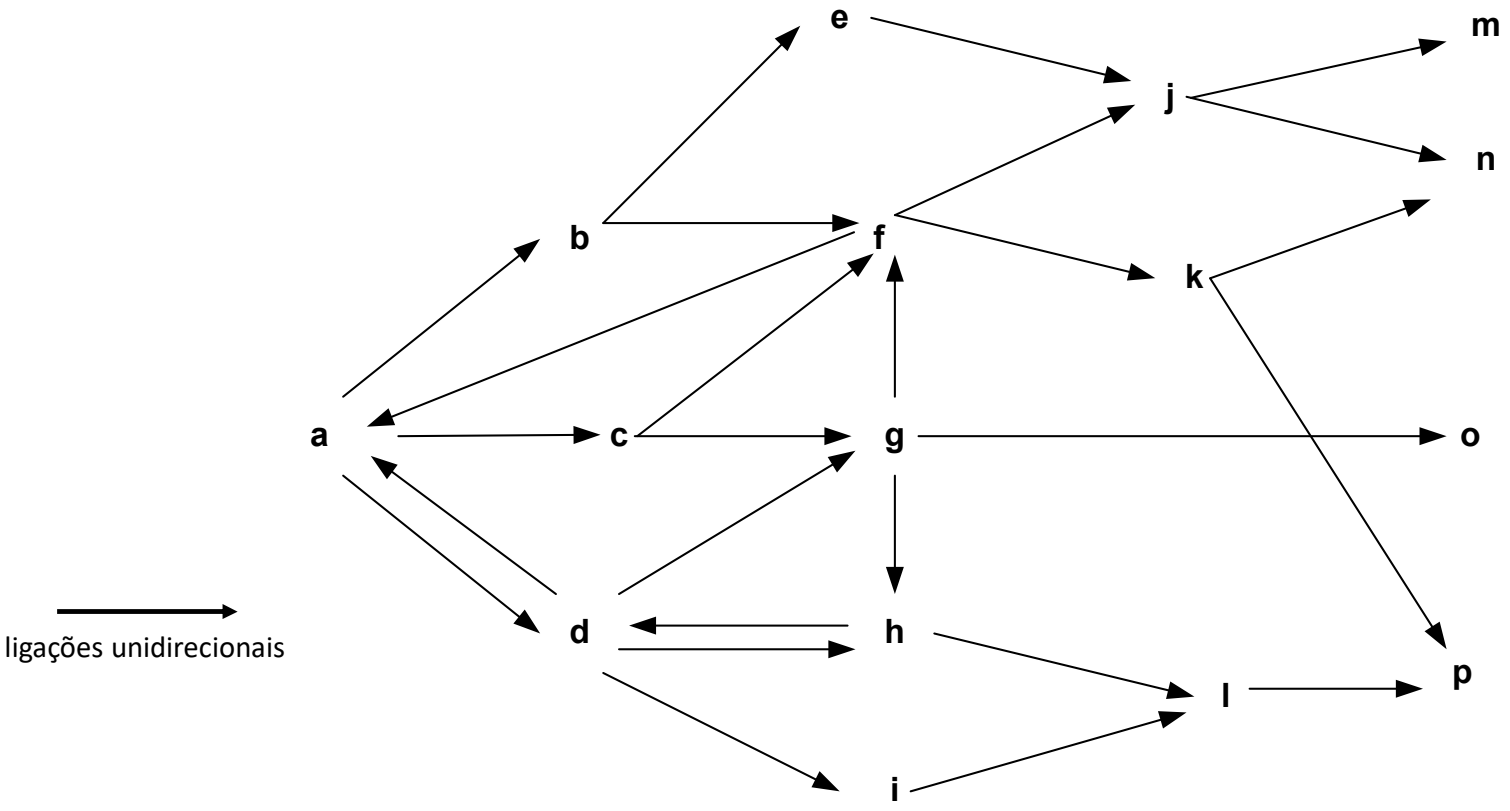
```
bfs(Orig,Dest,Cam):-
bfs2(Dest,[[Orig]],Cam).

bfs2(Dest,[[Dest|T]|_],Cam):-
        reverse([Dest|T],Cam).

bfs2(Dest,[LA|Outros],Cam):-
          write([LA|Outros]),nl,
           LA=[Act|_],
findall([X|LA],
(Dest\==Act,connect(Act,X),\+member(X,LA)),
Novos),
        append(Outros,Novos,Todos),
        bfs2(Dest,Todos,Cam).
```

```
?- bfs(a,f,Caminho).
[[a]]
[[b,a],[e,a]]
[[e,a],[c,b,a],[d,b,a],[e,b,a]]
[[c,b,a],[d,b,a],[e,b,a],[f,e,a],[b,e,a],[d,e,a]]
[[d,b,a],[e,b,a],[f,e,a],[b,e,a],[d,e,a]]
[[e,b,a],[f,e,a],[b,e,a],[d,e,a],[e,d,b,a]]
Caminho = [a, e, f] ;
[[f,e,a],[b,e,a],[d,e,a],[e,d,b,a],[f,e,b,a],[d,e,b,a]]
[[b,e,a],[d,e,a],[e,d,b,a],[f,e,b,a],[d,e,b,a]]
[[d,e,a],[e,d,b,a],[f,e,b,a],[d,e,b,a],[c,b,e,a],[d,b,e,a]]
[[e,d,b,a],[f,e,b,a],[d,e,b,a],[c,b,e,a],[d,b,e,a],[b,d,e,a]]
Caminho = [a, b, e, f] ;
[[f,e,b,a],[d,e,b,a],[c,b,e,a],[d,b,e,a],[b,d,e,a],[f,e,d,b,a]]
[[d,e,b,a],[c,b,e,a],[d,b,e,a],[b,d,e,a],[f,e,d,b,a]]
[[c,b,e,a],[d,b,e,a],[b,d,e,a],[f,e,d,b,a]]
[[d,b,e,a],[b,d,e,a],[f,e,d,b,a]]
[[b,d,e,a],[f,e,d,b,a]]
Caminho = [a, b, d, e, f] ;
[[f,e,d,b,a],[c,b,d,e,a]]
[[c,b,d,e,a]]
false.
```

O que acontece se trocarmos a ordem dos dois primeiros argumento do append?

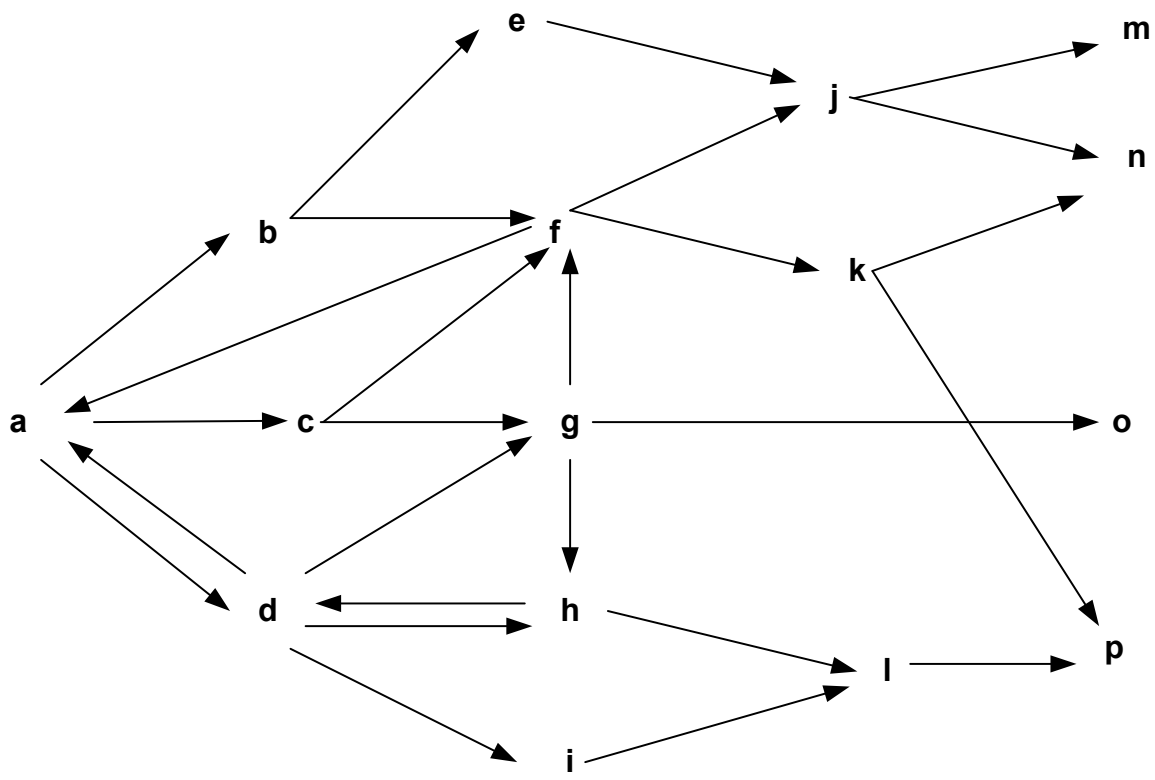
```
bfs(Orig,Dest,Cam):-bfs2(Dest,[[Orig]],Cam).

%condicao final: destino = nó à cabeça do caminho actual
bfs2(Dest,[[Dest|T]|_],Cam):-
        %caminho actual está invertido
        reverse([Dest|T],Cam).

bfs2(Dest,[LA|Outros],Cam):-
        LA=[Act|_],
        %calcular todos os nós adjacentes nao visitados e
        %gerar um caminho novo c/ cada nó e caminho actual
        findall([X|LA],
                (Dest\==Act,connect(Act,X),\+ member(X,LA)),
                Novos),
        %novos caminhos são colocados no inicio da lista
        %p/ exploracao imediata
        append(Novos, Outros, Todos),
        %append(Outros, Novos, Todos),
        %chamada recursiva
        bfs2(Dest,Todos,Cam).
```

Mais um exemplo para comparar o DFS com o BFS (com Retrocesso - Backtracking)




```
liga(a,b).
liga(a,c).
liga(a,d).
liga(b,e).
liga(b,f).
liga(c,f).
liga(c,g).
liga(d,g).
liga(d,h).
liga(d,i).
liga(e,j).
liga(f,a).
liga(f,j).
liga(f,k).
liga(g,f).
liga(g,o).
liga(g,h).
liga(h,d).
liga(h,l).
liga(i,l).
liga(j,m).
liga(j,n).
liga(k,n).
liga(k,p).
liga(l,p).
```

Mais um exemplo para comparar o DFS com o BFS (com Retrocesso - Backtracking)



Usando o Primeiro em Profundidade (com backtracking)

?- dfs(a,j,L).

L = [a, b, e, j] ;

L = [a, b, f, j] ;

L = [a, c, f, j] ;

L = [a, c, g, f, j] ;

L = [a, d, g, f, j]

?- dfs(a,p,L).

L = [a, b, f, k, p] ;

L = [a, c, f, k, p] ;

L = [a, c, g, f, k, p] ;

L = [a, c, g, h, d, i, l, p] ;

L = [a, c, g, h, l, p] ;

L = [a, d, g, f, k, p] ;

L = [a, d, g, h, l, p] ;

L = [a, d, h, l, p] ;

L = [a, d, i, l, p]

Usando o Primeiro em Largura (com backtracking)

?- bfs(a,j,L).

L = [a, b, e, j] ;

L = [a, b, f, j] ;

L = [a, c, f, j] ;

L = [a, c, g, f, j] ;

L = [a, d, g, f, j]

?- bfs(a,p,L).

L = [a, b, f, k, p] ;

L = [a, c, f, k, p] ;

L = [a, d, h, l, p] ;

L = [a, d, i, l, p] ;

L = [a, c, g, f, k, p] ;

L = [a, c, g, h, l, p] ;

L = [a, d, g, f, k, p] ;

L = [a, d, g, h, l, p] ;

L = [a, c, g, h, d, i, l, p]

4 – Teoria de Jogos

Carlos Ramos
csr@isep.ipp.pt

Unidade Curricular: Algoritmia Avançada
Aulas Teóricas
Curso: Licenciatura em Engenharia Informática (ISEP)

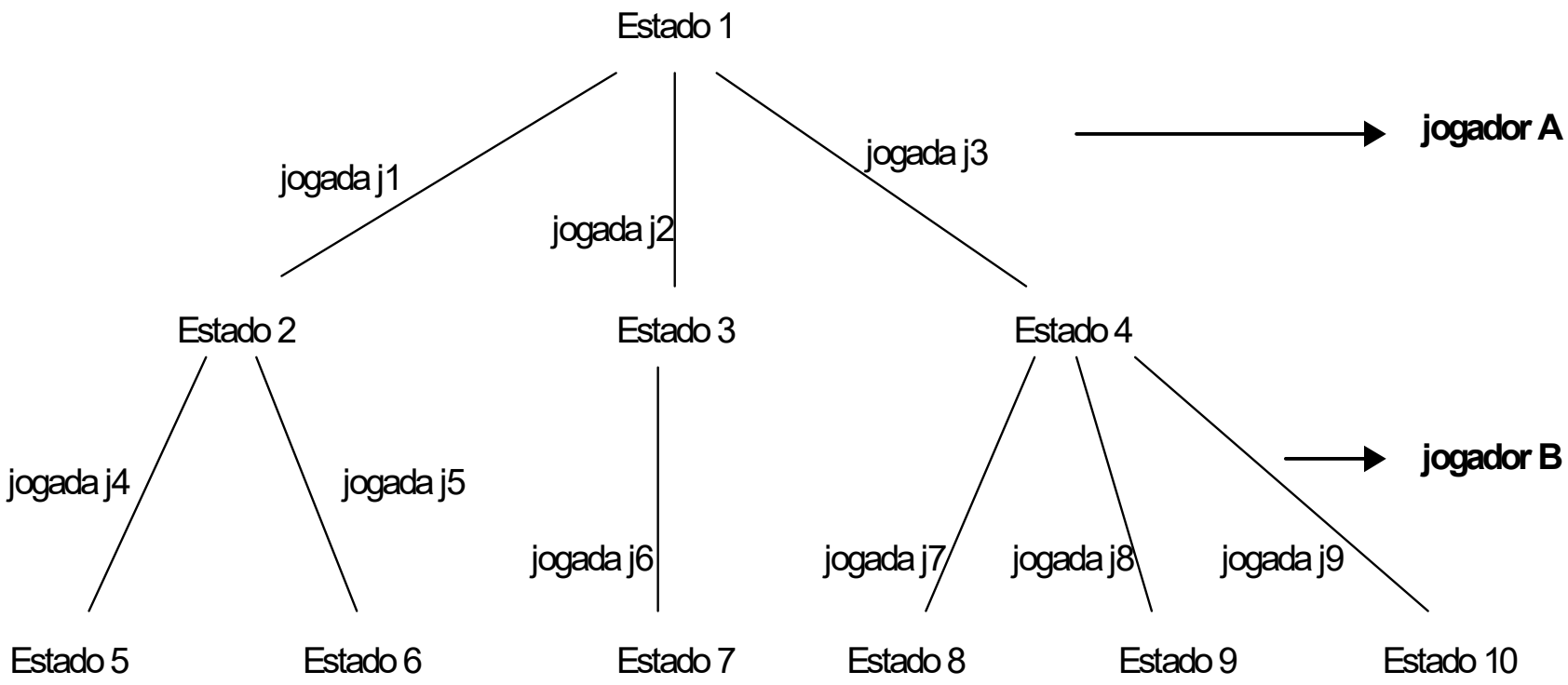
Teoria de Jogos

Os jogos constituem uma área privilegiada na Resolução Automática de Problemas.

- a indústria de jogos por computador movimenta verbas muito significativas, cada vez mais os jogos aparecem com "Inteligência Artificial" incorporada, isso é mais verdade para os casos em que o jogador tem que se defrontar contra um oponente comandado pelo computador (xadrez, futebol, etc)
- ao modelar o jogador e os seus oponentes a teoria dos jogos aproxima-se mais da realidade do que os métodos de pesquisa explicados anteriormente, os quais consideravam que os caminhos a seguir na pesquisa dependiam apenas de uma entidade (o computador), por exemplo, o funcionamento de um mercado concorrente pode ser visto como um jogo

Teoria de Jogos

Jogar pode ser visto como desenvolver uma árvore onde os nós representam estados e os ramos representam jogadas, como aparece ilustrado



Teoria de Jogos

Alguns factores a considerar:

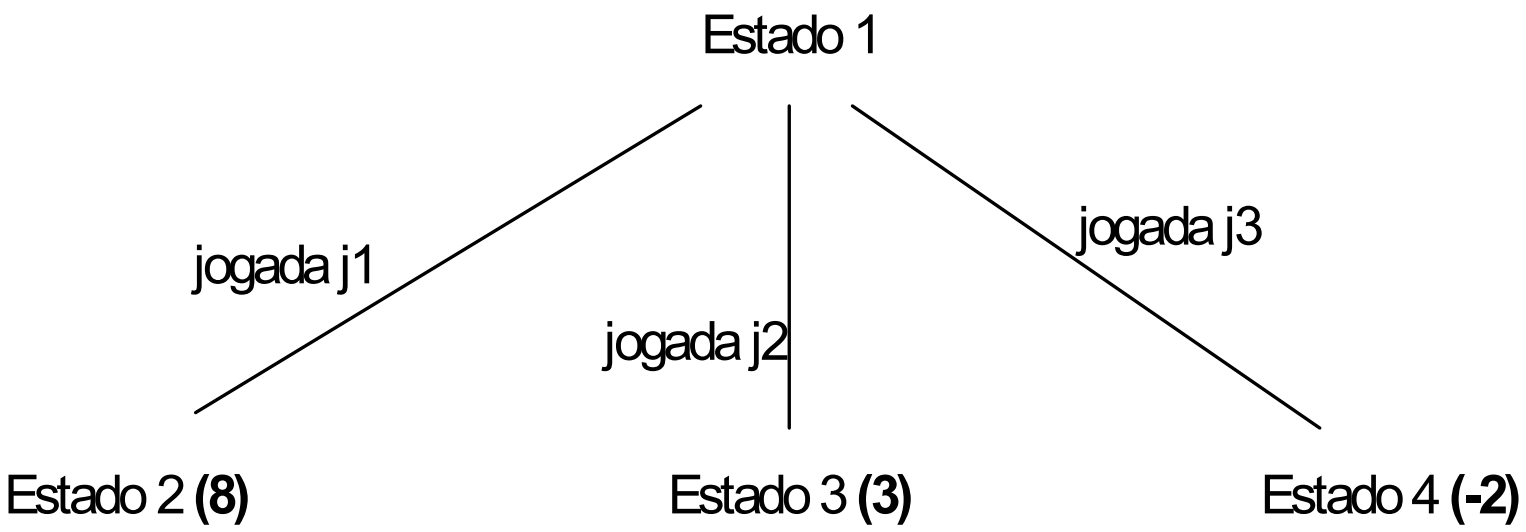
- Impossibilidade de explorar todos os caminhos possíveis em jogos reais e complexos
- Factor de ramificação
- Métodos Informados e uso de Heurísticas
- Quanto mais informado for o método maior será a sua capacidade de prever o que possa suceder mais adiante (maior "look ahead").
- Como avaliar a vantagem ou desvantagem de uma dada decisão?

O Método MINIMAX

- O método Minimax é o método mais conhecido para lidar com jogos. Admite-se que existe um gerador de estados e uma função que avalia a vantagem ou desvantagem de um dado estado.
- Vamos considerar que essa função de avaliação fornece:
  - Um valor positivo se a situação é vantajosa para o jogador que o sistema representa, quanto maior for o valor maior será a vantagem
  - Um valor negativo se a situação é vantajosa para o adversário, quanto mais negativo for o valor maior será a vantagem do adversário
  - Um valor nulo para uma situação de empate
- O objectivo é tentar maximizar o valor dado pela função de avaliação.

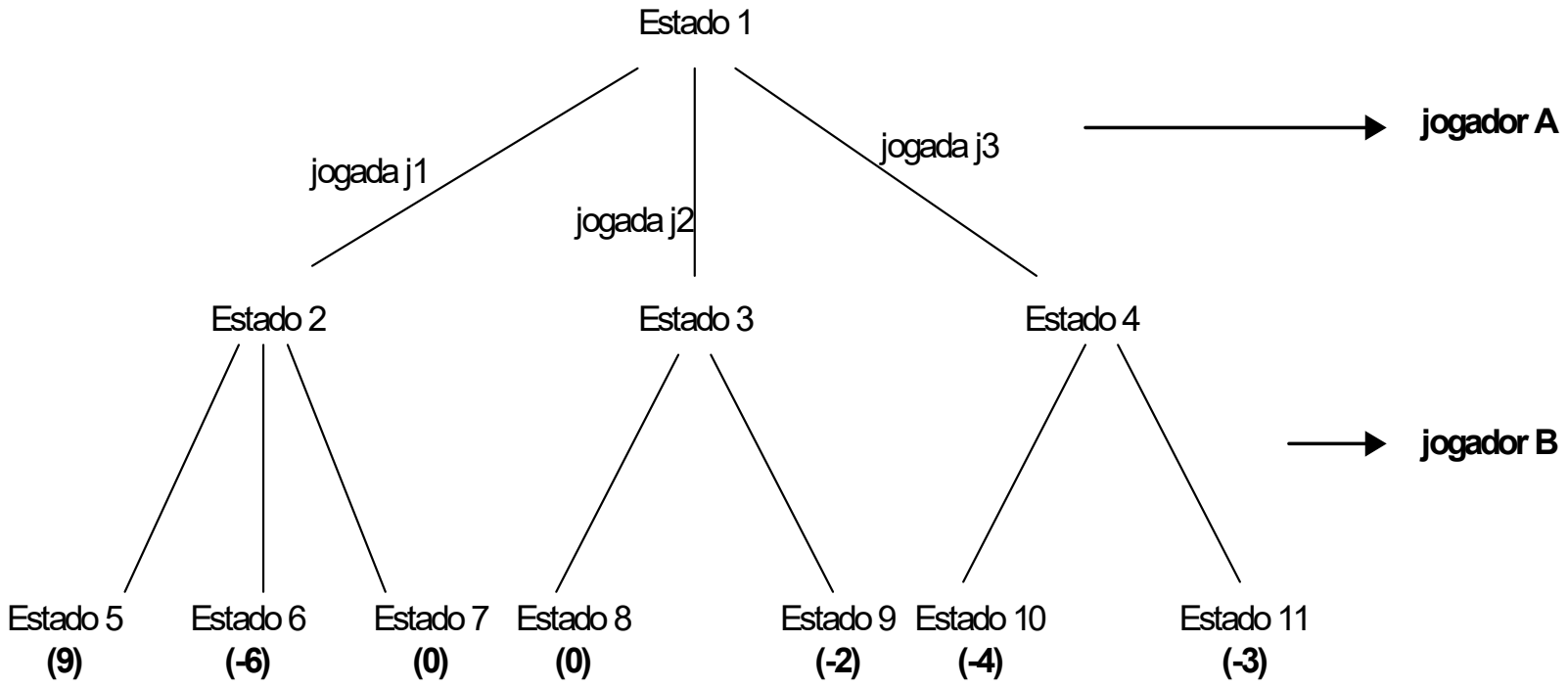
O Método MINIMAX

Neste exemplo, a profundidade de análise é 1. As avaliações dos Estados correspondentes às folhas dão valores 8, 3 e –2 respectivamente. Optamos pela primeira jogada que é aquela que origina o maior valor (8).



O Método MINIMAX

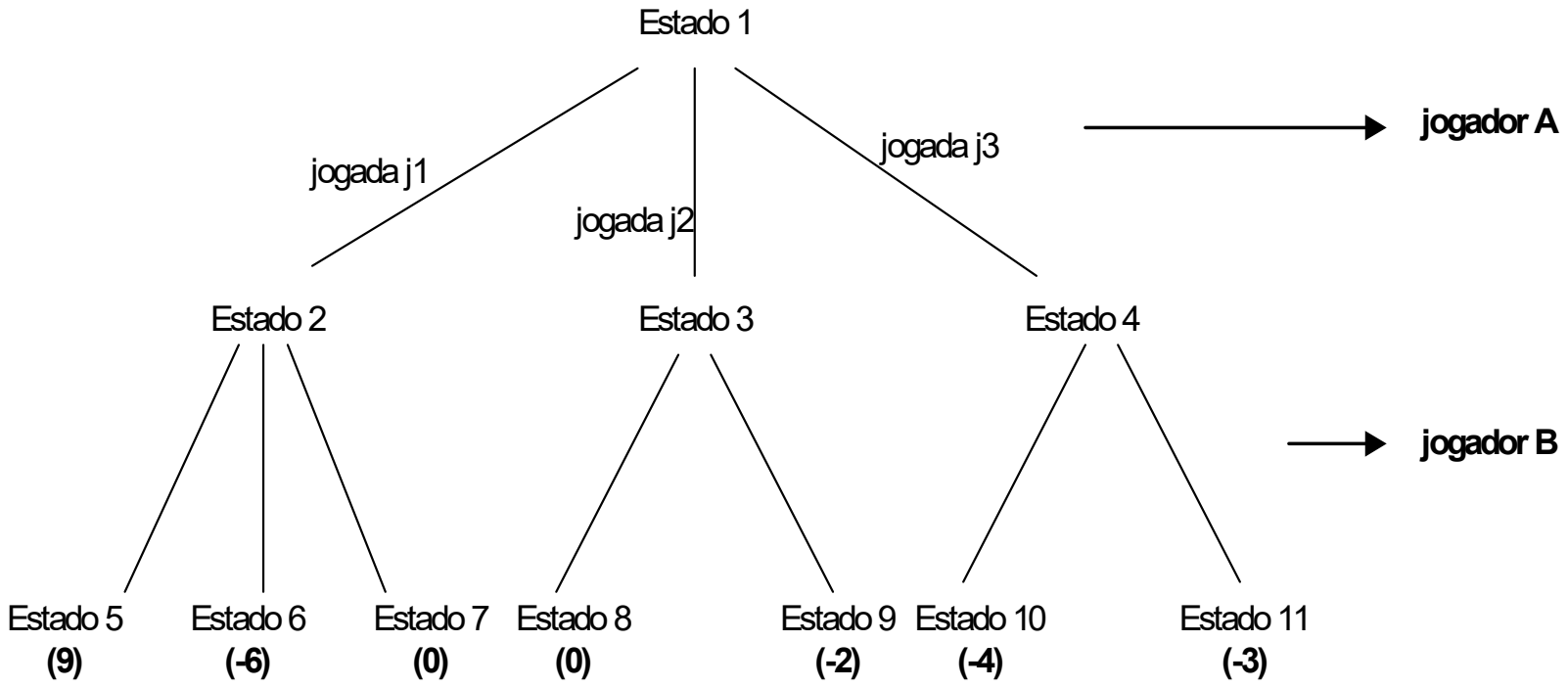
Neste outro exemplo com 2 níveis, o estado mais vantajoso para nós, ao fim de duas jogadas é o Estado 5, cuja função de avaliação dá 9. Se jogássemos a procura desse estado teríamos que ir para o Estado 2 através da jogada j1. Mas o adversário certamente que responderia indo para o Estado 6, cuja função de avaliação dá –6, que até é o estado mais desvantajoso para nós



O Método MINIMAX

O método MINIMAX permite que possamos escolher a jogada mais segura. O nome do método deriva do princípio que utiliza:

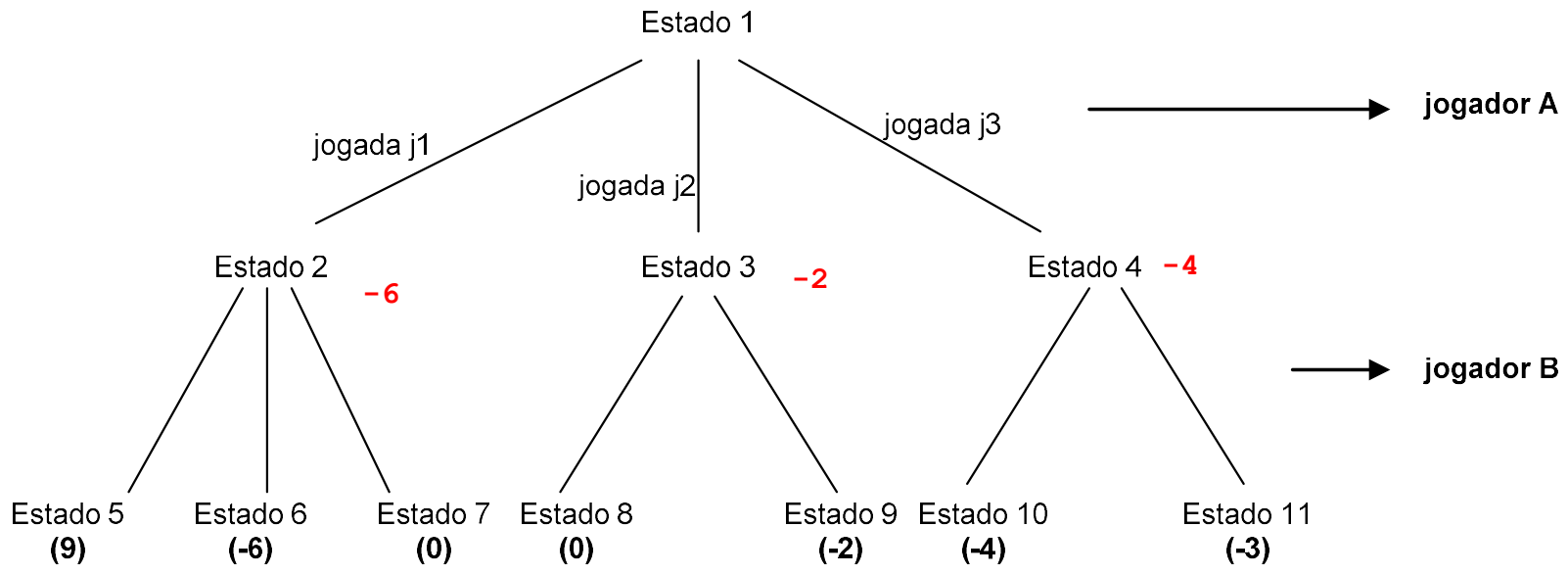
- MIN: minimiza o valor da função nos estados gerados após jogadas do adversário;
- MAX: maximiza o valor da função dos estados gerados pelas nossas jogadas



O Método MINIMAX

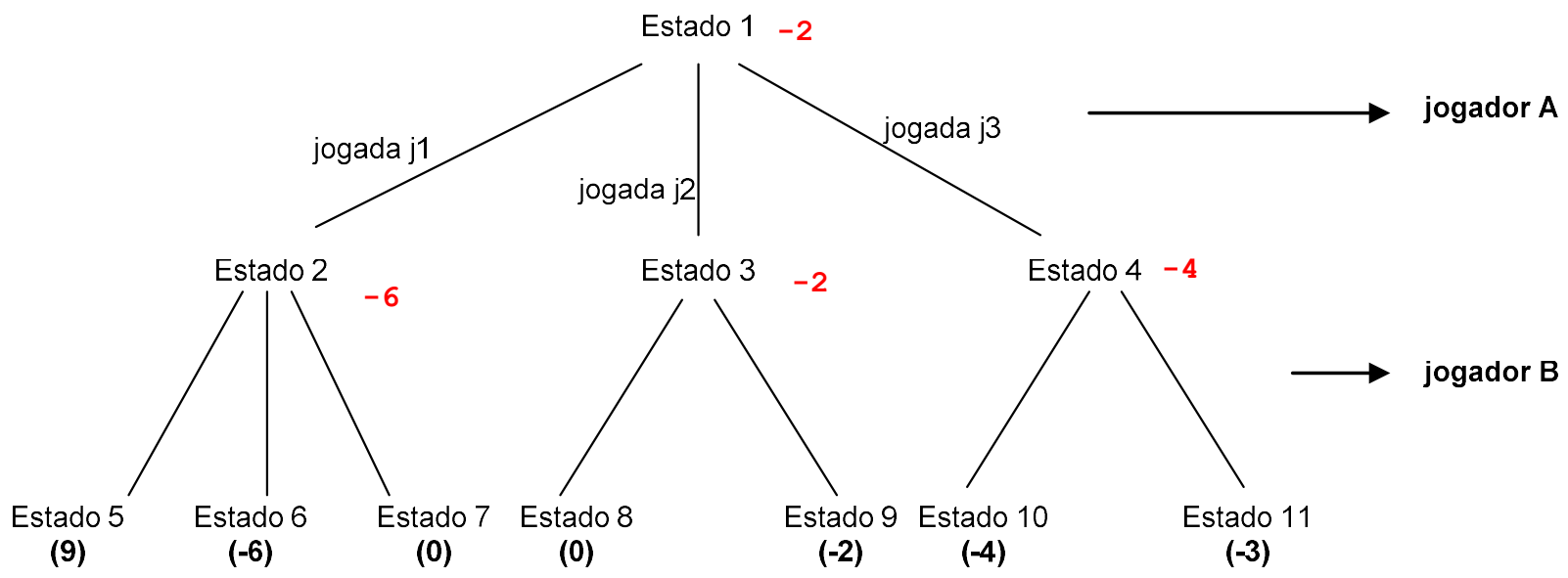
Começamos pelo nível inferior da árvore. Vamos aplicar minimizações, pois vamos admitir que o adversário joga bem e procura os estados mais vantajosos. Nesse exemplo observamos que:

- O Estado 2 toma o valor –6 que corresponde ao menor valor dos Estados 5,6 e 7
- O Estado 3 toma o valor –2 que corresponde ao menor valor dos Estados 8 e 9
- O Estado 4 toma o valor –4 que corresponde ao menor valor dos Estados 10 e 11



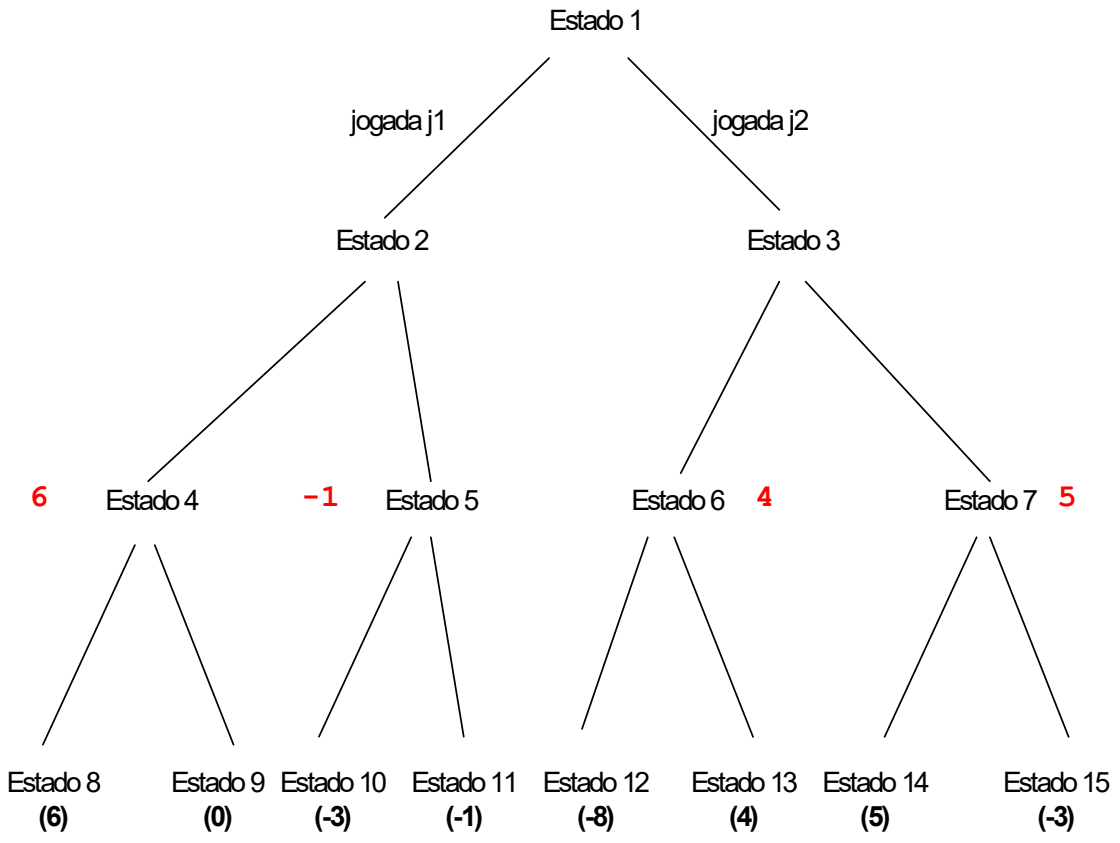
O Método MINIMAX

Agora temos que decidir para onde é que se joga a partir do Estado 1. Como se trata de uma jogada nossa iremos efectuar uma maximização, logo entre o Estado 2, que vale -6, o estado 3, que vale -2, e o estado 4, que vale -4, opta-se pelo Estado 3, que corresponde à jogada j2



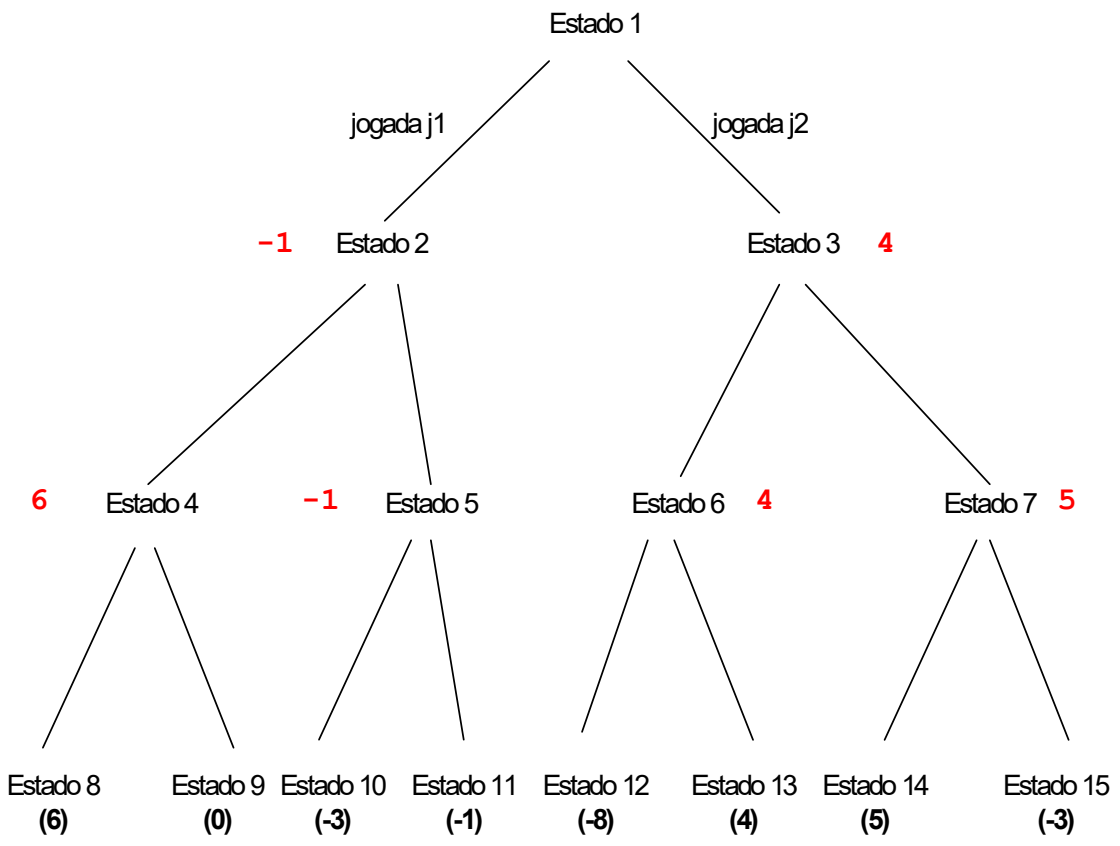
O Método MINIMAX

Vejamos agora um exemplo que desce 3 níveis (uma jogada nossa, outra do adversário e outra nossa). Logo, neste caso, do último nível para o anterior fazem-se maximizações



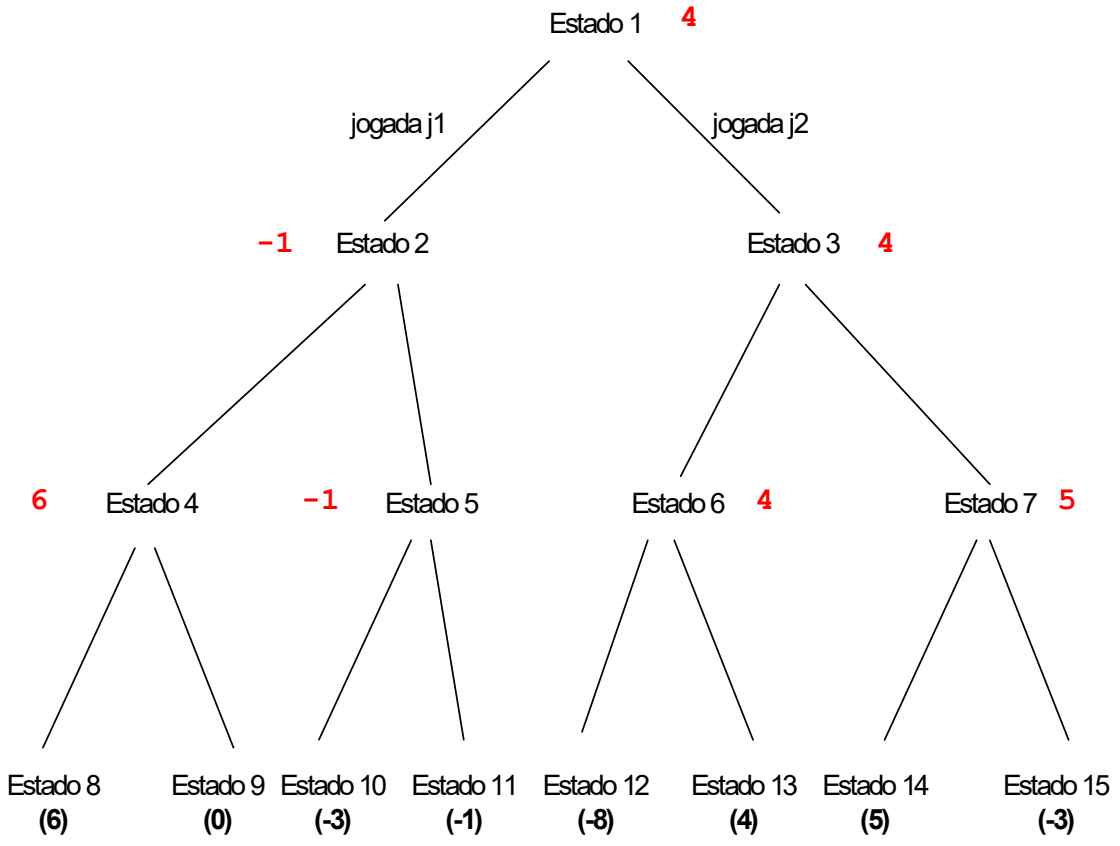
O Método MINIMAX

O passo seguinte é ver qual a jogada do nosso adverário, portanto fazemos minimizações, obtendo -1 para o Estado 2 e 4 para o Estado 3



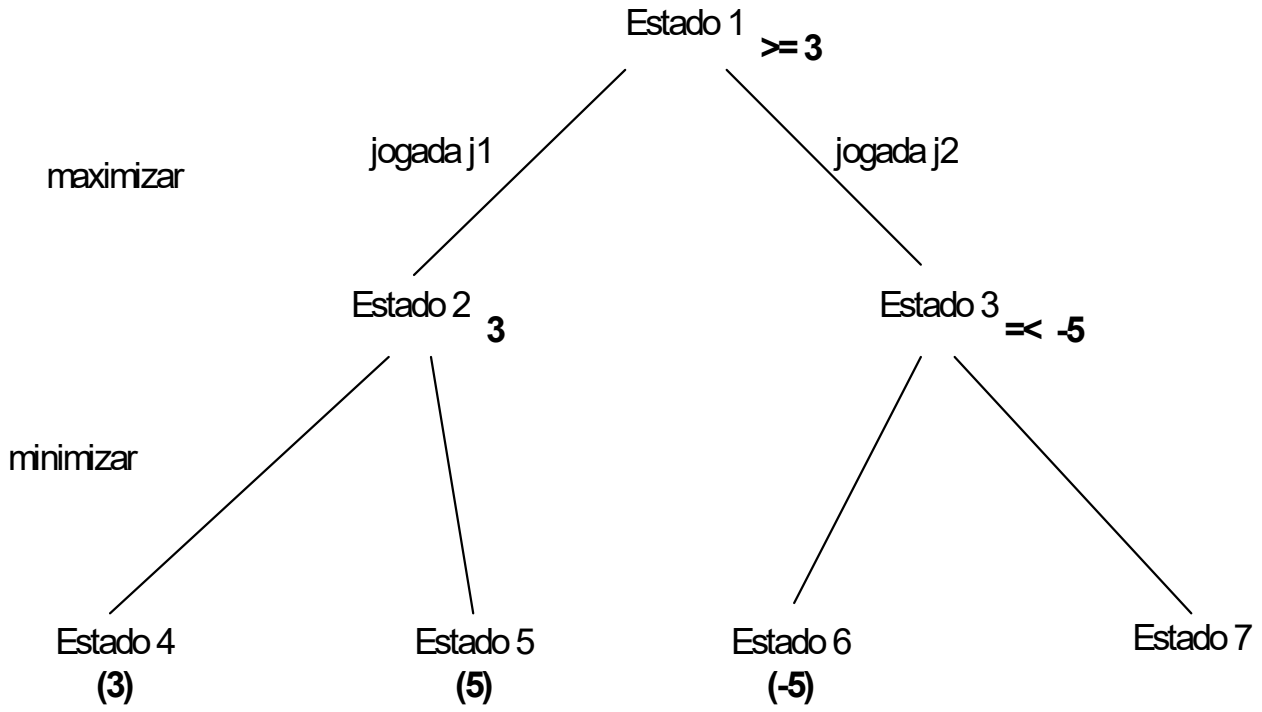
O Método MINIMAX

Finalmente, decide-se pela jogada j2, que é aquela que nos garante que após uma jogada do adversário e outra nossa vamos chegar a um Estado que vale pelo menos 4.



