CURSOS INTERSEMESTRALES

PROTECO

Minimax

Curso intersemestral de Inteligencia Artificial

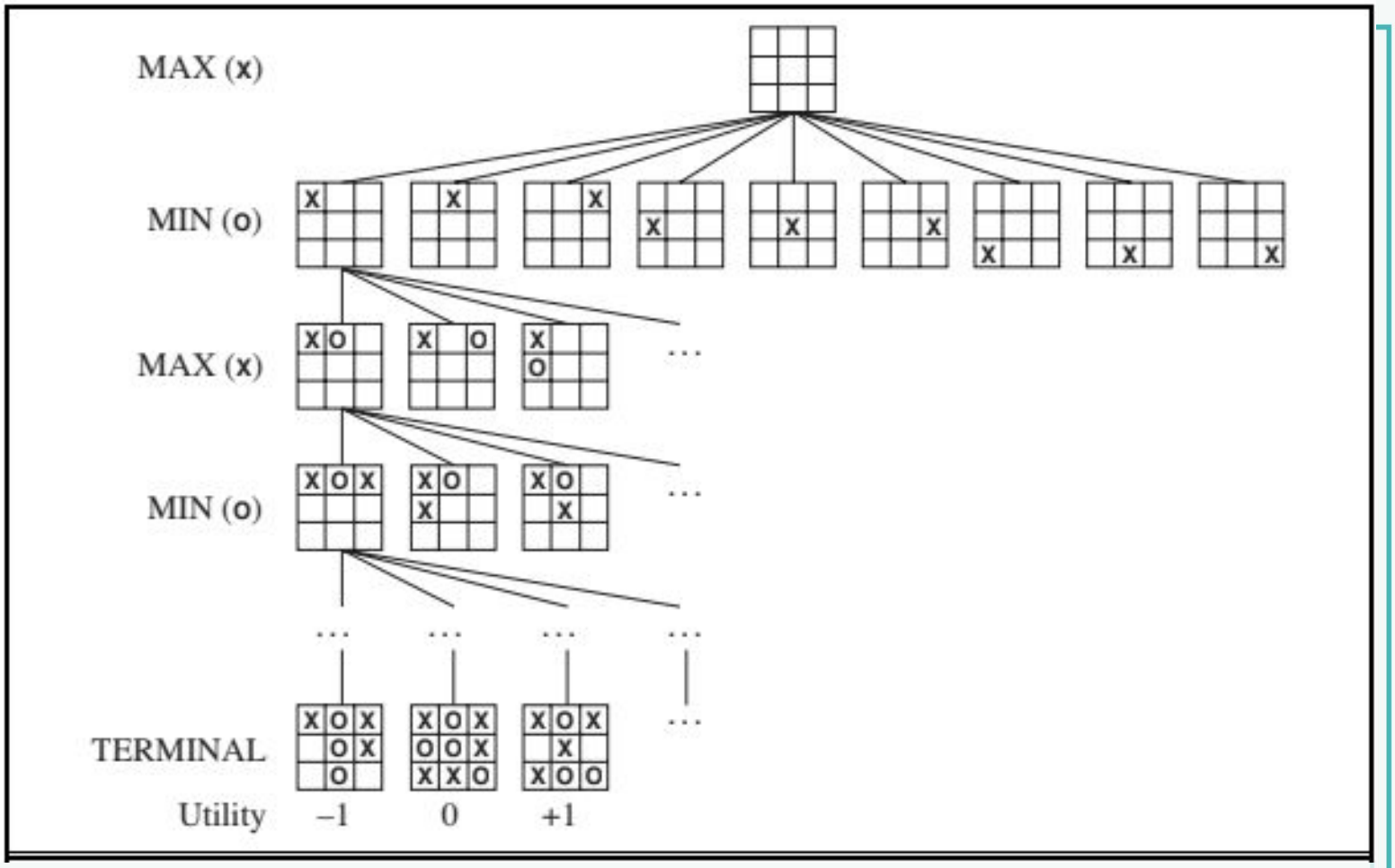
Búsqueda con adversarios

Además de estado inicial,acciones, resultados, estados finales, debemos tener en cuenta el número de jugadores (agente+adversarios) y la utilidad.

