FEUP IART SUMMARY

Problem Solving Strategies

Some search strategies:

1. Evaluation criteria:
    1. **completeness**: a search algorithm that is guaranteed to find a solution if there is one
    2. **time efficiency**:
    3. **space efficiency**:
    4. **optimality of solution**:
2. **Uninformed** or **blind** searches:
    1. **Breadth-first**: (FIFO queuing strategy)
        - complete: yes
        - time and space: O(branch_factor^depth) (empirically space a larger problem than time)
        - optimality: yes, if path cost non-decreasing with depth.
    2. **Uniform Cost (BB)** (expand lo cost fringe node)
        - optimal: yes, if no negative costs
        - a generalization of **Breadth-first**.
    3. **Depth-first** (uses a stack)
        - completeness: no! may go down and not come back
        - time: O(branch_factor^depth)
        - space: O(depth)
        - optimality: no! returns first found, not necessarily ideal
    4. **depth-limited** (DFS down to some cutoff)
        - completeness: yes, provided solution exists before cutoff
        - time: O(branch_factor^depth_limit)
        - space: O(depth_limit)
        - optimality: no!
    5. **Iterative Deepening** (successively increasing depth-limited)
        - **diameter of state space**: max anticipated path length for most problems.
        - completeness: yes (like Breadth-first)
        - time: O(branch_factor^diameter)
        - space: O(diameter) (like Depth-first)

    - optimality: yes (like Breadth-first)
3. **Informed** searches:
    1. **Best-first search** (expand nodes according to some cost function)
        1. **Greedy search** (expand according to h(n)–cost to goal)
            - completeness: no (could go down an endless branch)
            - optimality: no (ignores cost from initial node)
            - faster (on average) than uninformed algorithm
        2. **A* search** (combines uniform-cost search and greedy search)
            - expand nodes according to the estimated total cost function, f(n) = g(n) + h(n)
            - h(n) must always be optimistic (admissible)
            - completeness: yes
            - optimality: yes
            - optimally efficient: no better algorithm with same knowledge
    2. **Heuristic functions**
        - Relax the demands of the problem

        1. Admissible

            - it never overestimates the cost of reaching the goal, i.e. the cost it estimates to reach the goal is not higher than the lowest possible cost from the current point in the path - optimistic nature.

        2. Consistent

            - it estimates always less than or equal to the estimated distance from any neighboring vertex to the goal, plus the step cost of reaching that neighbor. $h(N) \leq c(N, P) + h(P)$, N is any node in the graph and P is any descendant of N.

Problem 1 - weak methods

Comapre in terms of complexity and characteristics the search strategies of: **depth first search** (dfs), **breadth first search** (bfs), **iterative deepening** (id):

	Time complexity	**Space complexity**	**Complete (guaranteed to find a solution)**	**Optimal (guaranteed to find the best solution)**
dfs	r^m	r * m (if it expands all sucessors before choosing the first), m otherwise	no	no
bfs	r^(p+1) (because it expands all the successors before navigating sideways)	same as time complexity (r^(p+1))	yes	yes (if the utility is the length of the path)
id	r^p (because it does not expand the next successors)	same as dfs space complexity (r * m if...)	yes	yes (if the utility is the length of the path)

Evolutionary

Problem 1 - container

Consider a container with **capacity = 100t**. Objects are represented as `(Object, Weight, Value)`.

