LISTA DE EXERCÍCIOS 2 – INTELIGÊNCIA ARTIFICIAL

1)

1a e 1b)

Probabilidade a priori: baseada na crença original do agente, antes de qualquer jogada, sem qualquer outra informação. *Probabilidade a posteriori*: o agente espera uma jogada e observa, aí atualiza sua crença através dos dados novos obtidos.

1c)

O *Teorema de Bayes*. Bayes expressa as probabilidades incondicionais em termos das probabilidades condicionais (mais fáceis de obter/estimar). Permite obter probabilidades desconhecidas a partir de probabilidades conhecidas.

4.1)

Valor da Informação:

Serve para calcular o valor de adquirir cada possível evidência, isto é, quanto o agente pagaria para obter uma informação que ele ainda não possui, mas que pode lhe ser útil para maximizar sua utilidade esperada.

---

5)

O *Aprendizado Supervisionado* supõe a existência de um "professor" que te ensina que tipo de comportamento/resposta você deve exibir em cada situação/padrão de treinamento, enquanto que o *Aprendizado Não-Supervisionado* não supõe a existência deste "professor".

O *Aprendizado por Reforço* se preocupa com o como um *agente* deve *agir* em um *ambiente* de forma que maximize alguma noção de *recompensa* a longo tempo e se diferencia no sentindo em que pares de input/output corretos nunca são apresentados, nem as ações sub-ótimas são explicitamente corrigidas.

---

6)

Em *classificação*, entradas são divididas em duas ou mais classes, e o aprendiz deve produzir um modelo que vincula entradas não vistas a uma ou mais dessas classes. A filtragem de spam é um exemplo de classificação, em que as entradas são as mensagens de e-mails e as classes são "spam" ou "não spam".

Em *regressão*, as saídas são contínuas, em vez de discretas.

---

7)

A diferença é que a *recompensa imediata* é um valor diretamente associado ao estado s_1, enquanto que em *Q* associamos a cada estado o valor preferência do agente por tomar cada uma das possíveis ações e vamos atualizando Q de forma recursiva, até encontrar o melhor valor.

---

8)

Política em Aprendizado por Reforço:

É uma função que modela o comportamento do agente, mapeia estados e ações e pode ser vista como um conjunto de regras. Como em problemas de aprendizado por reforço não se tem conhecimento prévio do ambiente, por isso a probabilidade de transição é desconhecida e deve-se utilizar a interação do agente com o ambiente para definir uma política ótima.

---

9)

Para ter a possibilidade de conhecer outros estados e talvez encontrar um estado com maior valor, aperfeiçoando assim seu aprendizado e obtendo uma solução ainda melhor que a anterior.

---

12)

Critério de Entropia:

É uma medida da aleatoriedade de uma variável. A entropia de um conjunto pode ser definida como sendo uma medida do grau de impureza do conjunto Este conceito define a medida de "falta de informação", mais precisamente o número de bits necessários, em média, para representar a informação em falta, usando codificação ótima.

---

13)

A razão é simples, o agente não utiliza todos os exemplos para treinar o modelo, ao invés disto, ele os divide em conjunto de treinamento e conjunto de teste. Esta divisão ocorre, pois o agente treina com o conjunto de treinamento e mede a qualidade de predição do modelo no conjunto de teste com seus exemplos "novos". Isto ajuda a medir de forma mais precisa a capacidade de generalização do modelo, isto é, o quão bem as regras aprendidas se aplicam a exemplos não vistos durante o treinamento, tentando evitar assim o fenômeno chamado *overfitting* (uso de atributos irrelevantes).

---

14)

Árvore de Decisão: uma árvore de decisão tem como entrada um objeto ou situação descritos por um conjunto de atributos e como saída uma "decisão". (previsão do valor de saída dada a entrada). Uma árvore de decisão toma as suas decisões através de uma sequência de testes.

Floresta Aleatória: pode treinar com um subconjunto dos exemplos e também usando apenas um subconjunto dos atributos. Treina K árvores de decisão, em vez de apenas uma, sendo que a i-ésima árvore é treinada usando um subconjunto aleatório dos exemplos e durante a decisão de qual atributo testar no próximo nodo, escolhe dentro um subconjunto aleatório dos atributos

disponíveis. Obtém assim resultados mais consistentes de performance do que treinando apenas uma árvore e é mais fácil de ser implementada.