O Método MINIMAX com cortes αβ

A avaliação de estados pode ser morosa e em certos casos não valerá a pena explorar certos nós visto que as situações daí derivadas não interessam



O Método MINIMAX com cortes αβ

O método αβ tira partido desse tipo de raciocínio e define dois níveis de corte:

- α é o limite inferior do valor que pode ser atribuído ao nó de maximização (no caso do exemplo anterior era o 3)
- β é o limite superior do valor que pode ser atribuído ao nó de minimização (no caso do exemplo anterior era o valor –5)

A pesquisa acaba quando nos níveis minimizadores se obtém algo abaixo de α ou quando nos níveis maximizadores se obtém algo acima de β

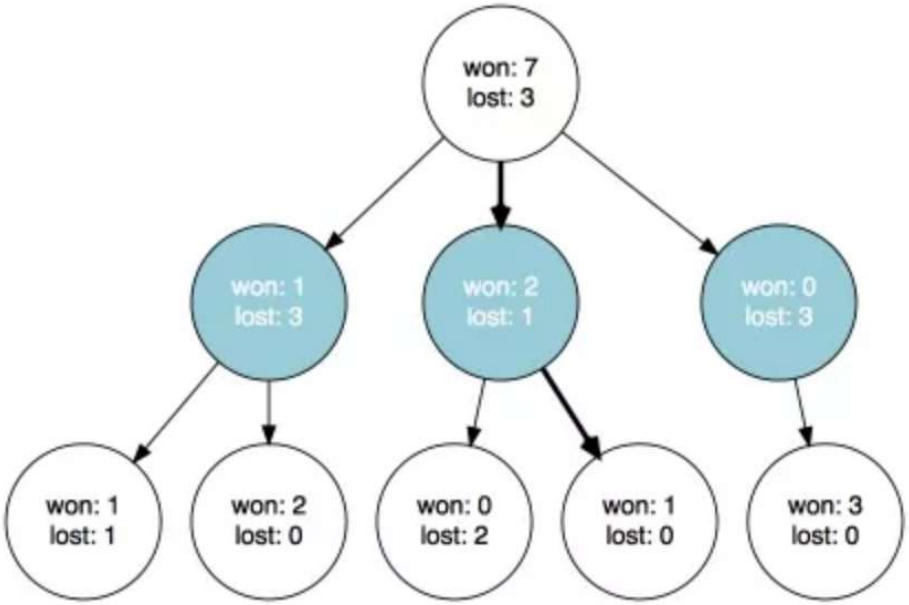
Monte Carlo Tree Search (MCTS)

O método de Pesquisa em Árvore Monte Carlo tem sido aplicado com sucesso em Teoria de Jogos, tendo ganho especial atenção por ter sido usado no AlphaGo, o sistema que derrotou o maior mestre de Go (um jogo mais complexo que o xadrez) em 2016

O fator de ramificação do Xadrez é ~30 enquanto que no Go é ~300

Monte Carlo Tree Search (MCTS)

O MCTS vai explorando a árvore de pesquisa associada ao jogo, registando os nós folha que correspondem a vitórias ou derrotas dos oponentes e retropropagando esses estados para contagem nos nós antecessores



Monte Carlo Tree Search (MCTS)

1-Seleção

Selecionar os bons nós "filho", a começar pelo nó raiz R, que representem estados que dão uma perspetiva maior de vitória

2-Expansão

Se *L* não for um nó terminal (ou seja, não finaliza o jogo), então criar um ou mais nós filho e selecionar um (*C)*.

3-Simulação

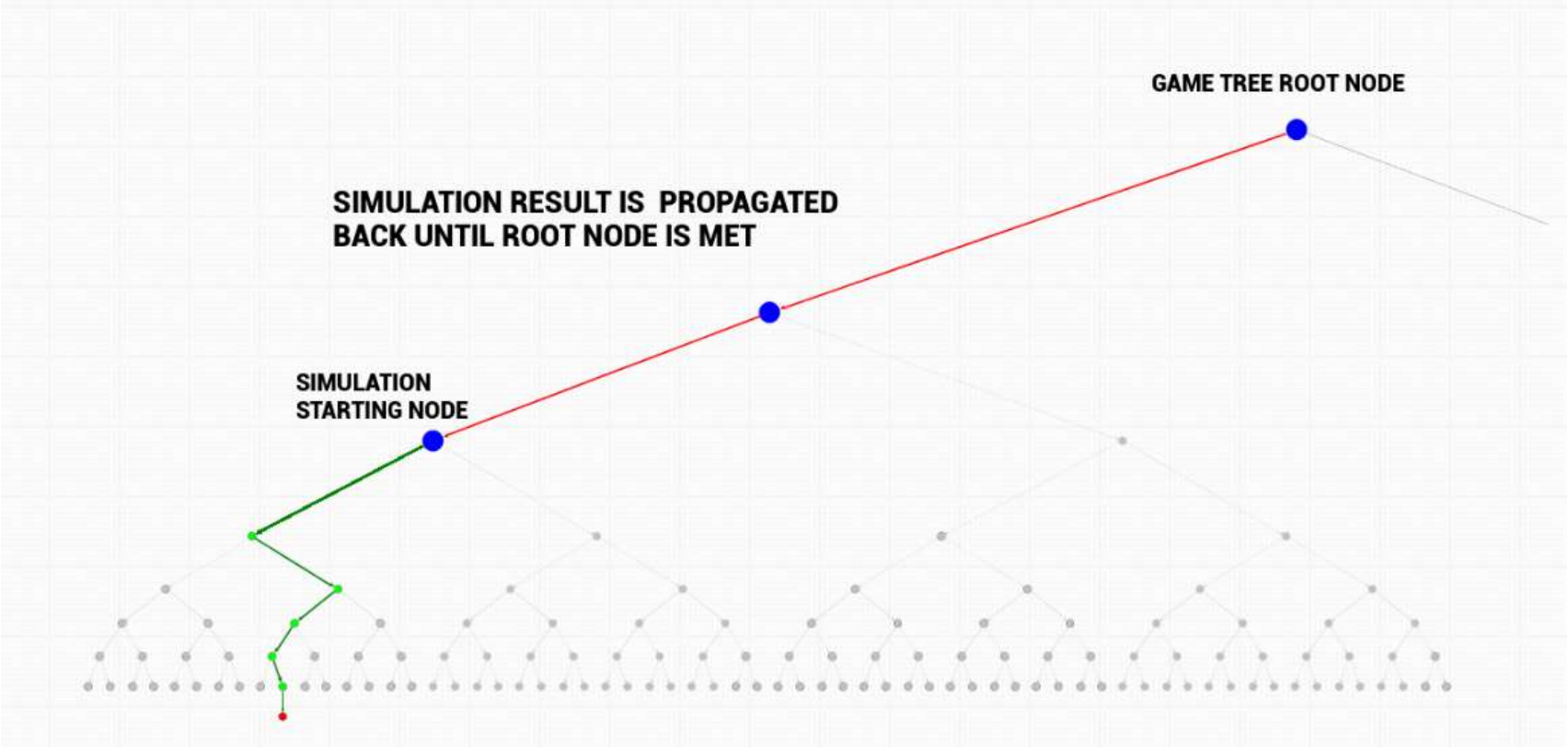
Simular jogadas a partir de *C* até atingir uma solução final para o jogo

4-Retropropagação

Atualizar a sequência de movimentos com o resultado da simulação

Monte Carlo Tree Search (MCTS)

Na fase inicial opta-se por um descendente e segue-se, a partir de um determinado ponto faz-se uma análise mais exaustiva (simulação) e depois faz-se a retropropagação



Monte Carlo Tree Search (MCTS)

Algumas notas:

- Nem todas as alternativas vão ser exploradas, o método é concebido para lidar com jogos complexos
- A ideia é no final optar por uma jogada tendo em conta o caminho que leva a um maior diferencial entre vitórias e derrotas na parte da árvore que foi explorada
- Devemos ter também em conta o modo como a subárvore correspondente à jogada foi mais ou menos explorada, quanto mais explorada estiver maior será a segurança na jogada que efetuarmos
- A retropropagação vai influenciar os nós antecessores dos que levam a vitórias e derrotas

5– Métodos de Pesquisa
Algoritmos Genéticos

Carlos Ramos
csr@isep.ipp.pt

Unidade Curricular: Algoritmia Avançada
Aulas Teóricas
Curso: Licenciatura em Engenharia Informática (ISEP)

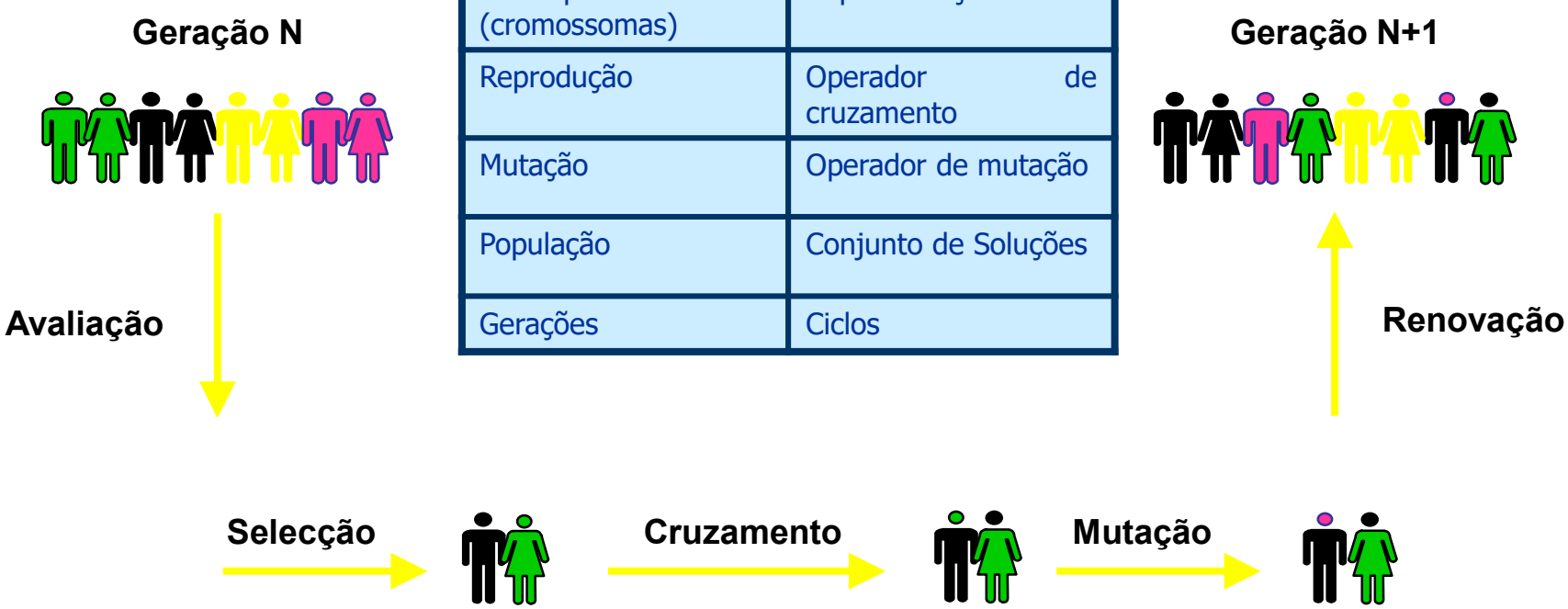
Algoritmos Genéticos

- No início da década 70 John Holland desenvolveu pesquisas pretendendo verificar se aspectos da evolução natural poderiam ser incorporados em algoritmos que permitissem a resolução automática de problemas
- Segundo John Holland, cada solução para um problema poderia ser entendida como um indivíduo de uma população, sendo assim todos os indivíduos da população equivaleriam a um conjunto de soluções possíveis para um problema, cada indivíduo seria representado pelo seu genótipo, ou seja, os indivíduos seriam representados pelos cromossomas
- Tais populações evoluiriam geralmente com base no processo de reprodução, admitindo-se que ocasionalmente pudessem ocorrer mutações
- Desse modo a população ia renovando-se através de gerações, a que corresponderiam ciclos do algoritmo
- O processo repetia-se até que chegássemos a um indivíduo (solução) que apresentasse as características mínimas pretendidas ou até que um limite temporal impusesse o final do algoritmo

Algoritmos Genéticos

Aspectos da evolução natural podem ser incorporados em algoritmos que permitem a resolução automática de problemas

Evolução Natural	*Algoritmos Genéticos*
Indivíduo	Solução
Genótipo (cromossomas)	Representação
Reprodução	Operador de cruzamento
Mutação	Operador de mutação
População	Conjunto de Soluções
Gerações	Ciclos



Algoritmos Genéticos

- O operador de cruzamento combina as características de dois indivíduos (progenitores) de modo a gerar dois novos indivíduos (descendentes), obtidos por composição do material genético dos seus progenitores
- O operador de mutação consiste na alteração arbitrária de 1 ou mais genes de um cromossoma, com uma probabilidade arbitrária. Este operador garante a existência de alguma diversidade genética na população, pretendendo prevenir a convergência prematura para óptimos locais, através da amostragem aleatória de novos pontos no espaço de pesquisa
- Efectuando ciclicamente cruzamentos e mutações e seleccionando os "indivíduos" que vão para a geração seguinte, implementamos os Algoritmos Genéticos

Operador de Cruzamento

- O operador principal dos Algoritmos Genéticos é o operador de cruzamento
- Dois indivíduos progenitores são escolhidos de entre os indivíduos da população através de um método de selecção bem definido
- Quando os cromossomas são representados apenas por sequências (por exemplo, de bits, números ou caracteres), o operador de cruzamento produz geralmente os dois descendentes pela escolha de um ou mais pontos de corte nos cromossomas dos progenitores e depois efectua a mistura das partes resultantes para gerar cada um dos cromossomas dos descendentes
- Alternativamente, os descendentes podem ser gerados através de máscaras que indicam se o gene de cada posição deve ser tomado de um dos progenitores ou do outro
- Há portanto 3 alternativas básicas de cruzamento:

  cruzamento num ponto de corte

  cruzamento em dois pontos de corte

  cruzamento uniforme (com máscara)

Operador de Cruzamento

- O operador de cruzamento num ponto faz a selecção aleatória de uma posição de corte
- O cromossoma do descendente D1 é obtido pela parte do cromossoma que está antes do ponto de corte no progenitor P1 e pela parte do cromossoma do progenitor P2 que está depois do ponto de corte
- O cromossoma do descendente D2 é obtido pela parte do cromossoma que está antes do ponto de corte no progenitor P2 e pela parte do cromossoma do progenitor P1 que está depois do ponto de corte

Operador de Cruzamento

Vejamos um exemplo considerando a codificação binária para os cromossomas dos progenitores P1 e P2:

P1 = 10110010

P2 = 11010110

Se considerarmos o ponto de corte entre o 5º e o 6º genes (ou bit, da esquerda para a direita) então os cromossomas dos progenitores ficam cortados da seguinte maneira:

P1 = 10110-010

P2 = 11010-110

Originando os seguintes descendentes:

D1= 10110-110 = 10110110

D2= 11010-010 = 11010010.

Operador de Cruzamento

- O operador de cruzamento em dois pontos de corte selecciona aleatoriamente dois pontos de corte, dividindo os cromossomas de cada progenitor em 3 partes
- O cromossoma do primeiro descendente é obtido pelas partes extremas do cromossoma do primeiro progenitor e pela parte central do cromossoma do segundo progenitor
- O cromossoma do segundo descendente é obtido pelas partes extremas do cromossoma do segundo progenitor e pela parte central do cromossoma do primeiro progenitor

Operador de Cruzamento

Vamos reconsiderar os progenitores P1 e P2 da secção anterior:

P1 = 10110010

P2 = 11010110

Admita-se que os pontos de corte situam-se agora entre o 2º e 3º genes e entre o 5º e 6º genes teremos os seguintes cortes nos progenitores:

P1 = 10-110-010

P2 = 11-010-110

Agora os descendentes são obtidos da seguinte forma:

D1 = 10-010-010 = 10010010

D2 = 11-110-110 = 11110110.

Operador de Cruzamento

Poderíamos estender o conceito anterior para mais pontos de corte (3, 4, etc)

Tal é possível com o operador de cruzamento uniforme que vai gerar aleatoriamente as posições dos genes que serão trocadas, o que corresponde a gerar uma máscara binária de igual dimensão a dos cromossomas, significando o 1 que o gene deve ser trocado na posição respectiva e o 0 que o gene não deve ser trocado.

Operador de Cruzamento

Vamos tomar novamente os progenitores P1 e P2:

P1 = 10110010

P2 = 11010110

Se a máscara gerada aleatoriamente for a seguinte:

M = 01110110

obteremos os seguintes descendentes:

D1 = 1**101**0**11**0

D2 = 1**011**0**01**0

Operador de Cruzamento

- Outro operador de cruzamento é o "Order Crossover"

- Vamos considerar o seguinte problema: temos um conjunto de 9 tarefas que podem ser executadas em qualquer ordem numa máquina. A função de avaliação poderia ter a ver com questões de optimização do tempo de processamento considerando tempos de espera para trocas de ferramentas na máquina, mas isso é apenas um detalhe não importante para o caso presente

- A codificação dos cromossomas poderia ser feita por sequências de 9 dígitos do seguinte alfabeto: {1,2,3,4,5,6,7,8,9}

Operador de Cruzamento

Dois potenciais progenitores poderiam ser os seguintes:

Pa = 247139865

Pb = 753218649

Se por ventura adoptássemos, por exemplo, o cruzamento em 2 pontos (entre o 2º e 3º genes e entre o 6º e 7º genes) teríamos os seguintes pontos de corte:

Pa = 24-7139-865

Pb = 75-3218-649

se efectuássemos o cruzamento especificado na secção com 2 pontos de corte teríamos os seguintes descendentes:

Da = 24-3218-865 = 243218865

Db = 75-7139-649 = 757139649.

Operador de Cruzamento

- Por essa razão surgem operadores de cruzamento como o operador de cruzamento com manutenção de ordem (Order Crossover) onde os genes a substituir de um dos progenitores aparecerão na ordem em que estão no outro progenitor. Essa troca poderá proceder-se a partir do primeiro ou do segundo ponto de corte

- Considerando L como a localização de um gene livre e admitindo que a parte central se mantém, teremos para o exemplo citado as seguintes sequências:

  Sequência obtida a partir de Pa = LL-7139-LLL

  Sequência obtida a partir de Pb = LL-3218-LLL

Operador de Cruzamento

Sequência a partir do 2º ponto de corte de Pb = 649753218

Sequência a partir do 2º ponto de corte de Pa = 865247139.

Vamos agora eliminar os elementos conhecidos na parte central do cromossoma de Pa (7139) na sequência a partir do 2º ponto de corte de Pb e os elementos conhecidos na parte central do cromossoma de Pb (3218) na sequência a partir do 2º ponto de corte de Pa, originando as seguintes sequências

Sequência a partir do 2º ponto de corte de Pb eliminando 7139 = 649753218 = 64528

Sequência a partir do 2º ponto de corte de Pa eliminando 3218 = 865247139 = 65479

Tais sequências vão preencher na ordem especificada as posições livres (L) anteriormente referidas, a partir do segundo ponto de corte, originando os descendentes Da e Db que se seguem:

Da = 28-7139-645 = 287139645

Db = 79-3218-654 = 793218654

Operador de Mutação

- O operador de mutação usado nos Algoritmos Genéticos tem o objectivo de trazer de volta para a população os genes perdidos durante o processo de selecção, de modo a que possam ser testados num novo contexto. Serve ainda para proporcionar novos genes que não estavam originalmente na população inicial
- Geralmente, a mutação é efectuada em apenas 1 gene, embora possa ser feita sobre mais genes. Na posição escolhida aleatoriamente o gene do cromossoma é substituído por um outro gene viável para aquela posição, oriundo do património genético da população (conjunto de todos os genes que poderão aparecer na população)

Operador de Mutação

Considere-se, por exemplo, o descendente D:

D = 10110010

Se o cromossoma deste descendente for sujeito a uma mutação no 5º gene, que continha um 0, o gene é substituído por um 1 (visto que não existe nenhuma outra alternativa de substituição dado que estamos a tratar de sequências binárias). Sendo assim a nova versão do descendente será:

D' = 1011**1**010

Operador de Mutação

Considere-se agora o descendente Da:

Da = 287139645

Se aplicássemos a mutação a um dado gene (por exemplo, o 2º gene que tem o valor 8) e originássemos o conjunto dos genes que poderiam aparecer nessa posição obteríamos o seguinte conjunto: {1,2,3,4,5,6,7,9}.

Observe-se que da mutação do gene da 2ª posição resultaria sempre uma solução não viável, onde o 8 não apareceria e uma tarefa estaria repetida.

Por essa razão a mutação aqui deveria ser efectuada em 2 genes, trocando os respectivos conteúdos. Por exemplo, se trocarmos o 2º e 5º gene de Da obteremos a seguinte mutação:

Da' = 2**3**71**8**9645

Algoritmos Genéticos

Um algoritmo genético para um determinado problema deve conter os seguintes componentes:

- uma representação genética para soluções potenciais do problema;
- uma forma de gerar uma população inicial de soluções potenciais;
- uma função de avaliação que desempenha o papel do ambiente, classificando soluções em função da sua aptidão;
- operadores genéticos que alteram a composição da descendência;
- valores para os diversos parâmetros que o algoritmo genético utiliza (tamanho da população, probabilidades de aplicação dos operadores genéticos, etc)

Algoritmos Genéticos

O algoritmo que se segue sumaria o funcionamento de um algoritmo genético típico:

Gerar população inicial de indivíduos

Enquanto o critério de paragem não for atendido

- *Avaliar a aptidão de todos os indivíduos*
- *Seleccionar os indivíduos mais aptos para reprodução*
- *Gerar novos indivíduos através do cruzamento e avaliar a respectiva aptidão*
- *Aplicar a mutação a alguns indivíduos*
- *Gerar nova população através da inserção de alguns bons indivíduos e eliminação de maus indivíduos*

Fim

Exemplo do Máximo de uma função

Este primeiro exemplo é simples, mas permitirá verificarmos como pode evoluir um algoritmo genético.

Vamos considerar a função $f(x)=x^2$, com x inteiro e pertencente ao intervalo [0,63]. O objectivo será maximizar o valor da respectiva função.

A codificação das soluções por uma string com 6 dígitos binários é adequada.

Vamos considerar o seguinte quadro que nos dá os elementos da população e o valor da função de avaliação (igual a f(x)).

Exemplo do Máximo de uma função

Elemento nº	Geração 1	Solução x	f(x)
1	100101	37	1369
2	001001	9	81
3	010011	19	361
4	001100	12	144

Exemplo do Máximo de uma função

Vejamos o que acontece após o 1º cruzamento:

Elemento nº	População após cruzamento	Solução x	f(x)
1	100101	37	1369
2	001001	9	81
3	010011	19	361
4	001100	12	144
5 - cruz.1,2	100001	33	1089
6 - cruz.1,2	001101	13	169
7 - cruz.3,4	000110	6	36
8 - cruz.3,4	011001	25	625

Vamos fazer os cruzamento usando o cruzamento uniforme com uma máscara M = 010101

Exemplo do Máximo de uma função

São então seleccionados os elementos 1, 3, 5 e 8, que são renumerados, originando a geração 2 que se segue:

Elemento nº	Geração 2	Solução x	f(x)
1	100101	37	1369
2	010011	19	361
3	100001	33	1089
4	011001	25	625

Exemplo do Máximo de uma função

Vamos repetir o ciclo de cruzamento mais uma vez:

Elemento nº	População após cruzamento	Solução x	f(x)
1	100101	37	1369
2	010011	19	361
3	100001	33	1089
4	011001	25	625
5 - cruz.1,2	110001	49	2401
6 - cruz.1,2	000111	7	49
7 - cruz.3,4	110001	49	2401
8 - cruz.3,4	001001	9	81

Exemplo do Máximo de uma função

Agora são seleccionados os elementos 1, 3, 4 e 5, que originam a geração nº 3:

Elemento nº	Geração 3	Solução x	f(x)
1	100101	37	1369
2	100001	33	1089
3	011001	25	625
4	110001	49	2401

Exemplo do Máximo de uma função

Não iremos continuar exemplificando as gerações seguintes. Vamos apenas observar alguns aspectos:

- Em primeiro lugar, e após 2 ciclos de cruzamento foi possível verificar que a população tem factores de avaliação superiores que a população original e que o melhor elemento da população tem um factor de avaliação superior
- Quererá isso dizer que rapidamente convergiremos para a solução óptima?
  - Infelizmente, esse não é o caso. Se continuarmos a aplicar apenas cruzamentos a partir da geração 3, até poderemos garantir que a melhor solução nunca será encontrada. A melhor solução é 111111 e note-se que no 5º bit, da esquerda para a direita todos os elementos da população têm o valor 0

Exemplo do Sequenciamento de Tarefas numa Máquina

- Considere-se o sequenciamento de 9 tarefas {1,2,3,4,5,6,7,8,9} numa máquina, onde as tarefas impares são executadas por uma ferramenta *fi* e as tarefas pares são executadas com uma ferramenta *fp*. O objectivo é minimizar o número de trocas de ferramentas
- A codificação será feita por uma sequência de 9 dígitos e a função de avaliação procurará minimizar o número de trocas de ferramentas, ou seja, contará o número de passagens de par para impar e de impar para par na sequência, sendo que agora se escolherão os elementos da população com menor valor dessa função
- Aqui aconselha-se a utilização de um operador de cruzamento com manutenção de ordem, pois os outros 3 operadores de cruzamento originariam soluções não viáveis (com tarefas repetidas e com tarefas ausentes)

Exemplo do Sequenciamento de Tarefas numa Máquina

Elemento nº	Geração 1	f(x)
1	394165287	6
2	821397465	3
3	136974285	4
4	754381296	7

Vamos aplicar o mesmo operador da secção 6.1.4, ou seja, o operador de cruzamento com manutenção de ordem e 2 pontos de corte (entre o 2º e 3º genes e entre o 6º e 7º genes). Novamente vamos cruzar o elemento 1 com o 2 e o elemento 3 com o 4.

Exemplo do Sequenciamento de Tarefas numa Máquina

Após o cruzamento teremos:

Elemento nº	População após cruzamento	f(x)
1	39-4165-287	6
2	82-1397-465	3
3	13-6974-285	4
4	75-4381-296	7
5 - cruz.1,2	97-4165-823	6
6 - cruz.1,2	65-1397-284	2
7 - cruz.3,4	81-6974-253	5
8 - cruz.3,4	97-4381-256	7

Exemplo do Sequenciamento de Tarefas numa Máquina

Por elitismo, chegamos à geração 2 que se indica de seguida:

Elemento nº	Geração 2	f(x)
1	821397465	3
2	136974285	4
3	651397284	2
4	816974253	5

Parametrização de um Algoritmo Genético

Os principais parâmetros de um Algoritmo Genético são:

- o tamanho da população
- a taxa de cruzamento
- a taxa de mutação
- a taxa de substituição
- os critérios de paragem

Parametrização de um Algoritmo Genético

O tamanho da população condiciona a qualidade da solução obtida e o tempo de processamento

- Se a população for muito pequena leva a que provavelmente tenhamos uma fraca diversidade das soluções que podem ser obtidas no espaço de pesquisa, geralmente chega-se rapidamente a uma situação na qual a população não é melhorada, a não ser que casualmente as mutações originem indivíduos mais interessantes
- Se a população for muito grande, temos uma melhor cobertura do espaço de pesquisa, e a probabilidade da população estabilizar um máximos locais é menor. Contudo, tal é conseguido com custos computacionais
- Geralmente, os algoritmos genéticos admitem que a população mantém-se constante, embora nada impeça a sua variação dinâmica, embora em termos práticos se deva ter cuidado em evitar a diminuição ou crescimento abrupto da população

Parametrização de um Algoritmo Genético

A Taxa de Cruzamento vai definir a probabilidade com a qual **1 individuo será** cruzado para gerar nova descendência

- Uma Taxa de Cruzamento baixa faz com que as gerações da população sejam pouco renovadas, fazendo com que a evolução seja muito lenta
- Uma Taxa de Cruzamento elevada origina uma grande renovação de geração para geração, o que se por um lado pode ser positivo para evitar o problema descrito anteriormente, pode também originar a necessidade de substituição de indivíduos com aptidão elevada
- Um valor típico para a taxa de cruzamento situa-se entre os 60% e 80%

Parametrização de um Algoritmo Genético

A mutação é uma operação importante pelo facto de poder gerar indivíduos com melhores aptidões, fugindo desse modo de máximos locais do problema. Dito de outro modo, com a mutação podemos introduzir um factor adicional de diversidade na população

- Uma taxa de mutação muito elevada conduz a uma pesquisa aleatória que poderá não originar convergência, visto que o mais provável numa mutação é até gerar indivíduos com pior aptidão
- Os valores típicos da taxa de mutação são bastante reduzidos (por exemplo, cerca de 1%)

Parametrização de um Algoritmo Genético

A Taxa de Substituição define a percentagem de população que será substituída em cada transição de gerações

- Um valor muito elevado leva a que possa ocorrer uma renovação excessiva da população enquanto que um valor muito baixo origina uma estagnação da população

Parametrização de um Algoritmo Genético

Geralmente adopta-se um dos seguintes critérios de paragem:

- quando se atinge uma solução com um determinado valor de aptidão (para o tipo de problemas em que se pretende encontrar uma solução com custo inferior a um dado valor mas sem necessidade de encontrar a melhor solução);
- quando se atinge um determinado limite temporal (para problemas que exigem soluções em tempo útil ou em tempo real);
- quando se atinge um determinado número de gerações

Avaliação e Aptidão

As noções de avaliação e de aptidão são por vezes intercambiáveis. No entanto, deve-se distinguir a noção de função de avaliação e função de aptidão, usadas nos Algoritmos Genéticos

- A Avaliação proporciona uma medida de desempenho em relação a um conjunto particular de parâmetros, sendo que, a avaliação de um dado indivíduo é independente dos outros indivíduos.
- A Aptidão transforma essa medida de desempenho na atribuição de oportunidades de sobrevivência e reprodução, sendo, portanto, definida em relação aos outros indivíduos da população
- O cálculo das funções de avaliação e de aptidão não deve ser muito complexo e demorado

Selecção

- A escolha dos indivíduos mais adaptados da população através de um operador de selecção determinístico pode parecer lógica
- Tal escolha, dita elitista, pode, no entanto, levar à redução da diversidade da população, fazendo com que uma boa solução obtida no principio se torne dominante e evitando a exploração adequada do espaço de pesquisa
- É importante que os indivíduos menos adaptados tenham probabilidade, mesmo que reduzida, de serem seleccionados

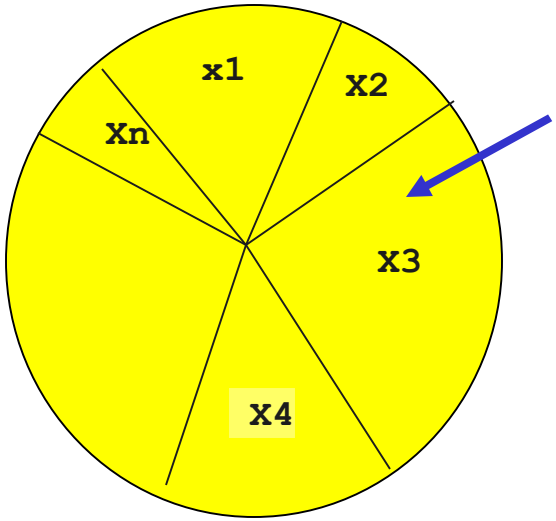
Selecção

Um dos métodos de selecção proporcional mais comuns é o método da roleta ("Roulette Wheel Parent Selection")

- segundo o qual a cada indivíduo *i* da população é atribuída uma probabilidade de selecção Ps(i), calculada através do quociente entre a sua avaliação e o somatório das avaliações de todos os indivíduos da população, ou por uma fórmula similar (por vezes subtrai-se ao numerador e denominador o mínimo valor obtido para a avaliação de um indivíduo da população)
- Segundo esse método, a selecção de indivíduos é realizada através da simulação de *n* rotações de uma roleta cujas fatias tivessem um ângulo proporcional ao resultado da fórmula referida anteriormente, constituindo assim a população pretendida. Ou seja, os indivíduos com maior valor de adaptação terão mais possibilidades de serem escolhidos, no entanto, nada garante que o sejam, nem nada implica que o elemento com menor valor de adaptação não possa ser escolhido

Seleção

Individuo *i* vai ter a seguinte probabilidade de ser escolhido $\frac{f(i)}{\sum_i f(i)}$



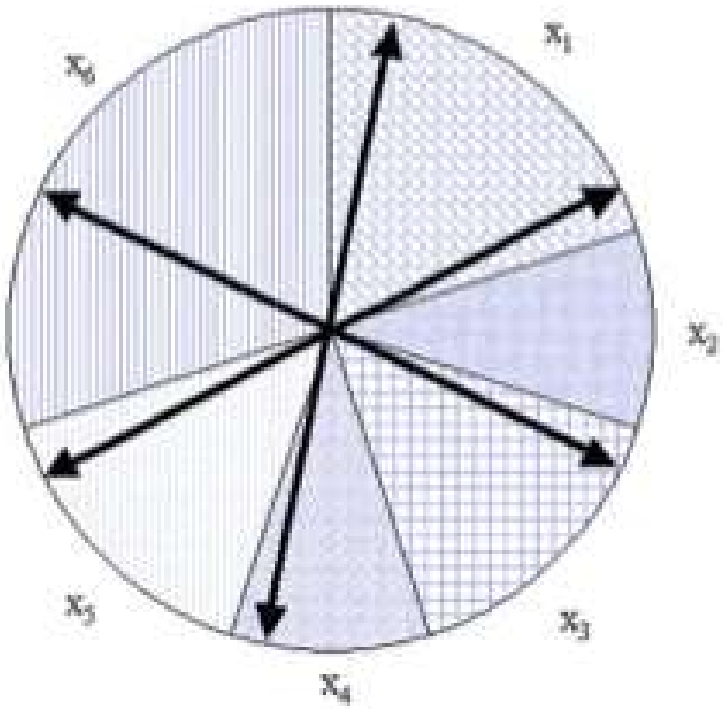
Area é proporcional ao valor da aptidão

Selecção

Além do método da roleta foram ainda estudados outros operadores de selecção proporcional, tais como, por exemplo, o Método de Selecção Universal Estocástica

- Rodamos a roleta apenas uma vez e temos um conjunto de "ponteiros" igualmente espaçados, os indivíduos cujas fatias estejam localizadas em tais ponteiros serão escolhidos.
- A evolução da população ao longo das gerações tende a uniformizar os valores de adaptação dos indivíduos, o que reduz a importância da selecção proporcional pois esta baseia-se no facto de existirem diferenças substanciais entre os valores de adaptação dos indivíduos

Método De Amostragem Estocástica Uniforme



Selecção

- O Método de Selecção por Torneios ("Tournament Selection") é um método de selecção por ordem de classificação que consiste na organização de *n* torneios entre 2 ou mais indivíduos da população, escolhidos aleatoriamente
- O indivíduo seleccionado é o que estiver em posição de superioridade no confronto directo entre os valores de adaptação

5– Métodos de Pesquisa

Primeiro o Melhor, Branch and Bound, A*

Carlos Ramos
csr@isep.ipp.pt

Unidade Curricular: Algoritmia Avançada
Aulas Teóricas
Curso: Licenciatura em Engenharia Informática (ISEP)

Primeiro o Melhor

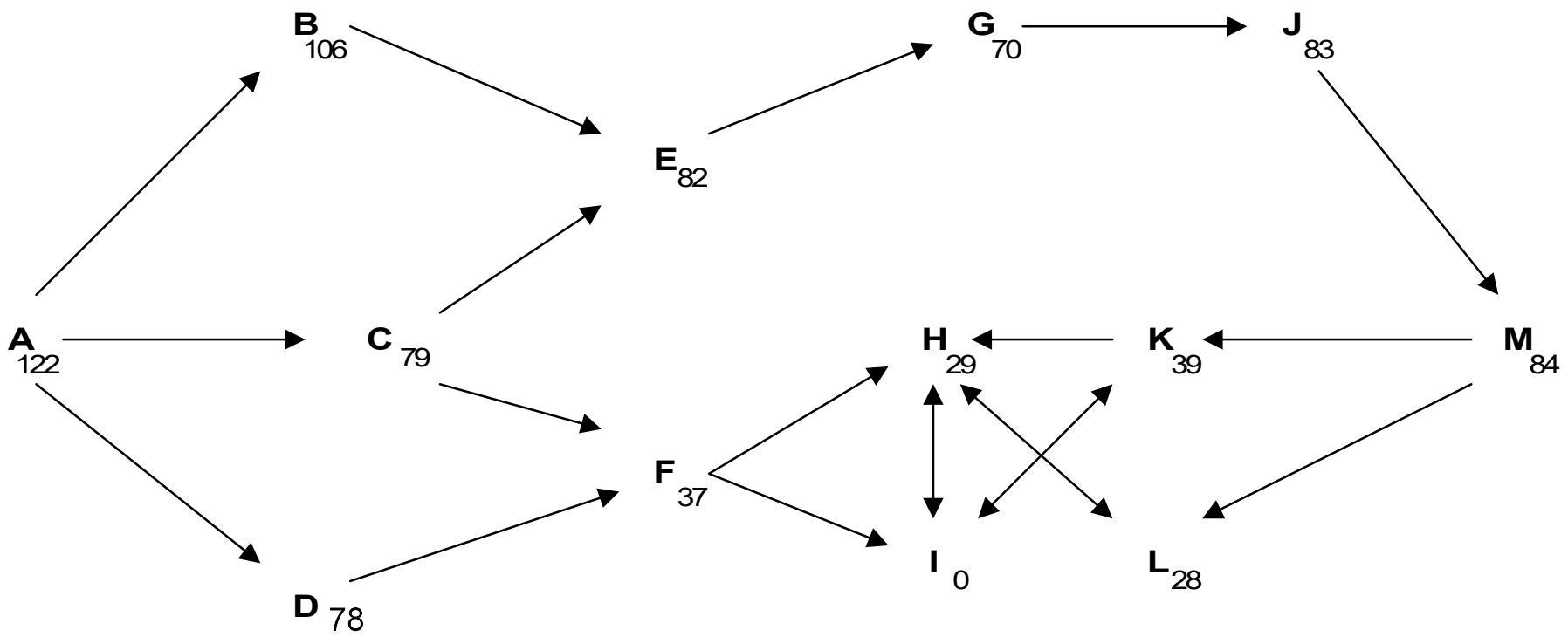
- Vamos descrever agora o primeiro método dito "informado" de pesquisa. O método "Primeiro o Melhor" assemelha-se ao Primeiro em Profundidade, a única diferença reside no facto da decisão sobre qual ramo seguir ser feita com base num critério de decisão local

- A ideia que está por trás é a de em caso de indecisão sobre qual caminho vamos seguir deve-se optar por aquele que pareça mais promissor. Trata-se de uma decisão feita localmente com base numa informação particular

- É então necessário dispor de um valor numérico que avalie o custo ou o ganho que se obtém pelo facto de seguir por um dado caminho

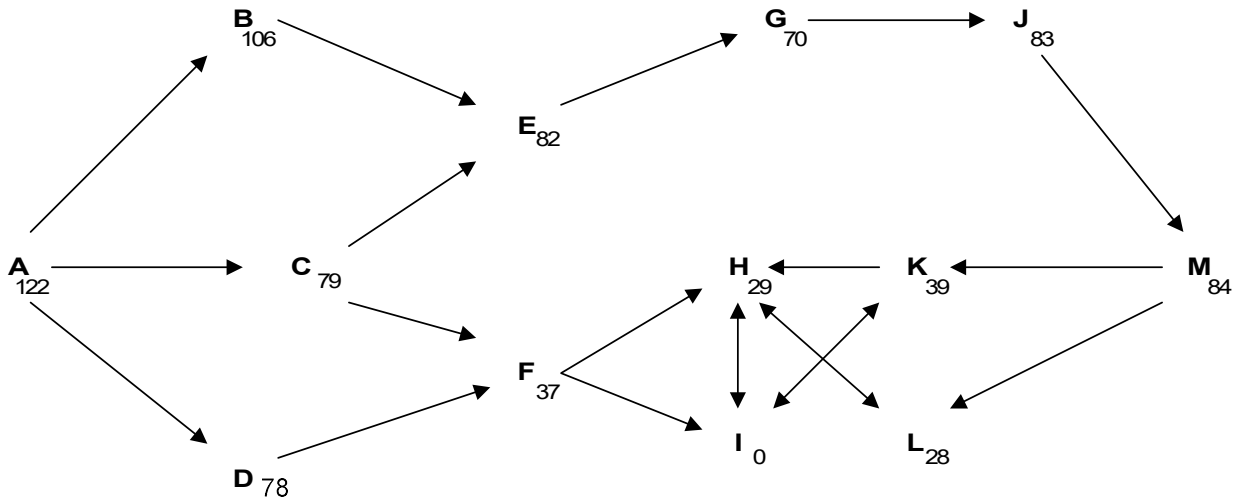
Primeiro o Melhor

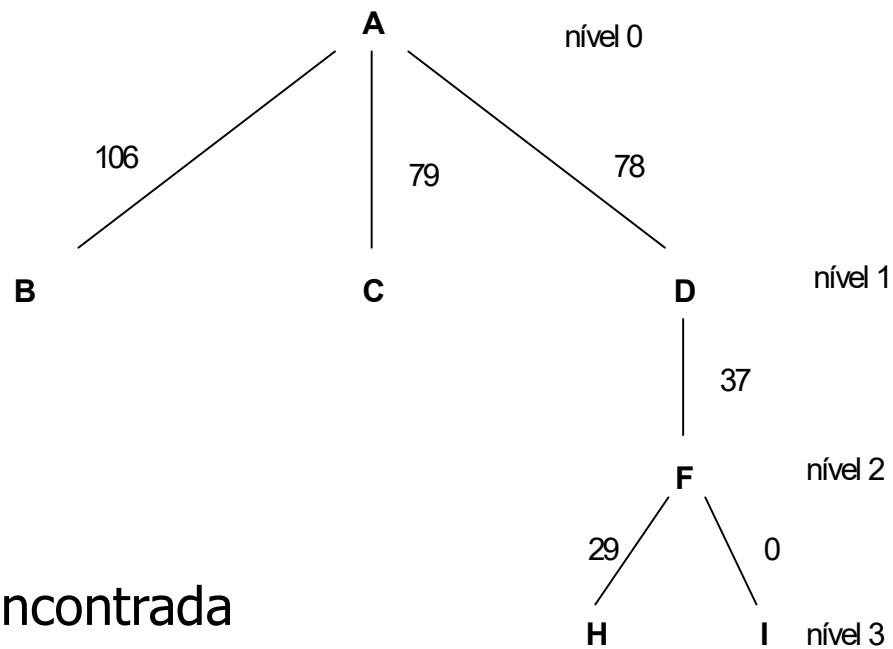
Na figura que se segue repete-se o grafo usado anteriormente, mas agora usamos estimativas do custo para ir desde cada uma das cidades até a cidade I (supostamente a cidade de destino)



Primeiro o Melhor

Vejamos a árvore que é gerada:





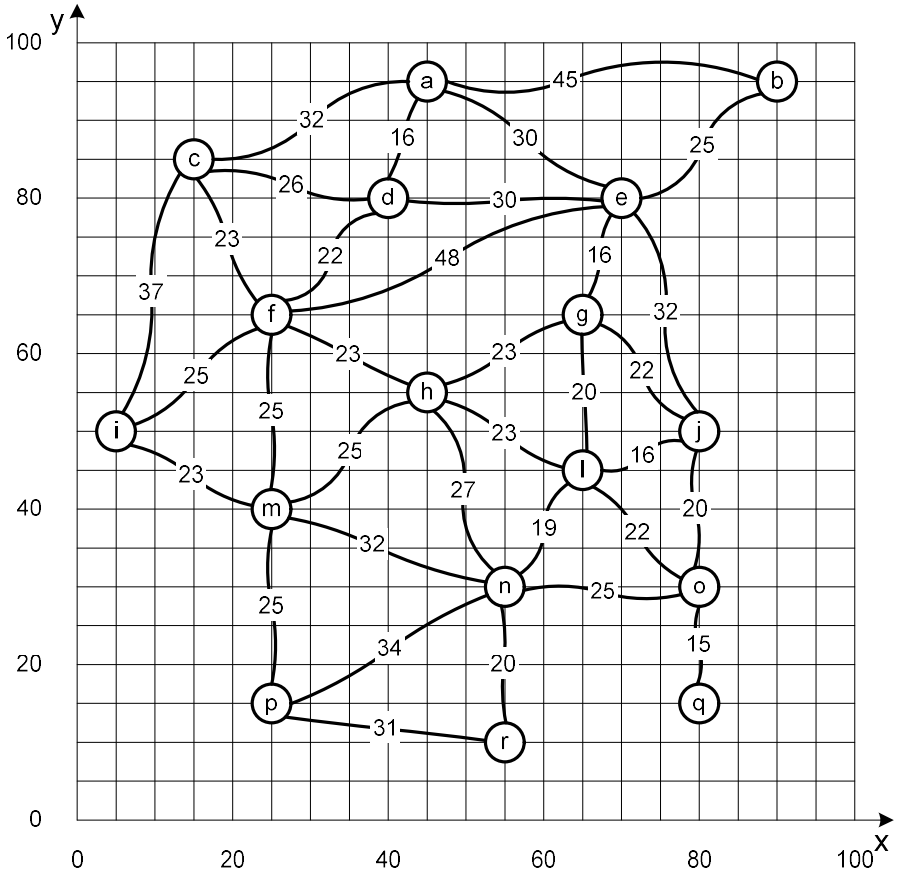
Nada garante que a melhor solução seja encontrada