- Objects: `(A, 10, 20)`, `(B, 70, 60)`, `(C, 30, 80)`, `(D, 80, 100)`
- The fitness function (*função de adaptação*) is: `f(Container_x) = Weight(Container_x) - Max(value[i]/weight[i])` for each `Object_i` **not in** Container_x. **OR** `f(Container_x) = 0` if Weight(Container_x) > 100
- Elististic strategy, with N = 1 (the 1 best subject is choosen for pairing automatically)
- 4 random numbers for pairing (*emparelhamento*): `0.6`, `0.9`, `0.09`, `0.2`
- **40%** chance of crossover (*cruzamento*) (only when the random number is below or equal to `0.4`) is the individual selected for crossover. The random numbers for crossover are: `0.6`, `0.9`, `0.1`, `0.2`, `0.8`
- Crossover point: between 2nd and 3rd bits
- Probability of mutation: 1%, only on the 10th RGN (randomly generated number) is a value smaller or equal to 0.01 generated.
- Initial Population:

- `C1 = (0010)` (only object C)
- `C2 = (1100)` (objects: A, B)
- `C3 = (1110)` (A, B, C)
- `C4 = (1001)` (A, D)
- `C5 = (0110)` (B, C)

Question: Which objects to include in the container if there are only 2 generations.

Answer:

1. [*preprocessing*] Calculate `value[i]/weight[i]` for each object:
   - `A: 20/10 = 2`
   - `B: 60/70 = 0.85`
   - `C: 80/30 = 2.67`
   - `D: 100/80 = 1.25`

   (2, 0.85, 2.67, 1.25)
2. Caclulate the **fitness** of each individual (C1, C2, C3, C4, C5) (converting into int):
   - `C1: fitness(0010) = [0 0 1 0] .* [10, 70, 30, 80] - max(2, 0.85, 1.25) = 30 - 2 = 28`
   - `C2: fitness(1100) = [1 1 0 0] .* [10, 70, 30, 80] - max(2.67, 1.25) = 80 - 2.67 = 77.33 = 77`
   - `C3: fitness(1110) = [1 1 1 0] .* [10, 70, 30, 80] - max(1.25) = 110 | 110 > 100 => fitness(C3) = 0`
   - `C4: fitness(1001) = [1 0 0 1] .* [10, 70, 30, 80] - max(0.85, 2.67) = 90 - 2.67 = 87.33 = 87`
   - `C5: fitness(0110) = [0 1 1 0] .* [10, 70, 30, 80] - max(2, 1.25) = 100 - 2 = 98` <- **best of this generation** (only one choosen automatically because it is elitistic with N = 1)
3. Normalize the fitness into 0...1, by calculating the desired probability:

   Sum of fitnesses is: `28 + 77 + 0 + 87 + 98 = 290`

   - `p(C1) = 28/290 = 0.096`
   - `p(C2) = 77/290 = 0.266`
   - `p(C3) = 0/290 = 0`
   - `p(C4) = 87/290 = 0.3`
   - `p(C5) = 98/290 = 0.338`
4. Construct the **probability scale** (C3 does not appear):

```
0-----0.096-----------0.362-----------0.662-----------------1

|----C1----|-------C2------|------C4-------|---------C5-----|
```

5. **Selection** of 2nd generation using the RGNs: (`0.6` , `0.9` , `0.09` , `0.2`) using the scale:
   - C1′ = **C5** = (0110) (elitistic)
   - C2′ = C4 (`0.6`) = (1001)
   - C3′ = C5 (`0.9`) = (0110)
   - C4′ = C1 (`0.09`) = (0010)
   - C5′ = C2 (`0.2`) = (1100)
6. Choose individuals for **crossover** from RGNs: `0.6` , `0.9` , `0.1` , `0.2` , `0.8` :
   Only (`0.1` and `0.2`) `<= 0.4` therefore only C3′ and C4′ are choosen for crossover.
7. Perform the **crossover** (in this case between the 2nd and 3rd bits):
   - `C3' = (01 10)` and `C4' = (00 10)`
   - `C3'' = (01 10)` and `C4'' = (00 10)`
8. After reproduction:
   - C1′′ = (0110)
   - C2′′ = (1001)
   - C3′′ = (0110)
   - C4′′ = (0010)
   - C5′′ = (1100)
9. The **mutation** only happens on the 10th RGN => on the 10th bit, which is the second bit of **C3''**
   - C1′′ = (0110)
   - C2′′ = (1001)
   - C3′′ = (0**0**10) <- mutation on the 2nd bit (overall 10th bit)
   - C4′′ = (0010)
   - C5′′ = (1100)
10. Recalculate the **fitness** function:
    - `C1'': fitness(0110) = [0 1 1 0] .* [10, 70, 30, 80] - max(2, 1.25) = 100 - 2 = 98` <- **best of this generation**
    - `C2'': fitness(1001) = [1 0 0 1] .* [10, 70, 30, 80] - max(0.85, 2.67) = 90 - 2.67 = 87.33 = 87`
    - `C3'': fitness(0010) = [0 0 1 0] .* [10, 70, 30, 80] - max(2, 0.85, 1.25) = 30 - 2 = 28`

