3
Aprendizado por reforço

Aprendizado por reforço é um ramo estudado em estatística, psicologia, neurociência e ciência da computação. Atraiu o interesse de pesquisadores ligados a aprendizado de máquina e inteligência artificial, e é um método de programação de agentes através do oferecimento de recompensas e punições, sem a necessidade de especificar como uma tarefa deve ser realizada.

É entendido como o problema encontrado por um agente que deve aprender como se comportar em um ambiente dinâmico através de interações do tipo "tentativa e erro". A abordagem que é utilizada nesse trabalho é feita usando-se técnicas estatísticas e métodos de programação dinâmica, buscando estimar qual a vantagem em se tomar determinadas ações em diferentes estados do ambiente.

Uma boa referência no assunto é [KAE], e neste capítulo são apresentados os principais aspectos necessários para a compreensão da abordagem prática, a ser apresentada no capítulo seguinte.

3.1.
Conceitos básicos

Em um ambiente de aprendizado por reforço, um agente está inserido em um ambiente *T* e interage com ele através de percepções e ações, conforme a Figura 1. A cada passo, o agente recebe como entrada *e*, uma indicação do estado (*s*) atual do ambiente. O agente escolhe, então, uma ação *a* a tomar, e gera sua saída. A ação altera então o estado do ambiente, e uma medida dessa mudança de estado é informada ao agente através de um valor de *sinal de reforço*, *r*. A função que mapeia os estados do ambiente nas ações que o agente deve tomar é definida como a *política* do agente. A política de comportamento do agente, *P*, deve escolher tomar ações que maximizem o valor final da soma dos reforços recebidos

em um intervalo de tempo. Tal política de comportamento pode ser aprendida através de um processo de tentativa e erro, guiado por diferentes algoritmos.

Figura 1 – Modelo padrão de aprendizado por reforço

Formalmente, o modelo é constituído por:

- Um conjunto discreto de estados que o ambiente pode assumir;
- Um conjunto discreto de ações que o agente pode tomar sobre o ambiente;
- Um conjunto de valores escalares de reforço; geralmente $\{0,1\}$, ou os números reais.

Na Figura 1 existe ainda a função de entrada *e*, que é a maneira como o agente "lê" o estado atual do ambiente.

A tarefa que o agente deve desempenhar é encontrar uma política π, que mapeia estados em ações, que maximiza a medida de reforço a longo prazo. Geralmente o sistema é não-determinístico; isso é, uma mesma ação tomada em um mesmo estado pode levar a diferentes estados, com diferentes valores de retorno percebidos.

Ao contrário dos métodos conhecidos de aprendizado supervisionado, no aprendizado por reforço não existem pares "entrada/saída", para serem utilizados no treinamento. Após tomar uma ação, o agente imediatamente recebe uma recompensa, mas não fica sabendo qual deveria ser a melhor ação para atingir o

objetivo (maximizar o retorno a longo prazo). Ele precisa obter experiência dos possíveis estados, ações, transições e recompensas do sistema para atingir a otimalidade.

3.1.1. Exploração x Intensificação

Um agente deve aprender, através de suas interações com o ambiente, qual a política que melhor o leva a atingir seu objetivo. Entretanto, como o agente não possui conhecimento sobre todas as variáveis envolvidas no processo, é necessário que ele explore, isso é, busque políticas alternativas àquelas que já utiliza e compare seus resultados com os já aprendidos até o momento. Digamos que, com o conhecimento adquirido até um determinado momento, o agente julgue que, para um dado estado s, a melhor ação a ser tomada seja a_1. Pode ser que exista uma ação alternativa a_2 que leve a um melhor resultado. Só será possível ao agente aprender essa nova informação se ele possuir um mecanismo de exploração que tome ações aleatórias com uma determinada taxa.

Essa taxa de exploração, se for muito alta, pode comprometer o alcance do objetivo, que é maximizar as recompensas. Um agente que decida explorar muito seu ambiente pode deixar de aproveitar o conhecimento já adquirido.

Soluções híbridas são interessantes: no início do processo, quando o agente ainda possui poucas informações sobre o ambiente, a taxa de exploração mantém-se alta; com o tempo essa taxa cai, levando a um maior aproveitamento do conhecimento adquirido e, consequentemente, maior probabilidade de obtenção de maiores retornos.