Best First Search (BestFS) Implementação Prolog

```
bestfs(Orig,Dest,Cam):-
          bestfs2(Dest,[Orig],Cam).

%condicao final: destino = nó à cabeça do caminho actual
bestfs2(Dest,[Dest|T],Cam):- !,
          %caminho actual está invertido
          reverse([Dest|T],Cam).

bestfs2(Dest,LA,Cam):-
          LA=[Act|_],
          %calcular todos os nodos adjacentes nao visitados e
          % guardar um tuplo com estimativa e novo caminho
          findall((EstX,[X|LA]),
                    ((edge(Act,X);edge(X,Act)),
                      \+ member(X,LA), estimativa(X,Dest,EstX)),
                    Novos),
          %ordenar pela estimativa
          sort(Novos,NovosOrd),
          %extrair o melhor que está à cabeça
          NovosOrd = [(_,Melhor)|_],
          %chamada recursiva
          bestfs2(Dest,Melhor,Cam).
```

Best First Search (BestFS)
(Outro) Grafo exemplo

```
%node(Nodo,PosX,PosY)
node(a,45,95).
node(b,90,95).
…
%edge(Nodo1,Nodo2,Custo)
edge(a,b,45).
edge(a,c,32).
edge(a,d,16).
…
```



Best First Search (BestFS) – versão c/custos
Implementação Prolog / Soluções

```
?- bestfs(a,r,Cam,Custo).
Cam = [a, d, f, m, n, r],
Custo = 115.
```

```
bestfs(Orig,Dest,Cam,Custo):-
          bestfs2(Dest,(0,[Orig]),Cam,Custo).

bestfs2(Dest,(Custo,[Dest|T]),Cam,Custo):-
          !,
          reverse([Dest|T],Cam).

bestfs2(Dest,(Ca,LA),Cam,Custo):-
          LA=[Act|_],
          findall((EstX,CaX,[X|LA]),
          ((edge(Act,X,CX); edge(X,Act,CX)),
            \+member(X,LA),estimativa(X,Dest,EstX),
          CaX is Ca+CX),Novos),

          sort(Novos,NovosOrd),
          NovosOrd = [(_,CM,Melhor)|_],
          bestfs2(Dest,(CM,Melhor),Cam,Custo).
```

Best First Search (BestFS) Estimativa

```
estimativa(Nodo1,Nodo2,Estimativa):-
        node(Nodo1,X1,Y1),
        node(Nodo2,X2,Y2),
        Estimativa is sqrt((X1-X2)^2+(Y1-Y2)^2).
```

Best First Search (BestFS) – versão c/custos
Implementação Prolog / Soluções

```
?- bestfs(a,r,Cam,Custo).
Cam = [a, d, f, m, n, r],
Custo = 115.
```

```
bestfs(Orig,Dest,Cam,Custo):-
          bestfs2(Dest,(0,[Orig]),Cam,Custo).

bestfs2(Dest,(Custo,[Dest|T]),Cam,Custo):-
          !,
          reverse([Dest|T],Cam).

bestfs2(Dest,(Ca,LA),Cam,Custo):-
          LA=[Act|_],
          findall((EstX,CaX,[X|LA]),
          ((edge(Act,X,CX); edge(X,Act,CX)),
            \+member(X,LA),estimativa(X,Dest,EstX),
          CaX is Ca+CX),Novos),

          sort(Novos,NovosOrd),
          NovosOrd = [(_,CM,Melhor)|_],
          bestfs2(Dest,(CM,Melhor),Cam,Custo).
```

Esta versão só encontra o caminho se as opções forem para uma alternativa que tenha conectividade com a solução

Experimente cortar as ligações de m para n, n para p e p para r

Dito de outro modo se ocorrer uma falha não há retrocesso para tentar outra alternativa
Isso porque LA=[Act|_] só dá a possibilidade de escolher o elemento que está à cabeça na lista

Mas a solução é simples basta substituir LA=[Act|_] por member(Act,LA)

Branch and Bound

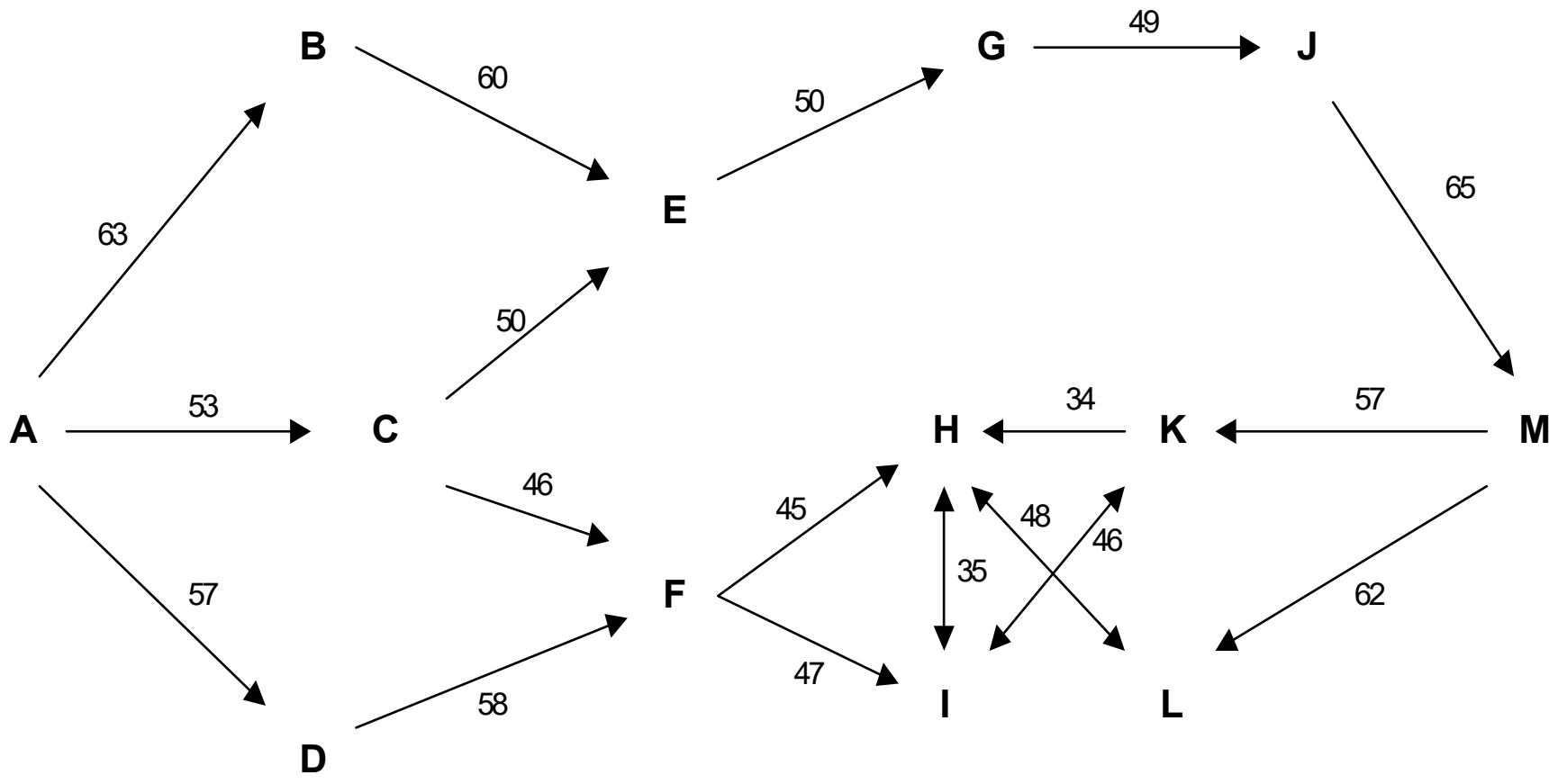
- O Branch and Bound é um método de pesquisa que corresponde ao método "Primeiro o Melhor" com avaliação de transições locais mas com possibilidade de alterar a qualquer momento o nó sobre qual se vai considerar a próxima expansão

- Quer isso dizer que o próximo nó que se expande não é obrigatoriamente um descendente do último nó expandido

- A comparação entre os nós candidatos à expansão é feita com base no valor acumulado do custo ou do ganho desde a raíz até esses nós

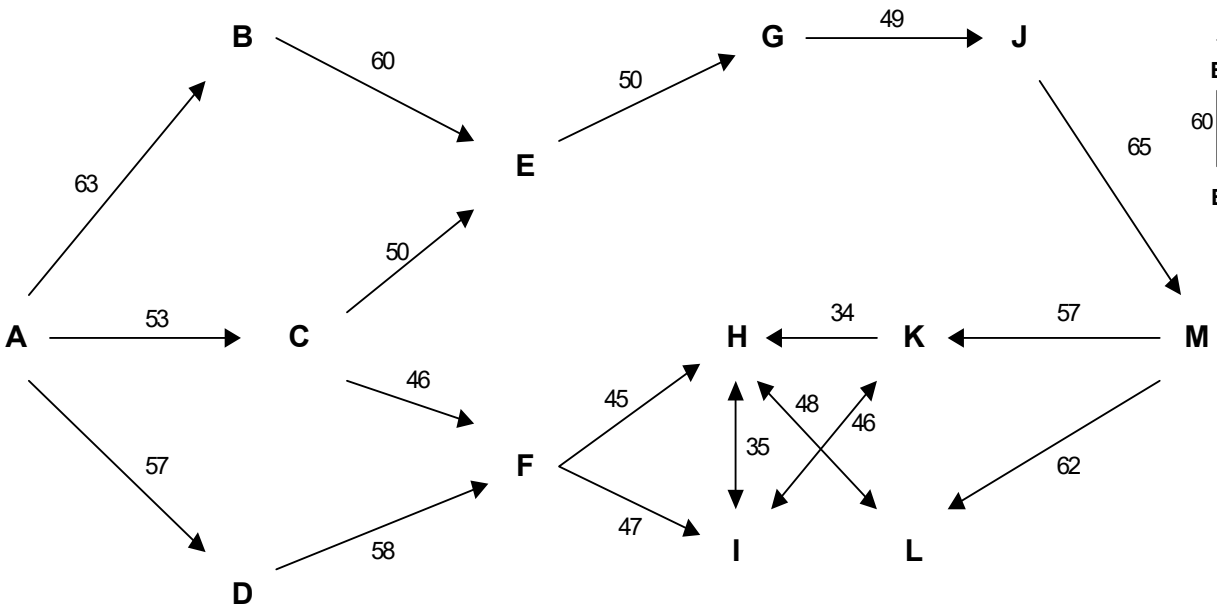
Branch and Bound

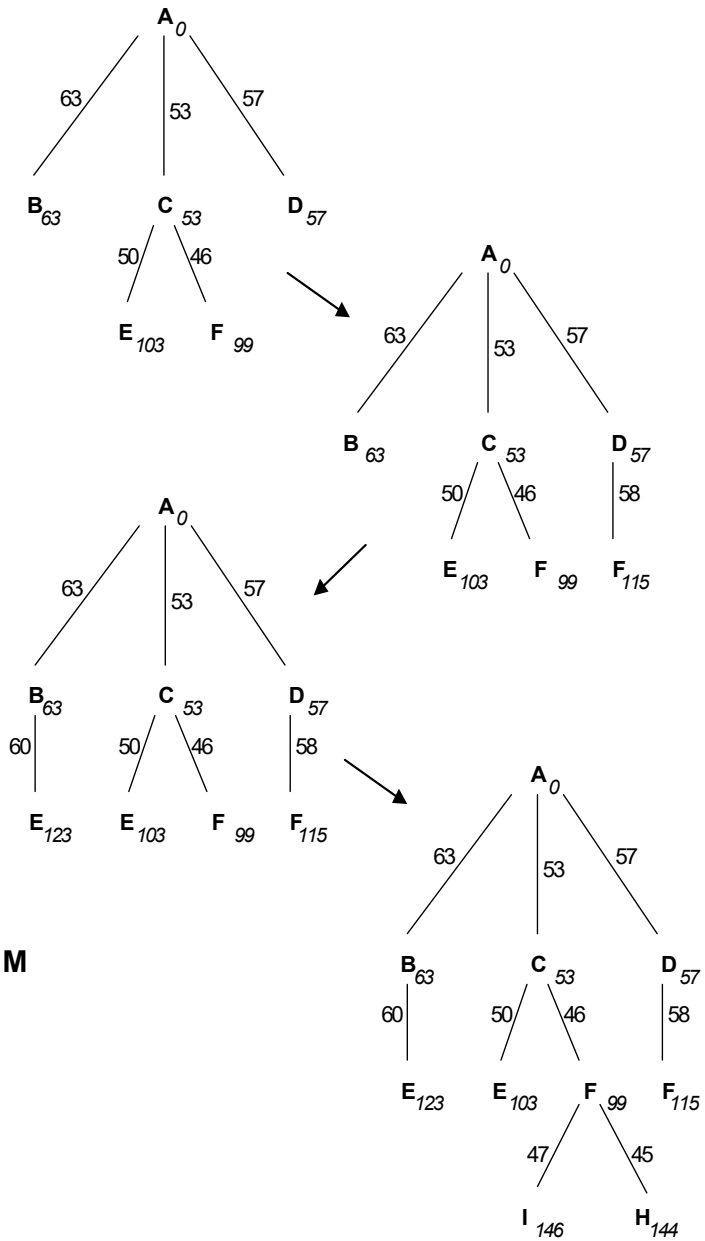
- Vejamos um exemplo de um grafo onde junto aos arcos aparecem as distâncias a serem percorridas entre 2 nós conectados



Branch and Bound

- Na figura ao lado ilustra-se a expansão da árvore de pesquisa usando o método Branch and Bound. Junto aos ramos estão os custos locais de transição enquanto que junto aos nós estão os custos acumulados desde a raíz até ao nó, e que podem ser obtidos pelo somatório dos custos dos ramos que conduzem desde a raiz até ao nó





Branch and Bound

- O principal inconveniente do Branch and Bound reside no facto de não ser sensível à distância que um dado nó se encontra da solução

- Será que se pode garantir que a primeira solução que se encontra usando o Branch and Bound é sempre a melhor solução?

- O que acontece se a solução estiver muito afastada da raiz, ou seja, estiver a um nível muito elevado?

Branch and Bound (Bnb)
Implementação Prolog

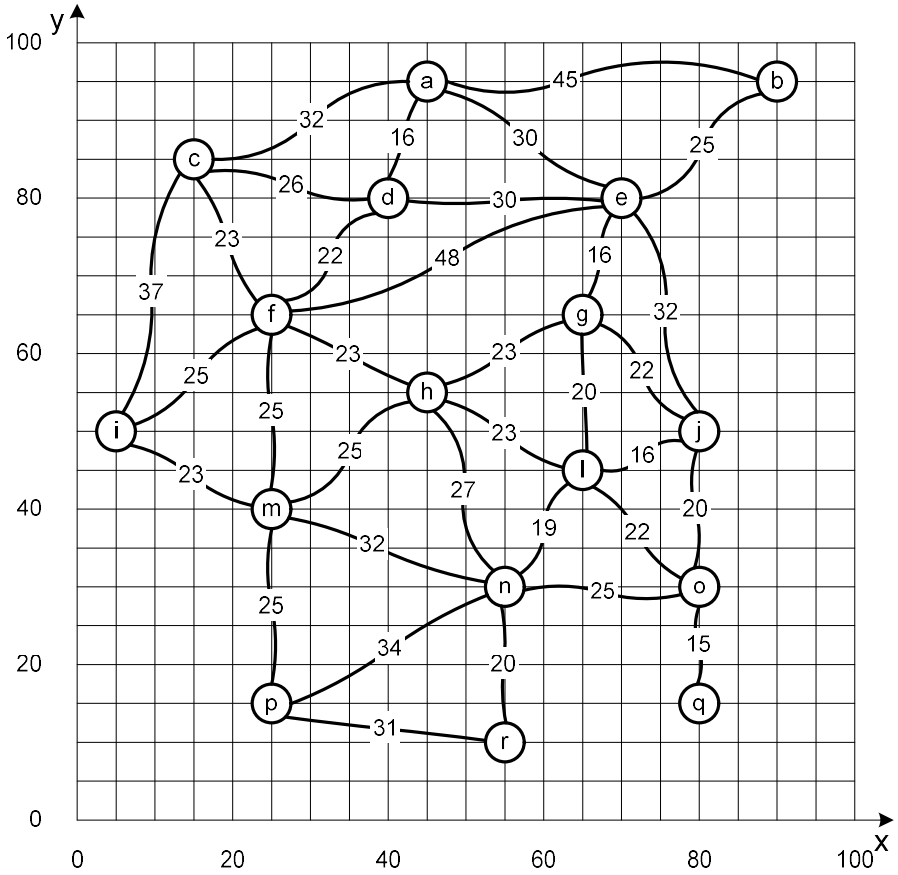
```
bnb(Orig,Dest,Cam,Custo):- bnb2(Dest,[(0,[Orig])],Cam,Custo).

%condicao final: destino = nó à cabeça do caminho actual
bnb2(Dest,[(Custo,[Dest|T])|_],Cam,Custo):-
          %caminho actual está invertido
          reverse([Dest|T],Cam).

bnb2(Dest,[(Ca,LA)|Outros],Cam,Custo):-
          LA=[Act|_],
          %calcular todos os nodos adjacentes nao visitados e
          %gerar tuplos c/ um caminho novo juntado o nodo + caminho actual
          %  o  custo  de  cada  caminho  é  o  custo  acumulado  +  peso  do  ramo
          findall((CaX,[X|LA]),
          (Dest\==Act, (edge(Act,X,CustoX); edge(X,Act,CustoX)),
           \+ member(X,LA),CaX is CustoX + Ca),Novos),
          %os  novos  caminhos  sao  adicionados  aos  caminhos  não  explorados
          append(Outros,Novos,Todos),
          %a ordenação (não sendo o mais eficiente) garante que
          % o melhor caminho fica na primeira posição
          sort(Todos,TodosOrd),
          %chamada recursiva
          bnb2(Dest,TodosOrd,Cam,Custo).
```

Branch and Bound (Bnb)
Implementação Prolog

```
bnb(Orig,Dest,Cam,Custo):- bnb2(Dest,[(0,[Orig])],Cam,Custo).

bnb2(Dest,[(Custo,[Dest|T])|_],Cam,Custo):-
       reverse([Dest|T],Cam).

bnb2(Dest,[(Ca,LA)|Outros],Cam,Custo):-
       LA=[Act|_],
       findall((CaX,[X|LA]),
       (Dest\==Act, (edge(Act,X,CustoX); edge(X,Act,CustoX)),
        \+ member(X,LA),
       CaX is CustoX + Ca),Novos),
       append(Outros,Novos,Todos),
       sort(Todos,TodosOrd),
       bnb2(Dest,TodosOrd,Cam,Custo).
```

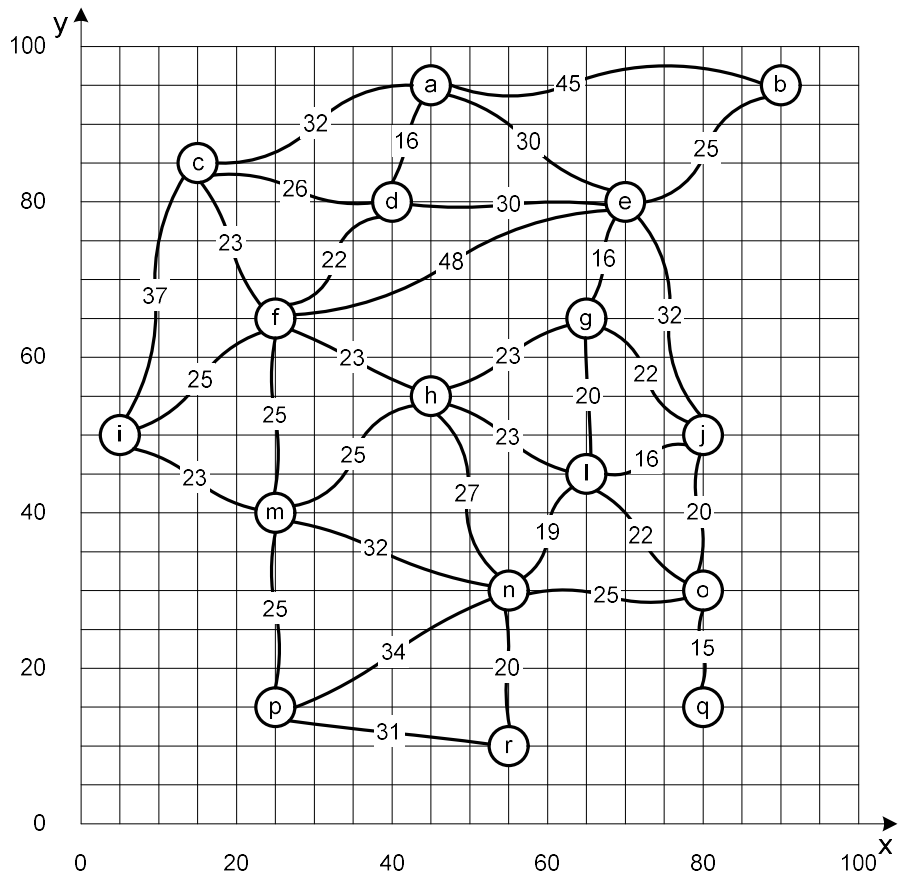
Branch and Bound (Bnb)
Grafo exemplo

```
%node(Nodo,PosX,PosY)
node(a,45,95).
node(b,90,95).
…
%edge(Nodo1,Nodo2,Custo)
edge(a,b,45).
edge(a,c,32).
edge(a,d,16).
…
```



Branch and Bound (Bnb)
Espaço do problema / soluções

```
?- bnb(a,r,Cam,Custo).
Cam = [a, e, g, l, n, r],
Custo = 105 ;
Cam = [a, d, f, h, n, r],
Custo = 108 ;
Cam = [a, d, f, m, n, r],
Custo = 115 ;
Cam = [a, e, g, h, n, r],
Custo = 116 ;
Cam = [a, e, j, l, n, r],
Custo = 117 ;
Cam = [a, d, f, m, p, r],
Custo = 119 ;
Cam = [a, d, e, g, l, n, r],
Custo = 121 ;
Cam = [a, d, f, h, l, n, r],
Custo = 123 ;
…
```



A*

- Interessa, portanto, juntar o que há de bom no Primeiro o Melhor, ou seja o uso de funções que estimam a distância à solução, com o que há de melhor no Branch and Bound, ou seja o uso de custos acumulados conhecidos e a possibilidade de comutar de um ponto para outro na árvore de pesquisa sem que o novo ponto seja um descendente do primeiro. No fundo isso corresponde ao método A*

- No A* consideramos uma função heurística f' tal que:
  - $f' = g + h'$
    - g é o custo conhecido para ir do estado inicial até ao estado do nó que se está a considerar, e que no momento é uma folha ainda não expandida
    - h' é uma estimativa do custo para ir desse nó até a solução, estando portanto sujeita a erro

A*

O que interessará que h' seja relativamente ao valor real?

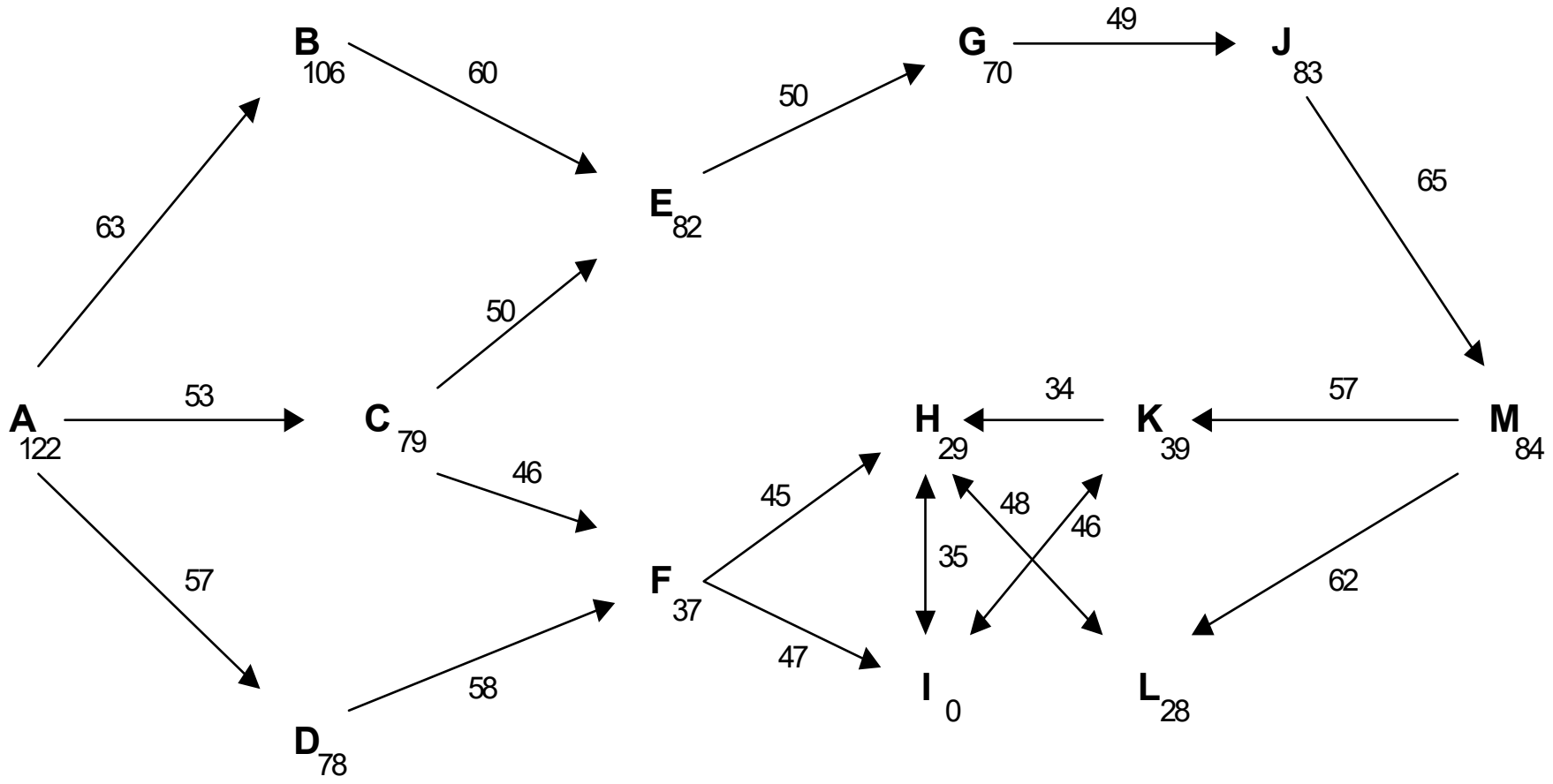
- Seja h' é uma estimativa da distância de um nó à solução no método de pesquisa A*, se estivermos a operar com custos convém que h' seja um minorante do custo real, ao passo que se estivermos a operar com lucros interessa que h' seja um majorante

- Porque?

- O que acontece se o minorante for 0?

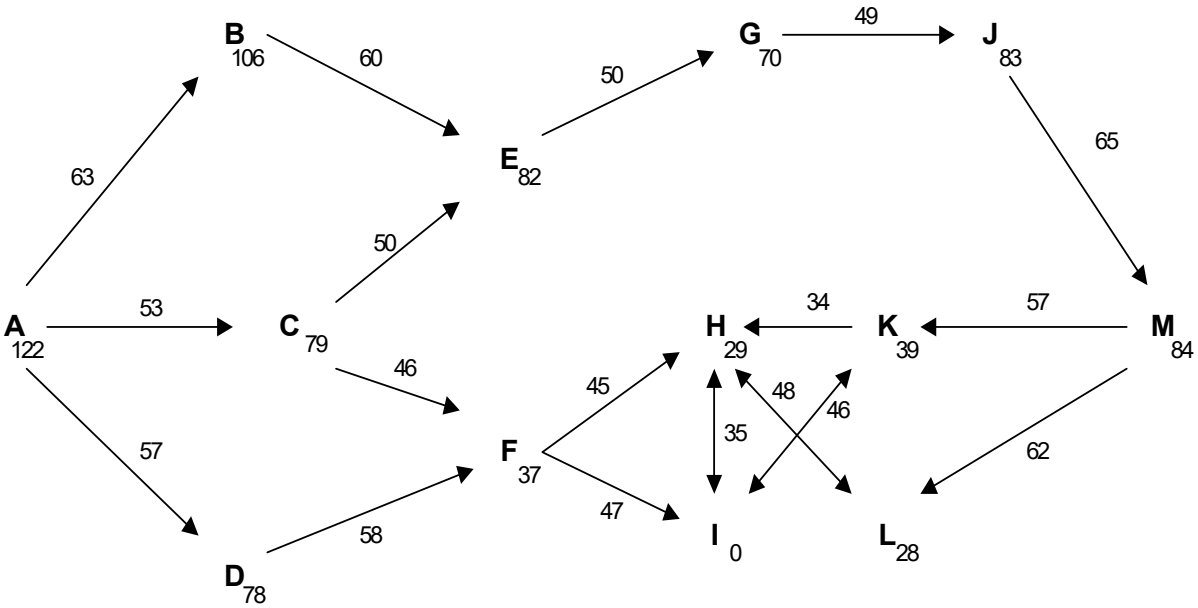
A*

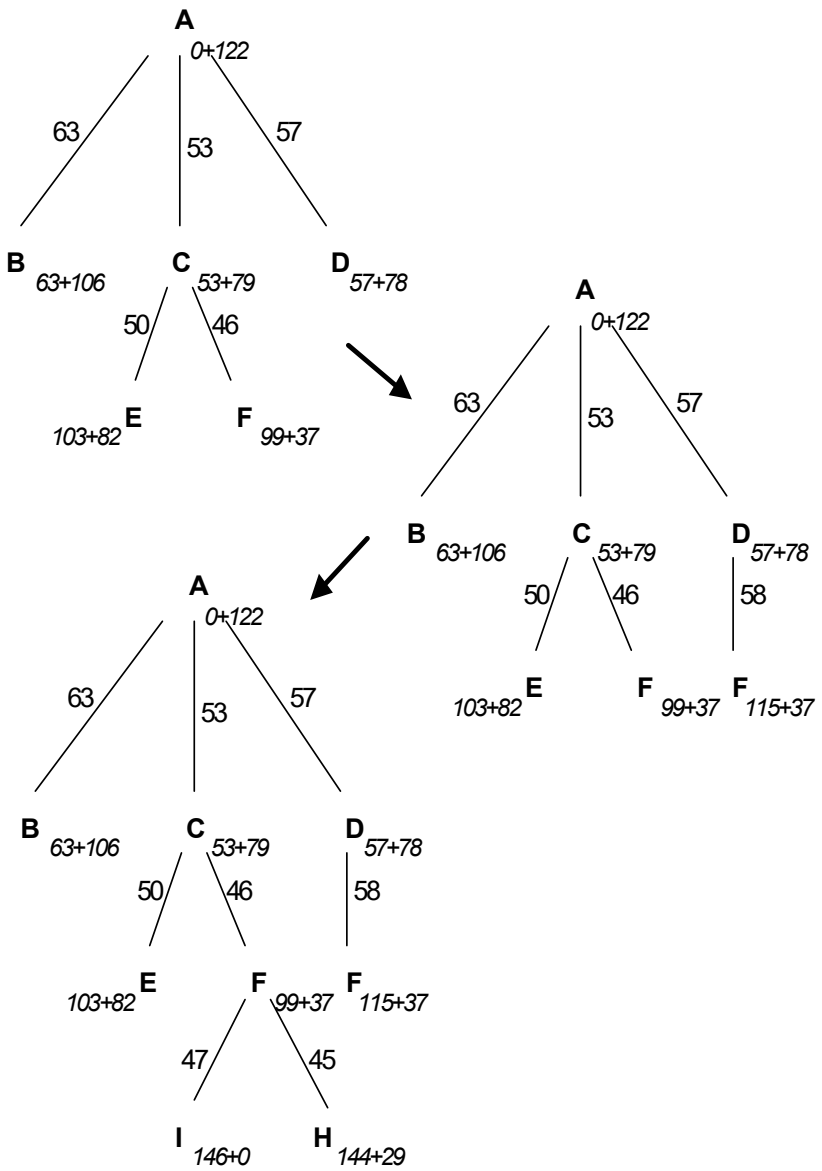
- Vejamos um exemplo de um grafo onde junto aos arcos aparecem as distâncias a serem percorridas entre 2 nós conectados e junto aos nós está a estimativa para ir desse nó até ao nó de destino (nó I)



A*

- Na figura ao lado ilustra-se a expansão da árvore de pesquisa usando o método A*. Junto aos ramos estão os custos locais de transição enquanto que junto aos nós estão as parcelas da soma g+h', onde a primeira parcela é o custo desde a raíz até ao nó (g) e a segunda parela é a estimativa (h') do custo desde o nó até o nó destino pretendido (nó I)





Domínios da IA
Língua Natural

Carlos Ramos
csr@isep.ipp.pt

Unidade Curricular: Algoritmia Avançada
Curso: Licenciatura em Engenharia Informática (ISEP)

Língua Natural

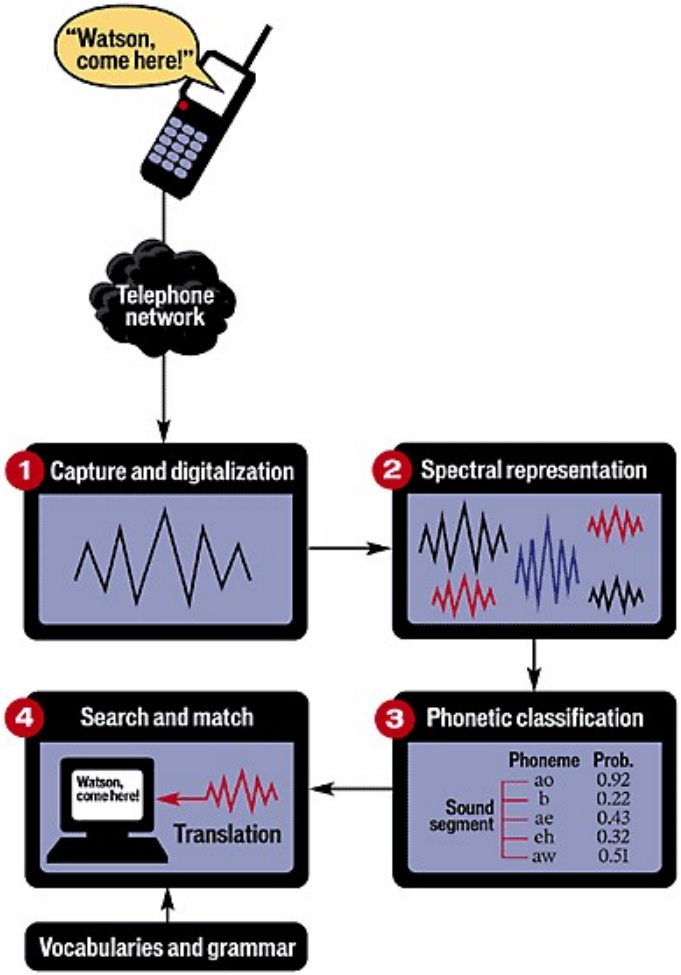
A Língual Natural, ou Linguagem Natural, é a área da Inteligência Artificial que se debruça sobre a compreensão textual, tarefa simples para os seres humanos, mas complicada para os sistemas computacionais

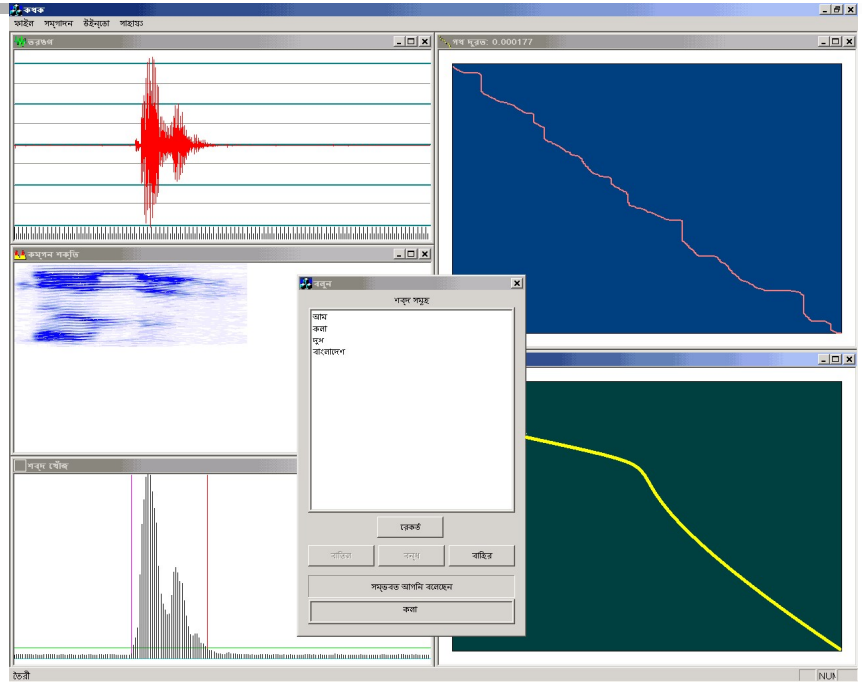
É entendido que a entrada de um sistema de Língua Natural é uma informação textual, onde já estão identificadas as palavras, por exemplo através de uma "string" ou lista de palavras. Vejamos um exemplo:

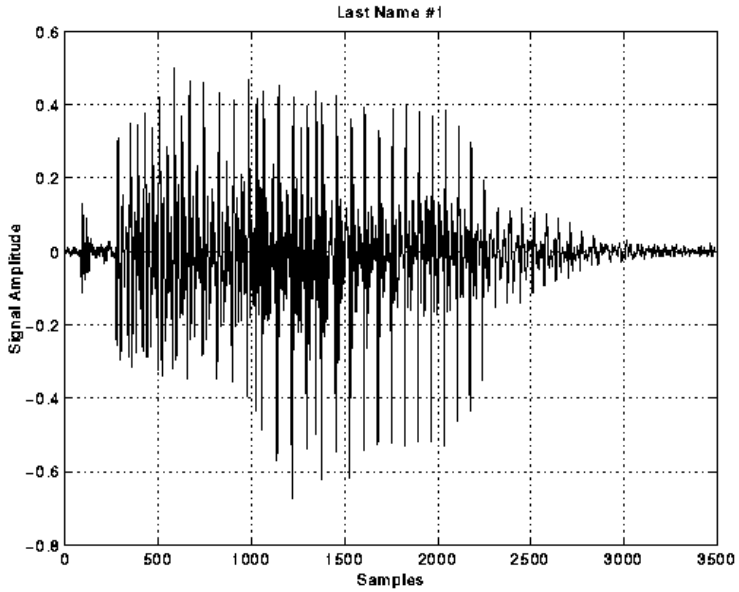
"o menino joga a bola" ou se preferirmos [o,menino,joga,a,bola]

O reconhecimento da voz e da fala pode gerar a entrada para um programa de LN, mas corresponde a uma área diferente, usando técnicas de Processamento de Sinal para proceder ao reconhecimento de fonemas e palavras. É usual em sistemas de legendagem automática e de interação com o utilizador (telemóveis, carros, assistentes pessoais como Alexa, Siri ou Google Assistant)

Reconhecimento da Voz e da Fala







Aplicações da Língua Natural

A Língua Natural pode ser usada para aplicações muito diversificadas, como por exemplo:

- Compreensão de frases isoladas
- Compreensão de Texto
- Geração de Texto
- Tradução Automática

- Apoio a Interfaces com o Utilizador (textuais ou por voz), como bots
- Interface com Linguagens de Interrogação de Bases de Dados (por exemplo SQL)
- Acesso à Informação (Information Retrieval)
- "Mineração" de Textos (Text Mining)

A interacção que se estabelece pode ser do tipo pergunta/resposta, frente-a-frente ou em diálogo livre

Sintaxe

- Na Língua Natural encontramos duas grandes áreas: a sintaxe e a semântica
- Na sintaxe definimos o modo como cada palavra é escrita, no fundo o vocabulário, e as sequências possíveis de palavras segundo o seu tipo, ou seja a Gramática
- Um programa em Língua Natural só conhece um vocabulário limitado e uma gramática limitada. Esse tem sido um dos problemas típicos nas interfaces por Língua Natural, pois uma falha na compreensão de uma frase pode dever-se ao facto da Base de Conhecimento do sistema ser limitada e não devido a erros de sintaxe

Semântica

A semântica analisa o significado das frases que construímos. As palavras de uma frase podem estar correctamente escritas e obedecerem a uma construção gramatical viável, mas a frase pode não fazer sentido

Convém notar que o problema é muito complexo. Há frases às quais podemos atribuir vários significados. Quando inseridas num texto podemos retirar algumas ambiguidades, mas isoladamente a sua compreensão não é possível

- Maria escolheu direito – pode ser interpretada como uma escolha acertada ou como a escolha de um curso de direito (embora nesse caso se devesse escrever Direito)
- O rapaz viu o homem na colina com o telescópio – quem estava na colina (o rapaz ou o homem) e quem tinha o telescópio (o rapaz ou o homem)

Identificação de Palavras Chave

Um dos primeiros sistemas de interface com o utilizador foi o ELIZA, desenvolvido nos anos 60 por Weizenbaum [Weizenbaum-1966]. Apesar de não ser dotado de grandes capacidades, por exemplo não guardava nenhuma representação das frases, este sistema aparentava um diálogo minimamente inteligente com base em alguns "truques"

USER: Man are all alike.
ELIZA: In what way?
USER: They´re always bugging us about something or other.
ELIZA: Can you think about a specific example?
USER: Well, my boyfriend made me come here.
ELIZA: Your boyfriend made you come here?
USER: He says I'm depressed much of the time.
ELIZA: I'm sorry to hear that you are depressed.
USER: It's true, I'm unhappy.
ELIZA: Do you think coming here will help you not to be unhappy?
USER: I need some help, that much seems certain.
ELIZA: What would it mean to you if you got some help?

Contudo, é fácil encontrar problemas no sistema

USER: My friend's sister likes me.
ELIZA: Tell me more about your family?