- C4'': fitness(0010) = [0 0 1 0] .* [10, 70, 30, 80] - max(2, 0.85, 1.25) = 30 - 2 = 28
- C5'': fitness(1100) = [1 1 0 0] .* [10, 70, 30, 80] - max(2.67, 1.25) = 80 - 2.67 = 77.33 = 77

11. The **answer** is: Objects B and C will go into the container (the best individual of this generation is **C1'' = (0110)**)

Uncertain Reasoning

Exercises Resolution steps

1. Meter os comandos da Ti-nspire

```
r(x):=x.log2(x)
rr(x,y):=r(x)+r(y)
```

2. Escrever a nomenclatura:

```
r(x) = x.log2(x)
r(x1,x2,...,xn) = sum(1,n)r(x)
```

- **ENTROPIA**: mede a **pureza** de um conjunto.

$$H(C/a_k) = \sum_{j=1}^{M_k} p(a_{k,j}) * [-\sum_{i=1}^{N} p(C_i|a_{k,j}) * \log_2 p(C_i|a_{k,j})]$$

- **informação média**: Se houver valores indefinidos, o denominador é decrementado (dá sempre o mesmo para todos os atributos a não ser que faltem valores).

$$info(C) = -\sum_{k=1}^{n} p(C_k) * \log_2(p(C_k))$$

> Informação média para identificar Classes Ck no Conjunto C de itens (independente do Atributo)

- **informação de separação**: infoS(A) - o que um dado atributo contribui para a separação da classe. Informação obtida dos resultados de um teste Tj com j=1 a n valores, independentemente de serem ou não da mesma classe. (Não confundir com info(C/Ai)). **Nota**: se faltar algum atributo o denominador mantém, mas a soma dos numeradores deixa de ser igual ao denominador (por causa dos que faltam).

$$infoS(C) = -\sum_{i=1}^{n} \frac{C_i}{C} * \log_2(\frac{C_i}{C})$$

- **ENTROPIA de atributo A em relação à classificação no conjunto de treino C**:

$$info(C|A_i) = \sum_{j=1}^{n} \frac{Cj}{C} * info(C_j)$$

- **ganho de informação**: o que escolher um dado atributo trás de melhoria para a entropia:

$$G(C|A_i) = info(C) - info(C|A_i)$$

se faltar conhecer algum valor de um atributo (usar freq dos conhecidos sobre conhecidos + desconhecidos):

$$G(C|A_i) = f(known/total) * (info(C) - info(C|A_i))$$

> ECO: Dá bons resultados mas sobrevaloriza testes com muitos valores possíveis.

- **razão do ganho**: Normalizar o ganho.

$$RG(C|A_i) = \frac{G(C|A_i)}{infoS(C)}$$

- **razão do erro**: (medida da **confiança** numa folha) `e` é o número de exemplos que não pertencem àquela folha e `n` é o número total de exemplos que acabam naquela folha.

$$Re = \frac{e+1}{n+2}$$

ID3

Algorithm for induction learning by Ross Quinlan (http://www.rulequest.com/Personal/). Generates Decision Trees.

