Universidade de Aveiro

Inteligência Artificial (*LEI, LECI*)

Tópicos de IA:
Aprendizagem Automática

Ano lectivo 2024/2025

Regente: Luís Seabra Lopes

Aprendizagem

- Aprendizagem é qualquer mudança num sistema que lhe permite ter um melhor desempenho ao executar pela segunda vez uma tarefa [Simon, 1983]
- Aprendizagem é um processo orientado por objectivos através do qual se melhora o conhecimento usando a experiência e o próprio conhecimento [Michalski, 1994]

Aprendizagem = Inferência + Memorização

Aprendizagem: tipos de inferência

- Inferência **dedutiva** – preserva a verdade
  - Especialização dedutiva – restringir o conjunto de referência
    Se $\{ \forall x \ \ x \in A \Rightarrow p(x), B \subset A \}$ |- $\forall x \ \ x \in B \Rightarrow p(x)$
  - Generalização dedutiva – alargar o conjunto de referência
  - Dedução simples
  - Abstracção
- Inferência **indutiva** – não preserva a verdade, mas é essencial para a aprendizagem
  - Generalização indutiva – alarga o conjunto de referência; é o inverso da especialização dedutiva
    Se $\{ \forall x \ \ x \in B \Rightarrow p(x), B \subset A \}$ |- $\forall x \ \ x \in A \Rightarrow p(x)$
  - Especialização indutiva – o inverso da generalização dedutiva
  - Abdução – gera uma premissa a partir da qual se poderá deduzir uma dada observação
  - Concretização – Adiciona detalhes sobre o conjunto de referência.

Aprendizagem: níveis de supervisão

- Supervisão
  - Aprendizagem **supervisionada** – cada exemplo contém uma instância do conceito a aprender, que está devidamente identificado
    - Redes neuronais, árvores de decisão, etc.
  - Aprendizagem **semi-supervisionada** – apenas uma (pequena) pequena parte dos exemplos contém informação do conceito a aprender
  - Aprendizagem **por reforço** – o agente aprende o seu comportamento tendo em conta as recompensas (positivas ou negativas) que recebe pelas suas ações
  - Aprendizagem **não supervisionada** – neste caso é o próprio processo de aprendizagem que descobre um novo conceito
    - Algoritmos de agrupamento (*clustering*)

Aprendizagem: os dados

- Quantidade de exemplos
  - Muitos exemplos
    - Redes neuronais, árvores de decisão
  - Um ou poucos exemplos
    - Aprendizagem baseada em explicações (EBL): usa generalização dedutiva
    - Aprendizagem analógica / baseada em casos (CBR)
- Utilização de símbolos e números
  - Aprendizagem simbólica – o conhecimento aprendido está representado numa forma equivalente a lógica proposicional ou de primeira ordem
    - Regras, árvores de decisão, EBL, CBR
  - Aprendizagem conectionista / sub-simbólica
    - Redes neuronais, redes de Bayes, árvores de regressão

Aprendizagem baseada em colecções de exemplos: motivação

- Como aprender a prever qual vai ser a evolução do lucro numa empresa de produtos informáticos?

Idade	Competição	Tipo	Lucro
Velha	Não	Software	Desce
Intermédia	Sim	Software	Desce
Intermédia	Não	Hardware	Sobe
Velha	Não	Hardware	Desce
Nova	Não	Hardware	Sobe
Nova	Não	Software	Sobe
Intermédia	Não	Software	Sobe
Nova	Sim	Software	Sobe
Intermédia	Sim	Hardware	Desce
Velha	Sim	Software	Desce

Aprendizagem com colecções de exemplos: protocolo básico (I)

- Um conjunto de **atributos** ou características
  - $A = \{ A_1, ..., A_n \}$
- Cada atributo pode assumir valores dentro de um conjunto finito de valores simbólicos.
  - $A_i = \{ a_{i1}, ..., a_{ik} \}$
- Os objectos do domínio estão organizados em **classes**
  - $C = \{ C_1, ..., C_m \}$
- O problema é aprender a **reconhecer a classe** (=classificar) do objecto dada uma descrição desse objecto em termos dos atributos em A