---

15)
15.1)
Naive Bayes: o agente utilizaria este modelo classificador em situações em que é preciso produzir *estimativas de probabilidade* em vez de simples classificações. Isto significa que, para cada rótulo de classe, o classificador pode gerar uma estimativa de o novo objeto pertencer à mesma. Utilizado quando se possui um conjunto de treinamento grande ou moderado e/ou os atributos que descrevem as instâncias forem *condicionalmente independentes dada a classe*.
Laplace Smoothing: como em Naive Bayes podemos resultar com probabilidades que são nulas durante o treinamento do modelo, e não queremos que isso aconteça, então para para nos livrarmos dese problema, temos o uso do Laplace Smoothing para somar 1 as características que não ocorem em uma amostra particular. Exemplo: ao recebermos um e-mail que contém uma palavra que nunca apareceu nos e-mails de amostras de teste, a probabilidade dele será sempre 0 para todos os valores, a não ser para os com predição baseada apenas nesta palavra. Isso é péssimo por desta forma ignoramos todas as outras palavras contidas no e-mail por conta apenas desta rara palavra.

---

16)
Regressão Linear: o Erro Médio Quadrático mede o quão preciso é o modelo de treinamento. É uma forma de avaliar a diferença entre um estimador e o verdadeiro valor da quantidade estimada, isto é, é a diferença entre o valor desejado e a saída obtida.

---

17)
17.1)
Perceptrons: os pesos do perceptron descrevem uma superfície de decisão linear em função das entradas. Todas as entradas "de um lado" da superfície geram saída +1, as demais geram saída 0. Redes com apenas uma única camada de neurônios só conseguem resolver problemas linearmente separáveis.
17.2)
Uma *rede neural multi-camada* (com, pelo menos, uma camada oculta de neurônios, além da camada de saída) é um classificador universal e pode resolver problemas não-linearmente separáveis podendo aproximar qualquer função (de valor discreto ou contínuo).

17.3)
Backpropagation dividido em duas fases:
Fase de propagação: sinal de entrada é propagado através de toda a rede, camada por camada gerando valores de saída da rede.
Fase de Adaptação/Treinamento: fase na qual ocorrem os ajustes dos pesos da rede, o fluxo de informação se dá da camada de saída em direção à camada de entrada e calcula-se as diferenças entre saída da rede e os valores desejados, podendo assim calcular parcelas individuais de erro para cada neurônio e corrigir os pesos segundo o algoritmo.

---

18)
18.1)
K-Médias: o algoritmo k-médias organiza os dados de acordo com a similaridade de conteúdo. É técnica mais simples de aprendizagem não-supervisionada. Consiste em focar k centróides de maneira aleatória, um para cada grupo (cluster). Associa cada exemplo ao centróide mais próximo, recalcula os centróides (a posição com base no centro de gravidade) com base nos indivíduos associados a ele e repete o processo.
18.3)
É preciso normalizar os atributos de entrada para que tenham a mesma faixa de valores, de modo que todos contribuam de maneira igual no cálculo da distância. Caso não ocorra a normalização, pode acontecer como o exemplo citado em aula, onde a produção de uniformes pode não ser boa, pois podemos ter uma pessoa com uma altura muito maior (outlier) que as outras ficando num cluster em que a roupa não vai ser do tamanho correto para ele.

---

19)
Overfitting: overfitting ocorre por uso de atributo que na verdade é irrelevante. Se a rede for muito complexa há maior chance de overfitting. É possível que o classificador faça uma indução muito específica para o conjunto de treinamento utilizado. Como este é apenas uma amostra de dados, é possível que a indução tenha bom desempenho no conjunto de treinamento, mas um desempenho ruim em exemplos diferentes daqueles pertencentes ao conjunto de treinamento. É possível detectar overfitting através do erro da hipótese sobre os dados de treinamento e do erro da hipótese sobre todos os dados. Para evitá-lo podemos obter mais dados de treinamento, utilizar um modelo mais simples, parar o crescimento quando a partição de dados não for estatisticamente significante, desenvolver uma árvore completa e então fazer uma poda (que pode ser feita diretamente na árvore ou ainda no conjunto de regras geradas pela árvore).