- Uses **Entropy** as the decision variable for branching a given tree node, until it has 0 entropy.
- Cannot handle unknown values
- Cannot handle continue values (0.1 to 0.195), only discrete (0.1,0.2,0.3)

- No further action taken to improve the results after the tree is ready

C4.5

Improvement on ID3, Quinlan as well.

- Uses **Gain Ratio** instead of Entropy
- Can handle unknown values by extrapolating from the others
- Can handle discrete values, by establishing interval values
- Prunes (through pessimistic pruning) by removing trees that do not suficiently contribute to the accuracy of the model

Chapa 7

Topic 1 - Search Methods

2016/2017 - Recurso - a) pdf (../exames/2017_R.pdf) [MONTECARLO TREE SEARCH, MINIMAX]

Question:

Explique duas vantagens do MCTS face ao Minimax.

Answer 1:

Vantagem 1: Não precisa de uma função de avaliação (o Minimax pode precisar se a profundidade for muito grande) dado que executa sempre jogos até ao fim e é em função disso que toma decisões, ou seja, só precisa de conhecer a mecânica de jogo e não de ter a noção do que é uma boa ou má jogada.

Vantagem 2: Não é determinístico. Esta é uma vantagem face ao MM quando se fala de árvores muito grandes (nas quais o MM tem uma performance limitada por ser exaustivo). Ao fazer amostragens aleatórias o MCTS consegue criar estimativas do valor de cada estado - um exemplo flagrante disto é quando o minimax tem de ser restringido a uma profundidade tal que fica cego a alterações drásticas no valor dos estados, ao passo que o MCTS poderá vir a ter essa visão mais global.

2016/2017 - Normal - a) pdf (../exames/2017_N.pdf) [A*, IDA*]

Question:

No IDA*, ao contrário do ID, aumenta-se o custo e não a profundidade. Como deve ser incrementado esse custo limite?

Answer 1:

O custo limite, para cada iteração do algoritmo IDA*, deverá tomar o valor do custo mínimo de todos os valores que excedem o custo limite, nesse momento. Formalizando, C(i+1) = min(all_values_exceding_C(i)).
Se, por exemplo, na iteração i do algoritmo, o custo limite for 5 e os valores obtidos foram 7,8 e 10, então na iteração i+1, o custo limite será 7.

2015/2016 - Normal - a) pdf (../exames/2016_N.pdf) [ARREFECIMENTO SIMULADO]

Question:

No algoritmo de pesquisa por arrefecimento simulado, a aceitação de um novo estado depende da sua qualidade. Explique de que forma.

Answer 1:

O agoritmo de arrefecimento simulado é um método de pesquisa para encontrar ótimos globais (**não é completo nem ótimo**, mas combate uma das limitações do hill climbing que é ficar preso em máximos locais). Caracterizado por considerar um valor de "temperatura" (que estabelece paralelismo com o arrefecimento de metais liquefeitos) e que vai diminuindo (arrefecendo) ao longo das iterações (tempo). Esse valor, doravente `T`, é usado para combater o problema "exploration vs exploitation", sendo que quanto menor é `T` maior é a probabilidade de a exploração ser beneficiada (face à potenciação), ou seja, `T` controla a probabilidade de selecionar estados que se afastam da melhor solução até ao momento. Quando `T=0` é retornado o valor atual. A fórmula que dita a probabilidade de aceitação de um estado é:

$$p(x) = e^{\frac{\Delta_{h(x)}}{T}}$$

Sendo `Δh = h(novo estado)- h(atual)` e `h(x)` é a função heurística de avaliação de um estado `x`. Para estados melhores que o atual, `Δh>0` e quanto maior `Δh` maior será `p(x)`. Para estados iguais ao atual, `Δh=0 => p(x)=1` (aceita sempre). Para estados piores que o atual, `Δh<0 -> p(x)<1` sendo que maior `T` faz com que o valor de `Δh` tenha menos efeito e aumenta a probabilidade.

