UNIDADE II: APRENDIZAGEM SUPERVISIONADA

- Sumário:
  - Programação lógica indutiva
  - Algoritmo FOIL

Aprendizagem de regras

- O algoritmo AQ permite a aprendizagem de um conjunto de regras a partir de exemplos
- As regras aprendidas se expressam em lógica proposicional

Limitações da LP

- A LP apresenta limitações quanto ao seu poder expressivo
- Se considera por isso a aprendizagem de regras expressas noutro tipo de lógica

Limitações da LP

- Considerar a aprendizagem do conceito ***Filha(x, y)***
  - Verdadeiro se x é filha de y
- Cada pessoa nos dados se expressa através dos atributos ***Nome***, ***Mãe***, ***Pai***, ***Homem***, ***Mulher***
- Exemplos de treino → dados pela descrição de duas pessoas, seguido do atributo classe ***Filha***
  - <$Nome_1$=Sharon, $Mãe_1$=Luísa, Pai_1=Roberto, $Homem_1$=Falso, $Mulher_1$=Verd, $Nome_2$=Roberto, $Mãe_2$=Ana, Pai_2=Víctor, $Homem_2$=Verd, $Mulher_2$=Falso, $Filha_{12}$=Verd>

Limitações da LP

- Se (Pai_1=Roberto) E ($Nome_2$=Roberto) E ($Mulher_1$=Verd) Então $Filha_{12}$ = Verd
  - Regras proposicionais aprendidas são muito específicas
- Em lógica de primeira ordem
  - Se ***Pai(y, x)*** E ***Mulher(y)*** Então ***Filha(x, y)***
- Permite também
  - Fazer referência nos antecedentes à variáveis que não aparecem no consequente
  - Fazer a descrição de regras de forma recursiva...

Programação Lógica Indutiva

- A aprendizagem de regras expressas em lógica de primeira ordem se designa *Programação Lógica Indutiva (ILP)*
- Pode ser vista como a inferência de programas em PROLOG a partir de exemplos

LPO

- Elementos básicos são os símbolos
  - Constantes: *Roberto, Luísa*
  - Variáveis: *x, y*
  - Predicados: *Casado, MaiorQue*
  - Funções: *idade*
- Com os símbolos se constroem expressões
  - Termos: qualquer constante, variável ou função aplicada a qualquer termo. Roberto, x, idade(Luísa)
  - Literal: qualquerpredicado ou a sua negação aplicado a qualquer termo. Casado(Roberto, Luísa), ¬MaiorQue(idade(Ana), 23)

LPO

- ***Cláusula***: qualquer disjunção de literais, onde todas as variáveis se assumem como sendo quantificadas universalmente
- ***Cláusula de Horn***: cláusula contendo um único predicado que se deduz de um conjunto de condições
  - ***Avô(x, y)*** <- ***Pai(x, z)*** E ***Pai(z, y)***
- ***Regra de Horn***: conjunto de ***cláusulas de Horn*** que deduzem o mesmo predicado
  - ***Avô(x, y)*** <- ***Pai(x, z)*** E ***Mãe(z, y)***

LPO

- Definição de um conceito pode ser feito de duas formas
  - Implícita: expressa através de ***Regras de Horn***
    - Definição anterior de avô
  - Explícita: através da listagem de todas as instanciações possíveis do conceito
    - ***Avô(João, Pedro)***, ***Avô(João, Luís)***, ***Avô(João, Maria)***

ILP

- O objectivo da ILP é encontrar uma descrição de um conceito alvo em termos relacionais (lógica de primeira ordem ou equivalente)
- Existem vários algoritmos de entre eles o FOIL desenvolvido por Ross Quinlan
  - Implementa uma estratégia de cobertura sequencial, parecida ao algoritmo AQ

FOIL

- Entradas:
  - Conjunto de exemplos positivos e negativos do conceito alvo
  - Conhecimento do meio, representado em forma explícita
- Saída:
  - Conjunto de regras de primeira ordem, semelhantes à cláusulas de Horn com duas excepções
    - As regras geradas por FOIL não contêm funções
    - As regras geradas por FOIL podem conter literais negativos

FOIL

- Desde o ponto de vista da busca:
  - Conjunto de estados: conjunto de regras de Horn formado a partir dos predicados definidos nos exemplos e na teoria do domínio
  - Conjunto de operadores
    - Acescentar um literal (condição) a uma cláusula de Horn
    - Criar uma nova cláusula vazia
  - Estado inicial: conjunto vazio de regras de Horn
  - Alvo: regra de Horn que descreve aos exemplos positivos e não aos negativos
  - Heurística: ganho de informação