3.1.2. Política ótima

Em um problema qualquer de aprendizado por reforço, deseja-se que o agente aprenda uma política que seja eficaz, no sentido de obter, ao final da execução de uma sequência (limitada ou não) de ações, o melhor valor possível como recompensa acumulada.

Quando o sistema encontra-se em determinado estado *s*, o valor recebido $V_\pi(s)$ como recompensa, após a execução das ações seguindo uma política π, é definido como as soma das recompensas r_t recebidas a cada ação tomada. Ações tomadas mais tarde podem ter peso menor, e isso é compensado com um fator de desconto temporal, γ.

$$V_\pi(s) = E\left(\sum_{t=0}^{\infty} \gamma^t r_t\right) \quad \text{(Eq 8)}$$

Para cada estado que o sistema assume, o valor máximo para a recompensa acumulada recebida após seguir-se a política ótima π^* é definido como:

$$V^*(s) = \max_\pi E\left(V_\pi^*(s)\right) \quad \text{(Eq 9)}$$

Seja $R(s,a)$ a recompensa percebida ao se tomar a ação *a* no estado *s*, $T(s,a,s')$ a distribuição de probabilidade para, ao se tomar a ação *a* no estado *s*, o ambiente migrar para o estado *s'*, $V^*(s')$ o valor esperado para a soma das recompensas das ações tomadas seguindo a política ótima π^*, a partir do estado *s'*, γ um fator de desconto no tempo, e *S* o conjunto de estados que o ambiente pode assumir. Quando as recompensas recebidas em instantes diferentes de tempo apresentam a mesma importância para o sistema como um todo, o valor de γ fica fixado como 1.

Para encontrar o valor máximo $V^*(s)$ de cada estado *s*, resolve-se iterativamente o sistema dado por:

$$V^*(s) = \max_a \left(R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')V^*(s')\right), \forall s \in S \quad \text{(Eq 10)}$$

Para exemplificar, consideremos o seguinte caso hipotético: um sistema pode assumir três estados distintos, s_1, s_2 e s_3. A qualquer instante, duas ações podem ser tomadas: a_1 e a_2. De acordo com o estado que o sistema se encontre, tomar uma ação pode levar a outro estado, com diferente distribuição de probabilidade. Por exemplo, podemos ter a seguinte distribuição de probabilidade:

Estado inicial s	Ação a	Novo estado s'	Probabilidade $T(s,a,s')$
s_1	a_1	s_1	0.1
s_1	a_1	s_2	0.7
s_1	a_1	s_3	0.2
s_1	a_2	s_1	0.2
s_1	a_2	s_2	0.5
s_1	a_2	s_3	0.3
s_2	a_1	s_1	0.3
s_2	a_1	s_2	0.3
s_2	a_1	s_3	0.4
s_2	a_2	s_1	0.1
s_2	a_2	s_2	0.3
s_2	a_2	s_3	0.6
s_3	a_1	s_1	0.8
s_3	a_1	s_2	0.1
s_3	a_1	s_3	0.1
s_3	a_2	s_1	0.1
s_3	a_2	s_2	0.8
s_3	a_2	s_3	0.1

Tabela 2 – Tabela de probabilidade de transição de estado

Da mesma forma, espera-se receber uma recompensa ao tomar-se uma ação em um determinado estado. A tabela 3 traz um exemplo.

Estado	Ação	Recompensa
s_1	a_1	1
s_1	a_2	0.5
s_2	a_1	0
s_2	a_2	1
s_3	a_1	1
s_3	a_2	0.5

Tabela 3 – Exemplo de recompensas por ações tomadas em diferentes estados

A cada estado é também associado um valor que espera-se obter ao tomar as ações determinadas pela política à partir dele. Podemos considerar a seguinte tabela:

Estado	Valor esperado V(s)
s_1	10
s_2	15
s_3	18

Tabela 4 – Valores esperados ao seguir uma política à partir de diferentes estados.

Resolver, então, para $V^*(s)$, requer que se resolva iterativamente um sistema de equações para cada estado, obtendo-se assim valores atualizados a cada iteração, e utilizando-os na estimativa seguinte. Vejamos como seria resolvido para s_1, na primeira iteração.