De referir que, dado `p(x)` é tirado um número `r` à sorte entre 0 e 1 e se `p(x)>r`, `x` passa a ser o novo estado atual e repete-se o processo de procurar e avaliar estados até mudar, sendo que `T` vai diminuindo (variações não necessarieamente constantes).

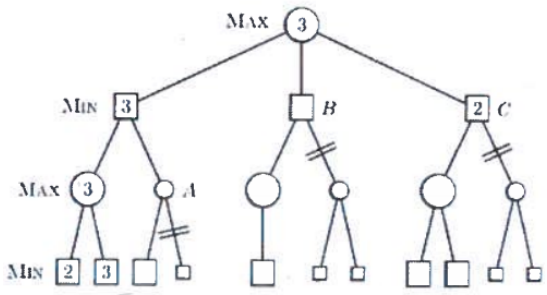
Posto isto, a afirmação original torna-se mais clara: a aceitação de um dado estado depende da sua qualidade - quanto melhor é um estado (em relação ao estado em que se toma a decisão!), maior é a probabilidade de ele ser selecionado (maior `Δh` -> maior `p(x)`) - sendo que a probabilidade de aceitação diminui com o dito arrefecimento, `T`.

> Nota ECO: SE variação baixar T suficientemente devagar, +provável encontrar um ótimo global.

2015/2016 - Normal - d) pdf (../exames/2016_N.pdf) [ALFA-BETA]

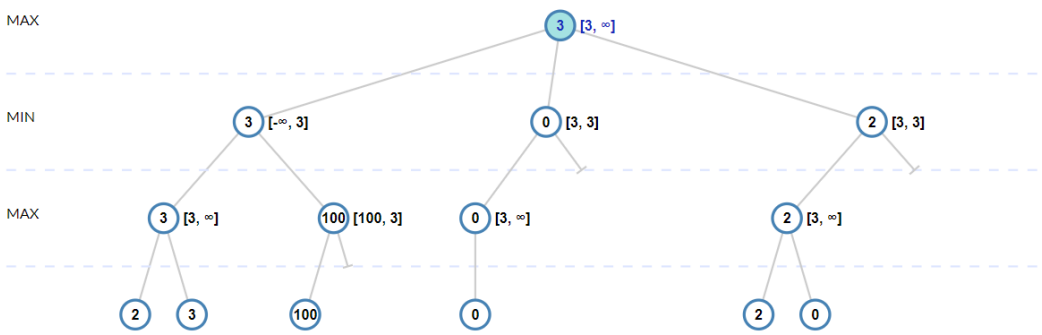
Question:

Ao aplicar o algoritmo minimax, aplicaram-se os cortes alfa-beta indicados na figura (e só esses). Indique que gamas de valores podem ter os nós da folha.



Answer 1:

Descrever a figura:



- No ramo A, o valor da folha teria de ser >= 3, de forma a que `alpha >= beta`.
- No ramo B, o valor da folha teria de ser <= 2, de forma a que `beta <= alpha`.
- No ramo C, o valor de uma das folha teria de ser 2 e o da outra <= 2 (se não houvesse a restrição do 2 já na árvore, também podiam ir até 3), de forma a que `beta <= alpha`.

2014/2015 - Normal - c) pdf (../exames/2015_N.pdf) [A*]

Question:

Explique as vantagens do algoritmo IDA* (Iterative Deepening) sobre o A*

Answer 1:

Como o A* tem de armazenar todos os estados encontrados até ao estado final, o algoritmo é bastante pesado a nível de memória utilizada. Apesar da qualidade da função heurísitica poder influenciar positivamente a complexidade temporal do A*, à medida que a escala (e espaço de pesquisa) aumentam, a utilização de IDA* torna-se mais viável de modo a reduzir a memória necessária. Isto deve-se ao facto do A* manter uma lista - tipicamente uma fila - de estados não visitados que rapidamente ocupam grandes níveis de memória. O IDA* mantém conhecimento apenas dos estados do caminho atual, de tal forma que a memória ocupada por essa lista aumenta linearmente apenas em relação ao tamanho da solução encontrada.