Aprendizagem com colecções de exemplos: protocolo básico (I)

- O processo de aprendizagem baseia-se numa colecção de exemplos de treino, *S*.
- É gerada uma função $f : A \rightarrow C$ tal que:
  - $\forall x \; x \in S \; f(x)=classe(x)$
- Por generalização indutiva chega-se a:
  - $\forall$x f(x)=classe(x)
- Isto chama-se **Aprendizagem por Indução**

Aprendizagem de regras: pesquisa em profundidade (gulosa)

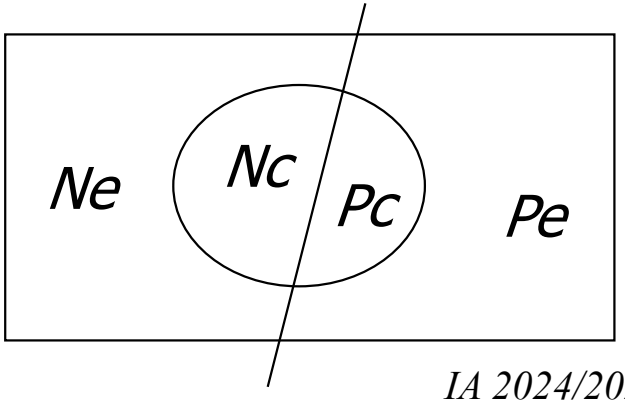
- Para cada classe *C*, fazer o seguinte
  1. *Regras* ← ∅
  2. *Pos* ← conjunto dos exemplos da classe *C*
  3. *Neg* ← restantes exemplos
  4. *Antecedente* ← *Verdade*
  5. Selecionar um novo *Teste*
  6. *Antecedente* ← *Antecedente* ∧ *Teste*
  7. Se *Antecedente* ainda cobre alguns exemplos em *Neg*, voltar a 5.
  8. *Regras* ← *Regras* ∪ { *Antecedente* ⇒ *C* }
  9. *Pos* ← *Pos* – { exemplos cobertos pelo *Antecedente* }
  10. Se *Pos* ≠ ∅, voltar a 4.

Em aprendizagem de regras, seleciona um teste do atributo

em regras, é comum gerar um conjunto de regras que cobre todas as classes.

Aprendizagem de regras: critérios

- Estão nesta categoria algoritmos bem conhecidos como o AQ e o CN2
- Critérios de comparação e selecção de regras para refinamento
  - *Pc*/*Tc* – sendo *Pc* o número de exemplos positivos cobertos e *Tc* = *Pc*+*Nc* o número total de exemplos cobertos
  - *Pc*+*Ne* – sendo *Pc* o número de exemplos positivos cobertos e *Ne* o número de exemplos negativos excluídos



Aprendizagem de árvores de decisão: algoritmo

- A árvore de decisão é gerada através de um processo recursivo descendente (*TDIDT – Top-Down Induction of Decision Trees*)

TDIDT(*Exemplos*)
  se todos os *Exemplos* pertencem a uma classe *C*,
    então *Arvore.classe* = *C*;
  senão:
    $A \leftarrow$ atributo de teste para *Exemplos*
    Arvore.teste $\leftarrow A$
    para cada valor a_i de A:
      $E_i \leftarrow$ subjconjunto de *Exemplos* em que $A=a_i$
      $Arvore.subarv_i \leftarrow$ TDIDT(E_i)
  retornar Arvore

Em árvores de decisão, cada nó testa um atributo e considera todos os valores possíveis

Em árvores de decisão, uma única árvore é geralmente gerada para lidar com todas as classes.

