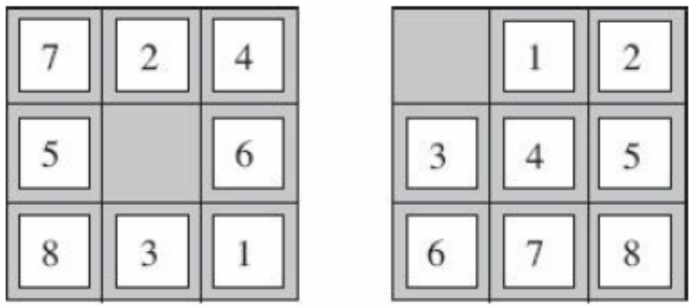
Data Science Academy

www.datascienceacademy.com.br

Introdução à Inteligência Artificial

Funções Heurísticas

Examinaremos a heurística para o quebra-cabeças de oito peças, a fim de esclarecer a natureza da heurística em geral. O quebra-cabeças de oito peças é um dos primeiros problemas de busca heurística. O objetivo do quebra-cabeças é deslizar as peças horizontal ou verticalmente para o espaço vazio até que a configuração corresponda à configuração objetivo, conforme a figura abaixo.



Estado inicial   Estado objetivo

O custo médio da solução para uma instância do quebra-cabeças de oito peças gerada aleatoriamente é de cerca de 22 passos. O fator de ramificação é cerca de 3 (quando a peça branca estiver no meio, é possível quatro movimentos, quando estiver em um canto, dois; quando estiver ao longo de uma borda, três). Isso significa que uma busca exaustiva da árvore de profundidade de 22 passos ficaria em cerca de 322 ≈ 3,1 × 1010 estados. Uma busca em grafos reduziria isso de um fator de cerca de 170.000, porque apenas 9!/2 = 181.440 estados distintos são alcançáveis. Esse é um número gerenciável, mas o número correspondente para o quebra-cabeças de 15 é de aproximadamente 1013, assim a próxima tarefa é encontrar uma boa função heurística. Se quisermos encontrar as soluções mais curtas usando A*, precisamos de uma função heurística que nunca superestime o número de passos até o objetivo. Há um longo histórico de tais heurísticas para o quebra-cabeças de 15 peças. Seguem as duas mais utilizadas:

• h1 = número de peças fora de lugar. Para a figura no estado inicial, todas as oito peças estão fora de lugar, de modo que o estado inicial teria h1 = 8. h1 é uma heurística admissível porque é claro que qualquer peça que esteja fora de lugar deverá ser movida pelo menos uma vez.

• h2 = soma das distâncias das peças de suas posições-objetivo. Devido às peças não poderem ser movidas ao longo de diagonais, a distância que vai contar é a soma das distâncias horizontal e vertical. Isso, às vezes, é chamado de distância Manhattan. h2 é também admissível porque todo movimento que pode ser feito move uma peça um passo mais perto do objetivo. As peças 1-8 no estado inicial dão uma distância Manhattan de:

h2 = 3 + 1 +2 + 2 + 2 + 3 + 3 + 2 = 18.

Como esperado, nenhuma delas superestima o custo da solução verdadeira, que é 26.

Uma função heurística h(n) deve ser capaz de estimar o custo de uma solução começando pelo estado do nó n. Como um agente poderia construir tal função? Ele poderia conceber problemas relaxados para os quais uma solução ótima pode ser facilmente encontrada. Outra solução é aprender com a experiência. “Experiência” aqui significa, por exemplo, resolver muitos quebra-cabeças de oito peças. Cada solução ótima para o problema do quebra-cabeças de oito peças fornece exemplos dos quais h(n) pode ser aprendido. Cada exemplo consiste em um estado do caminho da solução e do custo real da solução a partir desse ponto. A partir desses exemplos, pode ser utilizado um algoritmo de aprendizagem para construir uma função h(n) que pode (com sorte) prever os custos de solução para outros estados que surgirem durante a busca. É possível fazer isso utilizando técnicas como redes neurais, árvores de decisão e outros métodos.

Os métodos de aprendizagem indutiva funcionam melhor quando supridos de características de um estado que são relevantes para predizer o valor do estado, em vez de apenas uma simples descrição do estado. Por exemplo, a característica de “número de peças fora do lugar” pode ser útil em predizer a distância real de um estado a partir do objetivo. Vamos chamar essa característica de x1 (n). Poderíamos extrair 100 configurações geradas aleatoriamente do quebra-cabeças de oito peças e reunir estatísticas sobre seus custos reais de solução. Podemos considerar que, quando x1(n) for 5, o custo médio de solução será cerca de 14, e assim por diante. Tendo em conta esses dados, o valor de x1 poderá ser utilizado para prever h(n). Certamente poderemos utilizar várias características. Uma segunda característica x2(n) pode ser o “número de pares de peças adjacentes que não são adjacentes no estado objetivo”. Como deveríamos combinar x1(n) e x2(n) para prever h(n)? Uma abordagem comum é usar uma combinação linear: h(n) = c1x1(n) + c2x2(n).

Referências:

Livro: Inteligência Artificial
Autor: Peter Norvig