2012/2013 - Recurso - a) pdf (../exames/2013_R.pdf) [ARREFECIMENTO SIMULADO, HILL CLIMBING]

Question:

Comente a seguinte afirmação: "O algoritmo Arrefecimento Simulado, com uma temperatura constante positiva, é equivalente ao algoritmo Subir a Colina". Como se comporta o algoritmo Arrefecimento Simulado quando o parâmetro temperatura é sempre igual a zero?

Answer 1:

(Ver #20152016—normal—d-pdf-arrefecimento-simulado que descreve arrefecimento simulado)

Se, no arrefecimento simulado, a temperatura `T` for constante, a fórmula da probabilidade reduz-se a `e^(Δh/k)` que acaba por ser `e^Δh`. Nestas condições, temos ainda que quanto melhor um estado for maior será a probabilidade de ser o próximo e quanto pior for, menor também será essa probabilidade. Apesar de não haver uma tendência para arriscar menos à medida que as iterações aumentam, ainda existe o conceito de probabilidade, portanto a afiramação é **falsa**.

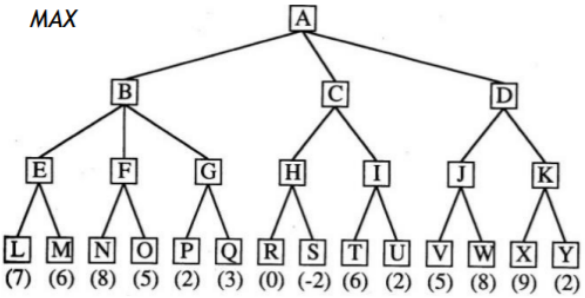
No caso de `T=0`, contudo, temos que `e^(Δh/0)` que acaba por ser um número não real, mas se considerarmos como infinito (potência de divisão por 0) podemos derivar algumas conclusões. No caso de `Δh<0` (o estado é pior que o atual) teremos `p(x)=-∞` - os piores estados nunca são selecionados. Se o estado em consideração tiver o mesmo valor que o

atual temos `0/0` o que acaba por ser uma indefinição e a intuição leva a crer que a probabilidade nesse caso seja considerada como nula. Caso contrário, teremos `Δh>0`, ou seja `p(x)=+∞`, nesse caso o estado será sempre selecionado. Portanto, perante `T=0` estamos perante o algoritmo de subir a colina, que avança apenas para estados de melhor valor. De realçar que há duas abordagens exclusivas no hill climbing: escolher o melhor ou expandir 1 a 1 até encontrar um melhor e que a implementação para a segunda opção era a que resultava de ter `T=0`, dado que não só se procuraria até a probabilidade ser 1 pela primeira vez.

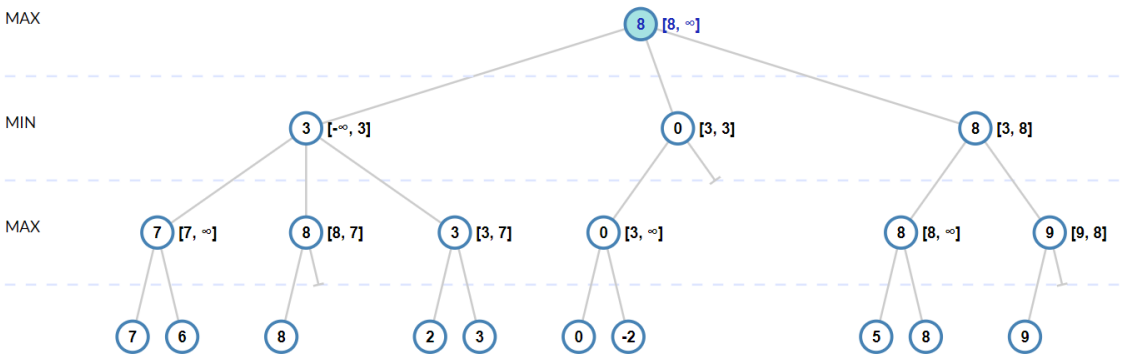
2012/2013 - Recurso - b) pdf (../exames/2013_R.pdf) [ALFA-BETA]