La utilidad es un valor numérico que obtiene un jugador en el estado final del juego.

Ganar	**100**
Empatar	**0**
Perder	**-100**

PROTECO



PROTECO

Algoritmo minimax

Se basa en búsqueda por profundidad

Por cada acción del jugador 1 debemos analizar las posibilidades del jugador 2.

Se asignan valores minimax a cada estado, cuyo valor para un estado final corresponde a la utilidad.

MiniMax(estado final) -> utilidad(estado final)

Asume que el contrincante actúa de manera optima.

PROTECO

MiniMax(estado):

MiniMax(estado final) -> utilidad(estado final)

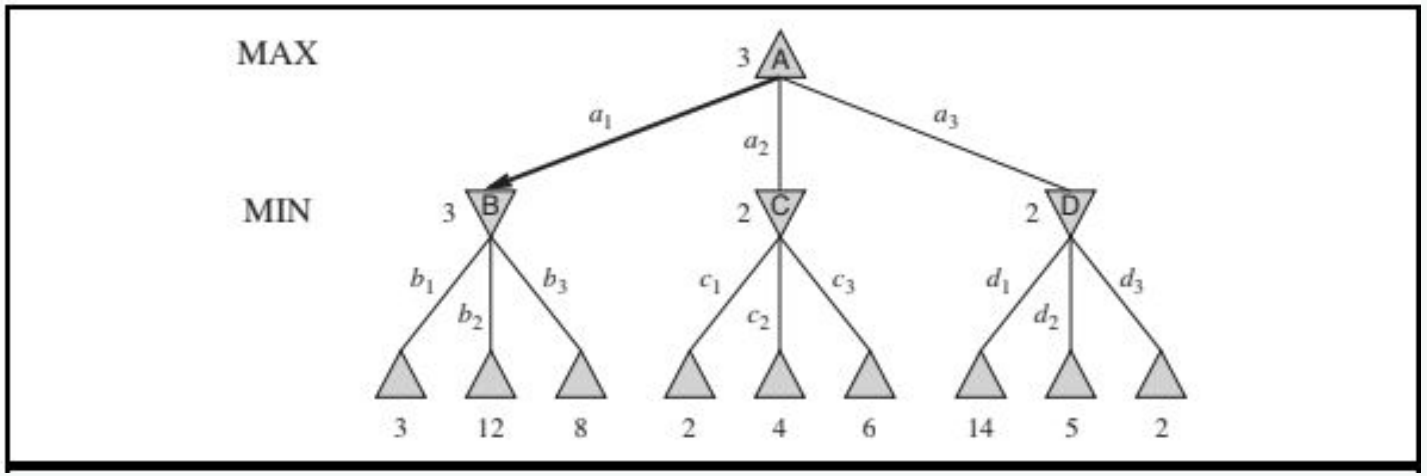
Jugador(estado):

    max   MiniMax(estado(resultado, acción)

Jugador(estado):

    min   MiniMax(estado(resultado, acción)

PROTECO

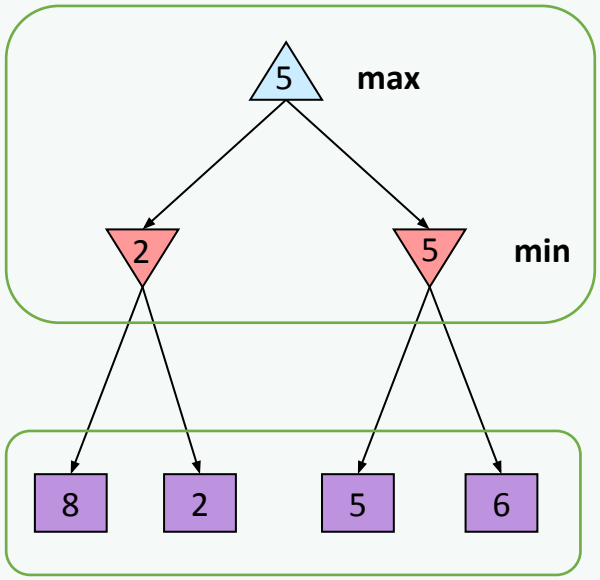


MAX nuestro agente inteligente queremos maximizar

MIN contrincante queremos minimizar

PROTECO

Valores minimax: computados de forma recursiva



Valores terminales: parte del juego

El valor minimax corresponde a la utilidad de MAX.