Da equação 10, temos que:

$$V^*(s) = \max_a \left(R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^*(s') \right), \forall s \in S$$

Então,

$$V^*(s_1)$$
$$= \max_a \begin{pmatrix} R(s_1,a_1) + \gamma\big(T(s_1,a_1,s_1)V^*(s_1) + T(s_1,a_1,s_2)V^*(s_2) + T(s_1,a_1,s_3)V^*(s_3)\big), \\ R(s_1,a_2) + \gamma\big(T(s_1,a_2,s_1)V^*(s_1) + T(s_1,a_2,s_2)V^*(s_2) + T(s_1,a_2,s_3)V^*(s_3)\big) \end{pmatrix}$$

Utilizando os valores das tabelas 2, 3 e 4, temos:

$$V^*(s_1) = \max_a \begin{pmatrix} 1.0 + \gamma(0.1 * 10 + 0.7 * 15 + 0.2 * 18), \\ 0.5 + \gamma(0.2 * 10 + 0.5 * 15 + 0.3 * 18) \end{pmatrix}$$

Considerando $\gamma = 1$, para os valores de ações tomadas no futuro tendo o mesmo peso das ações tomadas no presente, temos:

$$V^*(s_1) = \max_a \begin{pmatrix} 1.0 + (1 + 10.5 + 3.6), \\ 0.5 + (2 + 7.5 + 5.4) \end{pmatrix}$$
$$V^*(s_1) = \max_a \begin{pmatrix} 16.1, \\ 15.4 \end{pmatrix}$$

O valor de $V^*(s_1)$ passa a ser, então, 16.1, que deve ser alcançado tomando-se a ação a_1. Deve-se repetir o mesmo procedimento para $V^*(s_2)$ e $V^*(s_3)$. Após obter-se os valores esperados, repete-se a resolução do sistema, até que haja convergência dos valores.

Sabe-se que a política ótima $\pi^*(s)$, definida como aquela que maximiza o retorno esperado das recompensas futuras, ao ser seguida à partir do estado s, respeita a equação:

$$\pi^*(s) = arg \max_a \left(R(s,a) + \gamma \sum_{s' \in S} T(s,a,s')V^*(s') \right) \qquad \text{(Eq 11)}$$

No modelo geral de aprendizado por reforço, as ações determinam não apenas a recompensa imediatamente recebida pelo agente, mas também qual será o próximo estado do ambiente (se não deterministicamente, ao menos probabilisticamente). Assim, para um agente escolher qual ação tomar, não é apenas o valor da recompensa que deve ser considerado, mas também o próximo

estado a ser atingido, e o valor esperado da recompensa a longo prazo que será obtida tomando-se esse caminho.

Costuma-se chamar de *experimento* uma sequência de ações tomadas por um agente, desde um estado inicial (aleatório ou não), seguindo uma política, até um estado final predeterminado. Ao final do experimento, obtém-se o valor da recompensa total como a somatória das recompensas obtidas a cada ação realizada.

3.2. Métodos de aprendizado

Existem problemas nos quais a quantidade de estados que podem ser assumidos pelo ambiente e também a quantidade de ações que o agente pode tomar são computacionalmente tratáveis. Nesses casos, é possível realizar o treinamento armazenando-se todo o conjunto de estados, ações e tabelas de transições em memória, definir as recompensas para as ações e iniciar o treinamento, colocando o agente para interagir com o ambiente.

Os métodos que trabalham dessa forma diferem-se na maneira que realizam o aprendizado e a atualização da política. No método Monte Carlo, por exemplo, as ações são tomadas, as tabelas de mapeamento são atualizadas, as recompensas acumuladas. Após a realização de todo um experimento (início em um estado qualquer e término do sistema, no estado final), o valor da recompensa acumulada obtida é utilizado para reforçar as ações tomadas nesse experimento, positiva ou negativamente. Já no método Q-Learning, bastante semelhante, a cada ação tomada pelo agente o valor da recompensa obtida associado ao valor esperado para o estado alcançado são utilizados para reforçar a ação recém tomada. Não é possível apontar qual desses métodos é melhor; em um problema o resultado alcançado por um método é melhor, em outro problema, a situação se inverte.