Question:

Considere a seguinte árvore de jogo, em que os valores das folhas representam a avaliação do jogo nesse estado. Assumindo que os nós são analisados da esquerda para a direita, indique quais os nós que não são avaliados quando se usa o algoritmo Minimax Alfa-Beta.



Answer 1:



Não avaliados: O, I, T, U, Y

2012/2013 - Recurso - c) pdf (../exames/2013_R.pdf) [MINIMAX, EXPECTIMINIMAX]

Question:

Suponha agora que o adversário (representado nos níveis minimizadores) joga de forma aleatória (a probabilidade de efetuar qualquer jogada é a mesma). Como alteraria o algoritmo MiniMax se tivesse esta informação?

Answer 1:

Nestas condições, este deixa de ser um jogo de informação completa e passa a ser de informação perfeita (todos conhecem todo o estado do jogo) mas incompleta (parte do jogo depende do acaso). Posto isto, e sem necessidade de reinventar a roda, podemos considerar o expectiminimax de Donald Michie (https://en.wikipedia.org/wiki/Donald_Michie) que é em tudo semelhante ao Minimax, exceto nas decisões que envolvem probabilidade, nessas a abordagem é fazer uma média pesada (sendo os pesos as respetivas probabilidades) e assumir que é esse o valor da jogada (esta aboradagem converge para a melhor jogada possível). Tendo em conta que a probabilidade de o oponente efetuar qualquer jogada seria a mesma, basta apenas considerar pesos iguais o que resulta numa média aritmética das jogadas possíveis.

2012/2013 - Normal - b) pdf (../exames/2013_N.pdf) [ITERATIVE IMPROVEMENT: HILL CLIMBING, SIMULATED ANNEALING]

Question:

Explique o que caracteriza os algoritmos de pesquisa local (iterative improvement). Dê dois exemplos de algoritmos deste tipo, explicando sucintamente as suas diferenças.

Answer 1:

Os algoritmos de pesquisa local caracterizam-se pelo uso de heurísticas que permitem, para cada novo estado, testar a proximidade à solução e seleccionar a melhor opção, ignorando as outras. Dois exemplos de algoritmos deste tipo são o algoritmo “Hill Climbing”, que selecciona sempre o estado mais próximo da solução e o algoritmo “Simulated Annealing” que aceita, no início do processo de pesquisa da solução, algumas hipóteses mais afastadas da solução podem ser geradas, evitando ficar preso em máximos e mínimos locais.

2012/2013 - Normal - c) pdf (../exames/2013_N.pdf) [MINIMAX, ALFA-BETA]

Question:

Na aplicação do algoritmo minimax com cortes alfa-beta, explique que papel pode ter a ordenação dos nós gerados e avaliados pela função de avaliação.

Answer 1:

A ordenação dos nós gerados e avaliados pela função de avaliação permite reduzir a complexidade temporal da aplicação do algoritmo minimax com cortes alfa-beta. No pior caso, se a ordenação for exatamente da pior para a melhor jogada, todos os nós serão visitados e não haverá cortes (minimax normal com complexidade temporal O(n^m)) e no melhor caso, quando consideramos sempre a melhor resposta primeiro e portanto, podemos imediatamente cortar todas as jogadas seguintes, piores do que a do estado a avaliar, temos complexidade temporal O(n^(m/2)).

Topic 2 - Evolutionary Algorithms

Topic 3 - Uncertain Reasoning

2016/2017 - Normal - e) pdf (../exames/2017_N.pdf) [Näive-Bayes]

Question:

Explique qual é o compromisso subjacente ao uso da fórmula Näive-Bayes.