Análise da Sintaxe

Para a análise da sintaxe o mais habitual é a escrita de uma Gramática e de um Interpretador, tal como se costuma fazer para a implementação de uma linguagem de programação. O interpretador irá atribuir uma árvore de identificação (parsing) a cada frase

Frase → SN SV

SN → Artigo Nome

SN → Nome

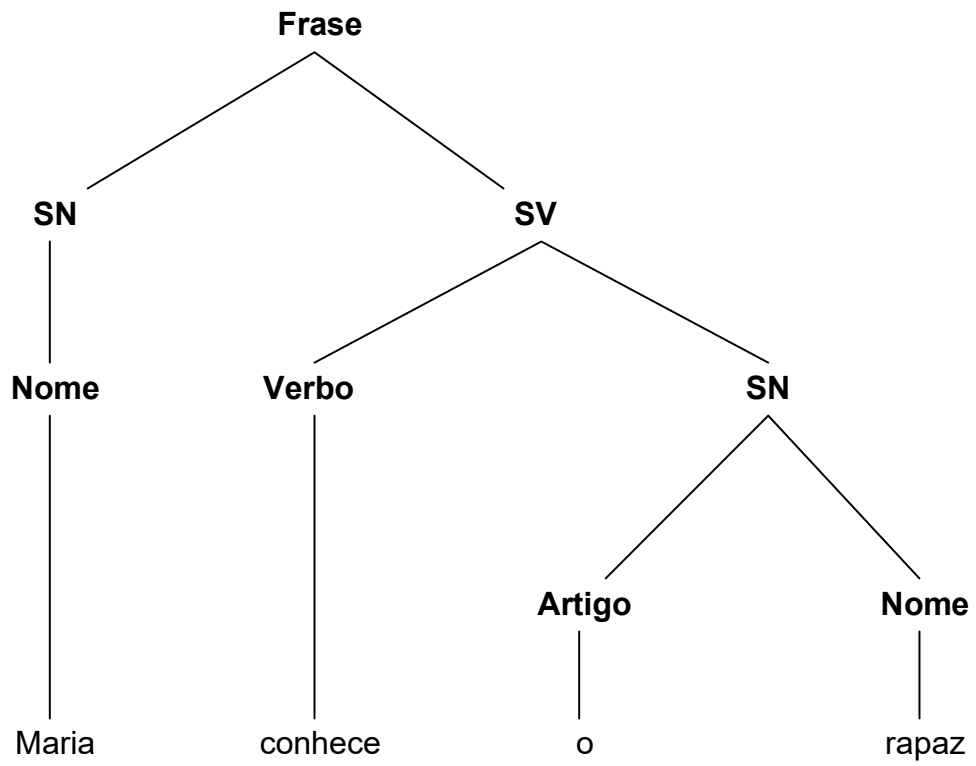
SV → Verbo SN

SV → Verbo

Artigo → o | a

Nome → rapaz | Maria | bola

Verbo → conhece | joga



Análise da Sintaxe

Exemplo de um tratamento simples sobre a Sintaxe feito em PROLOG (Gramática e Vocabulário), estamos a representar a Sintaxe por um sistema baseado em regras

```
frase --> sintagma_nominal(N), sintagma_verbal(N).
sintagma_nominal(N) --> artigo(G,N), nome(G,N).
sintagma_nominal(N) --> nome(_,N).
sintagma_verbal(N) --> verbo(N), sintagma_nominal(_,_).

artigo(m,s) --> [o].
artigo(f,p) --> [as].
nome(m,s) --> [rapaz].
nome(f,p) --> [cartas].
verbo(s) --> [joga].
verbo(p) --> [jogam].
```

Nota: --> é um operador do PROLOG que basicamente substitui o :- e coloca mais 2 argumentos em cada termo

A chamada poderia ser

```
?-frase([o,rapaz,joga,as,cartas],L).
L=[ ]
```

Análise da Semântica

Uma possibilidade de tratamento semântico pode ser feita através de uma Gramática Semântica. De seguida ilustramos uma Gramática Semântica possível

FRASE → Qual é a PROPRIEDADE_NAVIO do navio

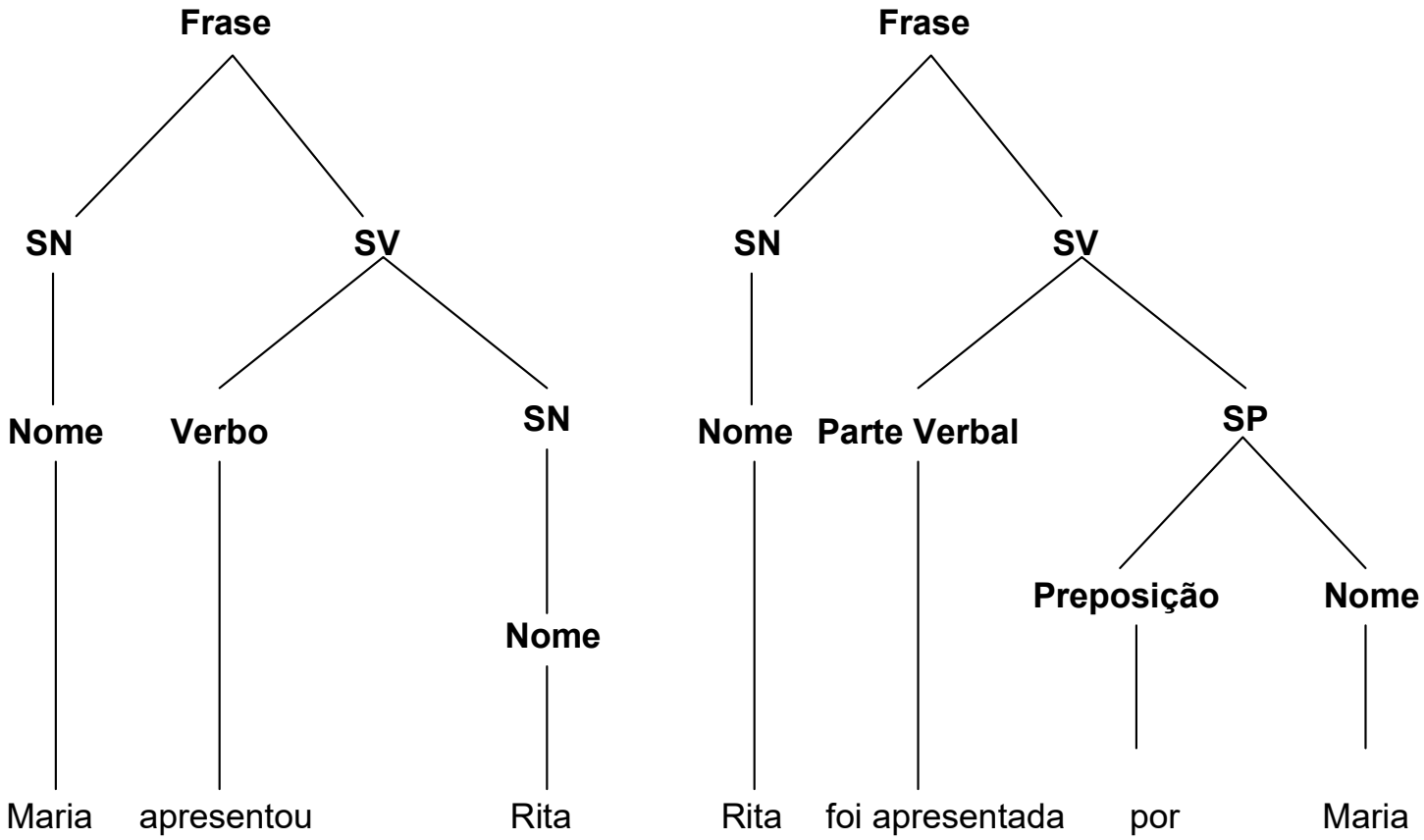
PROPRIEDADE_NAVIO → ARTIGO PROP_NAV | PROP_NAV

PROP_NAV → comprimento | largura | tipo

ARTIGO → o | a

Outra possibilidade é o uso de Gramáticas de Casos. As regras gramaticais descrevem a sintaxe, mas as estruturas criadas correspondem a relações semânticas (ver pág. seguinte)

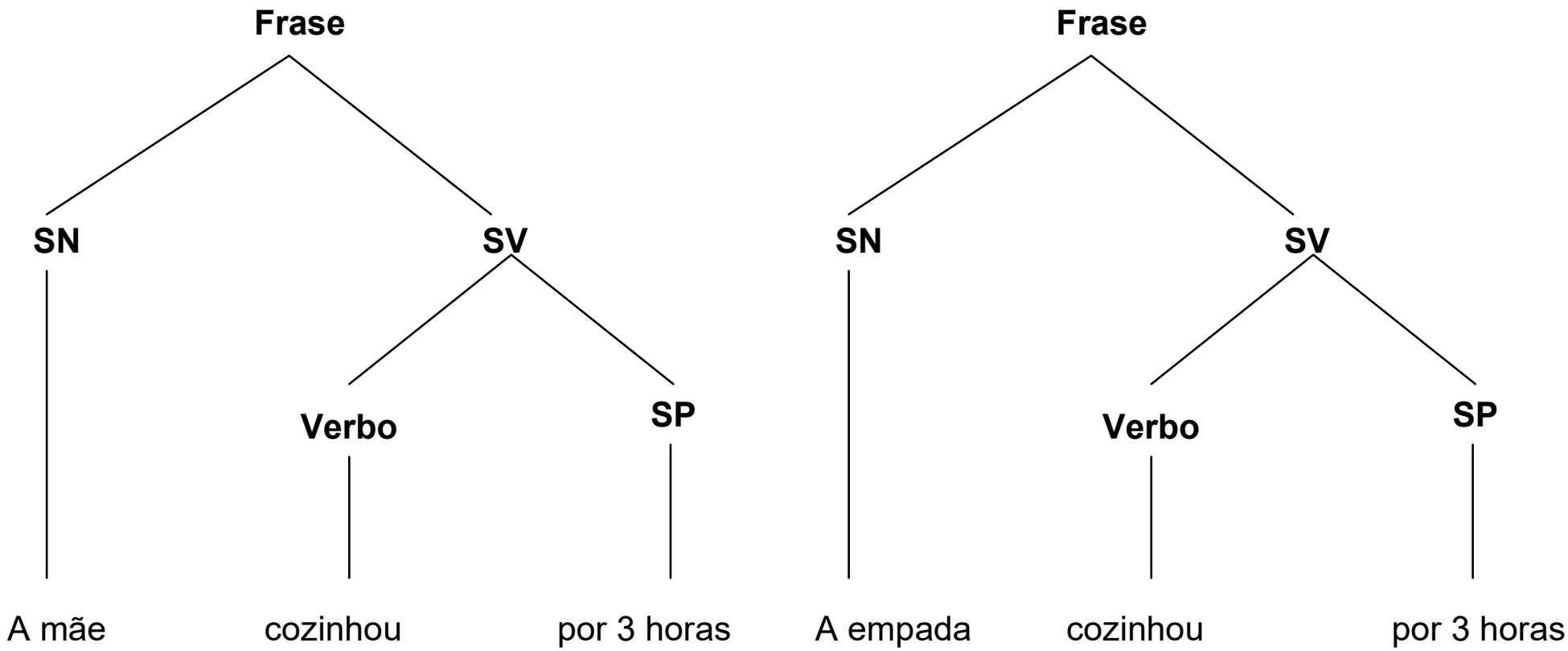
Análise da Semântica



- Em ambos os casos teremos uma estrutura criada do tipo:

  (apresentou (Atuante Maria) (Afetada Rita))

Análise da Semântica



Note-se que, apesar da árvore de identificação ser idêntica, a semântica da frase é diferente, visto que na primeira o sujeito é um Atuante (agente ativo – a mãe) enquanto que na segunda o sujeito é a empada, a qual é Afetada e não atuante

Análise da Semântica

Outra possibilidade é o uso da Dependência Conceptual. Por exemplo, quando aparece o verbo "esperar" podemos:

- Esperar que algo aconteça
- Esperar por um objecto
- Esperar por uma pessoa

Vejamos mais um exemplo. Considerem-se as duas frases que se seguem:

João foi ao parque com a namorada.

João foi ao parque com a fonte.

- No primeiro caso "namorada" refere-se a um ser animado, logo deve estar associada a João. Já no segundo caso a "fonte" não é um ser animado, devendo estar associada ao parque. Este exemplo demonstra o tipo de problemas que temos ao tentar dar significado (semântica) a uma frase.

Gramática de Cláusulas Definidas

Uma Gramática de Cláusulas Definidas (DCG – Definite Clause Grammar) permite a análise da sintaxe, e também da semântica, das frases

```
frase-->sintagma_nominal(Num),
                sintagma_verbal(Num).
sintagma_nominal(Num)-->artigo(Gen,Num),
                                nome(Gen,Num).
sintagma_nominal(Num)-->nome(_,Num).
sintagma_verbal(Num)-->verbo(Num),
                                sintagma_nominal(_).
```

Tratamentos Lógicos sobre a Semântica

Dois dos métodos básicos de tratamento lógico sobre frases são:

- os quantificadores de 3 ramos (3BQ – three branched quantifiers, de Dahl e Colmerauer)
- as cláusulas definidas do mundo fechado (DCW – Definite Closed-World, de Pereira e Warren)

Os 3BQ podem ser vistos como quantificadores de 3 partes (X,P,Q), onde X é uma variável, P está associado ao sintagma nominal e Q está associado ao sintagma verbal.

Por exemplo, a frase "Haddock despises every man who does not sail." Pode ser representada do seguinte modo:

every(X, and(isman(X), not(sails(X))), despises(haddock,X))

- onde *every* é o quantificador, *X* é a variável, *and(isman(X), not(sails(X)))* está associado ao sintagma nominal e *despises(haddock,X)* está associado ao sintagma verbal.
- P e Q podem ser verdadeiros, falsos ou indefinidos

Tratamentos Lógicos sobre a Semântica

Numa DCW, temos H ← G1&G2,... que corresponde em PROLOG a uma regra do tipo H:- G1,G2,...

Vejamos como fica a frase

"Há algum oceano que banhe todos os países da Europa?".

resposta(sim) ← *(∃ O) (oceano(O) &*

(∀ P) (país(P) & europeu(P) → banha(O,P)))

A questão "Quais são os oceanos que banham pelo menos três países europeus?" seria representada so segunite modo:

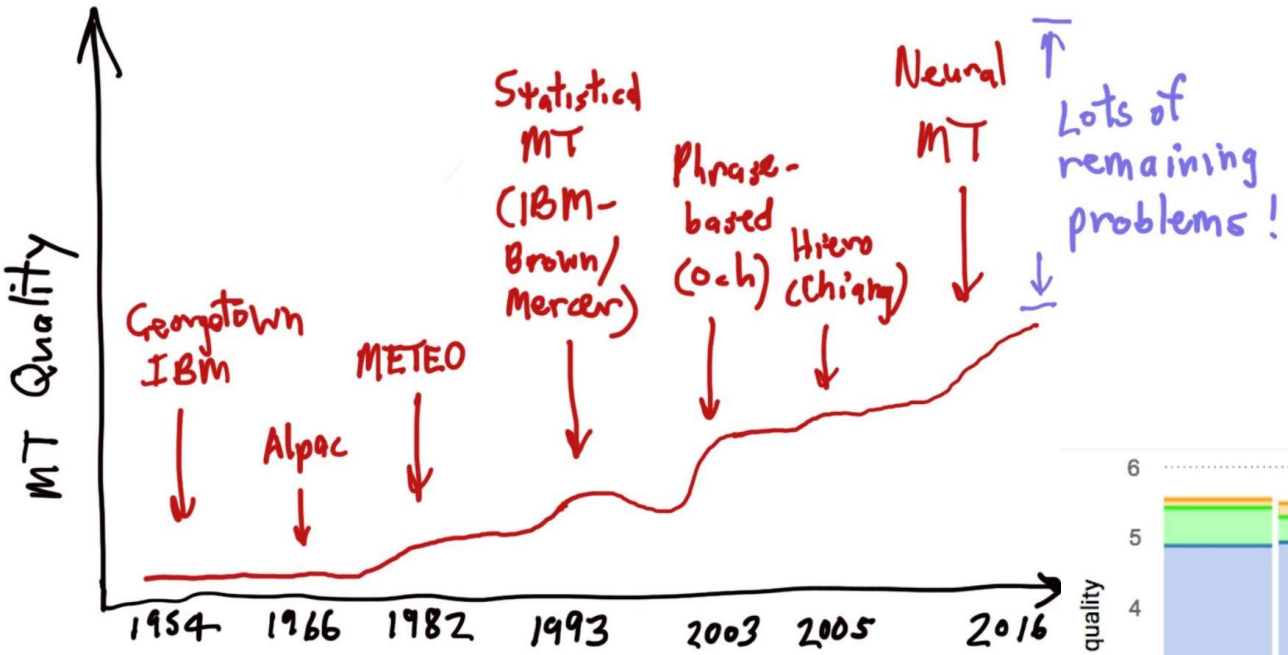
resposta(O) ← *oceano(O) &*

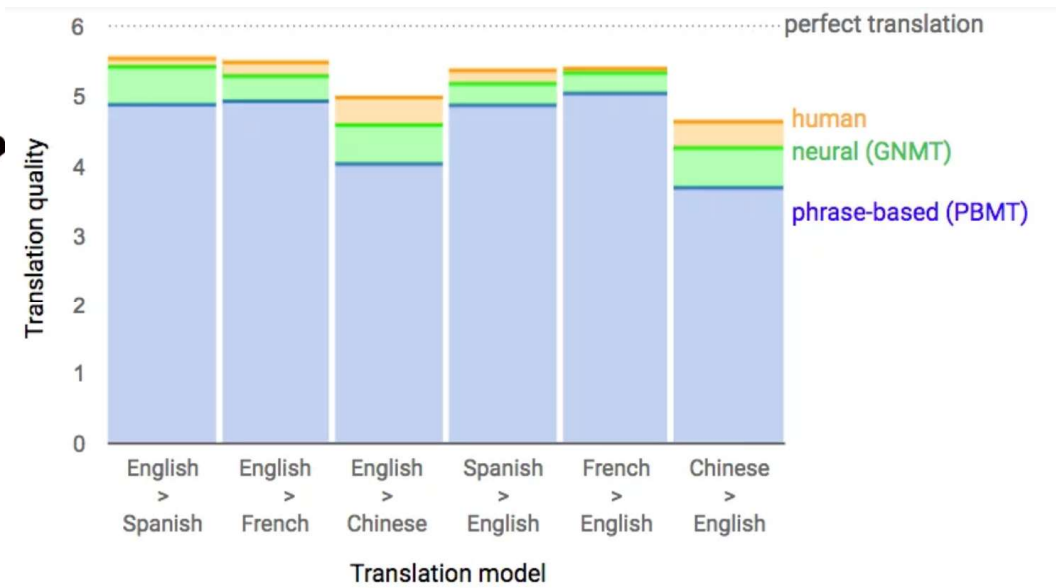
card {P : país(P) & europeu(P)& banha(O,P)} > 2

onde *card(S)* representa o cardinal de um conjunto S

Tradução Automática

A Tradução automática é um dos grandes desafios da IA desde os primórdios





Retirado de
https://nlp.stanford.edu/projects/nmt/Luong-Cho-Manning-NMT-ACL2016-v4.pdf

Tradução Automática

O Sistema **METEO** (1981) desenvolvido no Canadá faz a tradução bidirectional de previsões do tempo entre Francês e Inglês com base em Regras

Aujourd'hui, 26 novembre	Today, 26 November
Généralement nuageux. Vents du sud-ouest de 20 km/h avec rafales à 40 devenant légers cet après-midi. Températures stables près de plus 2.	Mainly cloudy. Wind southwest 20 km/h gusting to 40 becoming light this afternoon. Temperature steady near plus 2.
Ce soir et cette nuit, 26 novembre	Tonight, 26 November
Nuageux. Neige débutant ce soir. Accumulation de 15 cm. Minimum zéro.	Cloudy. Snow beginning this evening. Amount 15 cm. Low zero.

Nota: é bem mais simples fazer tradução num domínio específico que geral

Tradução Automática

A tradução automática baseada em conhecimento explicitado através de regras é elegante, mas tem grandes limitações:

- Se a compreensão de frases isoladas já levanta problemas de análise semântica o uso de mais do que uma língua levanta ainda mais problemas, a tradução não pode ser feita apenas traduzindo palavra a palavra e adaptando construções gramaticais diferentes entre línguas
- Na maioria dos casos, quer em texto escrito quer (sobretudo) em texto obtido a partir de voz, há erros nas palavras e nas construções gramaticais

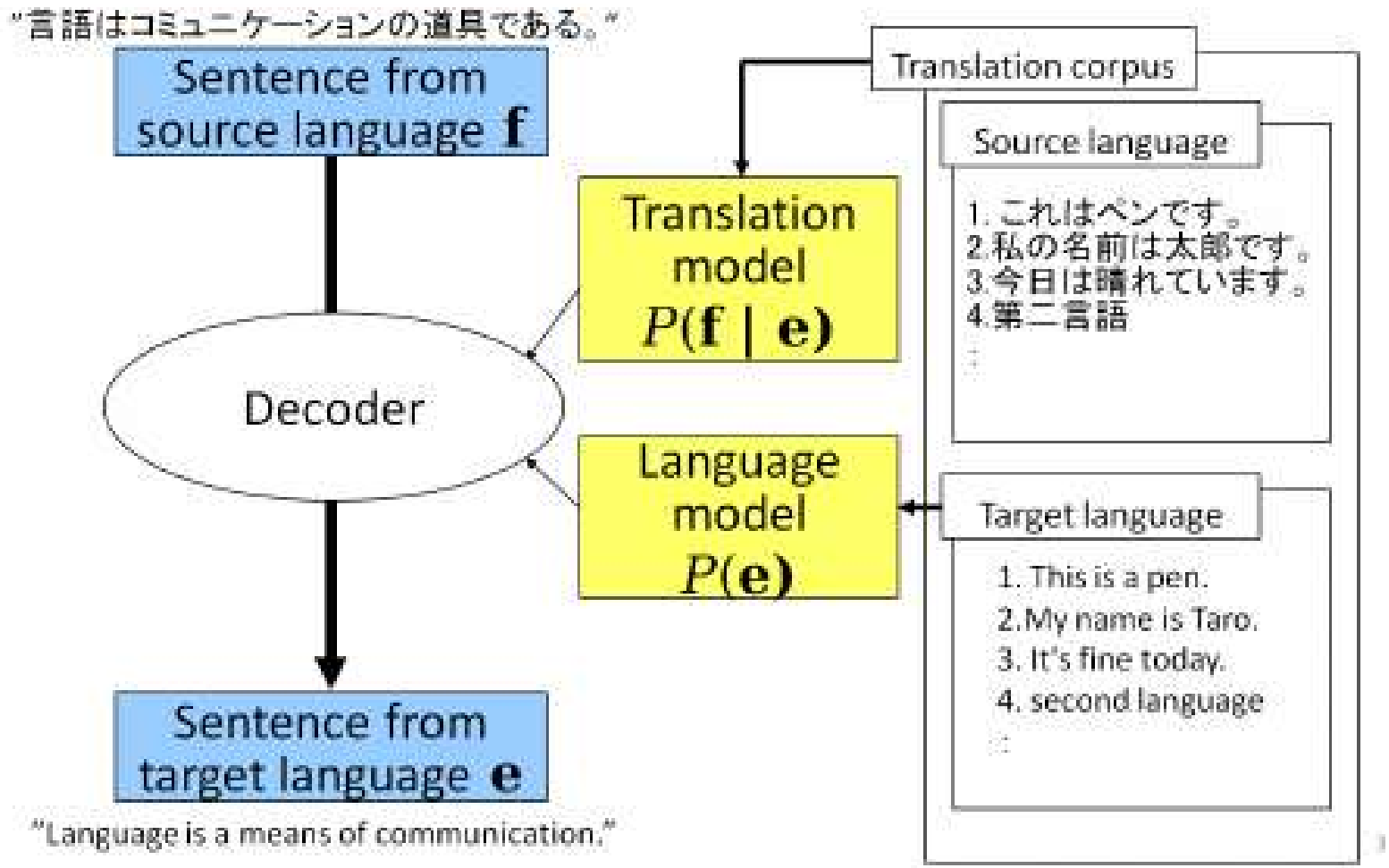
Surgem então os sistemas de Tradução baseada em Estatística que se popularizaram no início deste século e usam um Corpus

Tradução Automática

Tradução baseada em Corpus:

- Há um conjunto de textos na língua fonte e os mesmos textos na língua destino
  - Por exemplo a tradução de documentos nas Nações Unidas foi usada pela Google para constituir seus Corpora
- Os modelos estatísticos de Tradução Automática assumem que cada frase T numa língua é uma tradução da frase S de outra língua com uma dada probabilidade P(T|S) e a melhor tradução é a que tiver maior valor da probabilidade

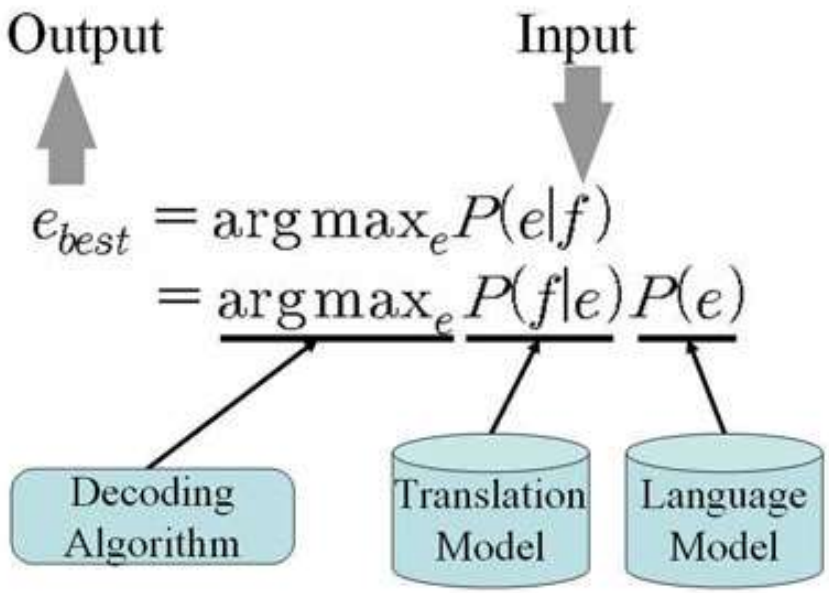
Tradução Automática



Retirado de
https://www.researchgate.net/figure/Statistical-Machine-Translation-system_fig3_267940056

Tradução Automática

- *P(f | e)* está relacionado com a identificação (matching) de palavras
- *P(e)* está relacionado com a correção da construção gramatical na linguagem pretendida, dá-nos a probabilidade de uma sequência de palavras ser viável
- *argmax* vai encontrar a tradução *e* com maior valor de probabilidade para traduzir *f*
- Obter *argmax (P(e | f))* é a função principal do Descodificador (Decoder)



Retirado de http://nlp.postech.ac.kr/research/previous_research/smt/

Tradução Automática

- Uma maneira de lidar com P(e) é o uso de n-gramas (n-grams), ou seja, ver num corpus a probabilidade de aparecer uma dada sequência de n palavras consecutivas de uma frase
  - Por exemplo, na frase "I have a black dog" temos os 3-gramas "I have a", "have a black" e "a black dog"
  - O 3-grama "I have a" é muito mais provável que o "a have I"
  - Vamos incorporando novas palavras no n-grama e calculando a probabilidade condicionada
  - A contagem de sequências de palavras numa query ao Google pode ser um bom indicador

Retirado de https://www.researchgate.net/publication/287196051_An_Overview_of_Statistical_Machine_Translation

Tradução Automática

Assunção de Markov de ordem n

- Numa sequência de palavras uma dada palavra depende normalmente de uma sequência de palavras anteriores, não é necessário considerar todas as anteriores

$$p(x_1, x_2, \ldots, x_T) = \prod_{t=1}^{T} p(x_t | x_1, \ldots, x_{t-1})$$

$$\approx \prod_{t=1}^{T} p(x_t | x_{t-n}, \ldots, x_{t-1})$$

Tradução Automática

ON VOIT JON À LA TÉLÉVISION

	good English? P(E)	good match to French? P(F|E)
Jon appeared in TV.		✓
It back twelve saw.		
In Jon appeared TV.		✓
Jon is happy today.	✓	
Jon appeared on TV.	✓	✓
TV appeared on Jon.		✓
Jon was not happy.	✓	

Retirado de https://pt.slideshare.net/hbnair080/statistical-machine-translation

Tradução Automática

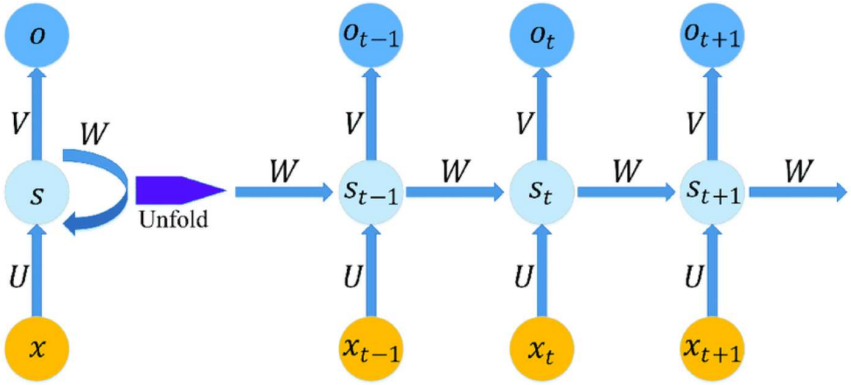
Os tradutores automáticos de maior sucesso hoje em dia usam a aprendizagem automática, em particular as Redes Neuronais com Deep Learninig, conduzindo ao que se conhece como Neural Machine Translation (NTM)

A arquitetura de rede mais usada em NMT é a arquitetura Encoder-Decoder, constituída por 2 Redes Neuronais Recorrentes

Tradução Automática

O modelo NTM representa as palavras como vetores densamente distribuídos que podem partilhar pesos em termos estatísticos entre palavras similares

A arquitetura de rede mais usada em NMT é a arquitetura Encoder-Decoder, constituída por 2 Redes Neuronais Recorrentes

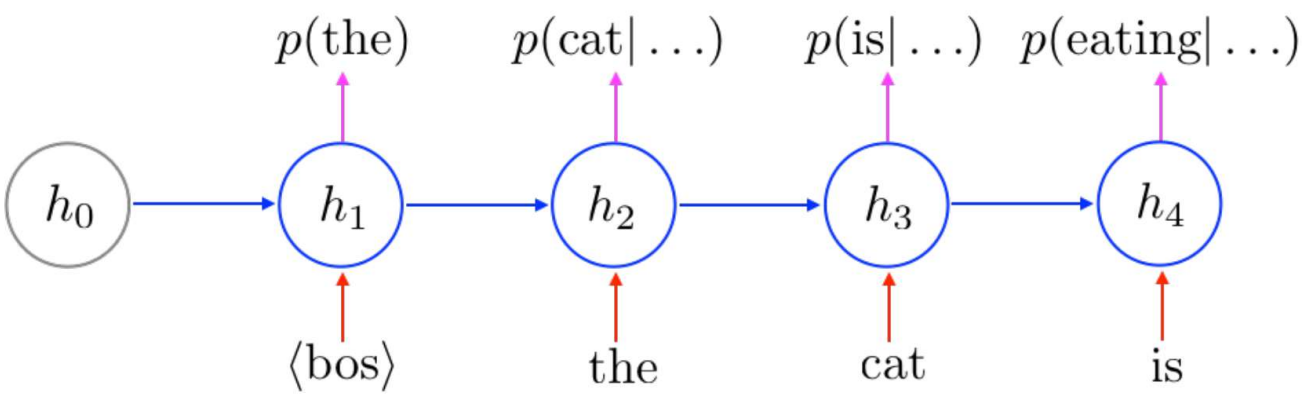


Rede Neuronal Recorrente

Tradução Automática

- As Redes Neuronais Recorrentes (RNN) vão adequar-se muito bem às cadeias de Markov, sem necessitarem considerar a assunção de Markov de ordem n

$$p(x_1, x_2, \ldots, x_T) = \prod_{t=1}^{T} p(x_t | x_1, \ldots, x_{t-1})$$

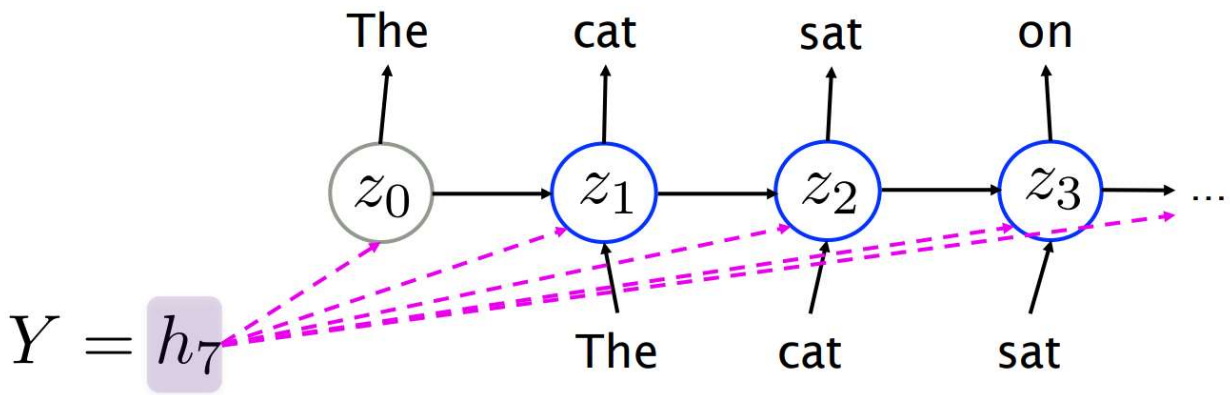


Mas as RNN também têm de ser treinadas e sujeitarem-se à Retropropagação
Constituem bons codificadores

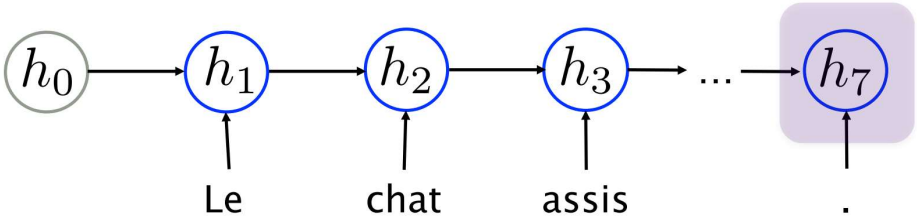
Tradução Automática

- Mas é necessário ter um par codificador-descodificador

Le chat assis sur le tapis.



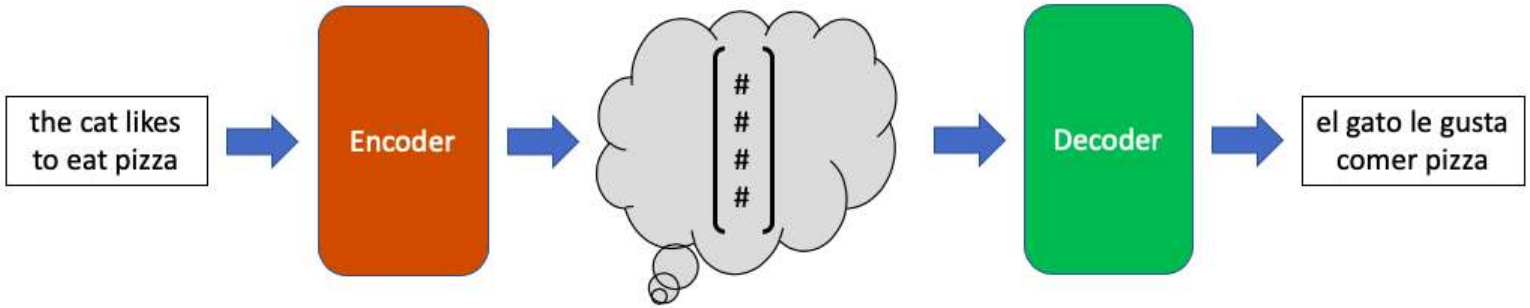
Descodificador



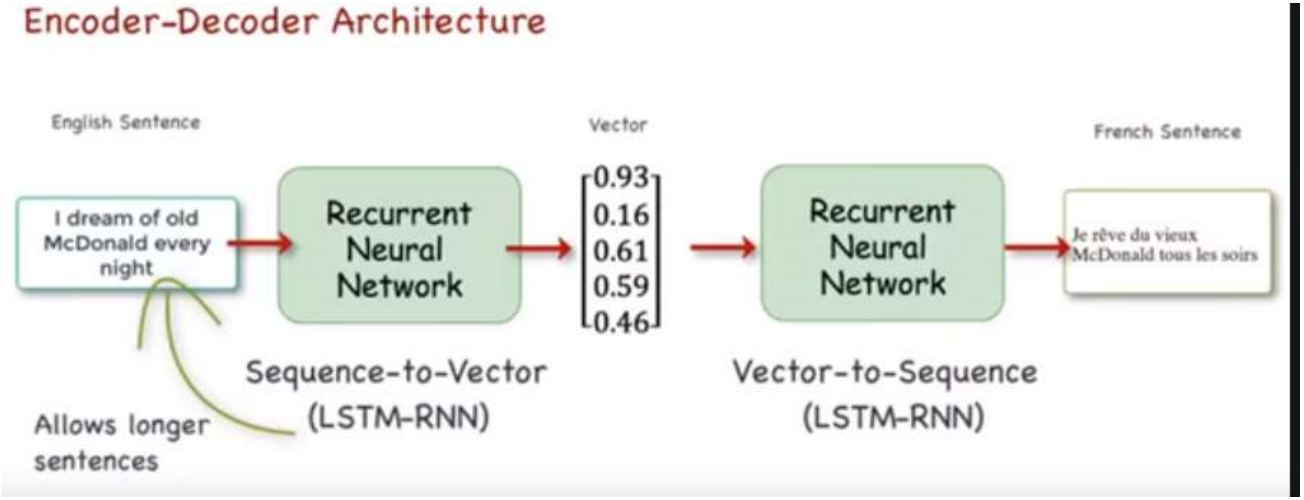
Codificador

Codificador e Descodificador são ambos RNN montadas em tandem

Tradução Automática

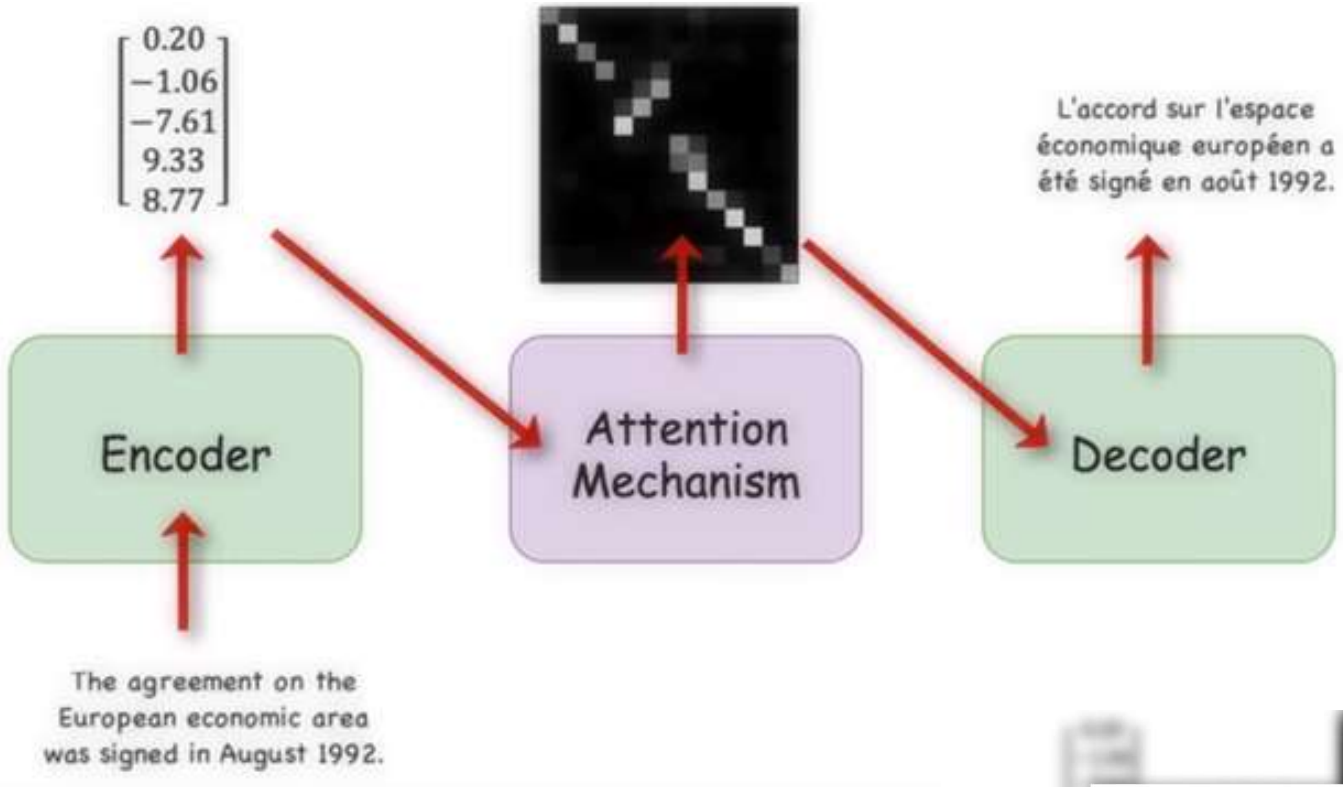


A RNN do Codificador converte uma sequência de palavras numa língua para um Vetor e a RNN do descodificador converte o vetor numa sequência de palavras noutra língua

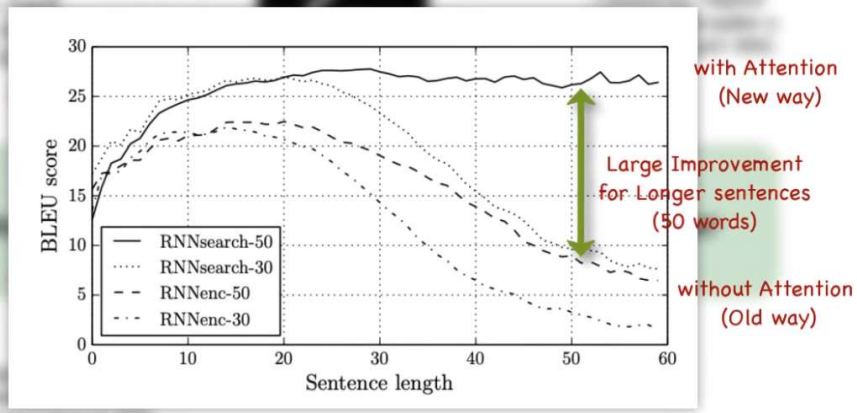


LSTM – Long Short-Term Memory, uma arquitetura de RNN

Tradução Automática



O Mecanismo de Atenção foca numa parte mais importante resultante da entrada, neste caso em palavras que são mais importantes



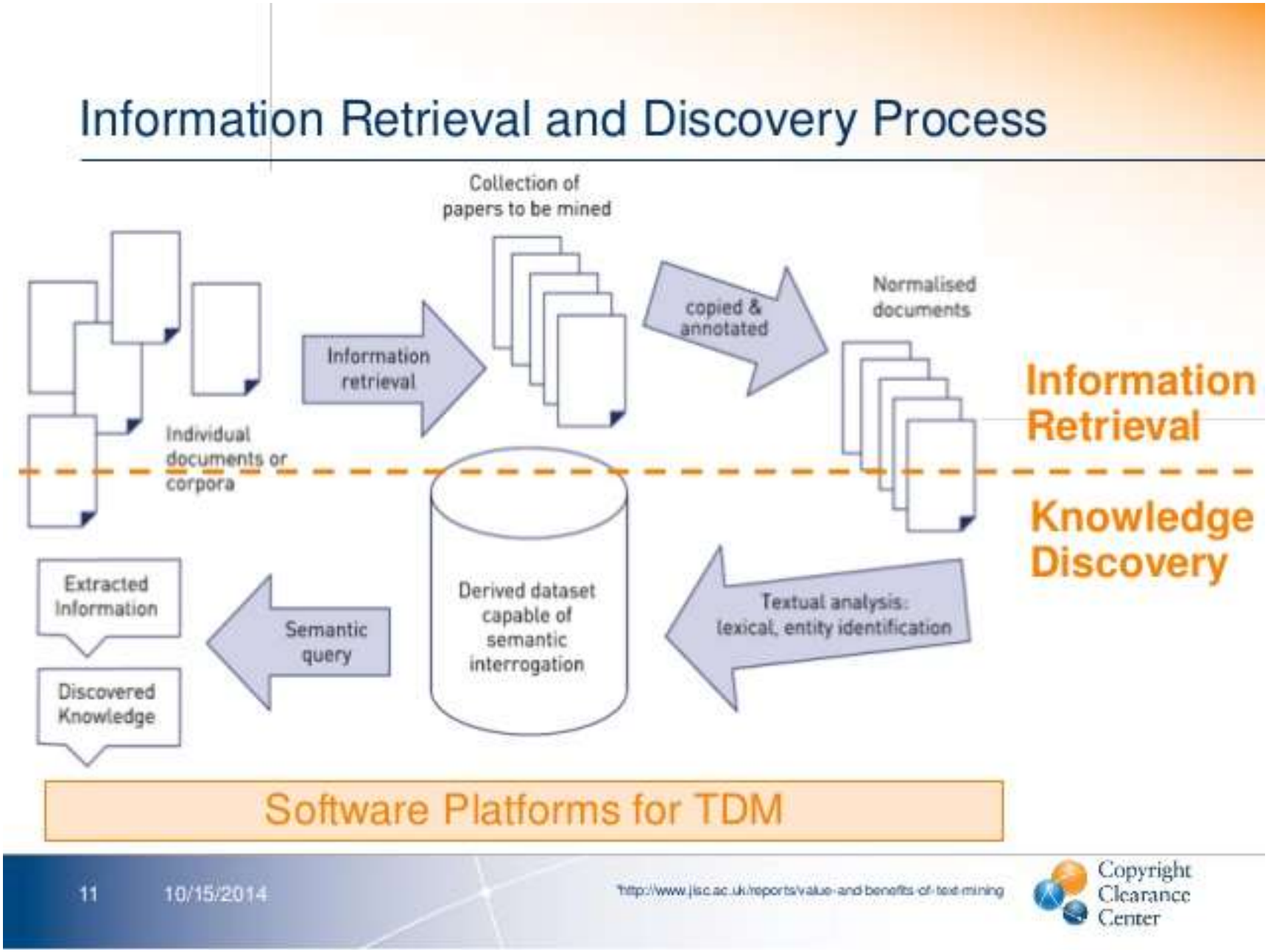
Source: Neural Machine Translation by jointly learning to align & translate (Bahdanau et al., 2016)

Information Retrieval e Text Mining

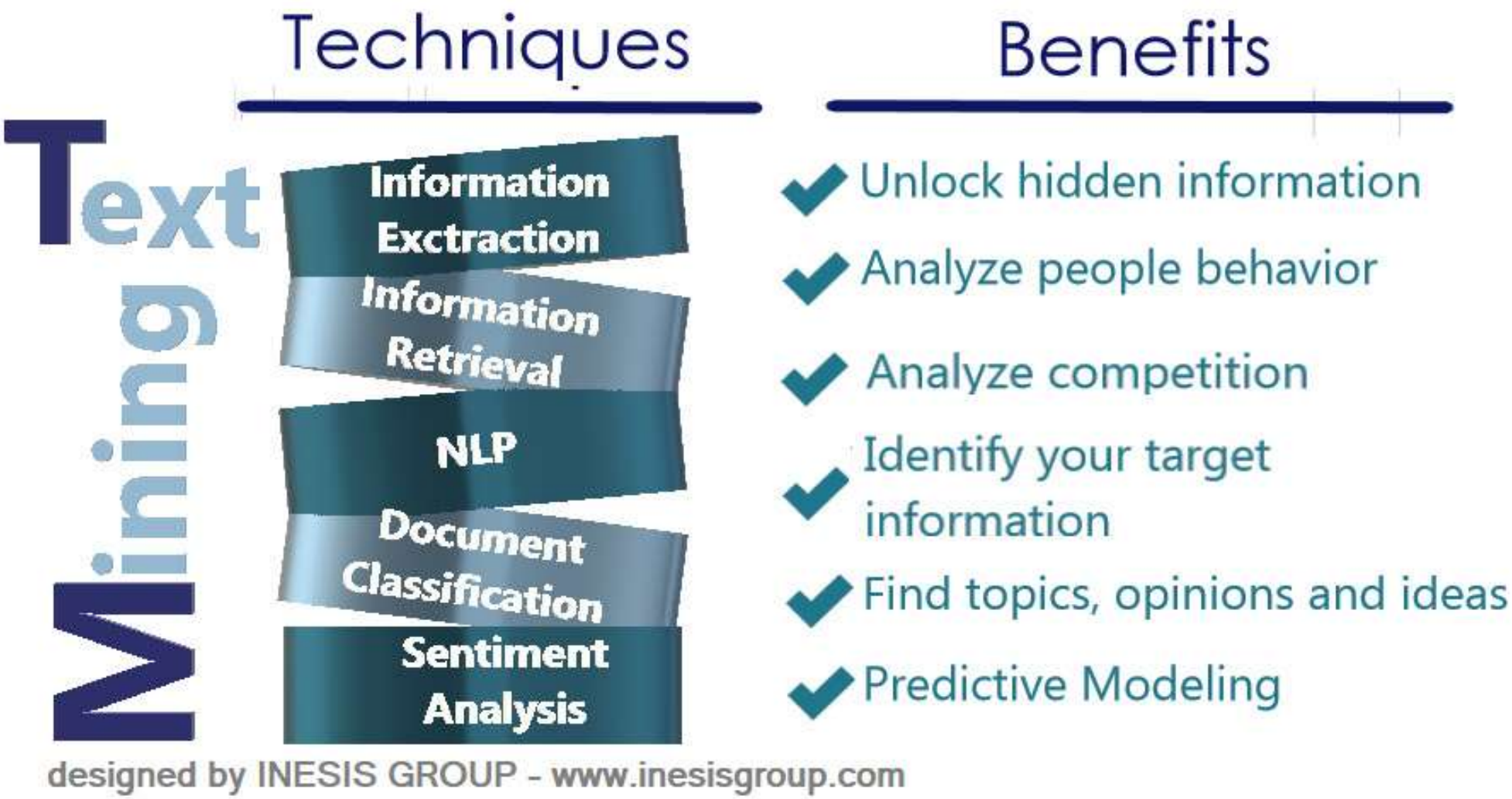
Information Retrieval - é a atividade de obter recursos relevantes a partir de uma coleção de recursos do sistema de informação