Árvores de decisão: selecção do atributo de teste (I)

- Podemos ver o domínio dos exemplos como uma fonte de mensagens, cada uma delas representando uma das classes possíveis
- Baseado na **Teoria da Informação**
  - Entropia *apriori*: $H(C) = -\Sigma\ p(C_i) \times log_2(p(C_i))$
  - Entropia *aposteriori*, dado o valor de um atributo: $H(C|a_{j,k}) = -\Sigma_i\ p(C_i|a_{j,k}) \times log_2(p(C_i|a_{j,k}))$
  - Entropia global *aposteriori*: $H(C|A_j) = \Sigma_k\ p(a_{j,k}) \times H(C|a_{j,k})$

Árvores de decisão: selecção do atributo de teste (III)

- **Ganho de informação**
  - Ou seja, redução da entropia

$$I(C;A_j) = H(C)\text{-}H(C|A_j)$$

- As probabilidades podem ser estimadas com base nos exemplos disponíveis
- Nota: Este método funciona mal quando os atributos têm muitos valores possíveis

Árvores de decisão: selecção do atributo de teste (II)

- **Razão do ganho**
  - $H(A_j) = -\Sigma\, p(a_{j,k}) \times log_2(p(a_{j,k}))$
  - $R(C;A_j) = I(C;A_j) \,/\, H(A_j)$
  - Resolve o problema dos atributos com muitos valores.
  - Quando $H(A_j)$ se aproxima de zero, a razão do ganho fica instável; por isso, são excluídos à partida os atributos cujo ganho de informação seja inferior à média

Árvores de decisão: selecção do atributo de teste (III)

- **Critério GINI**
  - Impureza *apriori*
    - $G = \Sigma_{m \neq n}\, p(C_m) \times p(C_n)$
  - Impureza *aposteriori*:
    - $G(A_j) = \Sigma\, p(a_{j,k}) \times \Sigma_{m \neq n}\, p(C_m \,|a_{j,k}) \times p(C_n|a_{j,k})$

Alguns problemas (I)

- Tratamento do **ruído** – por vezes, os exemplos de treino contém ruido, ou seja, particularidades não representativas do domínio que podem levar o algoritmo de aprendizagem a fazer uma generalização incorrecta.
- **Atributos numéricos** – como usá-los nas regras ou nas árvores de decisão?
- Atributos com valores não especificados nos exemplos

Alguns problemas (II)

- Levar em conta o **custo de cálculo** de cada atributo
- **Aprendizagem incremental**
- Aprendizagem por indução em **lógica de primeira ordem**
  - FOIL

Árvores de decisão: tratamento do ruído

- Parar a expansão da árvore quando o número de exemplos disponíveis é inferior a um dado limiar
- Ter um estimativa do erro, e parar a expansão quando a estimativa do erro começa a subir
- Ter um estimativa do erro, e parar a expansão quando essa estimativa sobe para além de um dado limiar.
- Expandir completamente a àrvore e no fim podá-la.

Avaliação de algoritmos de aprendizagem supervisionada

- **Complexidade computacional**
  - Tanto na aprendizagem como na utilização
- **Legibilidade** – a representação do conhecimento aprendido deve ser tão legível quanto possível
  - Especialmente relevante em sistemas de apoio à decisão
- **Precisão** – o conhecimento aprendido deve ser tão preciso quanto possível
  - No caso de problemas de classificação, a precisão é avaliada experimentalmente como a percentagem de erros de classificação num conjunto de exemplos de teste (não usados na aprendizagem).

Avaliação experimental da precisão em aprendizagem supervisionada (I)

- Partição dos exemplos disponíveis em dois subconjuntos:
  - Subconjunto de treino – exemplos usados para a aprendizagem (p. ex. 2/3 de todos os exemplos)
  - Subconjunto de teste – exemplos usados na avaliação experimental da precisão (p. ex. 1/3)

Avaliação experimental da precisão em aprendizagem supervisionada (II)

- **Validação-cruzada-*k***

k = subconjuntos
por exemplo para k=4,
pega num, usa esse para testar e os outros 3 para treinar

  - Divide-se o conjunto de exemplos disponíveis em k subconjuntos
  - Para cada subconjunto S_i, treinar usando todos os outros e testar em S_i
  - A precisão é dada pela percentagem global de erros (após todas as iterações treino-teste)

- **Um-de-fora**

k é o número de exemplos

  - Equivale à validação-cruzada-k, para o caso em que k é o número total de exemplos disponíveis