Aplicable en juegos por turnos.

Se asume que ambos agentes son óptimos y que es posible calcular todo el árbol.

PROTECO

Pseudocódigo.

```
function MINIMAX(problema) returns acción
    inicializar mejorAcción y mejorUtilidad
    inicial -> poblema.estadoInicial
            for each  acción  in problema.acciones (iniciar)
                resultado problema.resultado( inicial, acción)
            utilidad ValorMin(problema,resultado)
            if utilidad > utilidadMayor
            mejorAcción = accion
            mejorUtilidad = utilidad

        return  mejorAccion
```

PROTECO

```
function MIN(problema) returns utilidad
    if  problema.ESTADO_FINAL(estado)
         return  problema.utilidad(estado)
         menorValor
     for each  acción  in problema.acciones (iniciar)
                    resultado problema.resultado( inicial, acción)
               utilidad ValorMax(problema,resultado)
               if utilidad < valorMenor
               valorMenor = utilidad

        return  menorValor
```

PROTECO

```
function MAX(problema) returns utilidad
    if  problema.ESTADO_FINAL(estado)
         return  problema.utilidad(estado)
         mayorValor
     for  each   acción  in problema.acciones (iniciar)
                    resultado problema.resultado( inicial, acción)
              utilidad ValorMin(problema,resultado)
              if utilidad > valorMayor
              valorMayor = utilidad

        return  mayorValor
```

PROTECO

Eficiencia de minimax

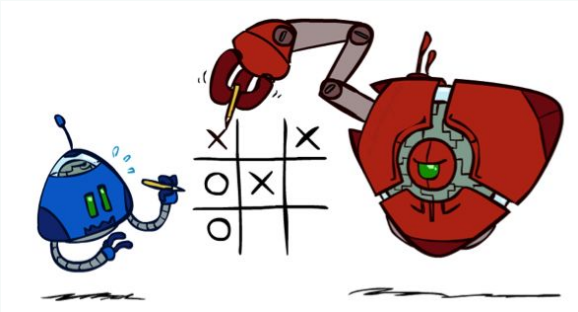
¿Qué tan eficiente es minimax?

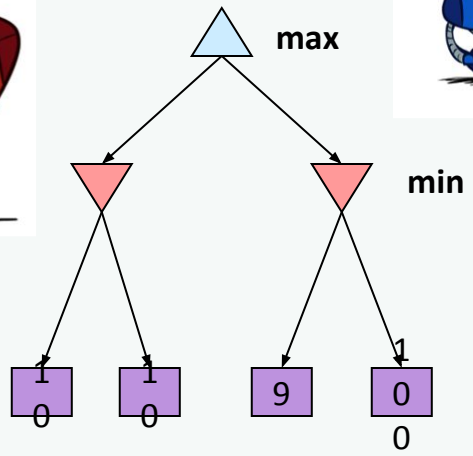
- Tanto como DFS exahustivo
- Tiempo: $O(b^m)$
- Espacio: O(bm)

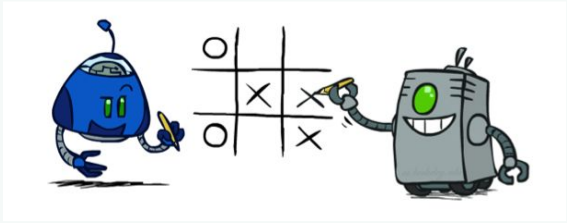
Ejemplo: Para ajedrez $b \approx 35, m \approx 100$

- Una solución completa es definitivamente imposible
- ¿Necesitamos explorar todo el árbol?