Text Mining - descoberta, por computador, de novas informações previamente desconhecidas, extraídas automaticamente de diferentes recursos escritos

Information Retrieval e Text Mining



Information Retrieval e Text Mining



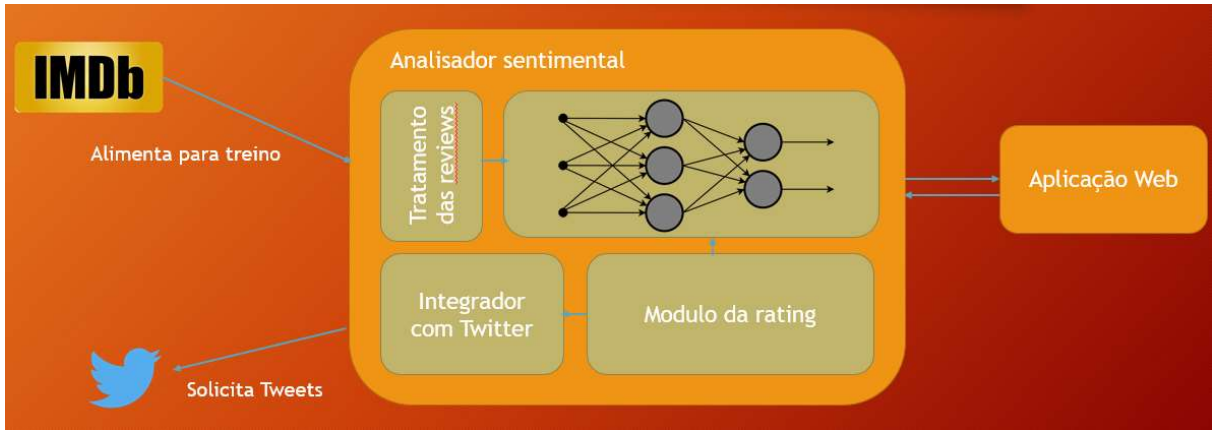
Para saber mais sobre Língua Natural

- Unidade Curricular de **Língua Natural e Sistemas Conversacionais (LNASCIA)** do **Mestrado em Engenharia de Inteligência Artificial (MEIA)** do ISEP
- Esta UC funciona em paralelo com a UC de Aprendizagem Automática 2 (AAUT2IA) que é dedicada a Deep Learning
- Dentro do Deep Learning as Redes Neuronais Recorrentes (RNN), em particular as Long-Short Term Memory (LSTM) são muito usadas para Língua Natural

MEIA
MESTRADO ENGENHARIA INTELIGÊNCIA ARTIFICIAL

Challenges 4Teams
MEIA

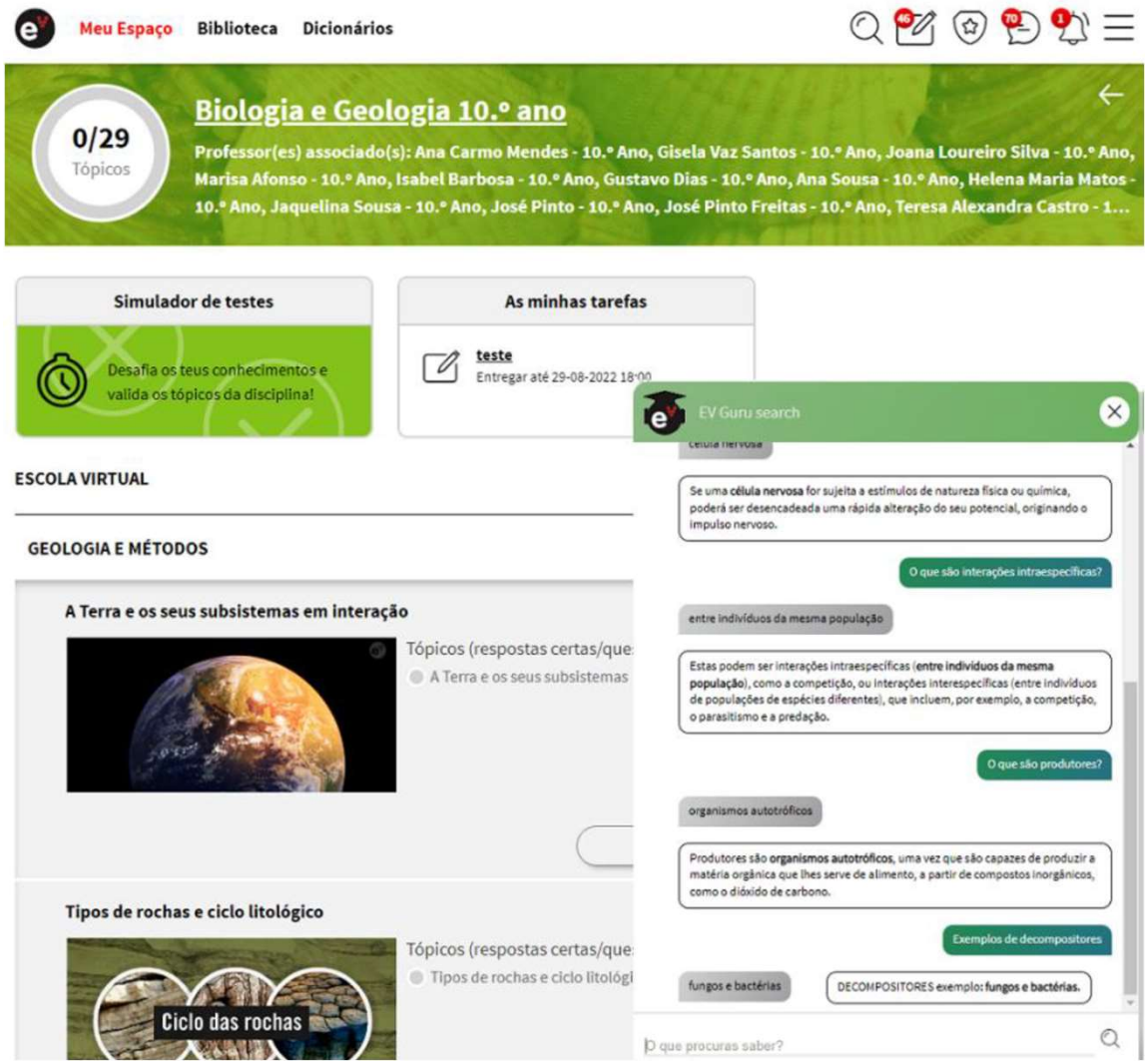
Demo Day
MEIA



Equipa de IA aplicada a Media do MEIA usa LSTM Para a recomendação de filmes com base na análise de sentimento feita a partir de tweets sobre filmes

Para saber mais sobre Língua Natural

- O sistema **EVGuru** foi desenvolvido numa dissertação de **Mestrado em Engenharia de Inteligência Artificial** do **ISEP** e será integrado no ambiente da Escola Virtual da Porto Editora
- O sistema tem capacidade de compreender texto obtido a partir de imagens de livros de Geologia e Biologia da Porto Editora, responder a perguntas sobre os temas do livro e gerar questões



Abel Ferreira, Respostas automáticas na pesquisa de conteúdos educativos, Dissertação de Mestrado em Engenharia de Inteligência Artificial, ISEP, 2022

6 - Domínios da IA
6.1 - Visão por Computador

Carlos Ramos

csr@isep.ipp.pt

Unidade Curricular: Algoritmia Avançada

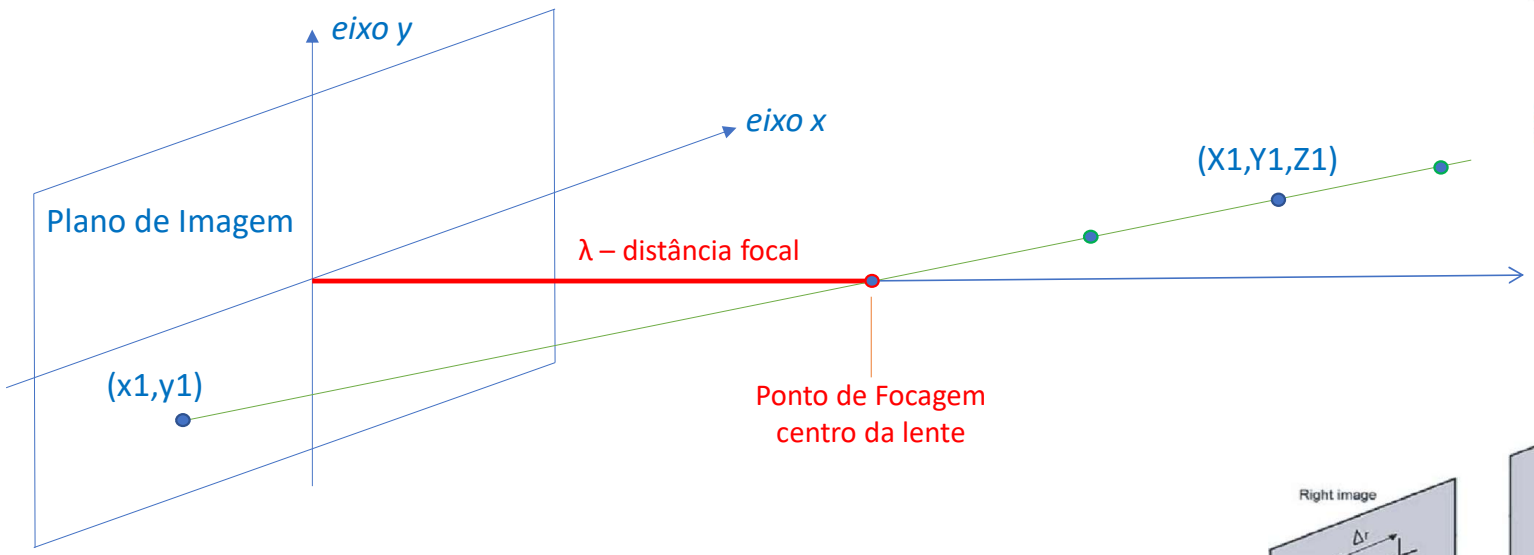
Curso: Licenciatura em Engenharia Informática (ISEP)

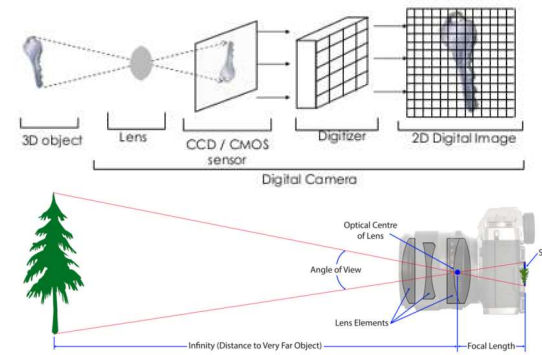
Processamento Digital de Imagem e Visão por Computador

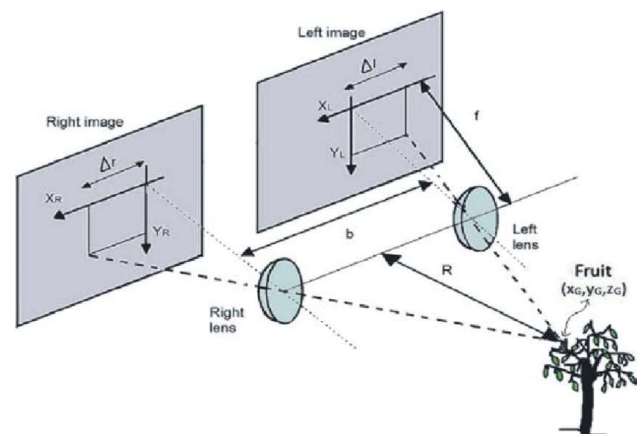
- As câmeras permitem a aquisição de imagem
- As câmeras digitais fazem a aquisição de uma imagem através de um sensor (tecnologia CMOS) e armazenam a imagem numa matriz de pixéis
- O Processamento Digital de Imagem é a área na qual se aplicam algoritmos que operam sobre uma imagem digital previamente armazenada para operações básicas como a redução de ruído, a compensação de distorções ou a compressão (por exemplo para jpeg)
- A Visão por Computador é a área científica na qual pretende-se efetuar o reconhecimento em imagem digital, por exemplo pela identificação de objetos numa imagem, por exemplo itens presentes numa estrada ou problemas identificados numa imagem médica

INPUT \ OUTPUT	IMAGE	DESCRIPTION
IMAGE	DIGITAL IMAGE PROCESSING	COMPUTER VISION
DESCRIPTION	COMPUTER GRAPHICS	ARTIFICIAL INTELLIGENCE

Formação da imagem







Plano Imagem – onde a imagem se forma (sensor CMOS)
Distância Focal – distância entre o centro da imagem e o ponto de focagem (centro da lente)
A um ponto (X1,Y1,Z1) no espaço corresponderá o ponto (x1,y1) no plano da imagem
Todos os pontos na semireta entre o ponto de focagem e passando por (X1,Y1,Z1) serão projetadas em (x1,y1)
Como o que conhecemos na imagem é (x1,y1) não sabemos exatamente onde está o ponto que lhe deu origem (X1,Y1,Z1)

Para podermos saber exatamente onde está um ponto no espaço vamos necessitar de 2 câmeras e relacionar os pixéis de projeção correspondentes, a isso designamos Visão Estéreo

Caracterícscticas das Imagens

Uma imagem é uma matriz de pixéis, no exemplo é usado 1 byte para representar cada pixel

0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	121	252	255	248	144	6	0
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	255	249	255	240	255	129	0	5
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0

0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	121	252	255	248	144	6	0
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	255	249	255	240	255	129	0	5
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	14	1	0	6	6	0	0

Fig. 2 A computer sees an image as an array of numbers. The matrix on the right contains numbers between 0 and 255, each of which corresponds to the pixel brightness in the left image. Both are overlaid in the middle image. The source image was downloaded via http://yann.lecun.com/exdb/mnist

Caracterícsticas das Imagens

- Resolução (nº de píxeis) – maior resolução implica mais memória, mas também mais detalhe
- Níveis de Intensidade - número de níveis diferentes que se pode representar num pixel que se relaciona com o espaço de armazenamento ocupado (por exemplo 256 níveis em 1 byte)

Uma imagem 256x256 em que cada pixel ocupa 1 byte vai ocupar 65536 bytes para representar os píxeis

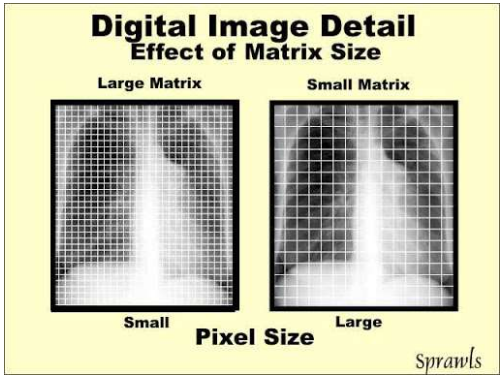
Com 1 byte podemos representar 256 níveis diferentes (níveis de cinzento)

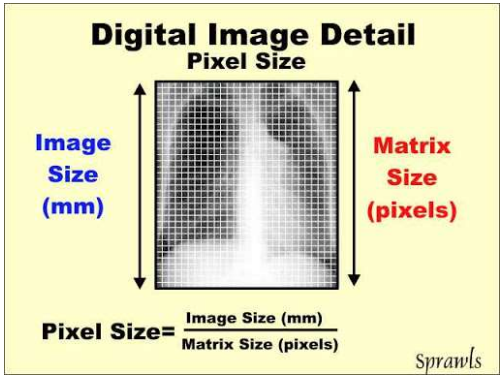
Imagens binárias ocupam 1 bit por pixel (branco,preto)

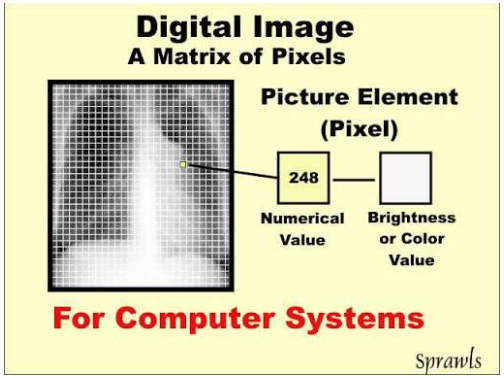
Se a cor for importante cada pixel pode ocupar 3 bytes (código RGB – Red, Green, Blue com 1 byte para cada cor, portanto 3 bytes/pixel)

Variação da resolução

Variação do nº de níveis de intensidade





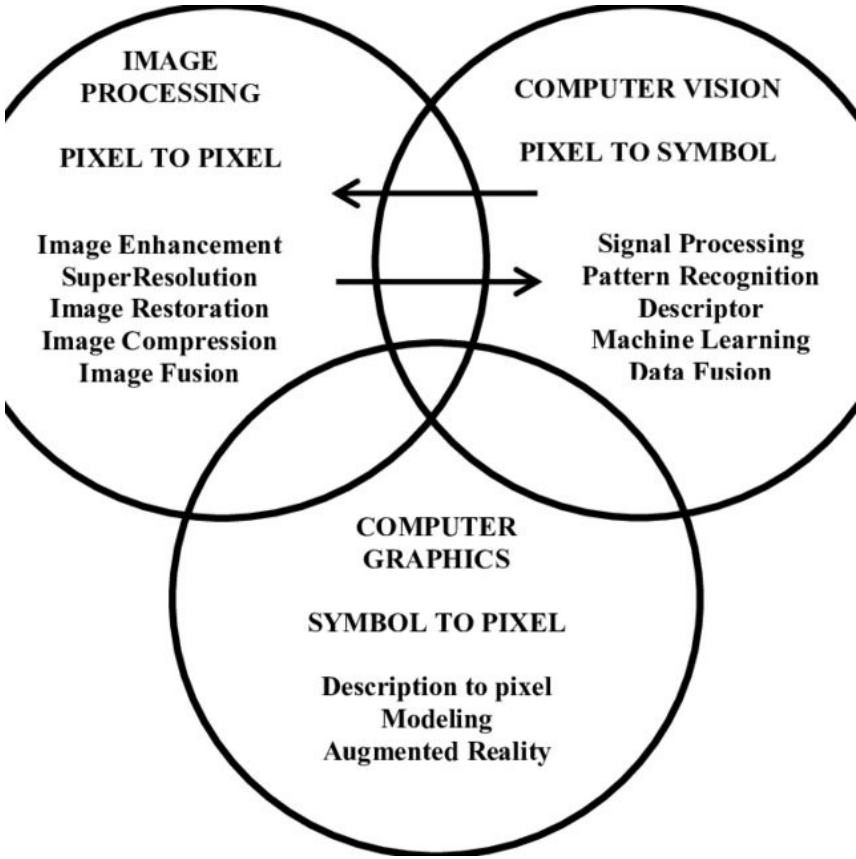


Processamento de Imagem, Visão por Computador e Computação Gráfica - diferenças

- Computação Gráfica – temos modelos de objetos e queremos criar uma imagem sintética que os envolvam em determinadas posições e orientações, do modo mais realista possível. Idealmente o ser humano até deveria ter dificuldade em distinguir uma imagem real de uma imagem sintética. Aspetos como oclusões entre objetos, sombras e reflexos são importantes para dar realismo à imagem

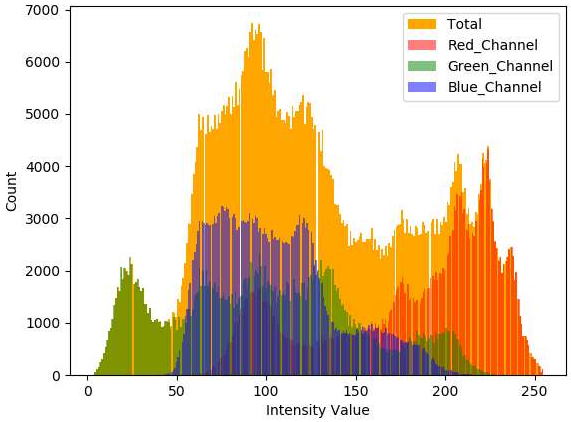
- Visão por Computador – o caminho é o inverso da Computação Gráfica, temos uma imagem real e queremos saber quais são os objetos presentes, suas posições e orientações (por exemplo usando visão estéreo). Oclusões, sombras e reflexos são aspetos com os quais temos de lidar e que criam complicações no processo

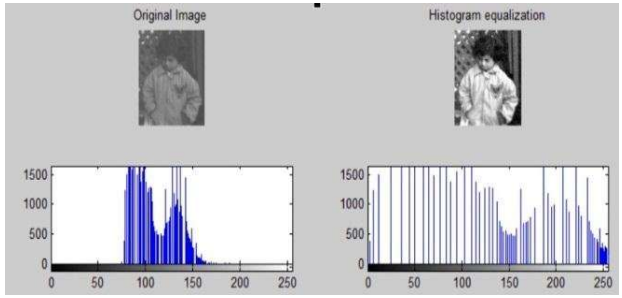
- Processamento de Imagem – o processo é feito dentro do espaço das imagens



Histogramas

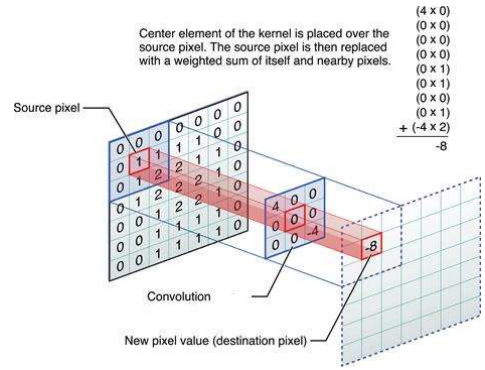
- Um histograma é um gráfico de barras que nos dá uma ideia do nº de ocorrências de uma dada intensidade (branco-preto, níveis de cinzento, código RGB) numa imagem
- Quando queremos fazer uma binarização numa imagem podemos socorrer-nos do histograma para decidir o threshold (nível de corte, ou seja o nível de intensidade abaixo do qual temos um valor, por exemplo preto, e acima do qual temos outro valor, branco)
- A equalização do histograma permite geralmente melhoria do brilho e contraste da imagem

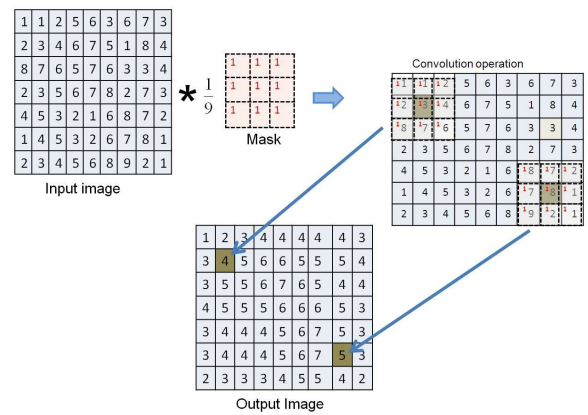




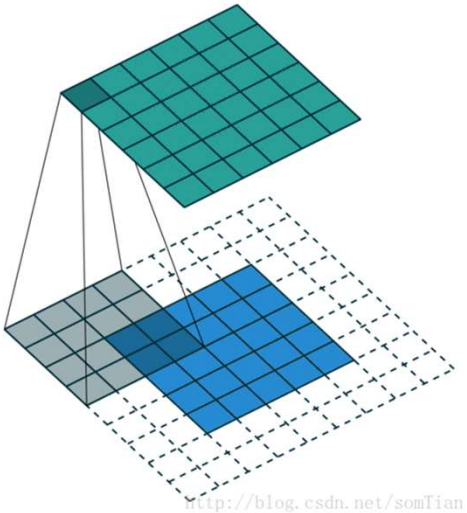
Convolução e Máscaras/Filtros

- Na Matemática a convolução é uma operação que a partir de 2 funções (*f* e *g*) produz *f*g*, traduz como uma das funções é moldada à outra. É muito usada em processamento de sinal (e imagem é um sinal visual)
- Em processamento de imagem faz-se frequentemente a convolução da imagem original com uma máscara/filtro de dimensão reduzida (por exemplo, uma matriz 3x3) sendo o resultado na imagem transformada o valor centrado do somatório dos produtos dos elementos da máscara com os valores da imagem original nas posições correspondentes. Essa operação é feita pelo varrimento completo da máscara/filtro sobre a imagem original





Convolução e Máscaras/Filtros



Resultado após convolução – matriz esverdeada 6x6

Filtro (ou máscara) – matriz cinzenta 4x4
Nas CNN recebe normalmente o nome kernel

Imagem original – matriz tracejada 9x9

Contudo é normal os filtros terem dimensão impar (3x3; 5x5) para ficarem centrados num pixel

http://cs231n.github.io/convolutional-networks/

Principais Máscaras/Filtros

-1	-1	-1
-1	8	-1
-1	-1	-1

Máscara para detetar
pixels isolados (possivelmente ruído)

1	1	1
1	1	1
1	1	1

Máscara de Média
para suavização da imagem

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

As máscaras podem ter diversas
Dimensões (Máscara de Média 5x5)

As máscaras podem definir apenas a zona de atuação
Por exemplo a máscara de mediana (também para suavização)
vai considerar o valor da imagem na matriz 3x3 que está a
meio de uma ordenação desses 9 valores

0	1	0
1	-4	1
0	1	0

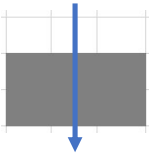
Laplaciano - Máscara para detetar
Transições (edge detection)

-1	0	1
-2	0	2
-1	0	1

Sobel H - Máscara para detetar
transições na horizontal

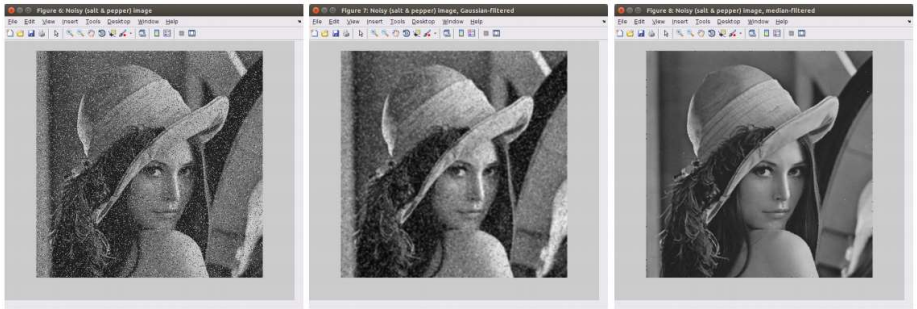
1	2	1
0	0	0
-1	-2	-1

Sobel V - Máscara para detetar
transições na vertical

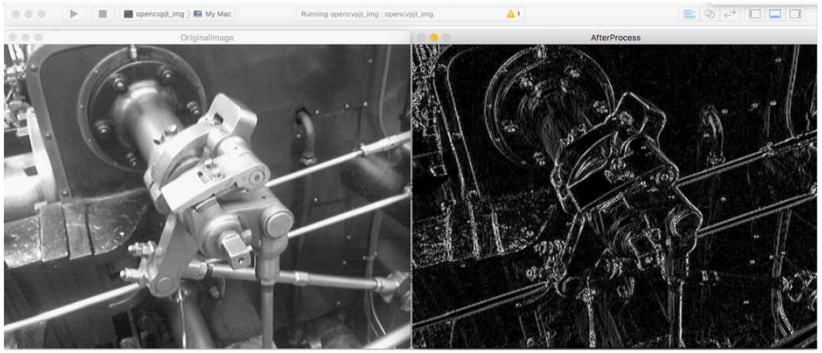


As máscaras de transições definem apenas onde estão
pixels de transição. Com o Sobel H e V temos alguma
ideia da direção da transição, o que é útil para agrupar
pixels de transição vizinhos

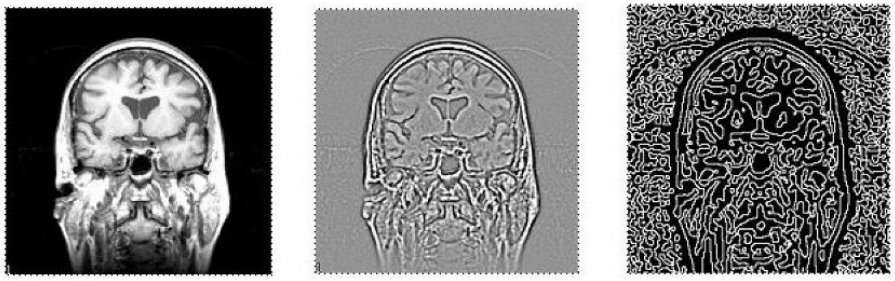
Principais Máscaras/Filtros



Suavização de imagem – à direita com filtro de mediana

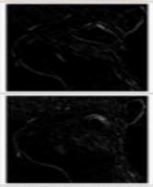


Deteção de transições com filtro de Sobel



(A) Original MR image (B) Laplacian results (C) Extraction of the zero crossing of the Laplacian (object edges)

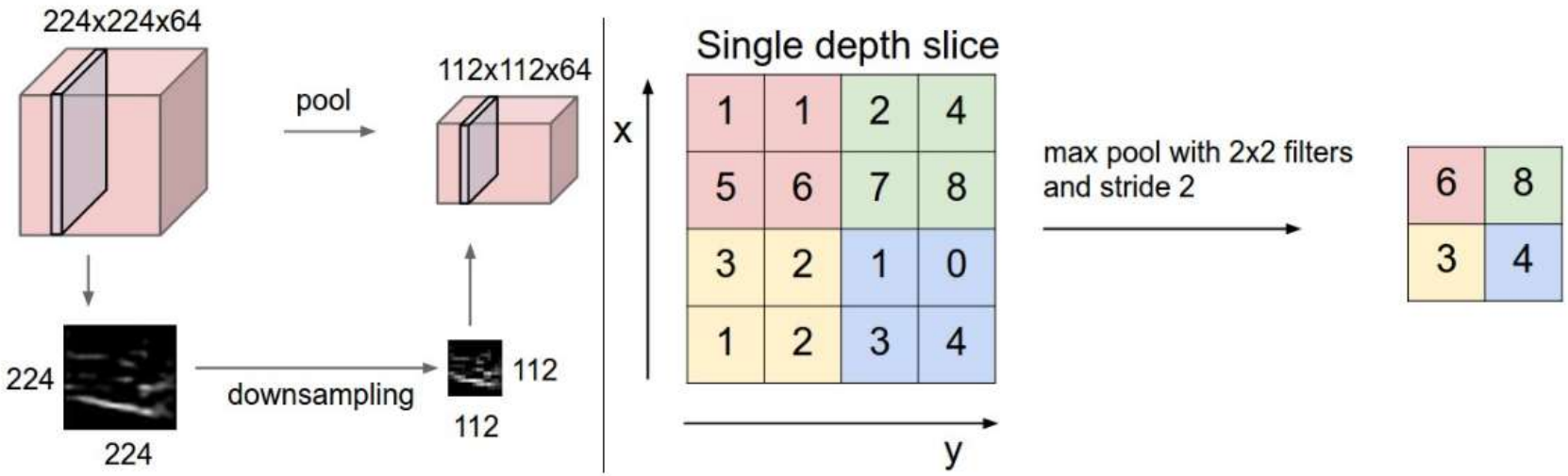
Deteção de transições com filtro Laplaciano

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Pooling

A operação de pool faz uma redução da dimensão da matriz imagem, ou seja, reduz a resolução da imagem e também o espaço de memória ocupado, por consideração dos píxeis da imagem original que ficam dentro da máscara considerada

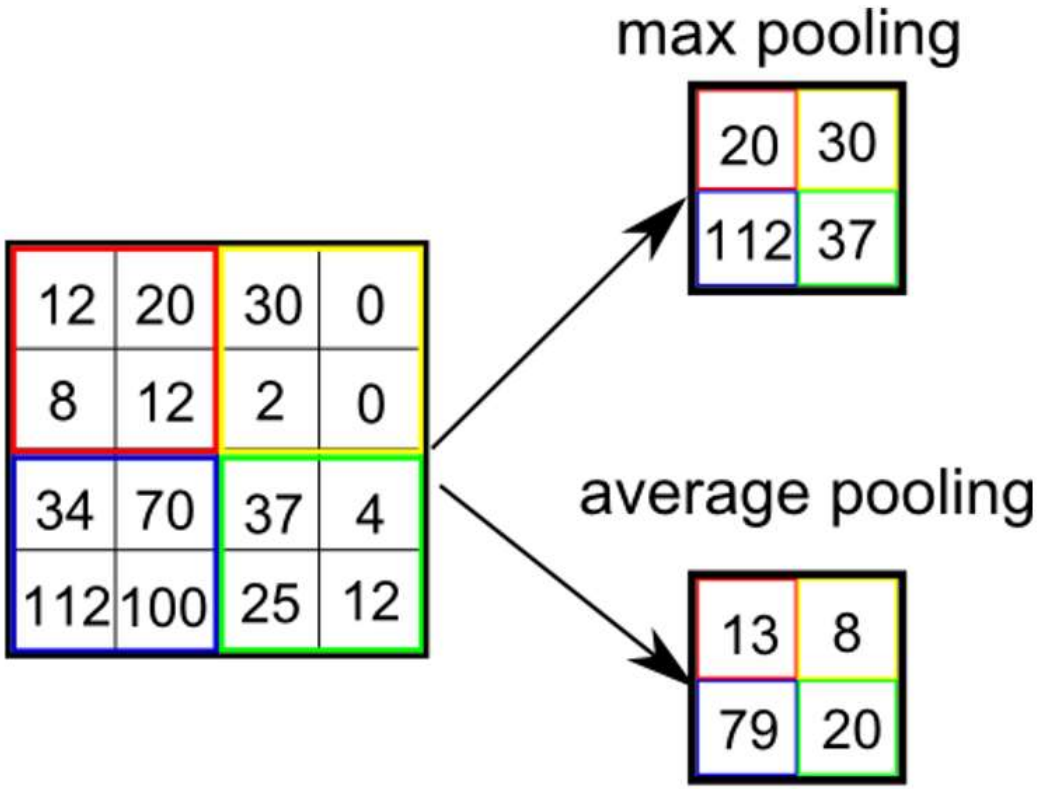
As situações mais normais são substituir pela média ou pelo valor máximo



Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume. **Left:** In this example, the input volume of size [224x224x64] is pooled with filter size 2, stride 2 into output volume of size [112x112x64]. Notice that the volume depth is preserved. **Right:** The most common downsampling operation is max, giving rise to **max pooling**, here shown with a stride of 2. That is, each max is taken over 4 numbers (little 2x2 square).

http://cs231n.github.io/convolutional-networks/

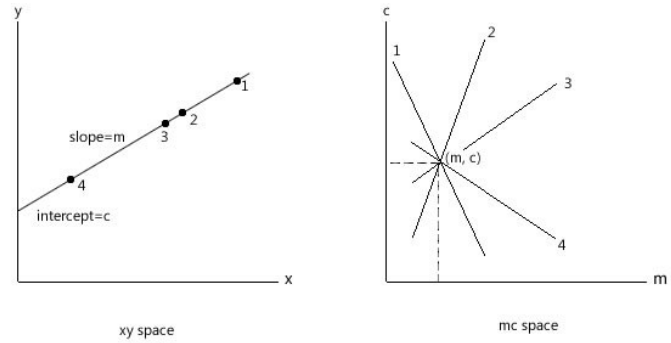
Pooling

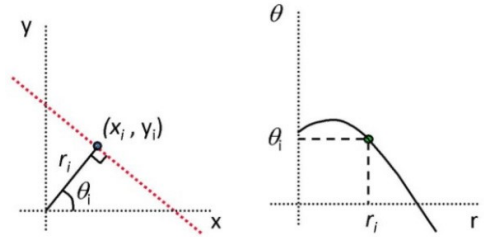


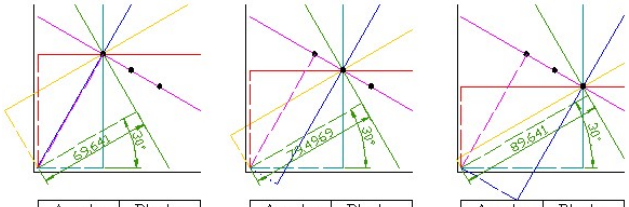
http://cs231n.github.io/convolutional-networks/

Identificando transições a partir de pixels de transição

- Transformada de Hough – com base em pixels de transição e direções de transição (Sobel) constrói um quadro onde as semiretas de transição têm um peso maior e portanto são mais facilmente identificadas
- Passamos do espaço x-y para o espaço m-c (equação da reta y=m*x+c). Vários pixels de transição de uma mesma semireta têm valores de m e c similares
- Por razão de dimensionalidade do quadro da Transformada de Hough usa-se o espaço r-Θ, onde r é a distância da reta à origem e Θ o ângulo







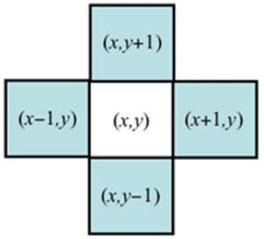
Angle	Dist.
0	40
30	69.6
60	81.2
90	70
120	40.6
150	0.4

Angle	Dist.
0	57.1
30	79.5
60	80.5
90	60
120	23.4
150	−19.5

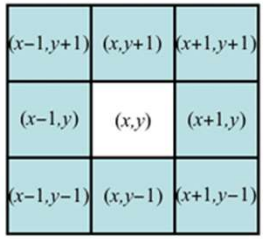
Angle	Dist.
0	74.6
30	89.6
60	80.6
90	50
120	6.0
150	−39.6

Segmentação

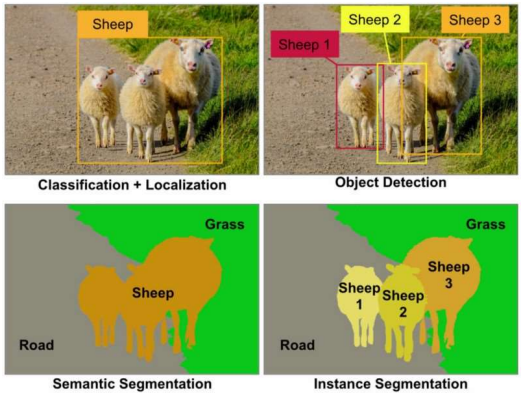
- A Segmentação consiste na divisão da imagem em conjuntos de pixels que m conjunto partilham algo de comum (por exemplo, a mesma cor, ou cores próximas, constituem uma mesma superfície ou objeto, etc)
- Pode ser feita por crescimento de regiões agregando pixels vizinhos (4 vizinhos, 8 vizinhos)
- Pode ser feita pela análise das transições (fronteira dos objetos na imagem)

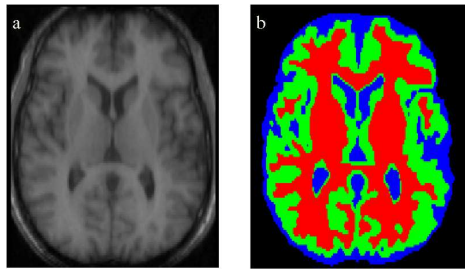


4-neighbourhood

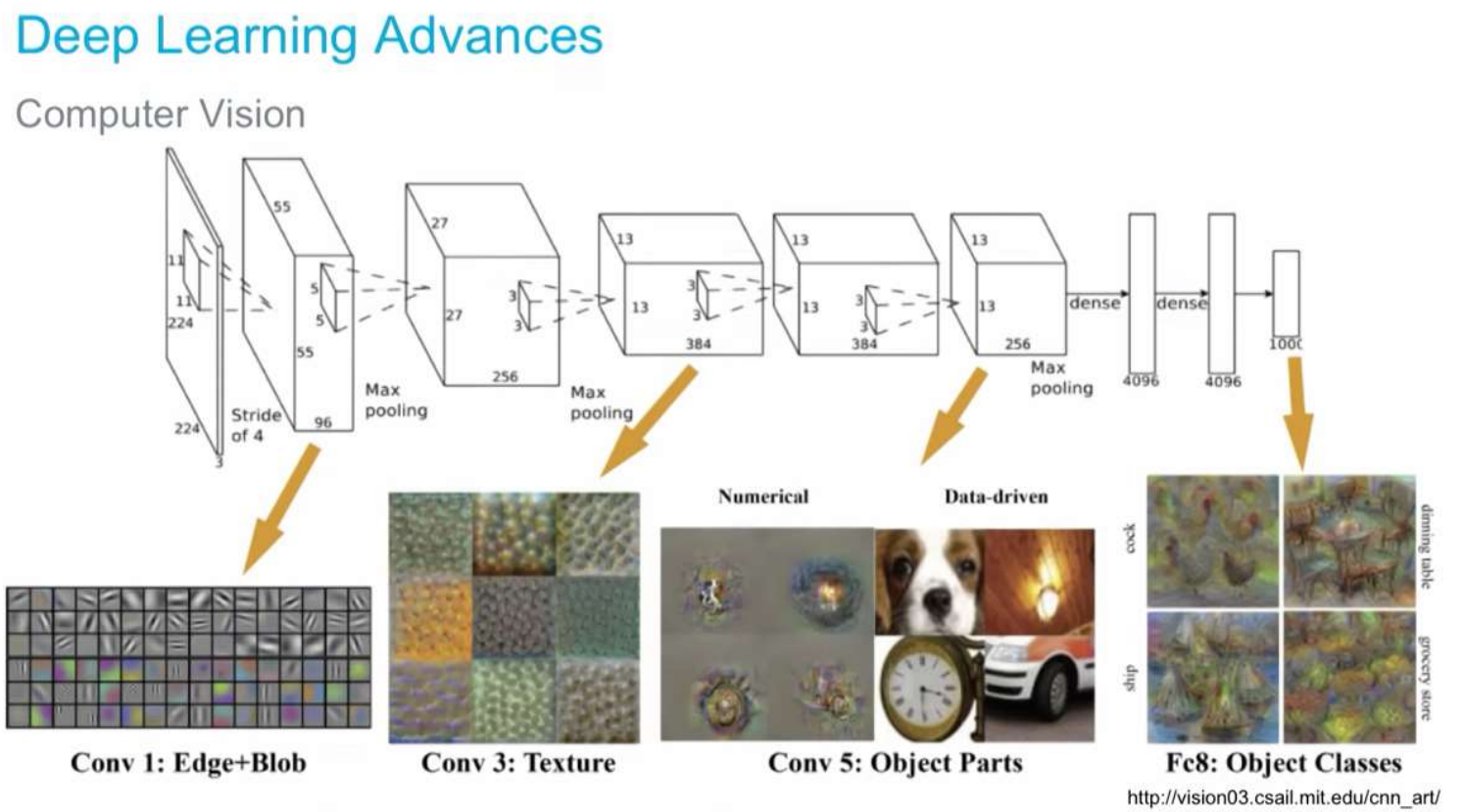


8-neighbourhood





Uso de Deep Learning (um tipo de Rede Neuronal)



Deep Learning: às vezes corre bem, outras nem tanto!

SAYS.COM

South Korea Now Has An AI-Powered News Anchor That Is Capable Of Working 24/7

IFLSCIENCE.COM

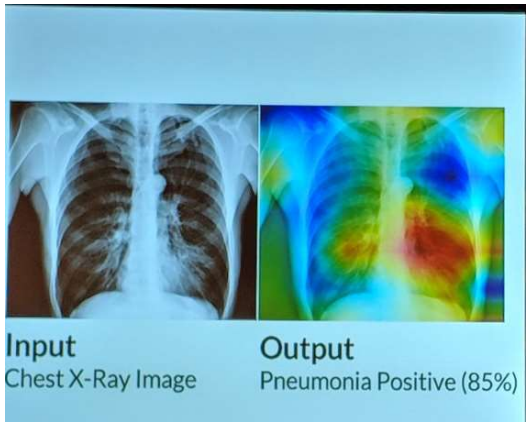
AI Camera Ruins Soccer Game For Fans After Mistaking Referee's Bald Head For Ball

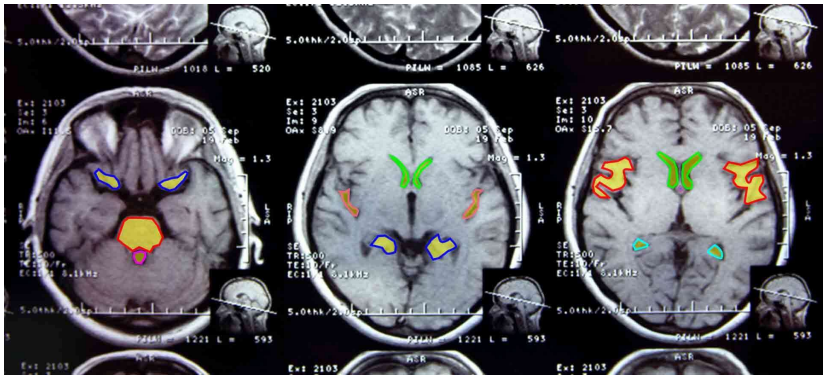
https://www.youtube.com/watch?v=k8X_Em-NQn0

https://www.youtube.com/watch?v=PfOJhxm3M8U

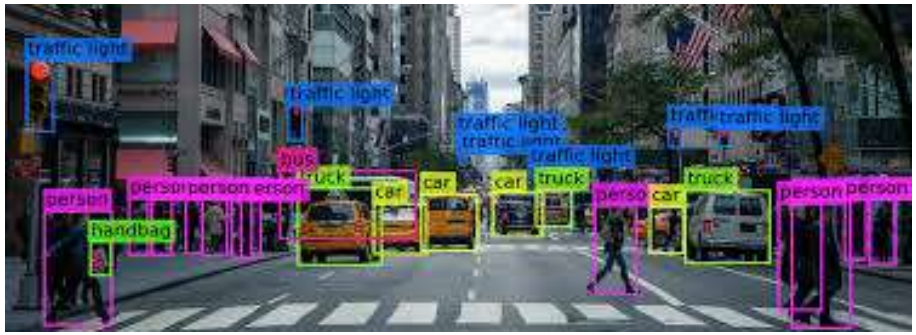
Aplicações de Visão por Computador

Indústria (exemplo bin picking)





Saúde



Veículos Autónomos

Segurança

.....

