Sheepdog Bot 1

1) 양이 이미 펜 안에 있는 상태 또는 로봇이 한 번에 포획하기 위해 움직일 수 있는 위치에 있는 상태입니다.

2) T_*(state) = min(T_*(봇이 a를 움직인 후 상태), T_*(양이 a를 움직인 후 상태), T_*(봇과 양이 모두 a를 움직인 후 상태) 여기서 a는 봇이 할 수 있는 가능한 행동 중 하나다.

3) T_*(state)의 가장 작은 값을 초래하는 행동을 선택하는 것만으로 양치기 로봇이 취할 최적의 행동을 결정할 수 있다.

4) 모든 상태에 대해 T_*(state)를 계산하는 것은 계산 비용이 많이 들지만 동적 프로그래밍을 사용하여 계산할 수 있다. 기본 아이디어는 양이 이미 펜 안에 있는 상태에서 시작하여 다른 모든 상태에 대해 T_*(state)의 값을 재귀적으로 계산하는 것이다. 양이 왼쪽 위에서 시작하고 로봇이 오른쪽 아래에서 시작하면 양을 포획하는 데 필요한 예상 이동 횟수는 120회다.

5) 로봇이 어느 위치에서든 시작할 수 있고 임의의 빈 공간에서 양이 도입될 경우 로봇이 시작하기에 최적의 장소는 그리드의 중앙. 여기서 시작할 때 양을 잡기 위한 예상 이동 횟수는 90회다.

6) T_*(state)와 최적의 동작을 기반으로 양치기 봇과 양의 상호 작용을 시뮬레이션할 수 있다. 시뮬레이션된 데이터는 임의성으로 인해 예상과 비슷하게 일치하나, 약간의 변동이 있을 것이다.

Sheepdog Bot 2

1) 각 상태가 314개의 0과 1의 벡터로 표현되는 one-hot 인코딩을 사용할 수 있지만, 계산적 비용이 많이 들 것이다. 우리는 양의 벡터와 로봇의 위치와 같은 저차원 표현을 사용할 수 있다. 덜 정확할 것이지만, 계산 비용도 덜 들 것이다.

2) MSE가 낮으면 훈련 데이터에 적합한 모델인데, 유효성 검사 데이터는 모델이 이전에 본 적이 없는 데이터의 집합이다. 유효성 검사 데이터의 MSE가 훈련 데이터의 MSE와 유사한 경우, 모델은 과적합되지 않는다. 우리는 모델의 정확도를 볼 수 있다. 정확도는 모델이 정확하게 예측하는 상태의 백분율이다. 정확도가 높으면 모델은 데이터에 잘 적합한 것이다.

3) 봇 2는 대부분의 경우 양을 안정적으로 잡는다. 봇 2는 대부분의 경우 봇 1보다 성능과 계산 효율이 좋다. 모든 경우 그런 것은 아니다.