Algoritmo

```
Função FOIL(Pred_alvo, Predicados, Exemplos)
Pos ← Exemplos para os quais Pred_alvo é verdadeiro
Neg ← Exemplos para os quais Pred_alvo é falso
Regras ← {}
While Pos do
  Aprender uma NovaRegra
  NovaRegra ← Regra que prediz Pred_alvo sem pré-condições
  NovaRegraNeg ← Neg
  while NovaRegraNeg do
      Especializar NovaRegra
      Literais_cand ← literais candidatos gerados
      Literal_melhor ← argmax Ganho(L, NovaRegra)
      Adicionar Literal_melhor aos antecedentes de NovaRegra
      NovaRegraNeg ← subconjunto de NovaRegra que satisfaz
        antencedentes de NovaRegra
  Regras ← Regra + NovaRegra
  Pos = Pos – {membros de Pos cobertos por NovaRegra}
Retorna Regras
```

Algoritmo

- Se estrutura igualmente na forma de dois ciclos
  - No ciclo externo
    - Aprende-se uma regra de cada vez, removendo os exemplos positivos cobertos antes de aprender a próxima regra
    - O efeito de cada nova regra é o aumento do número de instâncias positivas cobertas -> busca do específico para o geral
  - No ciclo interno
    - Se implementa uma busca refinada para definir a forma exacta de cada nova regra
    - Especializa a regra, acrescentando um literal de cada vez até que esta deixe de cobrir os exemplos negativos → Busca do geral para o específico

Geração de especializações

- Para a especialização das regras o algoritmo gera vários literais que podem ser acrescentados individualmente ao seu antecedente
- Considerando uma regra $P(x_1, x_2, ..., x_\kappa) \leftarrow L_1 ... L_n$, os literais gerados teriam uma das formas:
  - $Q(v_1, ..., v_r)$
    - Q é qualquer nome de predicado existente no conjunto de predicados
    - v_i é uma variável nova ou já presente na regra
    - Ao menos uma das variáveis v_i deve existir na regra
  - Igual(x_j, x_k), x_j e x_k são variáveis presentes na regra
  - A negação de qualquer das formas anteriores

Exemplo

- Consideremos a aprendizagem de regras para a predicção do literal ***Neta(x, y)*** sendo os outros predicados utilizados para a descrição dos exemplos ***Pai*** e ***Mulher***.
- Supor que *P(x, y)* significa "o *P* de *x* é *y*"
  - Início
    - ***Neta(x, y)***
  - Literais candidatos:
    - ***Igual(x, y)***, ***Mulher(x)***, ***Mulher(y)***, ***Pai(x, y)***, ***Pai(y, x)***, ***Pai(x, z)***, ***Pai(z, x)***, ***Pai(y, z)***, ***Pai(z, y)***
    - A negação de qualquer dos literais anteriores

Exemplo

- Supondo que o mais prometedor seja ***Pai(y, z)*** teríamos a regra
  - ***Neta(x, y)*** ← ***Pai(y, z)***
- Novos literais candidatos
  - Todos os anteriores
  - ***Mulher(z)***, ***Igual(z, x)***, ***Igual(z, y)***, ***Pai(z, w)***, ***Pai(w, z)*** e suas correspondentes negações
- Seleccionando nesta iteração ***pai(z, x)***
  - ***Neta(x, y)*** ← ***Pai(y, z) E Pai(z, x)***
- Regra completa???

Condução da busca

- Em cada iteração FOIL selecciona e acrescenta à regra o literal correspondente ao maior ganho de informação
- A utilidade da adição de um novo literal se baseia na quantidade de exemplos positivos e negativos cobertos antes e depois da referida adição
  - $G(L,R) = t(log_2 \frac{p_1}{p_1+n_1} - log_2 \frac{p_0}{p_0+n_0})$
  - P0, n0, p1, n1 → número de exemplos positivos e negativos antes e depois da adição
  - t → número de exemplos positivos cobertos antes que seguem sendo cobertos depois da adição

Aprendizagem de regras recursivas

- No exemplo anterior foi ignorada a possibilidade de inclusão do predicado alvo entre os novos literais gerados
- Permite a formação de regras recursivas
  - Se ***Pais(x, y)*** Então ***Ancestral(x, y)***
  - Se ***Pais(x, z)*** E ***Ancestral(z, y)*** Então ***Ancestral(x, y)***

Bibliografia

- Mitchell, pg. 283 – 291
- Borrajo Millán, 251 – 269