Plataformas/Linguagens para Visão por Computador

- OpenCV – muito completa, liga com C++ e Python e para diversos ambientes (Linux, Windows, Android). O processamento fica mais lento em grandes volumes de dados. Não permite de base desenvolvimento em GPU
- Matlab – permite código conciso, de leitura e debug fáceis. É bom para prototipagem e I&D, mas não é rápido. É pago
- Aforge.NET/Accord.NET – adequado para quem desenvolve em C# e .NET, com muitas operações sobre as imagens, mas não permite as capacidades de um OpenCV
- TensorFlow – hoje em dia muita da Visão por Computador é feita com recurso a técnicas de Machine Learning (ML), em particular Deep Learning (DL). Mas necessita dominar esses temas, além de ser uma plataforma muito exigente em termos de hardware, tirando partido de GPUs
- CUDA – plataforma da NVIDIA para computação paralela, tirando partido dos GPUs. Exige dominar bem computação paralela e GPU
- SimpleCV – adequada para prototipagem e quando não é necessário entrar em profundidade em Processamento de Imagem e Visão por Computador
- GPUImage – biblioteca iOS que aplica efeitos e filtros acelerados por GPU à imagem
- Há ainda plataformas de Visão por Computador como um serviço: Google Cloud Vision, Mobile Vision, Amazon Rekognition, Microsoft Azure Computer Vision
- Também pode ser feito desenvolvimento com várias linguagens e algumas bibliotecas, como por exemplo usando Python com SciPy e NumPy

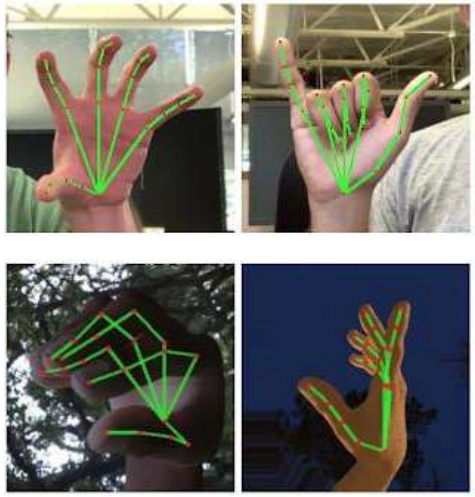
Para saber mais sobre Visão por Computador

- Unidade Curricular de **Aprendizagem Automática 2 (AAUT2IA)** do **Mestrado em Engenharia de Inteligência Artificial (MEIA)** do ISEP
- AAUT2IA estuda as Redes Neuronais Artificiais e a Aprendizagem Profunda (Deep Learning)
- Hoje em dia grande parte do que se faz em Visão por Computador usa Deep Learning, nomeadamente Redes Neuronais Convolucionais
- O tema é ainda abordado em AMBIIA (Ambientes Inteligentes) ligado à parte da Robótica

MEIA
MESTRADO ENGENHARIA INTELIGÊNCIA ARTIFICIAL

Challenges 4Teams
MEIA

Demo Day
MEIA



Equipa de IA aplicada à Saúde do MEIA usa Deep Learning para Reconhecer palavras de Língua Gestual

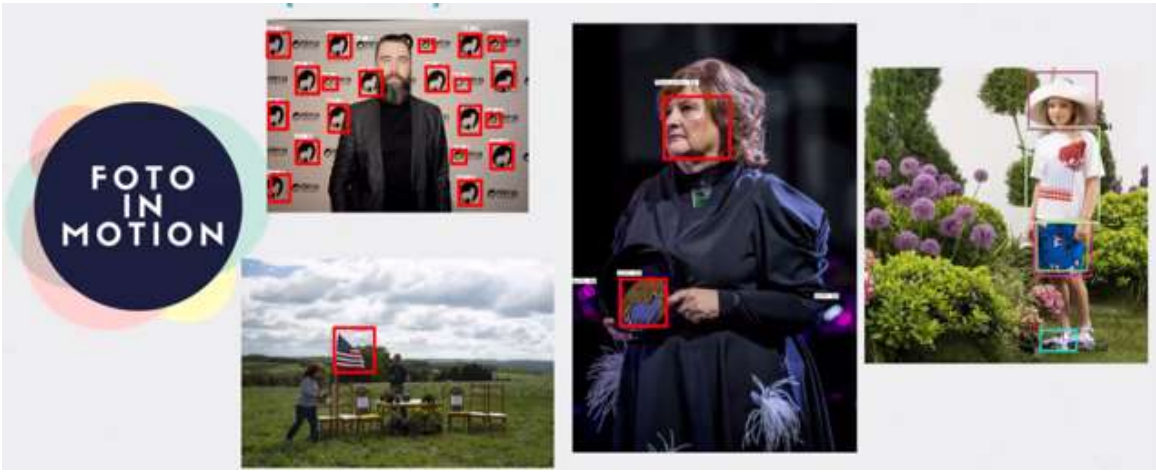
Equipa de IA aplicada a Recursos Humanos do MEIA usa Deep Learning para avaliação do Desempenho de um candidato numa entrevista

Para saber mais sobre Visão por Computador

- **Aprendizagem Automática 2** decorre em paralelo com a unidade curricular de **Língua Natural e Sistemas Conversacionais (LNASCIA)**
- Foi realizada no ano passado uma Master Classes on AI sobre Visão por Computador
  - Profª Paula Viana (ISEP e INESCTEC) – Uso de IA e Visão por Computador em Conteúdos Multimedia

MEIA
MESTRADO ENGENHARIA INTELIGÊNCIA ARTIFICIAL

AI Master Class
MEIA Series



Projeto Foto In Motion (Paula Viana) usa Redes Neuronais Convolucionais para detetar itens em vídeos e conteúdos Multemedia (por exemplo, detetar quando aparecer uma bandeira dos EUA num dado vídeo)

6 - Domínios da IA
6.2 - Robótica Inteligente

Carlos Ramos

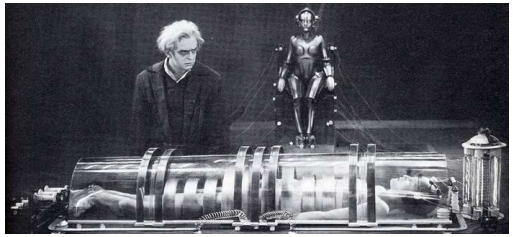
csr@isep.ipp.pt

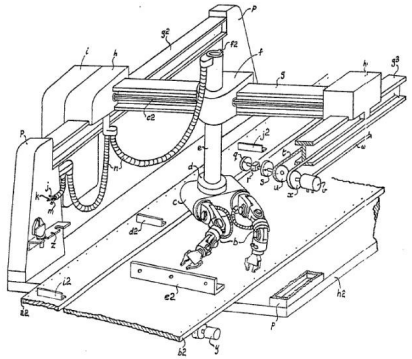
Unidade Curricular: Algoritmia Avançada

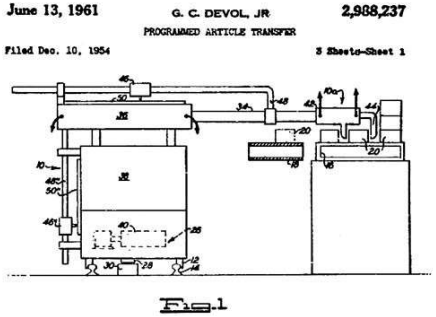
Curso: Licenciatura em Engenharia Informática (ISEP)

Origem da palavra Robot: da ficção aos primeiros robots

- Robota – palavra checa, significa trabalho, trabalho forçado
- Karel Capek escreve em 1921 a ficção “Rossum’s Universal Robot”
- Metropolis – filme de 1926 com robot Electro
- Isac Asimov escreve ficção “I Robot” e introduz o termo Robótica
- 3 leis de Asimov
  - 1ª lei: um Robot nunca poderá ferir um ser humano, ou permitir pela sua inatividade que um ser humano seja ferido
  - 2ª lei: um Robot deve obedecer a ordens dadas por seres humanos, exceto no caso de conflito com a 1ª lei
  - 3ª lei: um Robot deve proteger a sua própria existência, a menos de conflitos com as duas leis anteriores
- Em 1954 surgem 2 patentes ligadas ao surgimento dos Robots Industriais, de Cyril Kenward e de George Devol
- Em 1958 surge a primeira empresa robótica, a Unimation, e o robot UNIMATE é instalado na Ford em 1961

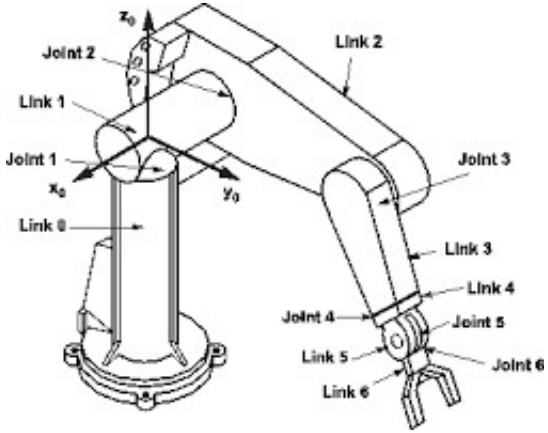


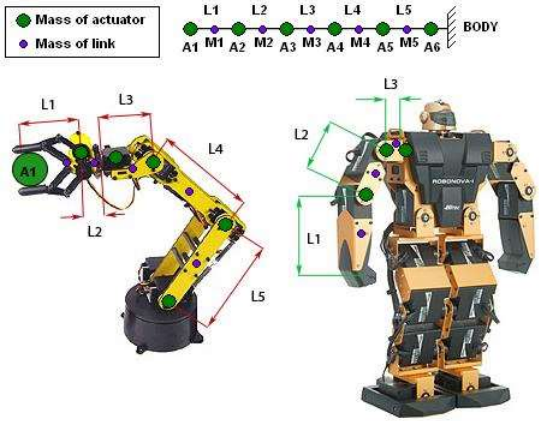




Anatomia Robótica

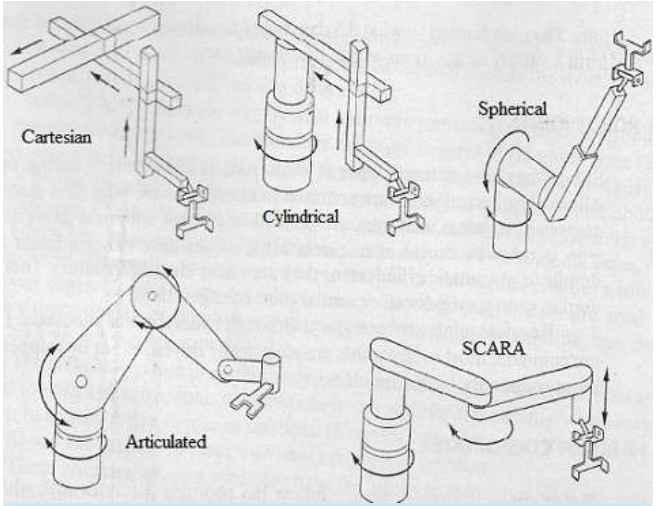
- Um Robot Industrial é constituído por várias partes rígidas (links) postas em série através de articulações (joints) de revolução (rotação) ou prismáticas (translação)
- Na extremidade encontra-se normalmente uma ferramenta (garra, pistola de pintura, ferramenta de corte, soldadura) relacionada com o trabalho útil do robot
- Cada articulação corresponde a um grau de liberdade (movimento)

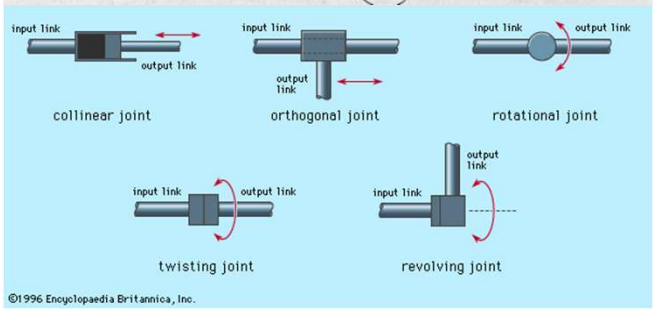


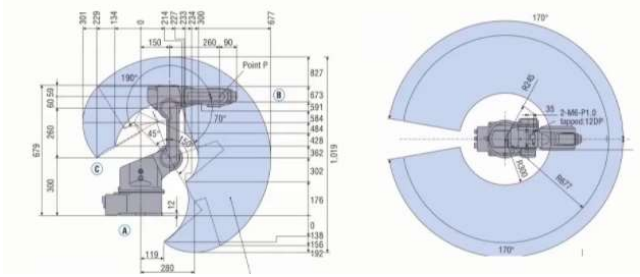


Anatomia Robótica

- Configurações Robóticas: Cartesiana, cilíndrica, esférica, articulada e tipo SCARA (adequada para montagens de precisão)
- As articulações permitem movimentos lineares (colinear ou ortogonal) ou rotacionais (eixos perpendiculares às 2 partes, colineares com as 2 partes ou colinear apenas com a parte de entrada)
- As configurações e tipos de articulação vão condicional o Espaço de Trabalho (Workspace) do Robot, o que implica possibilidades de colisão entre o robot e outras entidades do ambiente de trabalho, inclusive o ser humano (segurança)
- O movimento tem base em princípios elétricos (motor elétrico), hidráulicos ou pneumáticos

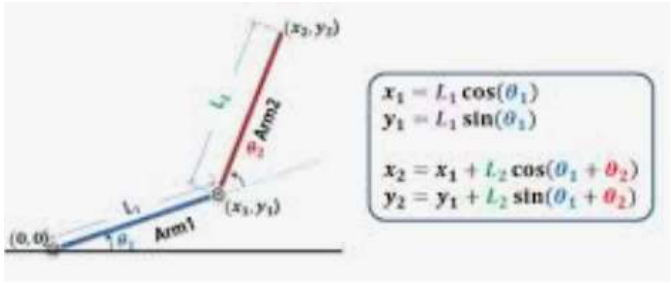


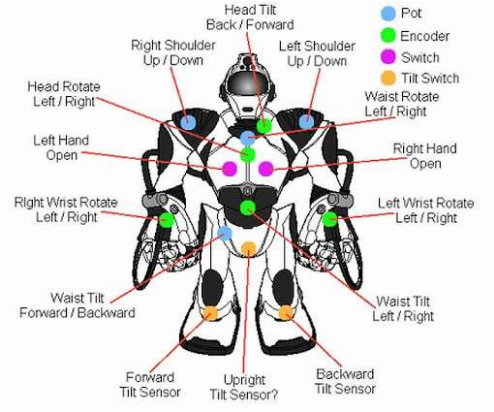




Movimento do Robot

- Cinemática Direta: Dadas as dimensões das partes dos robots (Li), seus tipos de movimentos, nomeadamente ângulos de rotação (Θi) saber qual a posição da extremidade do robot no espaço (x,y,z) da extremidade do robot
- Cinemática Inversa: Dada uma determinada posição no espaço pretendida para a extremidade do robot (x,y,z) saber qual ou quais as combinações de valores associados às partes (Li) e articulações (Θi) que levam a esse ponto
- Há ainda que lidar com a Física (Dinâmica direta ou inversa do robot) lidando com grandezas como a força ou o binário (torque) aplicados às várias partes do robot
- Há um conjunto de sensores internos do robot (baseados em potenciómetros, resolvers, codificadores) e sensores de grandezas como a velocidade (tacómetros) que vão monitorar os valores das diversas grandezas físicas dentro do robot





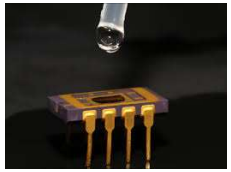
Robótica Inteligente

- Embora Robótica e IA sejam temas que por vezes aparecem juntos têm origem em "Escolas" diferentes:
  - Robótica - origem nas áreas de Engenharia Mecânica, Engenharia Eletrotécnica/Eletrónica, Mecatrónica, Engenharia de Produção, etc
  - Inteligência Artificial – origem em várias áreas, nomeadamente Ciências da Computação, Psicologia, Neurociências, etc
- Os robots das décadas de 50 e 60 eram essencialmente não-inteligentes, repetiam movimentos e operações de modo incansável, mas não envolviam nenhuma adaptação à situação nem eram capazes de produzir raciocínio elaborado
- Com a introdução de sensores nos robots (feita pelos grupos de IA nos anos 60 no MIT, nomeadamente com a Visão por Computador pelo grupo de Marvin Minsky) começa a fazer sentido dotar os robots de Inteligência
- A **Robótica Inteligente** é a área da conjunção da Robótica e da IA que permite dotar os Robots de capacidades inteligentes, permitindo nomeadamente que eles interpretem o meio envolvente a partir de informação dos sensores, efetuem raciocínios elaborados e atuem em mundos não estruturados, aprendendo com a interação com tais ambientes. Os Robots Inteligentes combinam comportamentos deliberativos com comportamentos reativos, de modo a atingirem seus objetivos e lidarem com situações não previstas

Sensores Externos de um Robot

Os sensores externos têm como objetivo obter dados e informação sobre o meio envolvente ao robot

- Câmeras: permitem obter imagens digitais, o canal mais rico de informação (iremos abordar em Visão por Computador)
- Sensores de Contato
- Sensores de Proximidade: Infravermelhos, ultrasons, fotoresistências
- Sensores de Distância: Ultrasons, Infravermelhos, Laser
- Sensores de Pressão
- Sensores de Tilt: para orientação e inclinação
- GPS: para posicionamento
- Sensores de Localização: com beacons que geram sinais
- Sensores de Aceleração: Força Estática e Dinâmica
- Sensores de Inércia
- Sensores com Giroscópio
- Sensores de Luz
- Sensores de Som
- Sensores de Componentes Químicos
- etc



Atuadores de um Robot

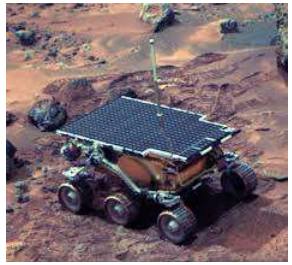
Os atuadores dos robots permitem que estes efetuem atuações sobre o meio envolvente do robot

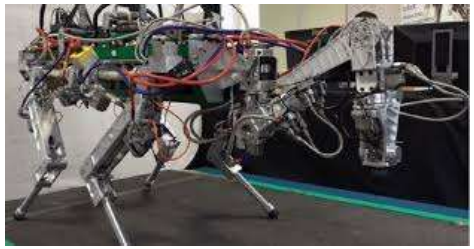
- Grippers, garras, mãos artificiais com menor ou maior destreza são usuais em braços robóticos para tarefas de manipulação ou montagem
- No contexto de robots móveis o próprio braço robótico pode ser visto como um atuador
- Em contexto industrial podem ser aplicadas ferramentas à extremidades dos braços robóticos (pistolas de pintura, ferramentas de corte ou de soldadura, sistemas de aparafusamento, etc)

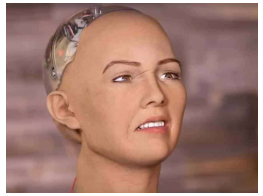
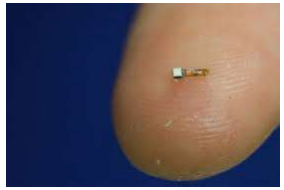
Diferentes implementações físicas e aplicações de Robots

Embora a Robótica tenha começado pelos braços robotizados, seguiram-se outras concretizações físicas

- AGV (Automatic Guided Vehicles) em contexto de espaços planos sobretudo em ambiente fabril para transporte
- Robots Móveis para exploração de espaços perigosos ou não acessíveis ao ser humano (espaço, submarino, centrais nucleares)
- Eletrodomésticos (Robot Aspirador)
- Legged Systems (com pernas)
- Robots Antropomórficos
- Carros Autónomos
- Drones
- Robots Cirurgiões
- Nanorobots

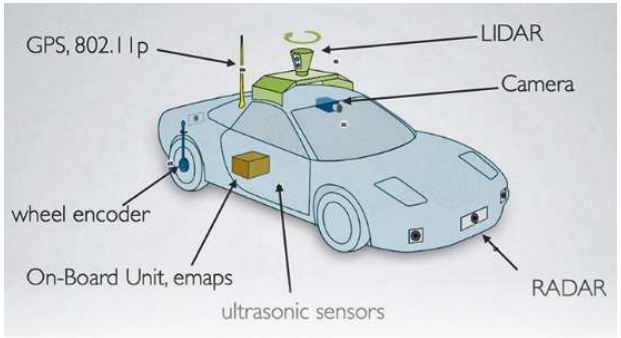


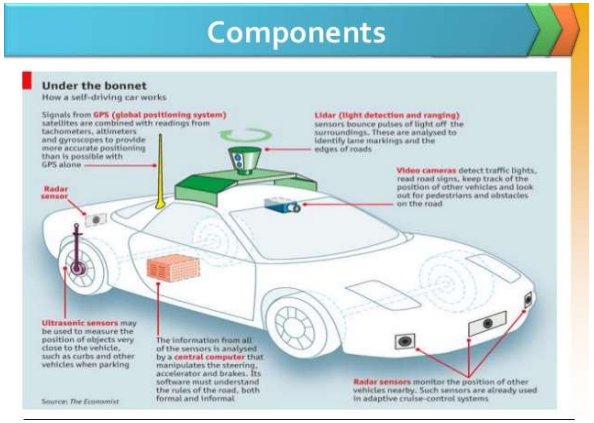


Veículos Autónomos

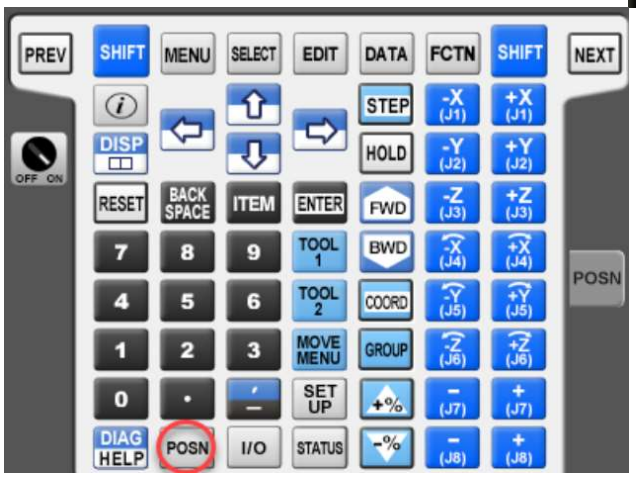
- Nas próximas décadas os carros conduzidos por humanos serão substituídos por carros autónomos
- Combinam diversos sensores (GPS, LiDAR, Radar, câmeras) efetuando a fusão sensorial, lidam com informação incerta, incorreta ou mesmo com ausência de informação
- Condução Autónoma mais segura que a condução humana





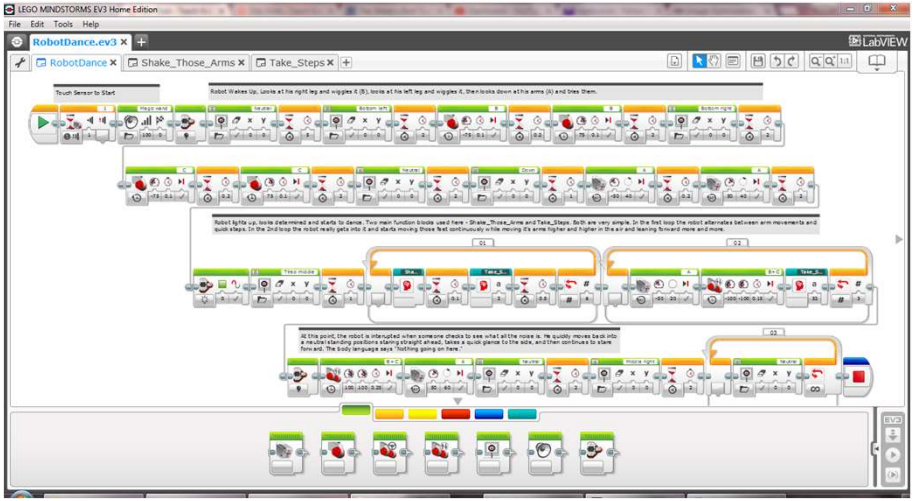
Programação de Robots por Aprendizagem

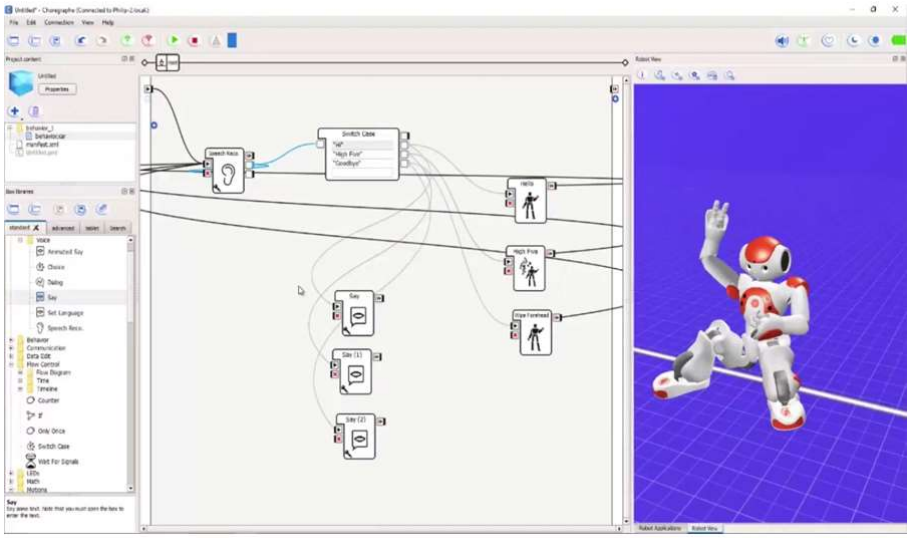
- Podem ser programados por aprendizagem, movimentamos as partes do robot através das suas articulações para uma determinada posição e memorizamos ponto a ponto. No final o robot repete a sequência de pontos memorizados (usual para programar tarefas repetitivas na indústria)



Programação Visual de Robots

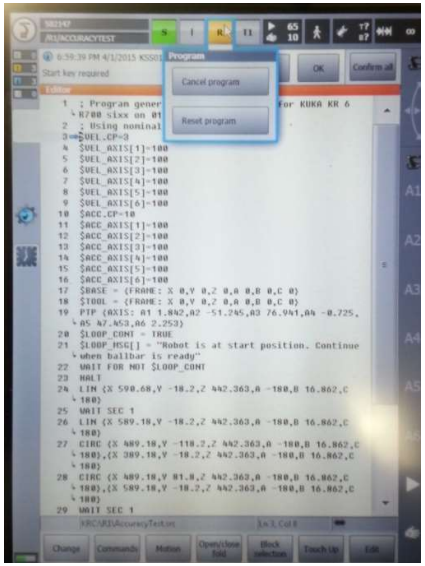
- A programação visual de robots é muito usual em robots de entretenimento (Mindstorms, NAO) e pretende de um modo simples exibir uma estrutura em lista ou grafo de instruções robóticas

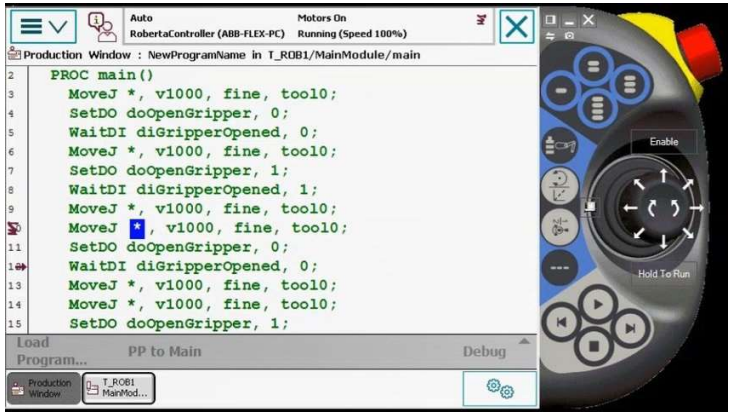




Programação com Linguagens Específicas de Robots

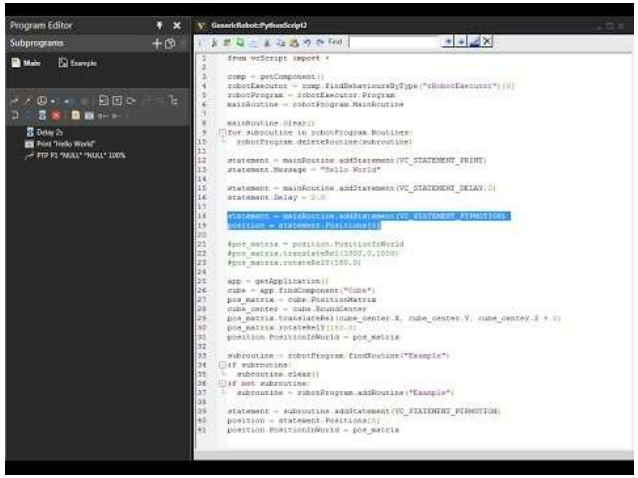
- As linguagens de programação específicas de robots dispõem de um conjunto de primitivas que comandam o movimento das diversas articulações do robot ou instruções para o robot mover a sua extremidade para uma determinada posição, abrir ou fechar uma garra, ler um sensor, etc

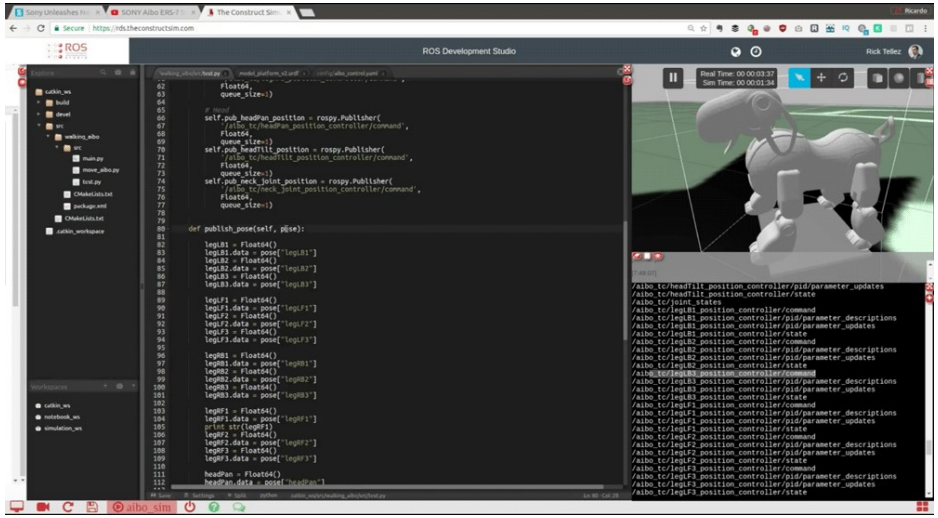




Programação de Robots com Linguagens Gerais

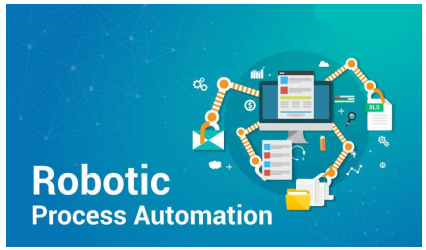
- Algumas linguagens como o Python já dispõem bibliotecas genéricas de programação de robots (Pyro, DART,…)
- ROS – Robot Operating System é uma linguagem/ambiente para a programação de robots de finalidade geral

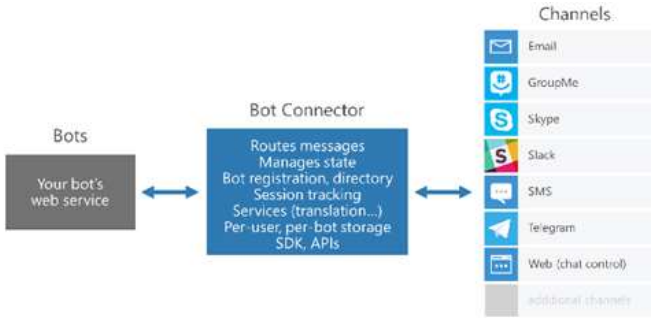




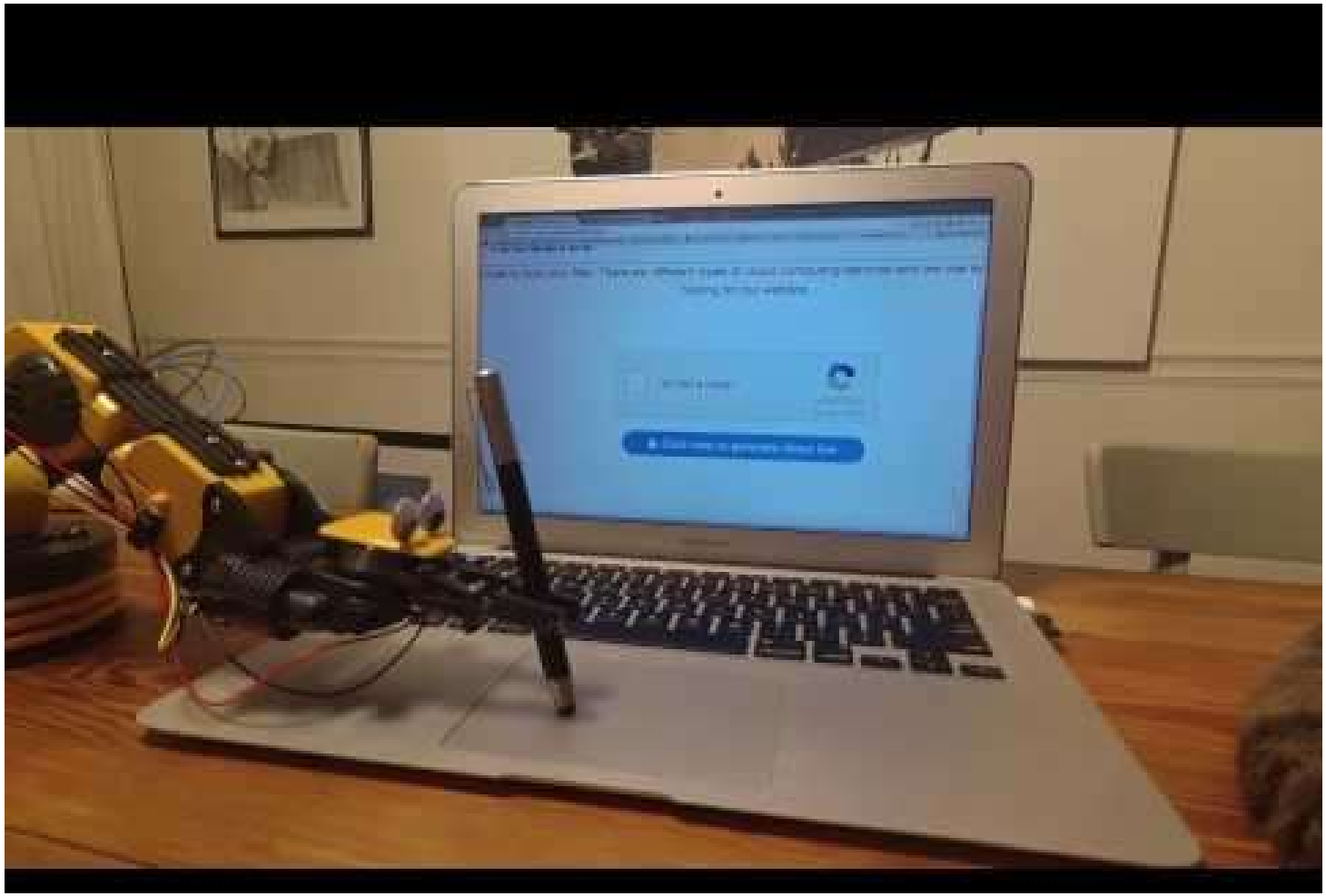
Nota Final: Isto não são os Robots que estivemos a falar!

- Robots têm existência física, ocupam espaço
- Interagem fisicamente com o meio envolvente
- Necessitam mecanismos de tração para movimentarem-se e sensores internos para ajudar no respetivo controlo
- Usam sensores externos para aperceberem-se da realidade física que os rodeia
- Usam atuadores (garras, ferramentas) para alterar o estado do mundo que os rodeia



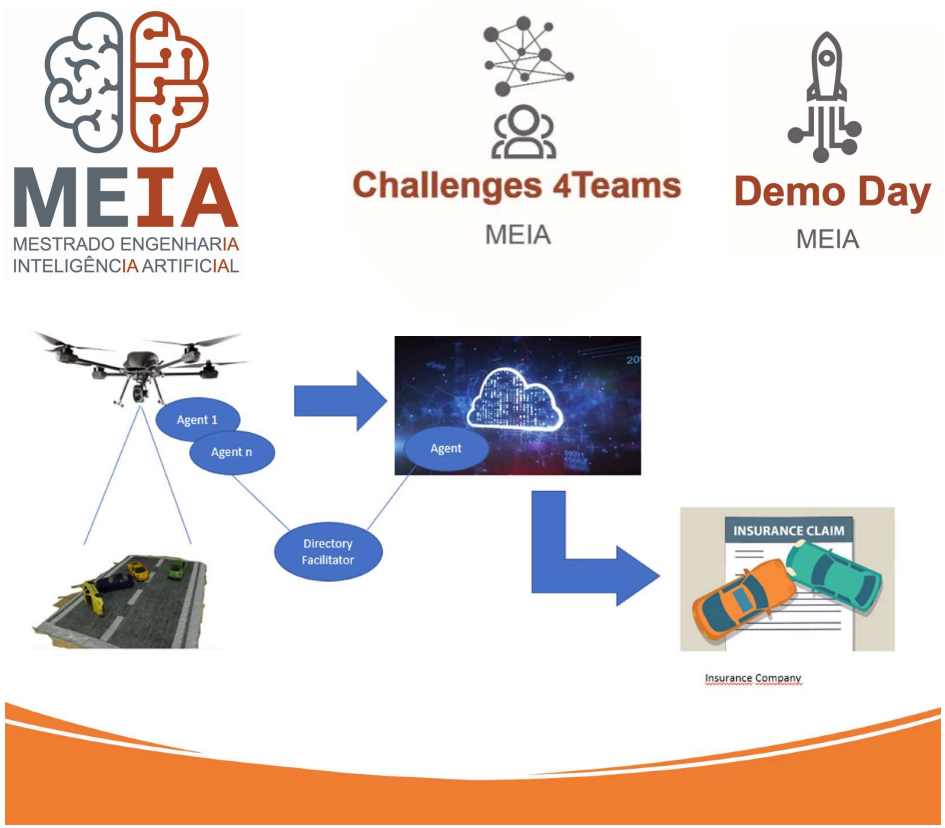


I’m not a Robot



Para saber mais sobre Robótica Inteligente

- Unidade Curricular de **Ambientes Inteligentes (AMBIIA)** do **Mestrado em Engenharia de Inteligência Artificial (MEIA)** do ISEP
- AMBIIA tem 2 partes principais: Robótica e IoT
- Na parte de Robótica é feito um estudo mais aprofundado dos conceitos introduzidos nesta aula, são introduzidos novos conceitos e são utilizadas várias ferramentas de programação de robots reais (linguagem RAPID para robots da ABB; linguagem ROS) ou ambientes simulados (MORSE para edição de ambientes tridimensionais)
- Dois docentes: Ramiro Barbosa (GECAD) e André Dias (LSA – Laboratório de Sistemas Autónomos, grupo de I&D que desenvolve robots de diversos tipos, nomeadamente Aquáticos, Submarinos e Drones)



Sistema Multiagente para análise de acidentes rodoviários desenvolvido pela equipa de IA aplicada a Seguros e Finanças do MEIA

Para saber mais sobre Robótica Inteligente

- **Ambientes Inteligentes (AMBIIA)** decorre em paralelo com a unidade curricular de **Sistemas Multiagente (SMAGIA)**
- Foram realizadas no ano passado 3 Master Classes on AI combinando Robótica com Sistemas Colaborativos e Multiagente
  - Ricardo Oliveira (ABB) – Robótica Colaborativa
  - Paulo Leitão (IPB) – Inteligência Distribuída em Sistemas Ciberfísicos
  - Luís Paulo Reis (FEUP) – Sistemas Multiagente em Futebol Robótico

MEIA
MESTRADO ENGENHARIA INTELIGÊNCIA ARTIFICIAL

AI Master Class
MEIA Series

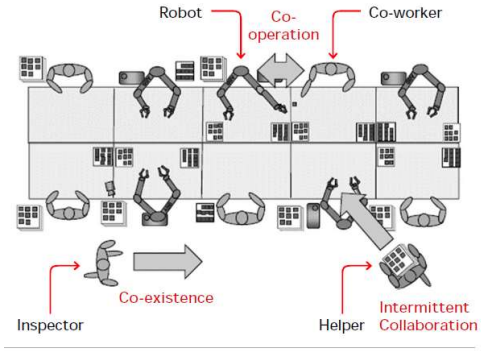


Imagem da Master Class de Ricardo Oliveira

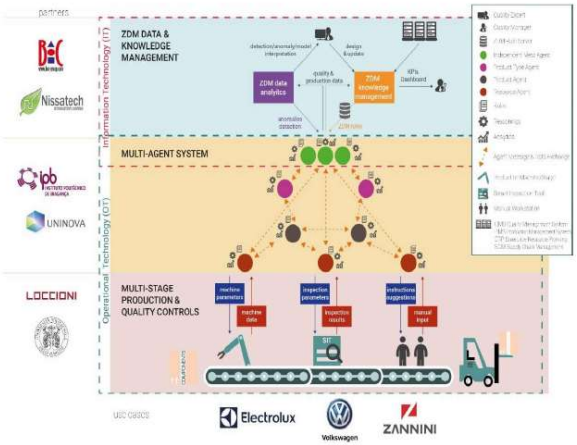


Imagem da Master Class de Paulo Leitão

Imagem da Master Class de Luís Paulo Reis

6 - Domínios da IA
6.4 - Aprendizagem Automática

Carlos Ramos

csr@isep.ipp.pt

Unidade Curricular: Algoritmia Avançada

Curso: Licenciatura em Engenharia Informática (ISEP)

Visão Geral da Aprendizagem Automática

- A Aprendizagem Automática (Machine Learning) é uma área da Inteligência Artificial
- Ao longo dos anos tem havido contributos de outros domínios científicos:
  - Reconhecimento de padrões (Pattern matching)
  - Estatística
  - Sistemas de Informação
- E…vários nomes
  - Extração e Descoberta de Conhecimento (Data Mining, Knowledge Discovery)

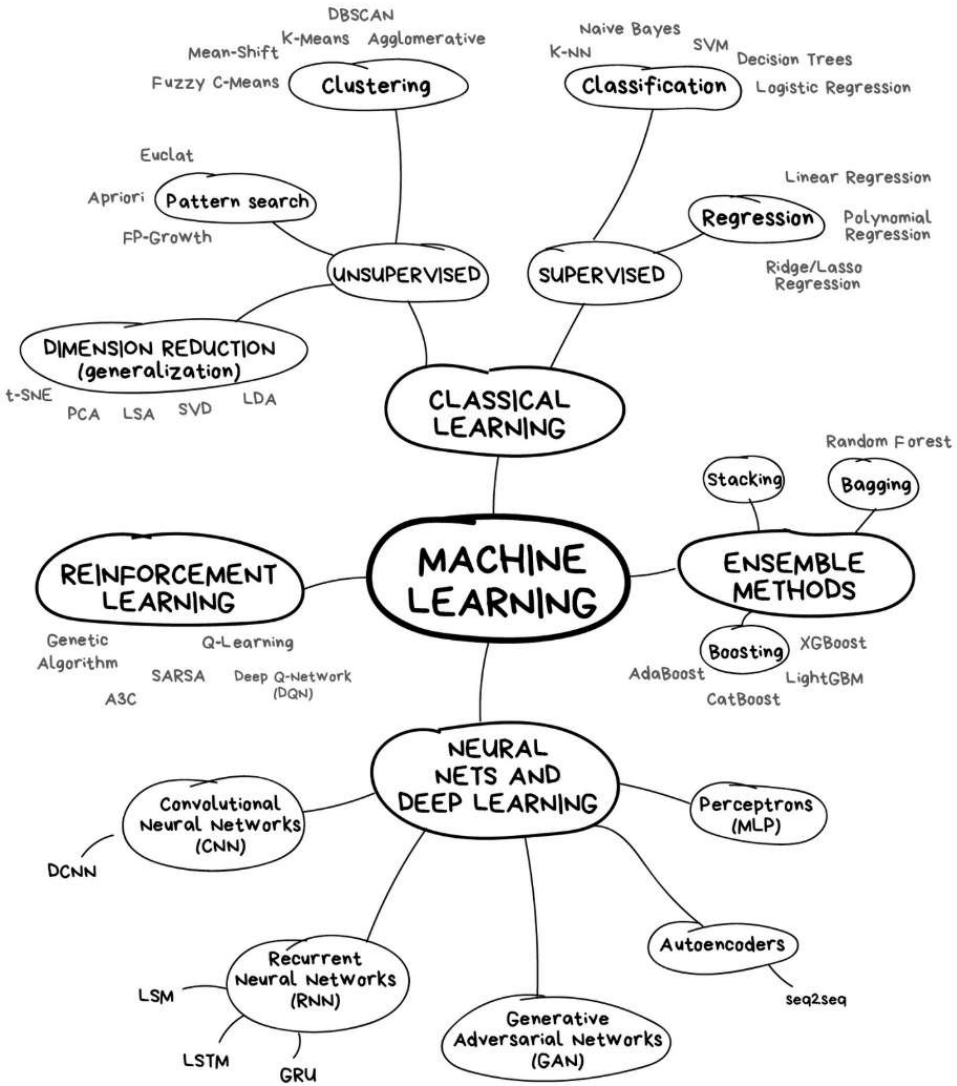


Figura retirada de https://vas3k.com/blog/machine_learning/

Visão Geral da Aprendizagem Automática

- As Redes Neuronais Artificiais (ANN), ou simplesmente Redes Neuronais são talvez o método mais aplicado e conhecido
- As Redes Neuronais com Aprendizagem Profunda (Deep Learning) têm tido grande sucesso nos últimos anos, mas são Redes Neuronais
- Por vezes ouvimos dizer "aplicamos técnicas de Inteligência Artificial, Machine Learning, Redes Neuronais e até Deep Learning", a figura ao lado esclarece que temos conceitos que estão embutidos uns nos outros. Deep Learning é um tipo de Rede Neuronal que é uma das abordagends de Machine Learning que é há muitos anos o termo usado pela Inteligência Artificial para as metodologias que levam a sistemas que têm a capacidade de aprender de forma autónoma, tal como o ser humano é capaz de fazer
- Mas os sistemas de Aprendizagem Automática também têm os seus problemas, nomeadamente:
  - Exigem um grande esforço no pré-processamento dos dados (dados erróneos, incompletos, incertos)
  - Dificuldade de lidar com situações para as quais não foram treinados
  - Dificuldade de Explicar como obtêm as suas conclusões