Propiedades de minimax





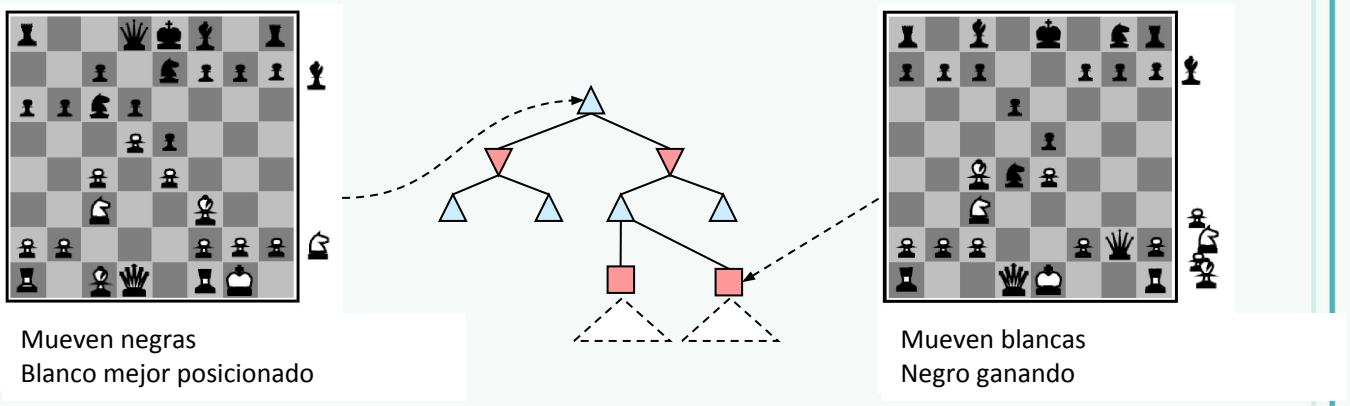


Óptimo contra un jugador perfecto.
¿De otro modo?

[Dibujos: UC Berkeley CS188 Materials http://ai.berkeley.edu.]

Funciones de evaluación

La función de evaluación puntúa no-terminales en la búsqueda de profundidad limitada



Mueven negras
Blanco mejor posicionado

Mueven blancas
Negro ganando

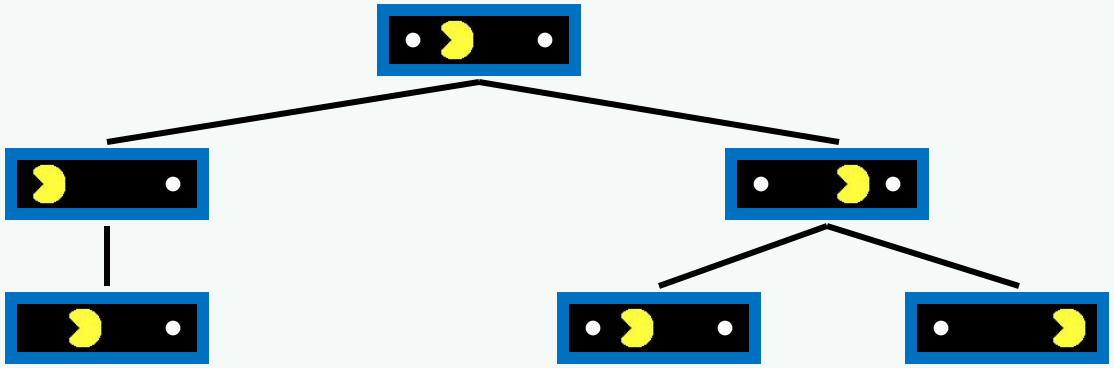
Función ideal: regresa el valor minimax *exacto* de esa posición

En la práctica: Típicamente, una suma lineal ponderada de los factores:

$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$

e.g. $f_1(s)$ = (num de reinas blancas– num reinas negras), etc.

Porque pacman se mata de hambre



Un peligro de los agentes re-planificadores!

- Sabe que su puntaje subirá comiendo ahora el punto (oeste, este)
- Sabe que su puntaje subirá igual si lo come después (este, oeste)
- No hay oportunidades de anotar puntos después de comerlo, (dentro del horizonte, 2 aquí)
- Por lo tanto, esperar parece tan bueno como comer:

PROTECO