Answer 1:

O compromisso associado à fórmula do Teorema de Bayes:

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

é que pressupõe a independância entre os acontecimentos A e B e, como tal, estamos limitados a conclusões que assumem independência. Se tal não se verificar, as conclusões serão erradas. Um exemplo deste fenómeno é o de encontrar a palavra “viagra” e as palavras “blue pill” em emails e, no processo de os classificar em spam ou não, assumir que `P(viagra & bluepill) = P(Viagra) * P(bluepill)`, quando na verdade os emails que contém a palavra “viagra” costumam também conter “blue pill”, de certa forma, a mesma informação acaba por ser contabilizada duas vezes.

Topic 4 - Natural Language Processing (NLP)

rip

Topic 5 - Symbolic Learning

2016/2017 - Recurso - c) pdf (../exames/2017_R.pdf) [C4.5 INFORMAÇÃO DE SEPARAÇÃO]

Question:

No C4.5 para que serve a informação de separação?

Answer 1:

Uma das barreiras que o C4.5 ultrapassou face ao ID3 foi a de sobrevalorização de testes com muitos valores possíveis através da introdução da noção de razão do ganho face à medida do ganho. A razão do ganho, por sua vez, resulta do quociente entre o ganho de informação e a informação de separação - correspondendo portanto a uma medida normalizada do ganho. A razão pela qual o denominador é a informação de separação deve-se a este valor representar o que um dado atributo contribui para a separação da classe. Desta forma, o C4.5 permite que a comparação de atributos não seja enviesada para os que contêm muitos valores possíveis.

2012/2013 - Normal - e) pdf (../exames/2013_N.pdf) [SYMBOLIC vs SUBSYMBOLIC]

Question:

Em aprendizagem supervisionada, existem métodos simbólicos e métodos sub-simbólicos para a construção de modelos representativos dos exemplos pré-classificados. Indique o que distingue estes dois tipos de métodos e nomeie um exemplo de cada um deles.

Answer 1:

Estes dois tipos de métodos têm características muito diferentes, nomeadamente:

- Simbólicos: Passam pela aprendizagem estruturada de nova informação simbólica. Focam-se na aquisição/extração de conhecimento. Têm como vantagens: regra geral, permitir uma mais fácil inspeção e explicação dos resultados, dado que é possível rastrear os passos que levaram a uma dada conclusão. Exemplo: Árvores de Decisão
- Sub-simbólicos: Passa por refinar um procedimento através de tentativas sucessivas até atingir uma proximidade com o objetivo. Focam-se no refinamento de habilidades. Têm como vantagens: robustez face a ruído nos dados, melhor performance, menor necessidade de ter conhecimento de caso no início. Contudo, são mais difíceis de explicar, analisar e de fazer *debug*. Exemplo: Redes Neuronais

Topic 6 - Artificial Neural Networks (ANN)

2015/2016 - Normal - g) pdf (../exames/2016_N.pdf) [RNN]

Question:

Construiu-se uma rede neuronal com **30** neurónios de entrada, 1 camada escondida com 20 neurónios, e 2 neurónios na camada de saída. Os neurónios de cada camada ligam a todos os neurónios da camada seguinte. Em termos teóricos, quantos exemplos de treino são necessários para que a rede consiga generalizar?

Answer 1:

Se considerarmos o paralelismo entre arquitetura e treino de redes neuronais para com sistemas de equações, em que:

1. o número de ligações independentes = nº variáveis
2. nº saídas x nº exemplos = nº equações

Então, tentando verificar a equação `nº saídas x nº exemplos >= nº ligações independentes`, temos que `nº exemplos >= (nº ligações independentes / nº saídas)`. No caso concreto: `nº ligações independentes = 30*20 + 20*2 = 640`, ou seja, `NE >= (640/2) >= 320`.

Em termos teóricos, são necessários pelo menos 320 exemplos de treino para que a rede consiga generalizar.