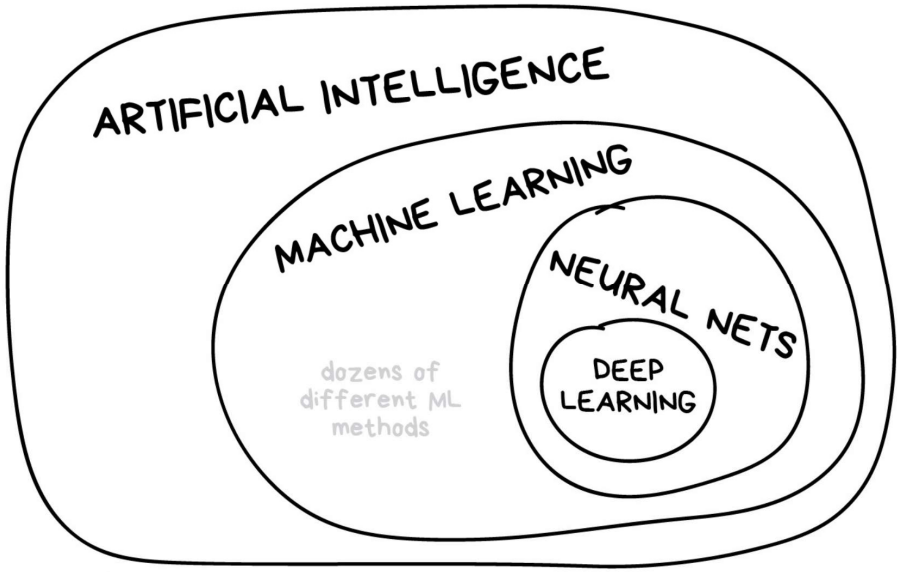
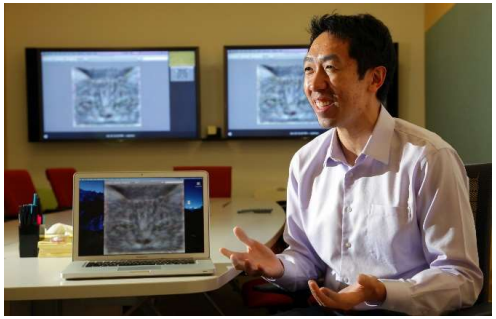


Figura retirada de https://vas3k.com/blog/machine_learning/

Áreas da IA (algumas…)

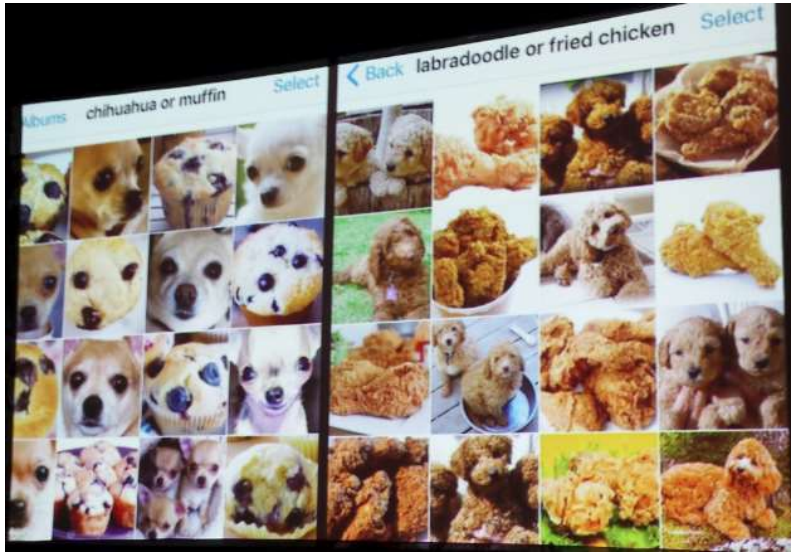
- Inteligência Artificial
  - Língua Natural
  - Teoria de Jogos
  - Robótica Inteligente
  - Visão por Computador
  - Otimização e Planamento Automático
  - Sistemas Baseados em Conhecimento
    - Sistemas Periciais
  - Aprendizagem Automática (Machine Learning)
    - Redes Neuronais
      - Aprendizagem Profunda (**Deep Learning**)
  - Agentes
  - Ambient Intelligence
  - …



Aqui há gato !
Andrew Ng – Stanford University, Coursera, Baidu
1000 computadores para treinar uma rede com 1 B conexões capaz de detetar itens em videos

As técnicas de Aprendizagem Automática têm tido muito sucesso, mas também têm também suas falhas e limitações

Incluindo Deep Learninig!
2 Examplos de Aprendizagem Automática para Imagem/Visão por Computador:



Apenas identificam aquilo para que foram treinadas

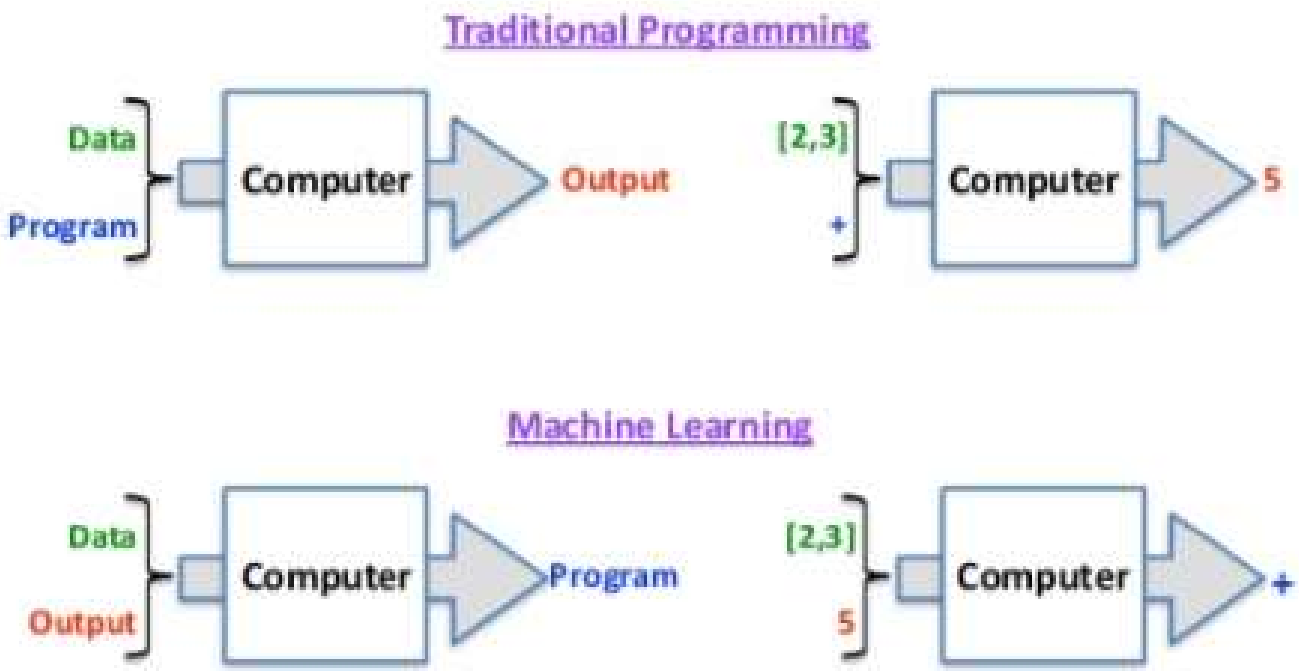
São capazes de efetuar classificação, previsão, etc
Mas têm dificuldade em compreender ou explicar

São um tipo de Black Box

Não são propriamente uma caixa mágica onde se colocam os dados na entrada e sai conhecimento na saída

Aprendizagem Automática (Machine Learning)

- ML representa uma maneira diferente de desenvolver sistemas



Métodos Clássicos de Aprendizagem Automática

- **Aprendizagem Supervisionada** – alguém indicou previamente o que é importante encontrar
  - **Classificação** – Prever se um determinado item cai numa ou noutra categoria já identificada anteriormente
    - Por exemplo. classificar numa imagem de uma cão qual é a sua raça (chiuaua ou labrador)
  - **Regressão** – usamos os dados que temos para adaptar a melhor forma que os representa
    - Por exemplo, dado um conjunto de pontos num gráfico y=f(x) determinar qual a reta que melhor os aproxima
- **Aprendizagem não Supervisionada** – não foi indicado previamente o que se pretendia encontrar, o sistema terá de encontrar padrões por si só
  - **Clustering** – Não tem classes pré-definidas, o objetivo é criar essas classes
    - Por exemplo, segmentar os clientes de um determinado produto
  - **Redução de Dimensão ou Generalização** – agrupa categorias conhecidas em outras mais gerais
  - **Associação** – associa itens
    - Por exemplo, quem compra uma impressora também comprará toners ou quem compra carnes vermelhas provavelmente comprará vinhos tintos

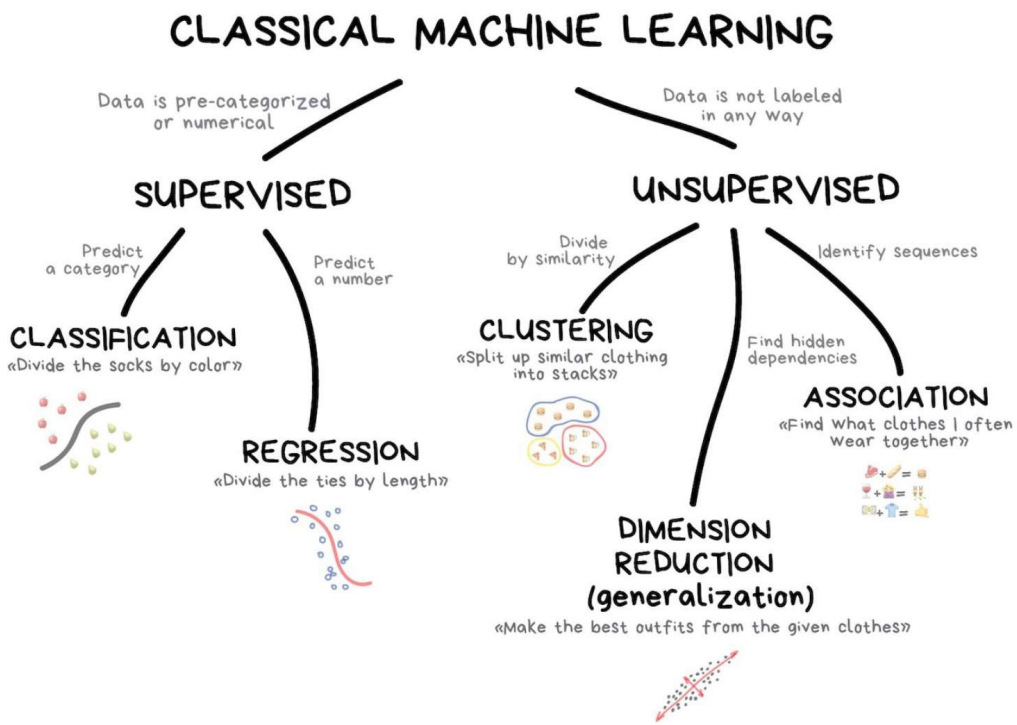


Figura retirada de https://vas3k.com/blog/machine_learning/

Uma boa listagem de algoritmos de ML está em https://machinelearningmastery.com/a-tour-of-machine-learning-algorithms/

Árvores de Decisão

- É usada uma representação em árvore
- São muito usadas em Sistemas Periciais para representar o conhecimento de especialistas num dado domínio
- Mas podem ser criadas automaticamente a partir de métodos de aprendizagem automática, sobretudo para problemas de classificação, nos quais usa-se um conjunto de dados rotulados para criar a árvore de decisão que contém uma sequência de perguntas se-então-senão sobre caraterísticas individuais
- Os principais métodos são o CART, ID3, C4.0, C5.0, CHAID, M5
- Permitem um certo nível de explicação

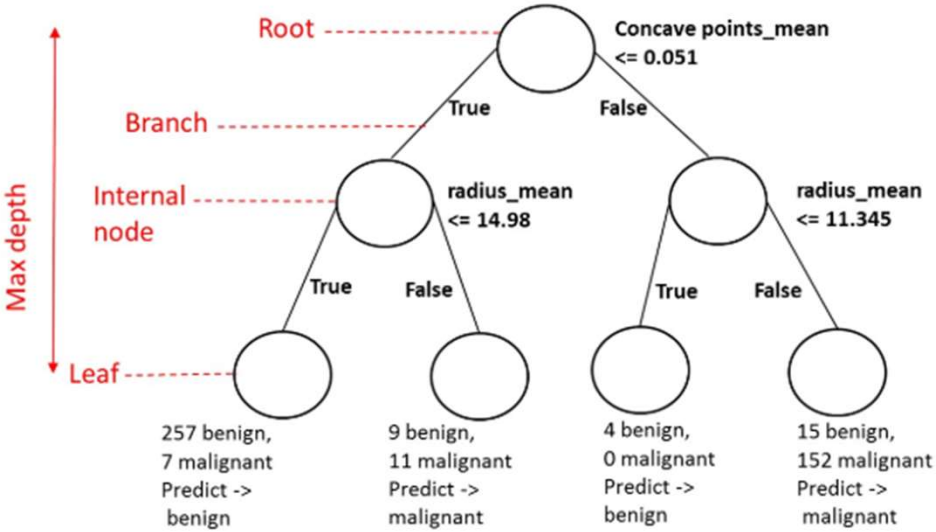


Figura retirada de http://cican17.com/machine-learning-with-tree-based-models/

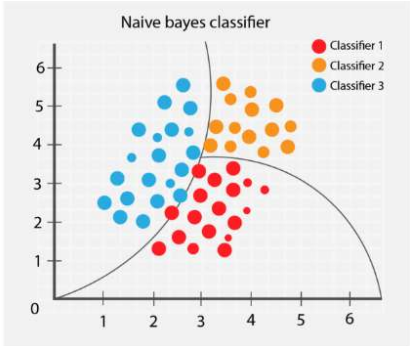
Naïve Bayes

- o classificador Naïve Bayes pertence à família de classificadores de probabilidade, usando o teorema de Bayes
- o motivo pelo qual é chamado de "Naïve" é porque requer uma rígida suposição de independência entre variáveis de entrada
- é um dos métodos populares para resolver problemas de categorização de textos (classificação), como o problema de julgar os documentos como pertencentes a uma categoria ou outra, como a deteção de spam no email
- outros métodos Bayesianos para aprendizagem automática são: Gaussian Naive Bayes; Multinomial Naive Bayes; Averaged One-Dependence Estimators (AODE); Bayesian Belief Network (BBN); Bayesian Network (BN)

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

using Bayesian probability terminology, the above equation can be written as

$$\text{Posterior} = \frac{\text{prior x likelihood}}{\text{evidence}}$$



Source: https://thatware.co/naive-bayes/

Obs	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
X1	A	A	A	A	A	B	B	B	B	B	C	C	C	C	C
X2	S	M	M	S	S	S	M	M	L	L	L	M	M	L	L
Y	0	0	1	1	0	0	0	1	1	1	1	1	1	1	0

Prior Probability

P(Y=1) = 9/15 P(Y=0)=6/15

Conditional Probability

P(X1=A|Y=1)=2/9 P(X1=B|Y=1)=3/9 P(X1=C|Y=1)=4/9
P(X2=S|Y=1)=1/9 P(X2=M|Y=1)=4/9 P(X2=L|Y=1)=4/9
P(X1=A|Y=0)=3/6 P(X1=B|Y=0)=2/6 P(X1=C|Y=0)=1/6
P(X2=S|Y=0)=3/6 P(X2=M|Y=0)=2/6 P(X2=L|Y=0)=1/6

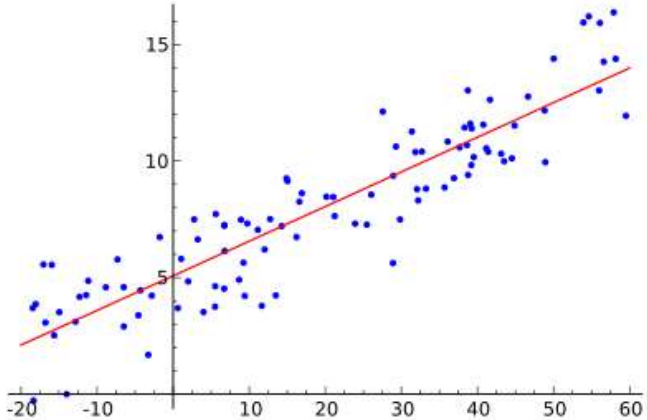
P(Y=1)P(X1=B|Y=1)P(X2=S|Y=1)=1/45
P(Y=0)P(X1=B|Y=0)P(X2=S|Y=0)=1/15

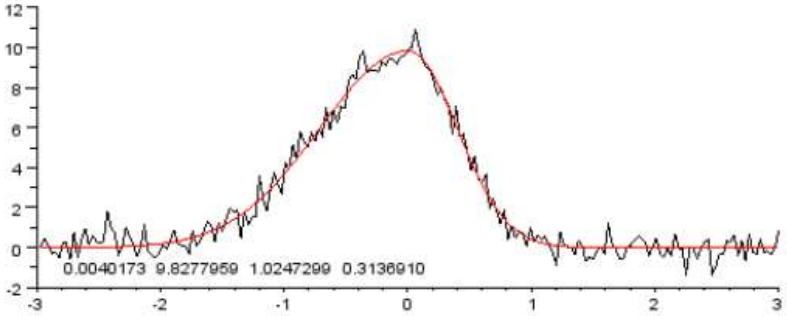
*P(Y=0)P(X1=B|Y=0)P(X2=S|Y=0)> P(Y=1)P(X1=B|Y=1)P(X2=S|Y=1) → **Y=0***

Exemplo retirado de https://towardsdatascience.com/introduction-to-na%C3%AFve-bayes-classifier-fa59e3e24aaf

Regressão

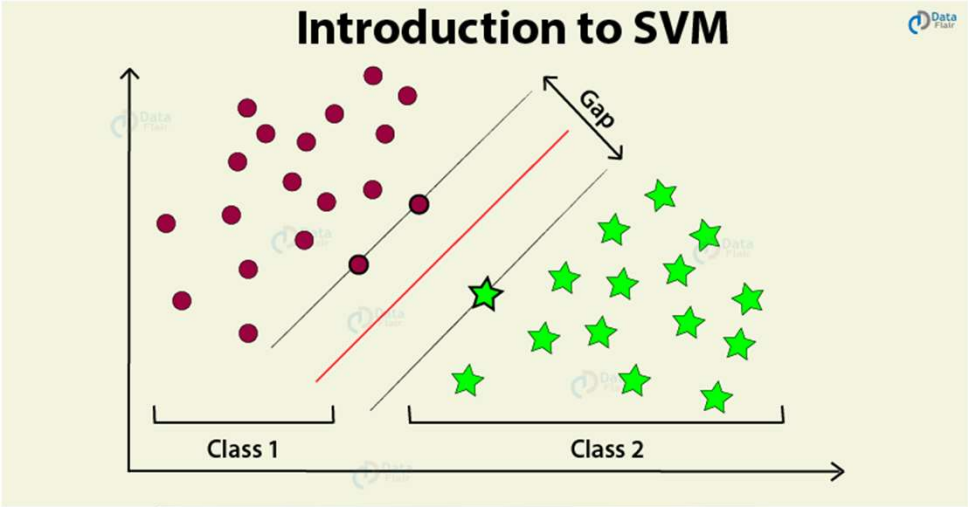
- Regressão Linear – aproximar um conjunto de pontos a uma reta, pode ser feita através do método dos mínimos quadrados
- Regressão Não-Linear – aproximação não linear, por exemplo usando uma função polinomial
- é costume remover outliers, pontos que se afastam muito do que seria previsível, geralmente devido a erros
- outros métodos de regressão são: Logistic Regression; Stepwise Regression; Multivariate Adaptive Regression Splines (MARS); Locally Estimated Scatterplot Smoothing (LOESS)





Support Vector Machines

- As máquinas de vetores de suporte (SVM) são um tipo de algoritmo de aprendizagem automática supervisionada que fornece análise de dados para classificação e regressão
- Embora possam ser usados para regressão, o SVM é usado principalmente para classificação
- Se analisarmos as características dos nossos dados num espaço n-dimensional o valor de cada uma terá coordenadas específicas
- Podemos encontrar o hiperplano ideal que diferencia entre as duas classes
- Dá-nos uma medida de segurança de separação entre duas classes



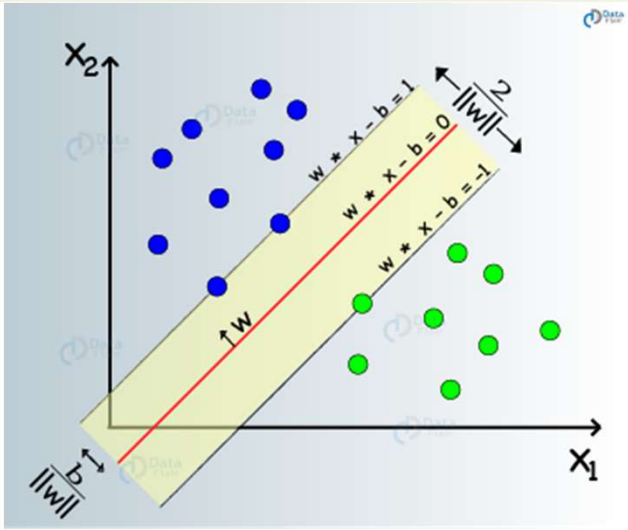


Figura retirada de https://data-flair.training/blogs/svm-support-vector-machine-tutorial/

K-Means

- O k-Means é um método de Clustering
- É definido o nº de clusters pretendido (k)
- o agrupamento é feito minimizando a soma das distâncias ao quadrado (distâncias euclidianas) entre os itens e o centróide correspondente
- Outros métodos de clustering são o k-Medians, Expectation Maximisation (EM) e Hierarchical Clustering

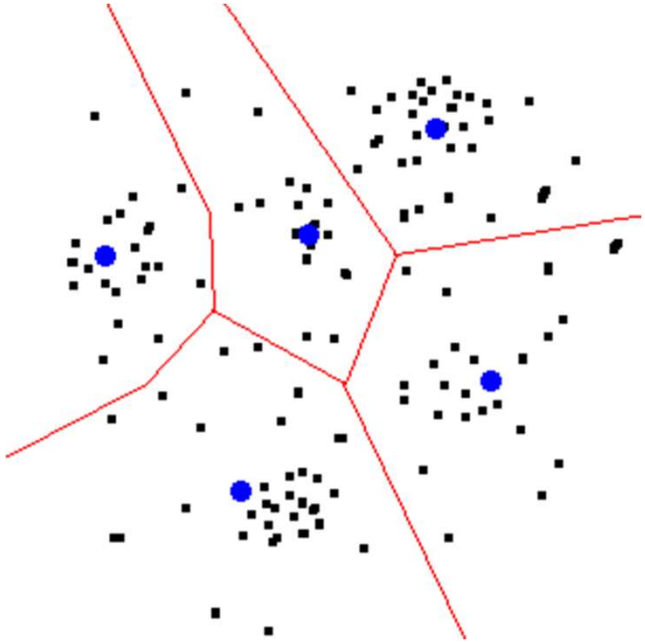


Figura retirada de http://mnemstudio.org/clustering-k-means-introduction.htm

Algoritmo Apriori

- O algoritmos Apriori é um método de Associação
- Um bom exemplo é fazermos associações de compras num supermercado, por exemplo, quem compra o produto X também compra o produto Y
- Define-se o nível de suporte, ou seja, o mínimo de itens nos dados que podem suportar uma associação (2 no exemplo ao lado, mas para exemplos reais teria de ser muito superior)
- Vê-se o suporte de produtos individuais, eliminam-se os que estão abaixo do nível de suporte, depois associam-se 2 a 2 e vê-se nos itens o suporte, eliminando-se os que estão abaixo do nível de suporte e assim por diante, até chegar ao maior nº de produtos que atendem ao nível de suporte
- Outro método alternativo para Associação é o Eclat

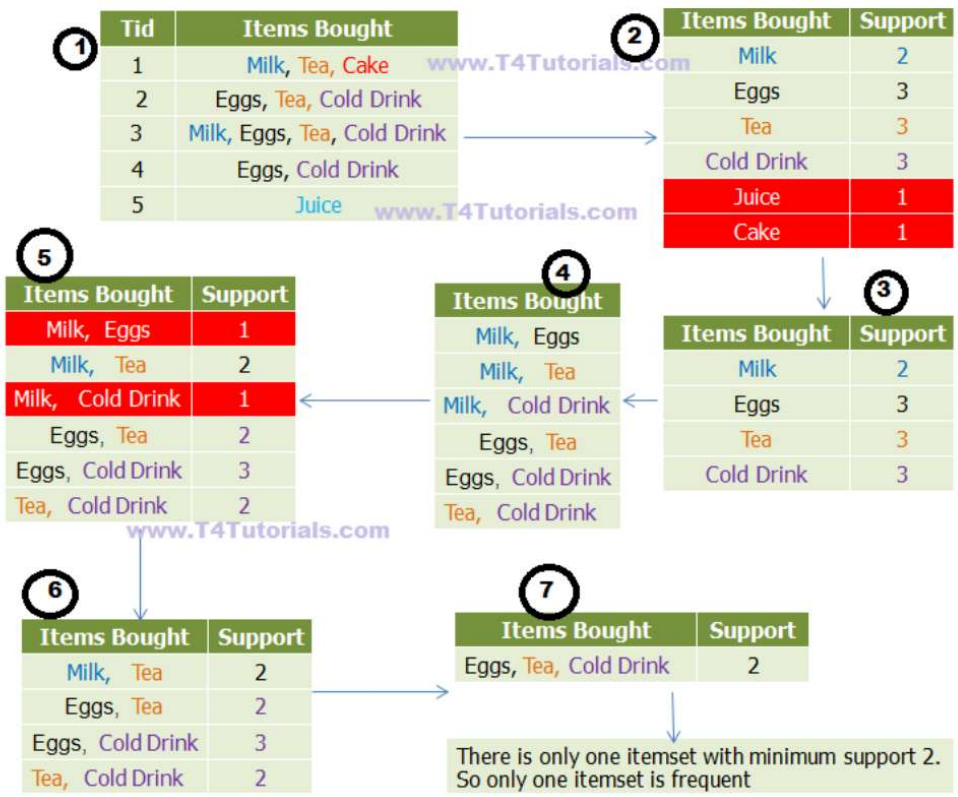


Figura retirada de https://t4tutorials.com/apriori-algorithm-in-data-mining-with-examples/

Análise de Componentes Principais (PCA)

- A Análise de Componentes Principais ou PCA é uma técnica usada para redução de dimensionalidade de um grande conjunto de dados
- Reduzir o número de componentes ou recursos custa alguma precisão e, por outro lado, torna um conjunto de dados grande mais simples, fácil de explorar e visualizar
- O método começa por fazer uma normalização, obtém vetores e valores próprios, aplica a matriz de covariância e seleciona as componentes principais, constrói a matriz de projeção e projeta-a no novo espaço com as componentes principais

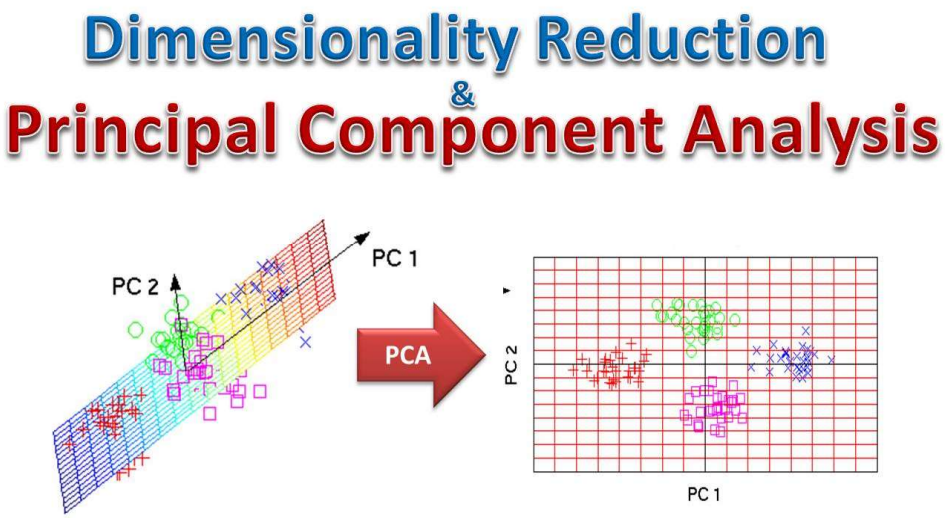
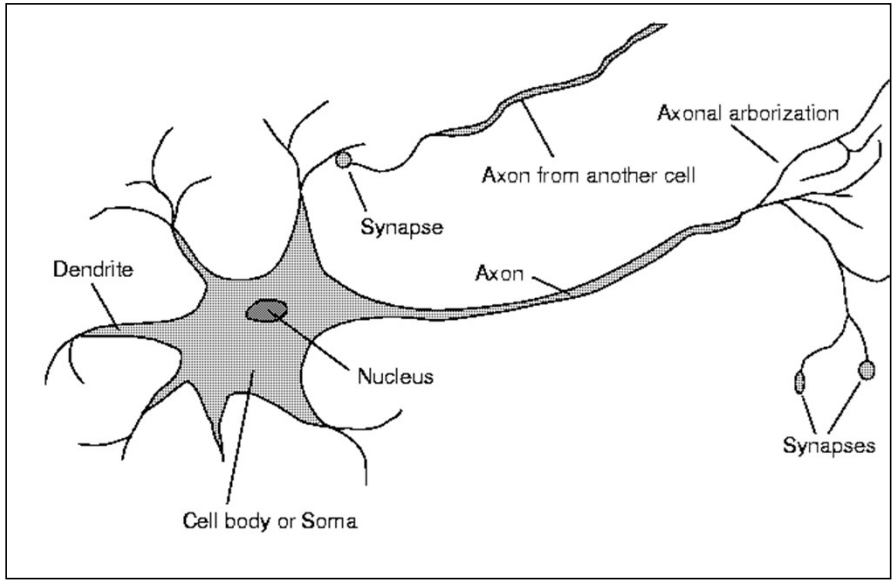


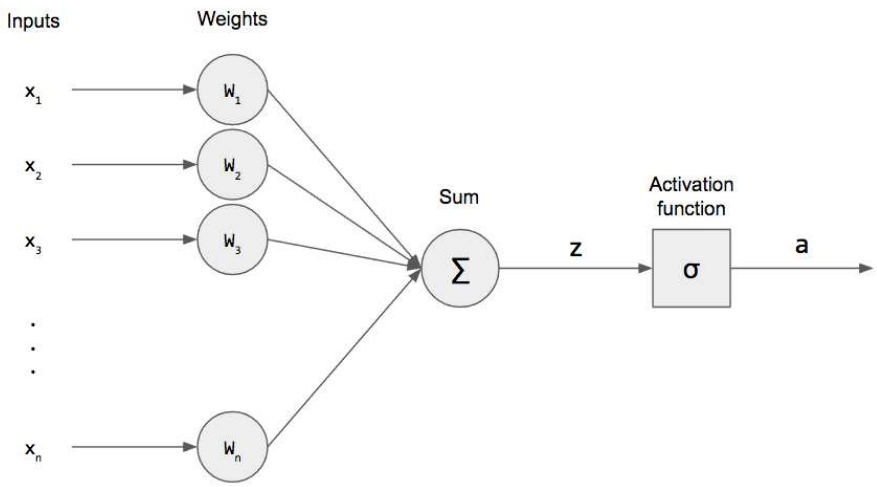
Figura retirada de https://ashutoshtripathi.com/2019/07/11/a-complete-guide-to-principal-component-analysis-pca-in-machine-learning/

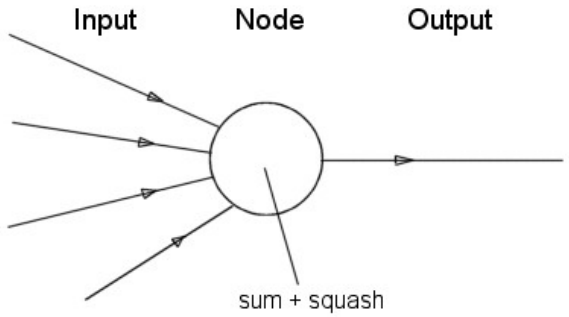
Redes Neuronais (Neural Networks)

- Nascemos com ~100 mil milhões de neurónios
- Um neurónio pode conectar cerca de 100 mil outros neurónios através de sinais (eletroquímicos)
- As sinapses (que ligam os neurónios) podem libertar uma dada quantidade de transmissores químicos
- O valor acumulado que vem de várias sinapses pode ativar o neurónio quando este ultrapassa um determinado valor de threshold



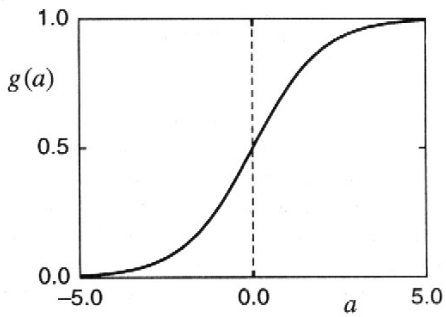
Perceptron e Funções de Ativação



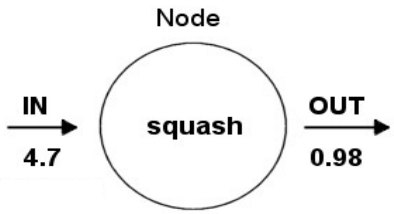


Simplificando a representação

Perceptron: produz uma saída resultante da aplicação de uma função de ativação sobre a somatório de um conjunto de entradas afetados por pesos



Função de Ativação



Funções de Ativação

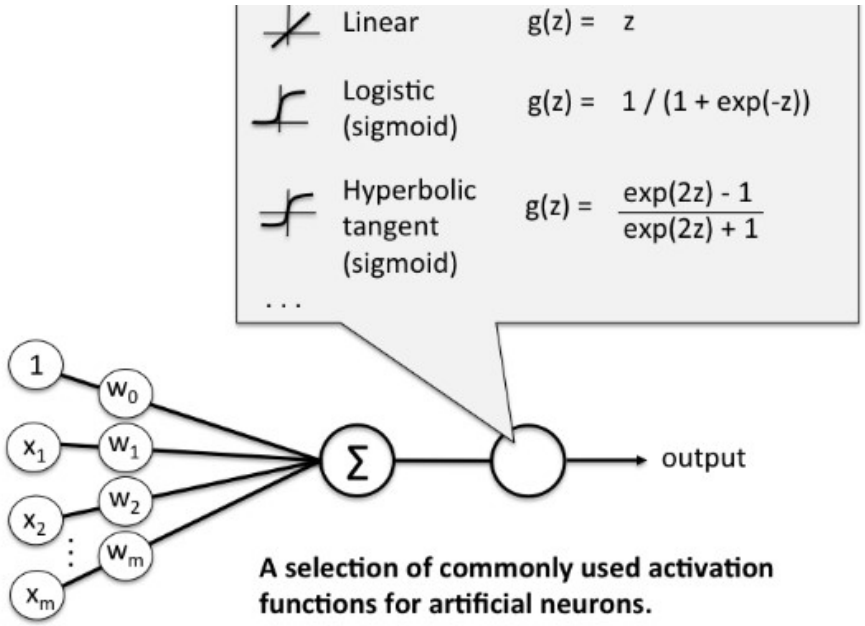
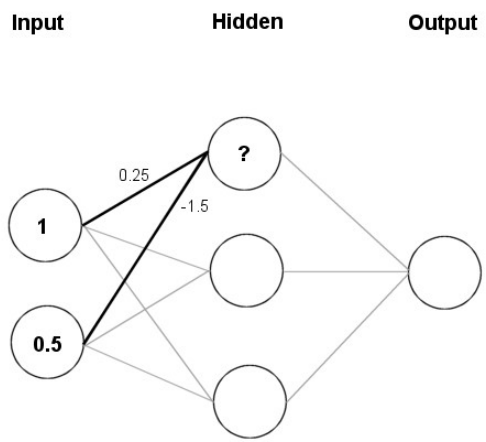


Imagem retirada de https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f

Alimentando a rede com dados:



$$(1 \times 0.25) + (0.5 \times (-1.5)) = 0.25 + (-0.75) \quad = \; - \mathbf{0.5}$$

Assumindo uma função de ativação

Sigmóide o valor à saída do neurónio é: $\dfrac{1}{1+e^{0.5}} = 0.3775$

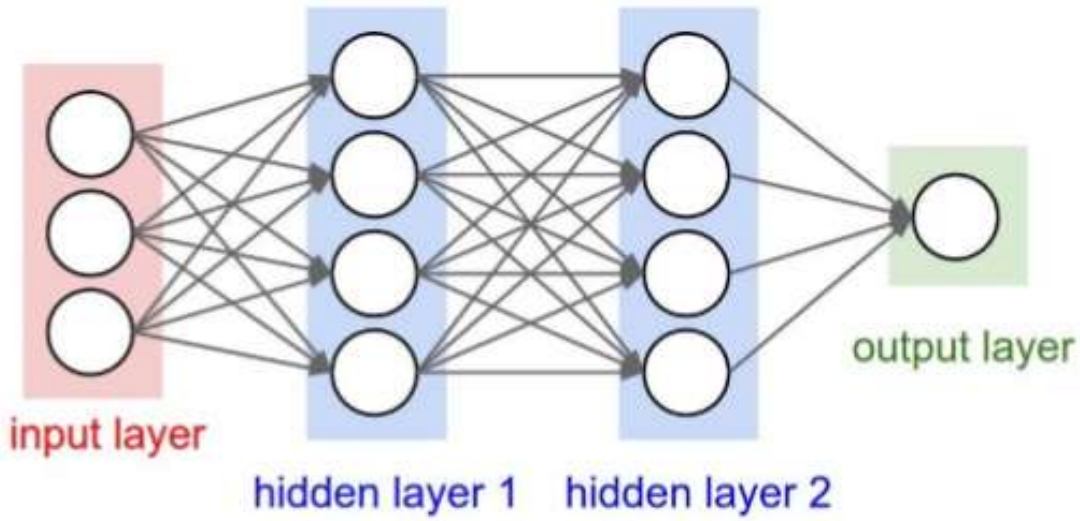
Redes Diretas (Feed-Forward)

O fluxo da informação é unidirecionall

- Os dados são apresentados na camada de entrada
- passam pelas camadas intermédias
- e ativam a camada de saída

A informação é distribuída

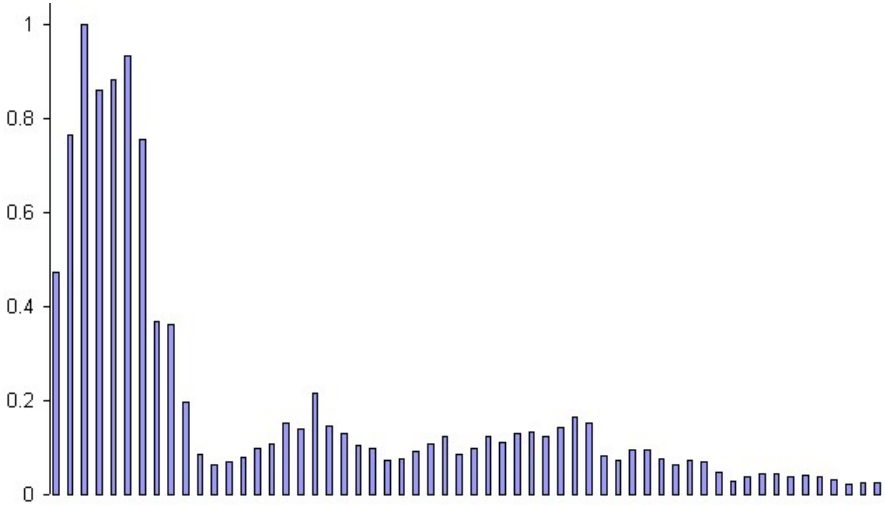
O processamento da informação pode ser em paralelo



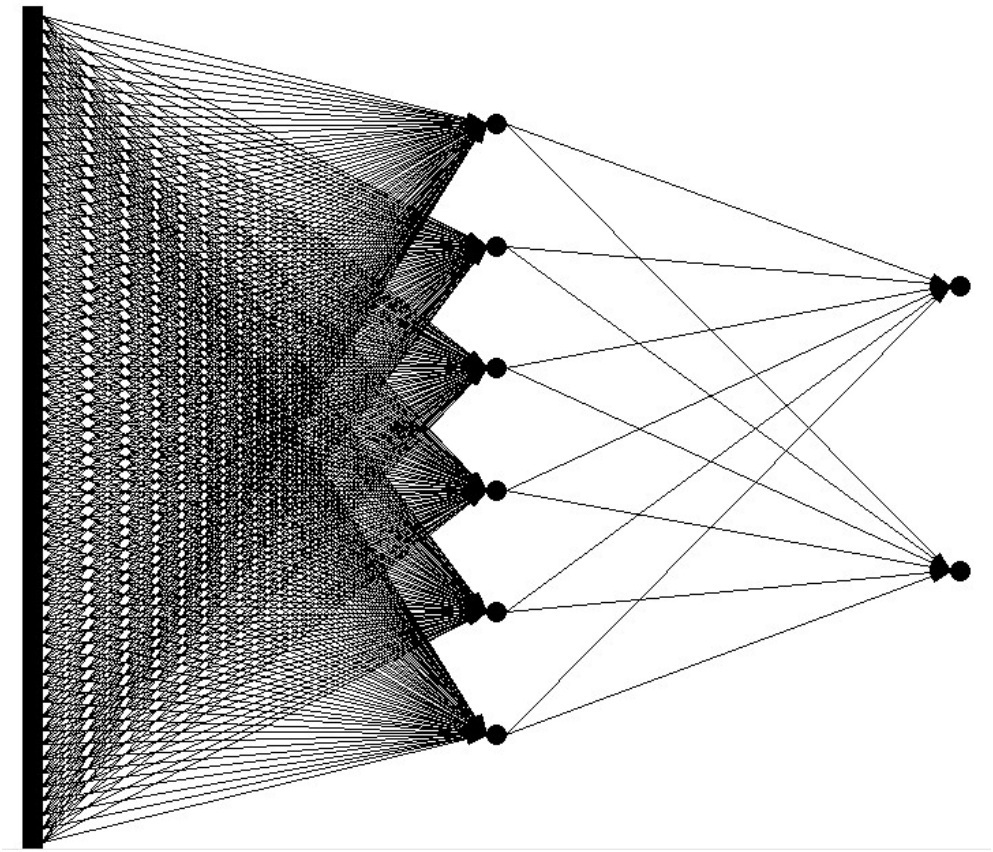
Exemplo: Reconhecimento de Voz

- Tarefa: Aprender a distinguir duas vozes diferentes dizendo “Hello”

- Dados
  - Fontes
    - Steve Simpson
    - David Raubenheimer
  - Formato
    - Distribuição no espetro de frequências(60 bins)

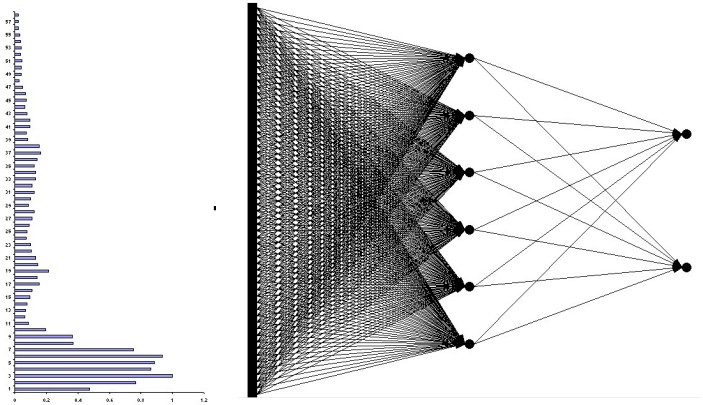


- Arquitetura da ANN
  - Feed forward
  - 60 entradas (uma para cada nível do espetro de frequências)
    - 6 camadas escondidas (hidden layers)
    - 2 saídas(0-1 para o Steve, 1-0 para o David)

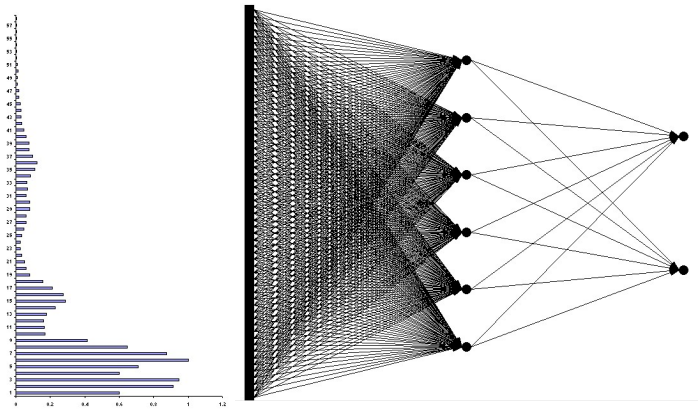


- Introduzindo os Dados

Steve

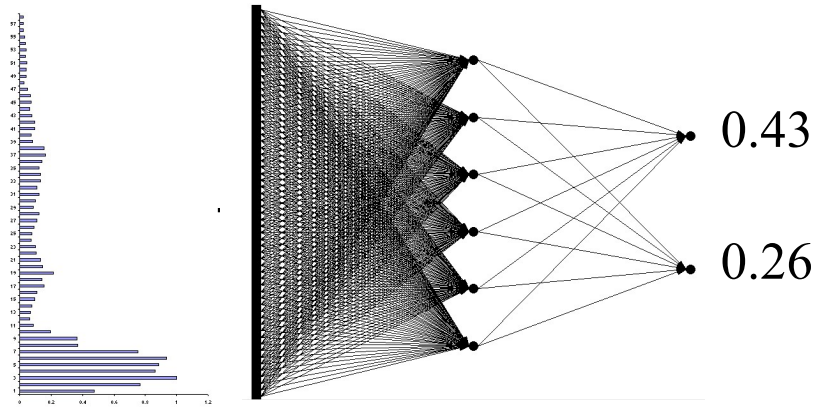


David

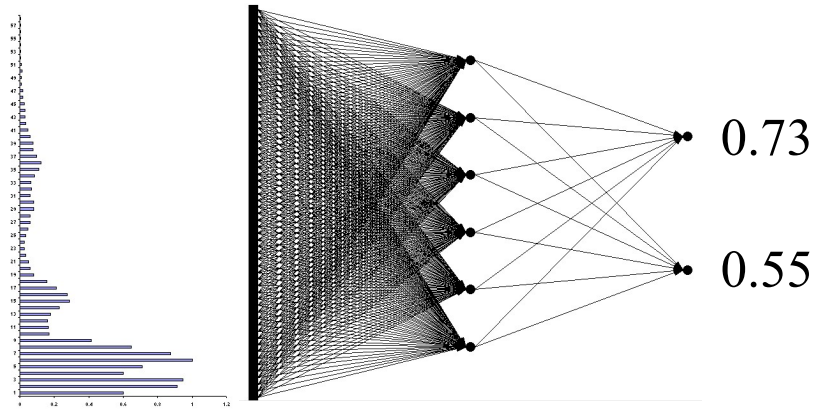


- Introduzindo os dados (rede não treinada)

Steve

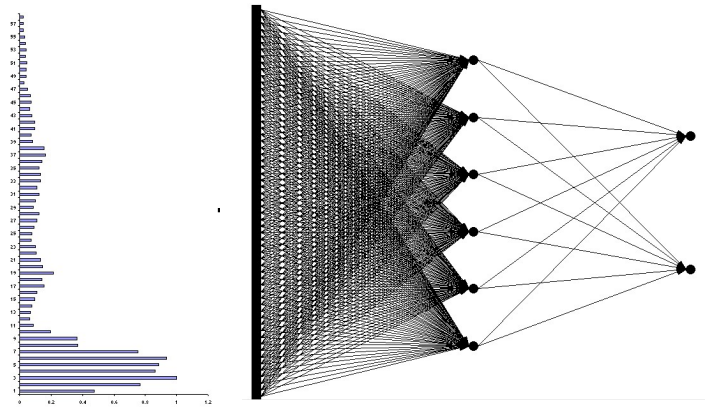


David



- Cálculo do erro

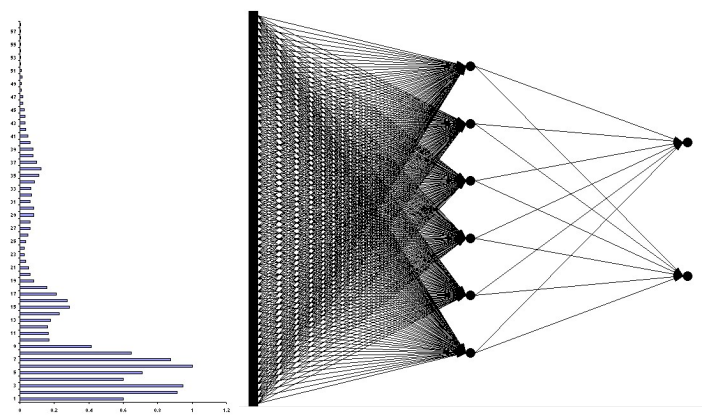
Steve



$$|0.43 - 0| = 0.43$$

$$|0.26 - 1| = 0.74$$

David

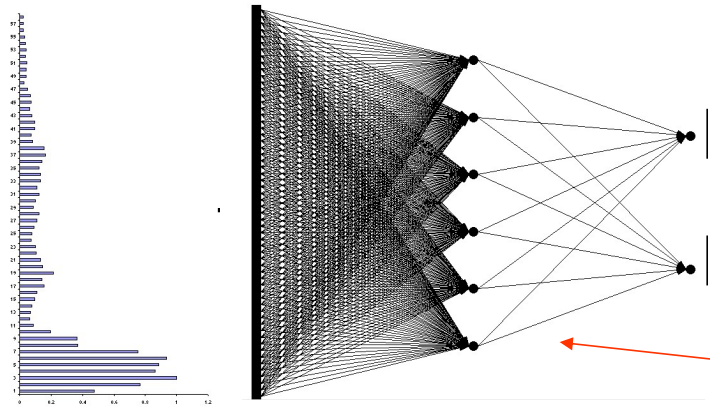


$$|0.73 - 1| = 0.27$$

$$|0.55 - 0| = 0.55$$

- Propagar para trás os erros e ajustar os pesos (Backpropagation)

Steve

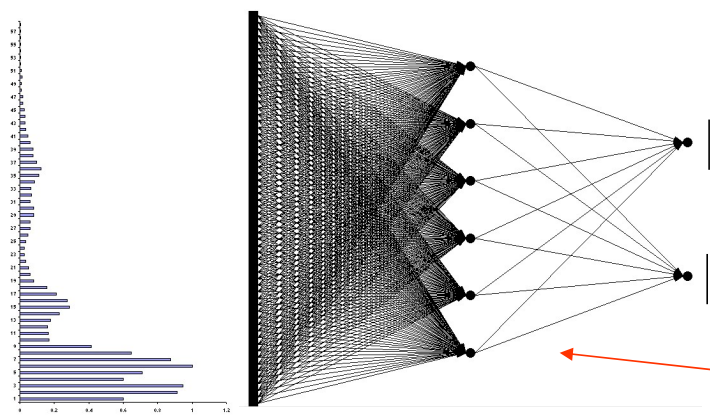


$$|0.43 - 0| = 0.43$$

$$|0.26 - 1| = 0.74$$

$$1.17$$

David

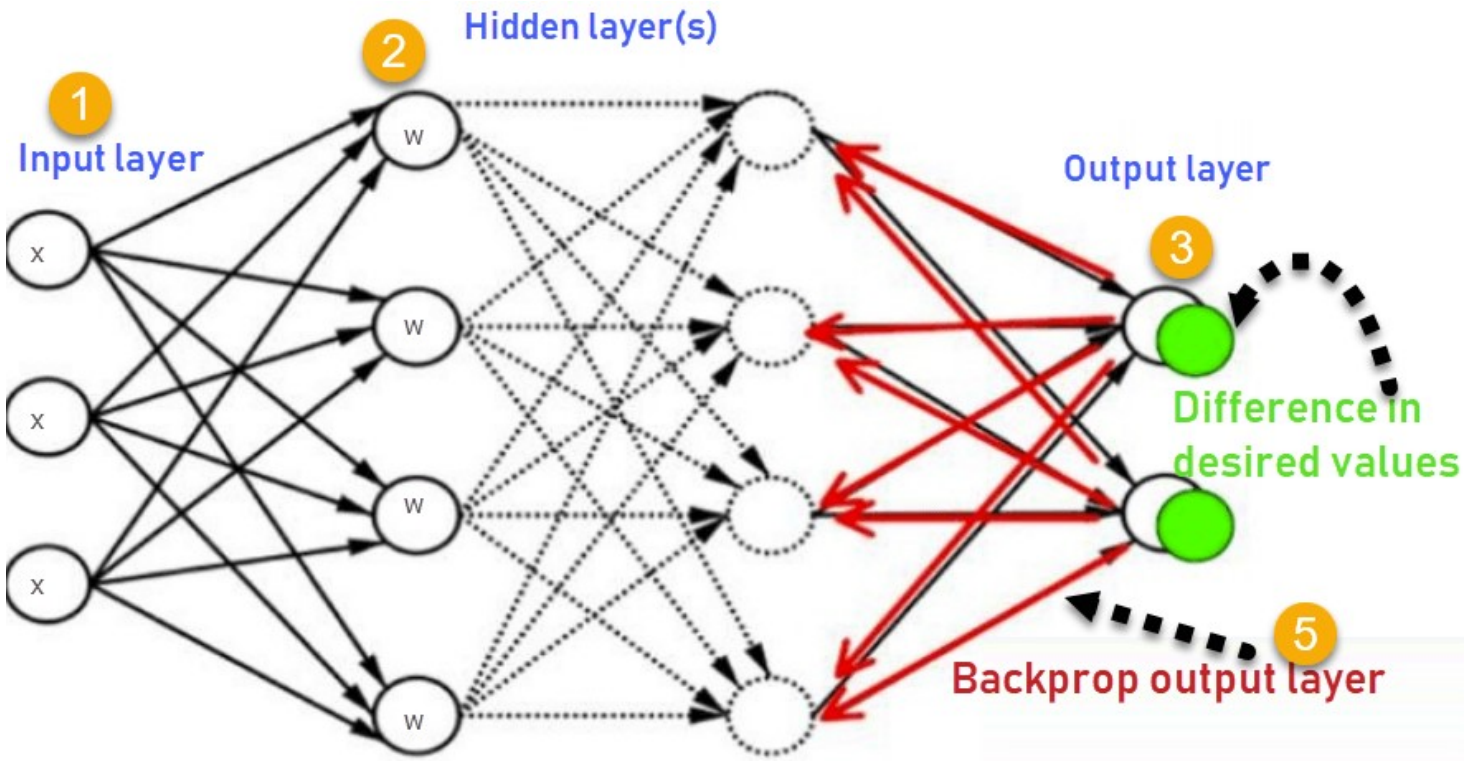


$$|0.73 - 1| = 0.27$$

$$|0.55 - 0| = 0.55$$

$$0.82$$

Backpropagation



Gradiente Descendente (Gradient Descent)

Há vários métodos para a Backpropagation – o mais conhecido é o Gradient Descent

D={<(1,1),1>,<(-1,-1),1>,
<(1,-1),-1>,<(-1,1),-1>}

Gradient:
$\nabla E[w]=[\partial E/\partial w_0,\ldots \partial E/\partial w_n]$

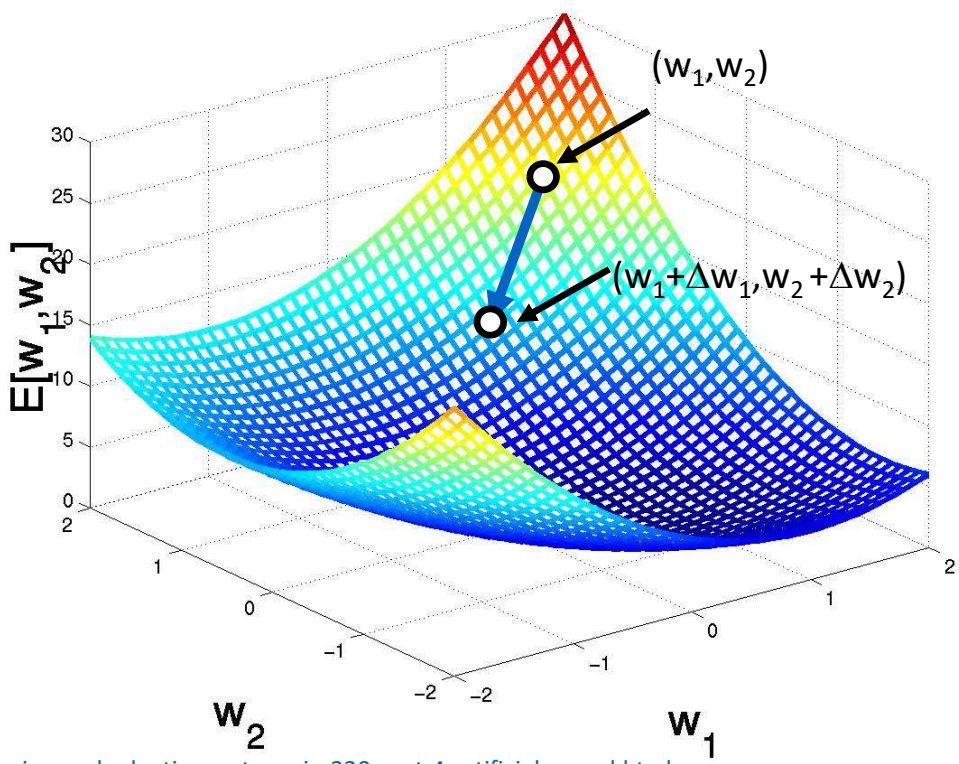
$\Delta w=-\eta\ \nabla E[w]$

$\Delta w_i=-\eta\ \partial E/\partial w_i$

$=\partial/\partial w_i\ 1/2\sum_d(t_d-o_d)^2$

$=\partial/\partial w_i\ 1/2\sum_d(t_d-\sum_i w_i x_i)^2$

$=\sum_d(t_d-o_d)(-x_i)$



Retirado de: https://dokumen.tips/documents/foundations-of-learning-and-adaptive-systems-ics320-part-4-artificial-neural.html

Gradient Descent

Gradient-Descent(*exemplos_treino*, η)

Cada exemplo de treino é um par na forma **<(x_1,…x_n),t>** onde **(x_1,…,x_n)** é o vetor com os valores de entrada e **t** é o valor de saída pretendido, η é a taxa de aprendizagem (por exemplo 0.1)

Initializar cada $\mathbf{w_i}$ com um pequeno valor aleatório

Até que a condição de término seja atendida, Fazer

Inicializar cada $\Delta \mathbf{w_i}$ a zero

Para cada **<(x_1,…x_n),t>** dos *exemplos_treino* Fazer

Introduzir **(x_1,…,x_n)** na unidade linear e calcular a saída **o**

Para cada peso da unidade linear $\mathbf{w_i}$ Fazer

$\Delta \mathbf{w_i} = \Delta \mathbf{w_i} + \eta\ (t-o)\ x_i$

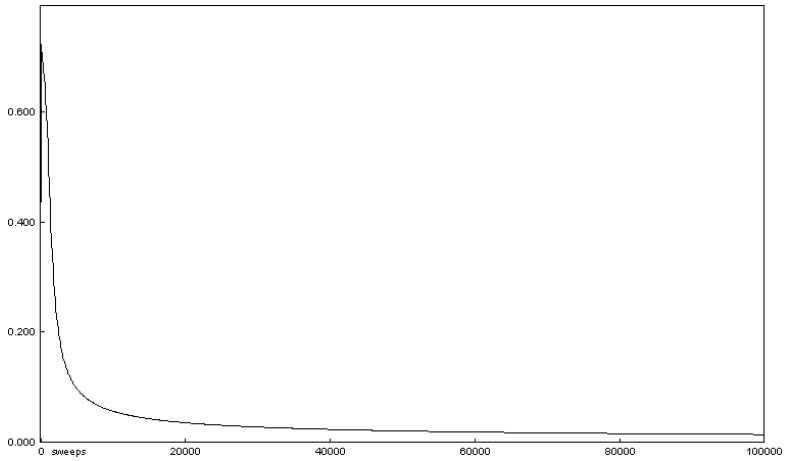
Para cada peso da unidade linear $\mathbf{w_i}$ Fazer

$\mathbf{w_i} = \mathbf{w_i} + \Delta \mathbf{w_i}$

Retirado de: https://dokumen.tips/documents/foundations-of-learning-and-adaptive-systems-ics320-part-4-artificial-neural.html

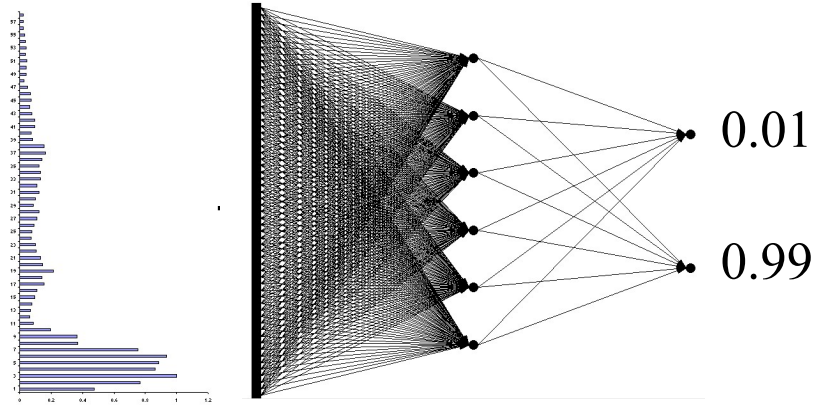
- Repetir o processo para todos os conjuntos de dados de treino
  - Introduzir os dados na entrada
  - Verificar os resultados na saída e calcular os Erros
  - Propagar os erros para trás (Backpropagation)
  - Ajustar os pesos entre nós da Rede Neuronal

- Repetir o processo muitas vezes (o erro vai diminuindo)

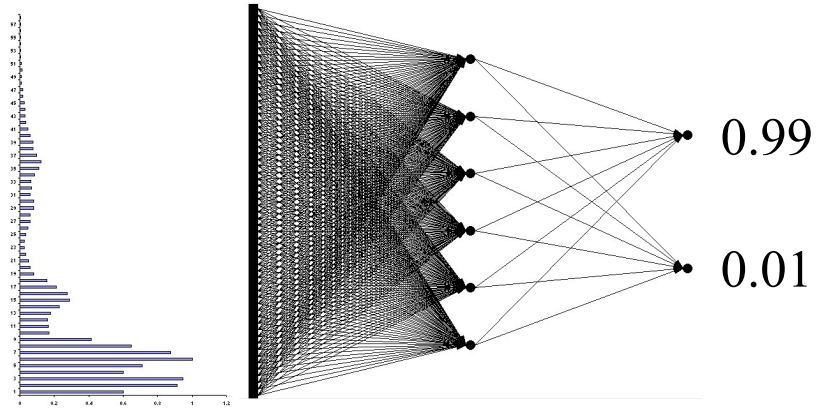


- Após o treino da Rede temos nas saídas valores próximos dos desejados

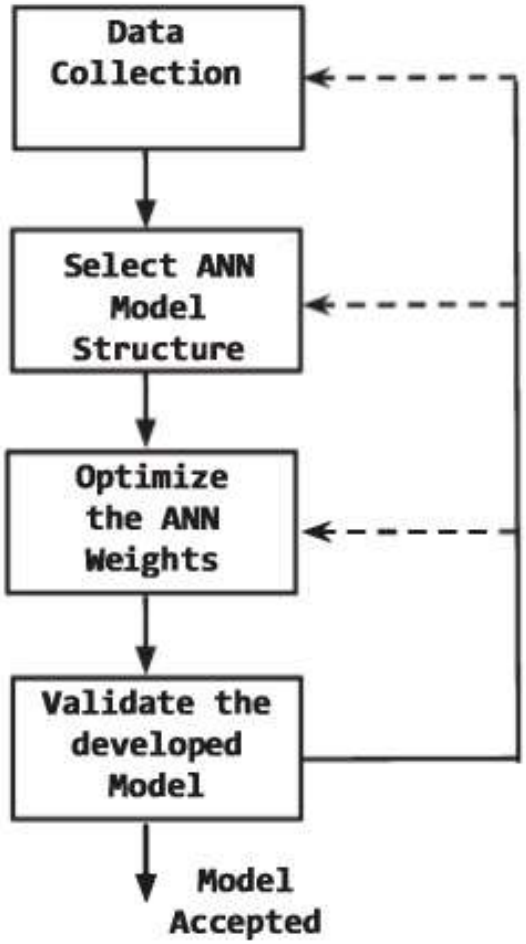
Steve

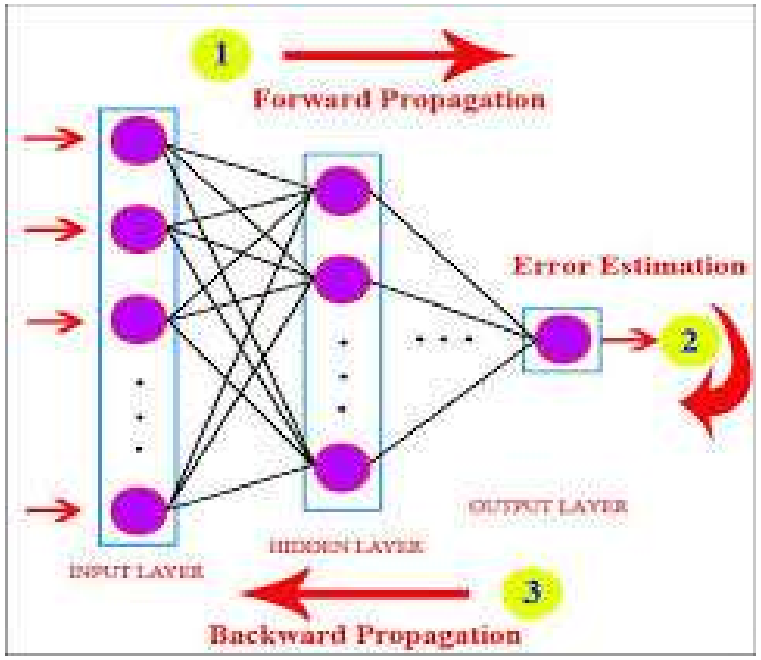


David



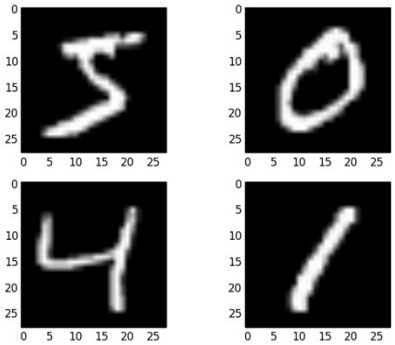
Processo de Desenvolvimento de uma ANN

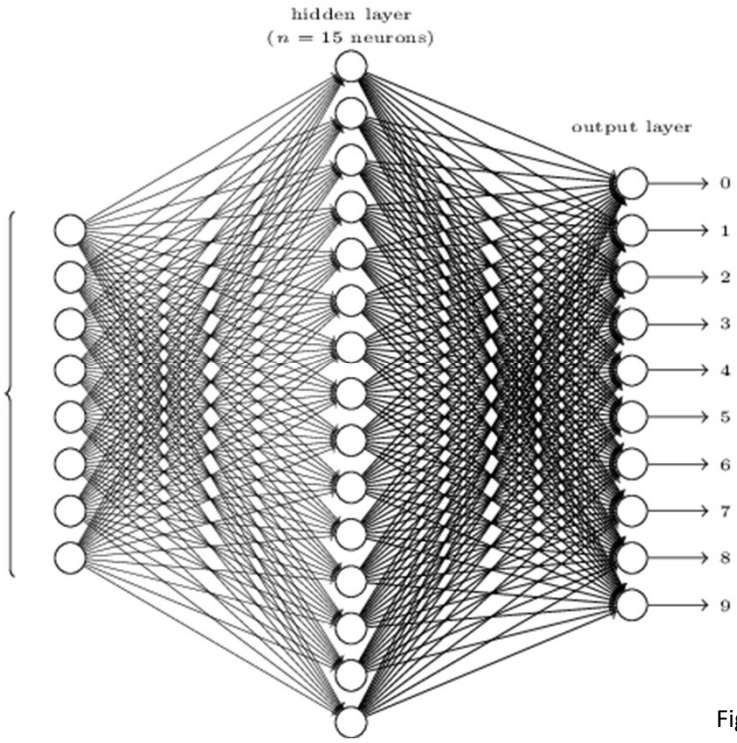


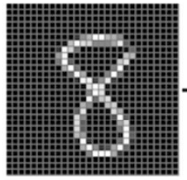


O número de nós na camada de entrada e na camáda de saída depende do nosso problema

Exemplo: Rede Neuronal para reconhecimento de dígitos manuscritos

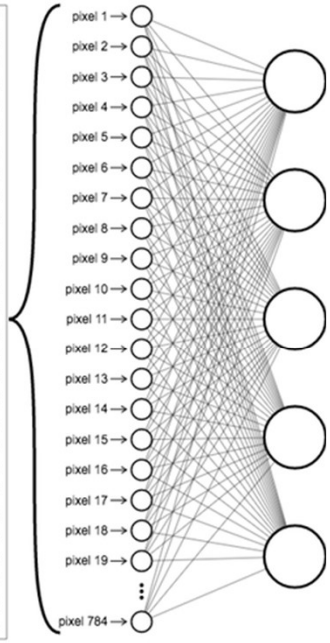






28 x 28
784 pixels

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	4	62	146	182	254	254	181	176	139	15	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	34	186	253	217	208	136	136	136	166	232	99	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	61	242	208	111	3	0	0	0	0	0	18	32	107	43	0	0	0	0	0
0	0	0	0	0	0	0	0	156	242	23	0	0	0	0	0	0	0	13	191	181	6	0	0	0	0	0
0	0	0	0	0	0	0	0	121	255	98	3	0	0	0	0	0	8	194	225	12	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	169	253	120	3	0	0	0	0	128	247	51	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	3	111	244	169	19	0	14	131	249	117	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	59	241	235	72	142	229	66	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	25	218	254	231	36	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	133	253	221	33	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	19	237	111	196	217	19	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	174	138	0	23	193	204	18	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	96	224	0	0	0	25	218	169	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	215	138	0	0	0	0	86	253	99	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	3	162	214	11	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	215	97	0	0	0	0	0	118	253	68	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	185	157	0	0	0	0	0	40	254	98	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	50	244	61	0	0	0	0	112	244	58	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	174	251	142	59	83	167	244	111	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	6	133	253	253	253	169	61	3	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0



Figuras retiradas de http://neuralnetworksanddeeplearning.com/chap1.html

Mais alguns exemplos

Rede Neuronal para previsão do preço de uma viagem de avião

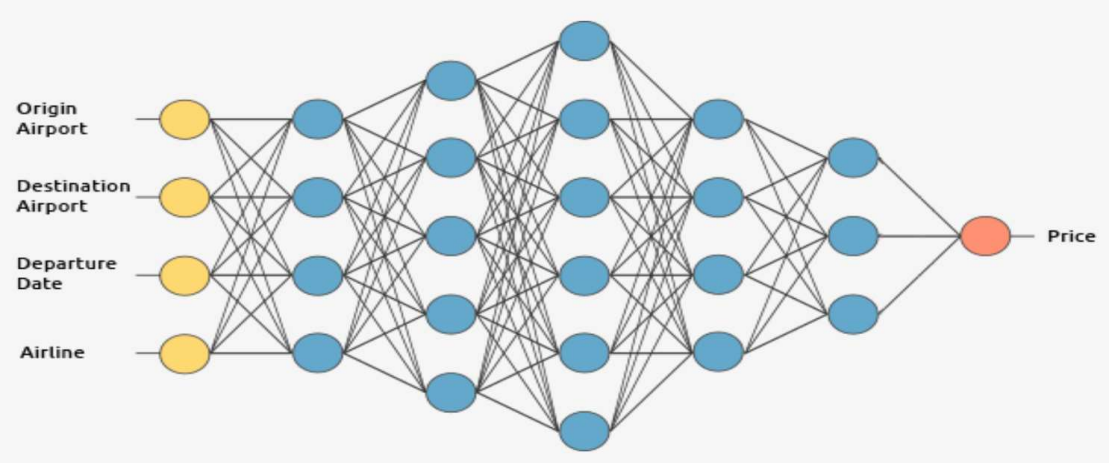


Figura retirada de https://www.houseofbots.com/news-detail/11733-4-know-how-deep-learning-works-heres-a-quick-guide-for-all-engineer

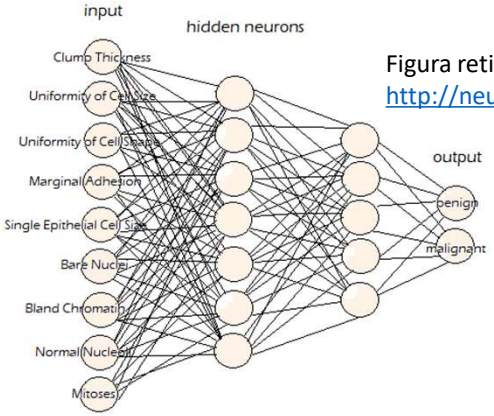


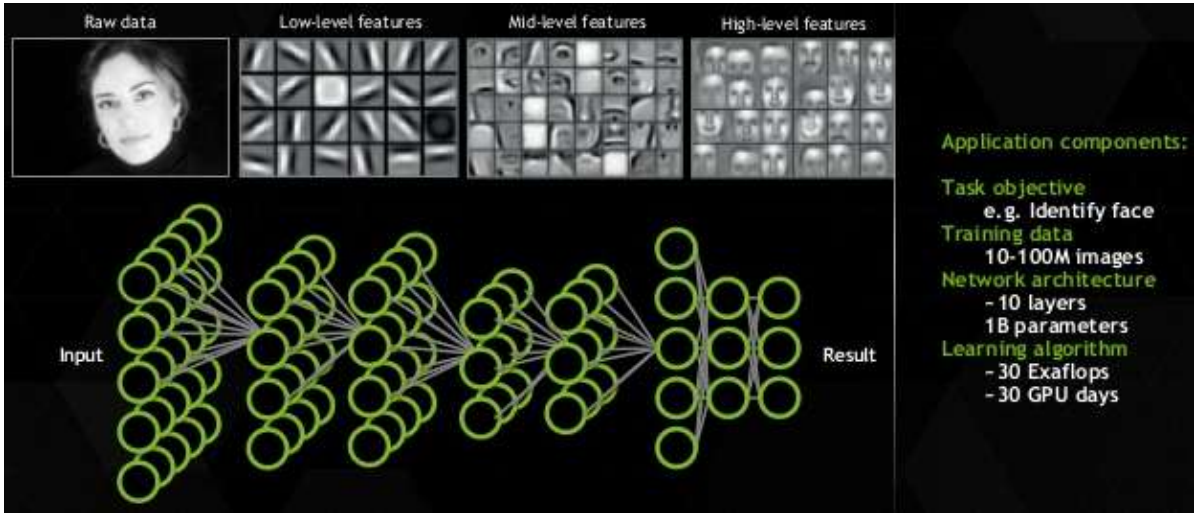
Figura retirada de
http://neuroph.sourceforge.net/tutorials/PredictingBreastCancer/PredictingBreastCancer.html

Aprendizagem Profunda (Deep Learning)

Deep Learning

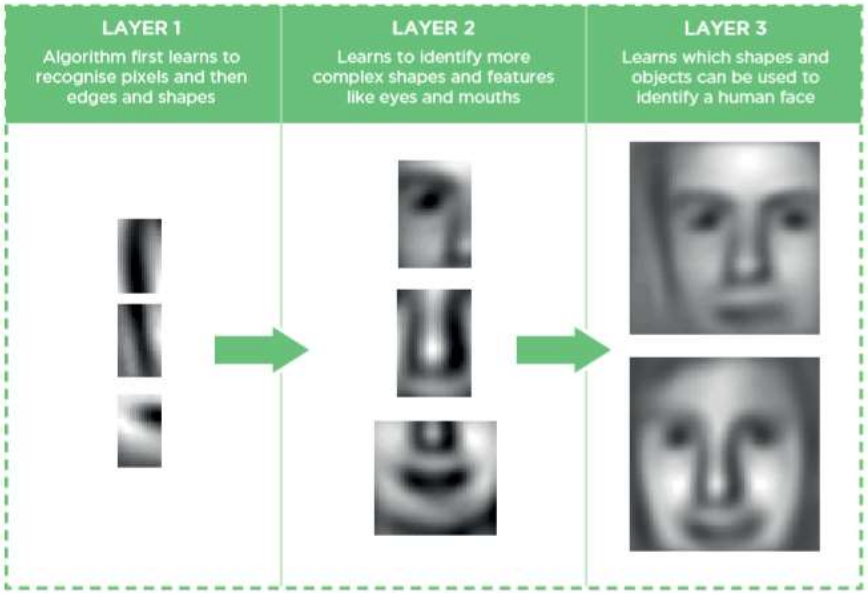
Hoje é a máquina de cozinha da IA!

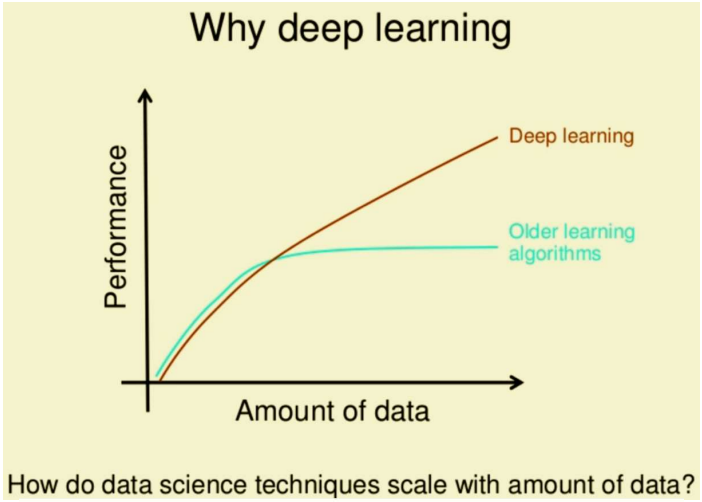
- Faz Data Analytics
- Faz Image/Video Understanding
- Faz Língua Natural e Tradução Automática
- …e muito mais

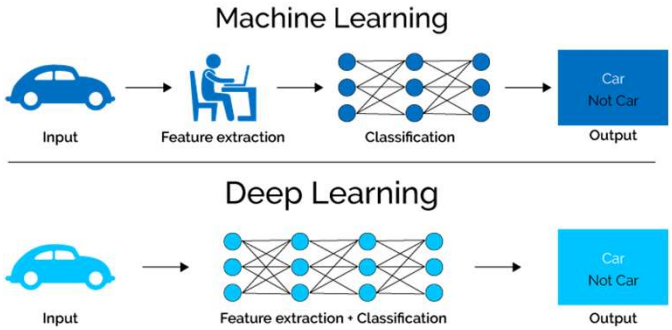


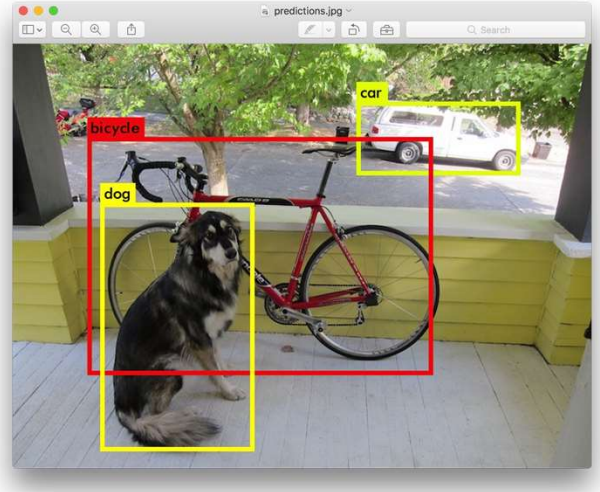
Deep Learning é um tipo de Arquitetura de Rede Neuronal:
- mais camadas
- associadas a alguma semântica
- habilidade para extração de caraterísticas
- supera outras abordagens no que toca ao aumento do desempenho em função da dimensão
- exige uma estrutura de hardware considerável envolvendo paralelismo, GPUs ou FPGAs
- exige muito tempo de treino (dias, meses) mas pouco tempo para teste
- adequado para problemas complexos com enormes volumes de dados (Visão por Computador, Língua Natural, Análise Grandes Volumes de Dados)
- arquiteturas específicas em função do objetivo Convolucionais para Imagem, Recorrentes para Texto

Aprendizagem Profunda (Deep Learning)





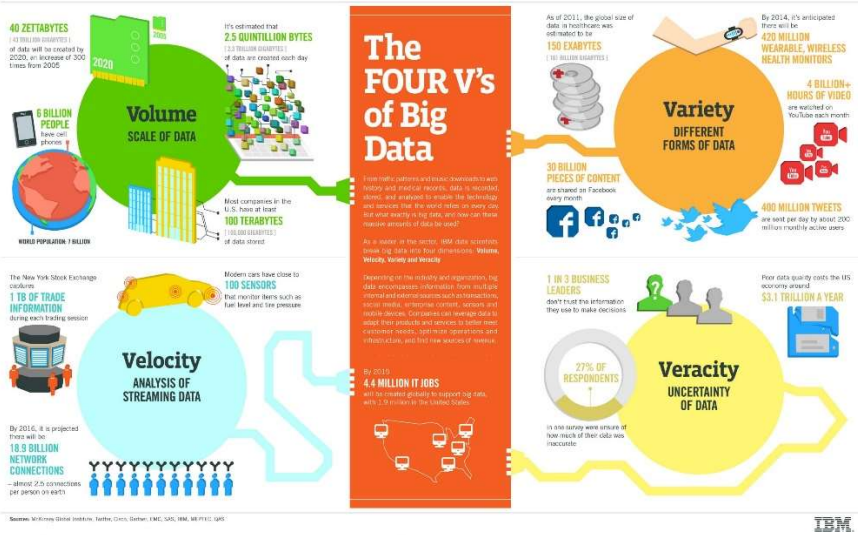


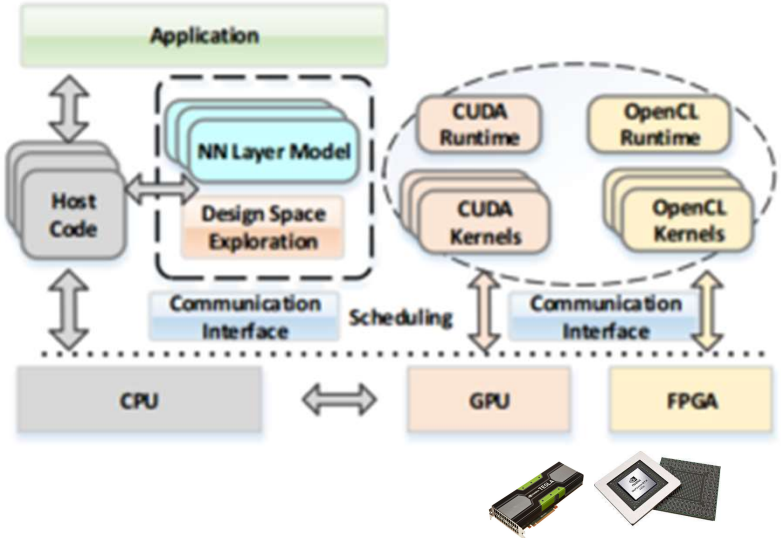


Figuras retiradas de https://towardsdatascience.com/why-deep-learning-is-needed-over-traditional-machine-learning-1b6a99177063

E o que levou a este sucesso da evolução do Deep Learning?

- Evolução da I&D à volta das Redes Neuronais
  - Nomeadamente a inclusão de mais camadas nas redes e associação dessas camadas a alguma semântica
- Disponibilidade de grandes volumes de dados
  - Big Data
- Evolução no Hardware
  - Disponibilidade de memória e tempos de processamento que permitem criar redes de grande dimensão
  - Treino das redes passou a ser bem mais rápido
  - Possibilidade de termos redes neuronais com milhões de nós (neurónios)





ImageNet

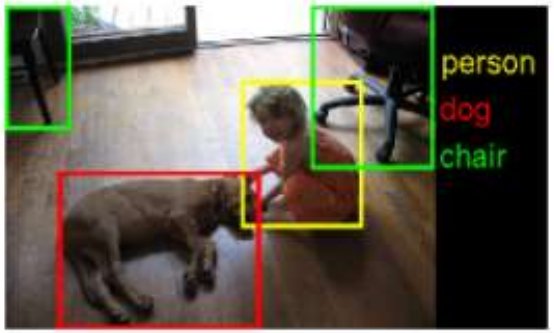
O projeto ImageNet constitui um enorme repositório de imagens para serem usadas na investigação na área de reconhecimento de objetos

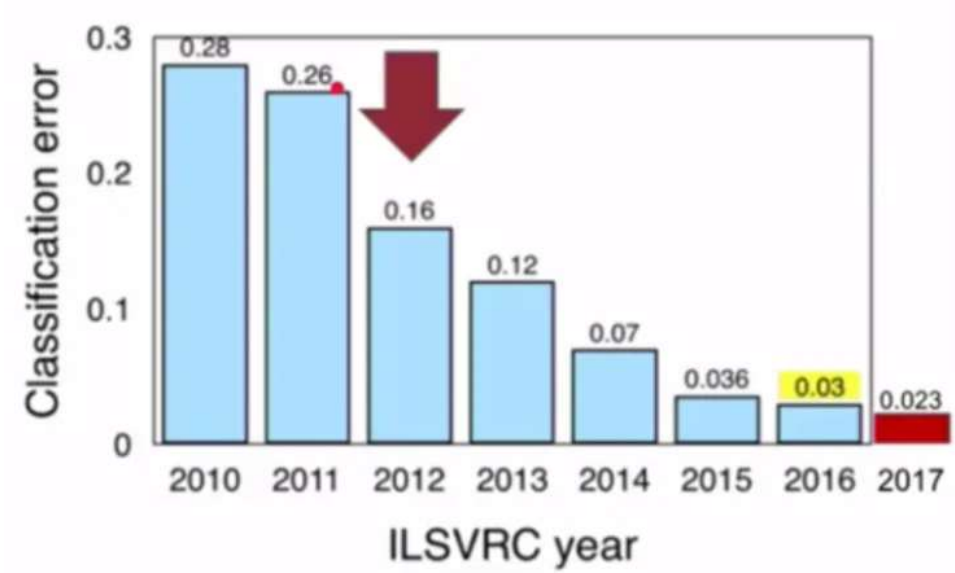
Mais de dez milhões de imagens anotadas!

ILSVRC – ImageNet Large Scale Visual Recognition Challenge, desde 2010

ILSVRC 2017

- LOC – localização de objetos entre 1000 categorias
- DET – deteção de objetos entre 200 categorias classificadas em imagens
- VID - deteção de objetos entre 200 categorias classificadas em vídeos





Algumas Ferramentas para desenvolvimento de sistemas com Aprendizagem Automática

	Platform	Cost	Written in language	Algorithms or Features
Scikit Learn	Linux, Mac OS, Windows	Free.	Python, Cython, C, C++	Classification Regression Clustering Preprocessing Model Selection Dimensionality reduction.
PyTorch	Linux, Mac OS, Windows	Free	Python, C++, CUDA	Autograd Module Optim Module nn Module
TensorFlow	Linux, Mac OS, Windows	Free	Python, C++, CUDA	Provides a library for dataflow programming.
Weka	Linux, Mac OS, Windows	Free	Java	Data preparation Classification Regression Clustering Visualization Association rules mining
KNIME	Linux, Mac OS, Windows	Free	Java	Can work with large data volume. Supports text mining & image mining through plugins

	Platform	Cost	Written in language	Algorithms or Features
Apache Mahout	Cross-platform	Free	Java Scala	Preprocessors Regression Clustering Recommenders Distributed Linear Algebra.
Accors.Net	Cross-platform	Free	C#	Classification Regression Distribution Clustering Hypothesis Tests & Kernel Methods Image, Audio & Signal. & Vision
Shogun	Windows Linux UNIX Mac OS	Free	C++	Regression Classification Clustering Support vector machines. Dimensionality reduction Online learning etc.
Keras.io	Cross-platform	Free	Python	API for neural networks
Rapid Miner	Cross-platform	Free plan Small: $2500 per year. Medium: $5000 per year. Large: $10000 per year.	Java	Data loading & Transformation Data preprocessing & visualization.

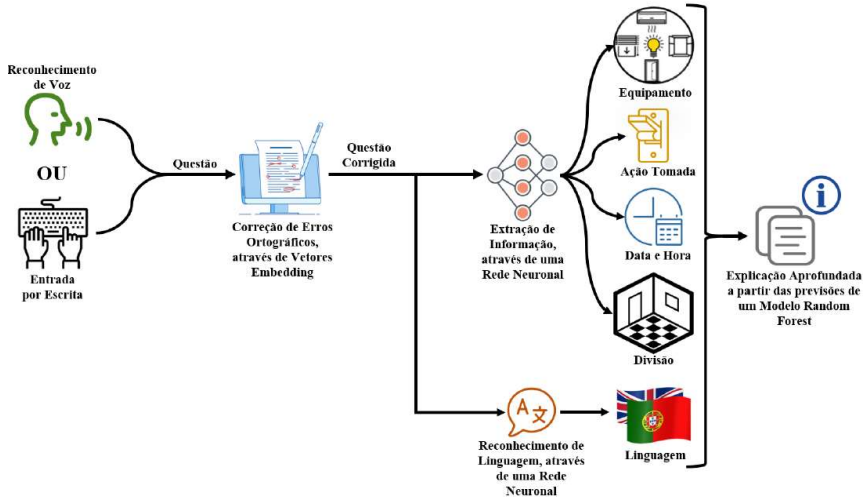
Para saber mais sobre Aprendizagem Automática

- Unidade Curricular de **Análise de Dados** da **Licenciatura em Engenharia Informática** do ISEP (métodos clássicos)
- Unidade Curricular de **Aprendizagem Automática 1 (AAUT1IA)** do **Mestrado em Engenharia de Inteligência Artificial (MEIA)** do ISEP (métodos clássicos, Reinforcement Learninig, Ensemble Learning, introdução a Redes Neuronais Artificiais)
- Unidade Curricular de **Aprendizagem Automática 2 (AAUT2IA)** do **Mestrado em Engenharia de Inteligência Artificial (MEIA)** do ISEP (Redes Neuronais Artificiais detalhadas e Aprendizagem Profunda – Deep Learning, também Big Data, Hardware para Deep Learning, Explainable AI, Automated Machine Learning,…)

MEIA
MESTRADO ENGENHARIA INTELIGÊNCIA ARTIFICIAL

Challenges 4Teams
MEIA

Demo Day
MEIA



Equipa de IA aplicada à Energia do MEIA usa Deep Learning para reconhecer várias perguntas após um sistema de Deep Learning ter sugerido decisões para poupança energética em contexto habitacional

Equipa de IA aplicada à Cibersegurança do MEIA usa Redes Neuronais Convolucionais, SVM e K-Means para detetar ataques a sistemas Ciber-Físicos (Sistema de Tratamento de Água)