3.2.1. Q-Learning

Em um problema de aprendizado por reforço, a política ótima π^* satisfaz:

$$\pi^*(s) = arg\max_a \left(R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^*(s') \right) \quad \text{(Eq 12)}$$

O método Q-Learning é um algoritmo de aprendizado no qual uma tabela Q é preenchida, para os pares (s,a), com os valores esperados a serem obtidos ao final de um experimento ao se tomar, num estado *s*, a ação *a* e prosseguir escolhendo as ações determinadas pela política ótima.

$V^*(s)$ é o valor considerado ótimo para *s*, admitindo-se que a melhor ação é tomada inicialmente, e então $V^*(s) = \max_a Q^*(s,a)$.

Assim, os valores ótimos da tabela Q podem ser determinados recursivamente da seguinte maneira:

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \max_{a'} Q^*(s',a') \quad \text{(Eq 13)}$$

Buscam-se os valores ótimos para a tabela Q, de estados e ações, iterativamente através de execuções de experimentos. A tabela de transições, $T(s,a,s')$, pode não estar disponível. E isso é o que ocorre neste trabalho. Entretanto, esta tabela geralmente é esparsa, e a execução de diversos experimentos no ambiente pode capturar satisfatoriamente as transições. A atualização dos valores de $Q(s,a)$ é realizada, então, seguindo-se a seguinte equação:

$$Q(s,a) := Q(s,a) + \alpha \left(R(s,a) + \gamma \max_a Q(s',a') - Q(s,a) \right) \quad \text{(Eq 14)}$$

onde $\alpha \in (0,1)$ é a taxa de aprendizagem – o quanto a nova experiência (tupla <*s,a,r,s'*>) influencia o valor já aprendido até o momento. Idealmente, no início do aprendizado, essa taxa deve ser maior e, após alguns experimentos, o peso da experiência pode ficar maior que o do aprendizado.

3.2.2.
Métodos de aproximação por função com atualização por gradiente

Quando o espaço de estados ou de ações é muito grande, fica complicado manter uma tabela com todos os valores esperados para os pares estado x ação. Para resolver esse problema, a tabela é aproximada por uma função que simula seu comportamento. Assim, para obter o valor ótimo esperado para um determinado estado ($V^*(s)$), ao invés de recorrer à tabela realizamos uma operação em uma função, que recebe como entrada um conjunto de características do estado e retorna o valor ótimo para esse estado.

Essa função de aproximação é construída, como foi dito, sobre um conjunto de características do estado. Dessa forma, estados "parecidos" são agrupados pela própria modelagem da função, e o tratamento do grande conjunto de estados passa a ser computacionalmente viável.

Determinar quais são as características do estado a serem consideradas na função faz parte da modelagem do problema, e não é fácil acertar na sua escolha. Essas características são combinadas (linearmente ou não) na função, e o aprendizado se dá na determinação e atualização dos coeficientes que ponderam, na função, as características do estado.

No treinamento desse método, os coeficientes reais da função são agrupados em um vetor coluna $\overrightarrow{\theta_t} = \big(\theta_t(1), \theta_t(2), \dots, \theta_t(n)\big)^T$ (T denota "transposta"), e $V_t(s)$ deve ser uma função suave diferenciável de $\overrightarrow{\theta_t}$ para todo $s \in S$. Assume-se que a cada passo t do algoritmo, um novo estado $s_t \mapsto V^\pi(s_t)$ é observado. Mesmo que o valor $V^\pi(s_t)$ obtido seja exato, o problema ainda é complicado porque a função de aproximação possui recursos e resolução limitados. Não há conjunto $\vec{\theta}$ que contemple todos os estados, nem mesmo os exemplos varridos no treinamento, corretamente. Ainda mais porque a função deve generalizar os estados que não apareceram no conjunto de exemplos.

A referência [SUT] apresenta como minimizar o erro quadrático médio da função de aproximação minimizando o erro no conjunto de exemplos. A cada novo exemplo obtido no algoritmo ajusta-se o vetor de parâmetros com uma pequena variação na direção que melhor reduz o erro obtido no exemplo (o gradiente da função de aproximação, com as derivadas parciais nos coeficientes do vetor $\overrightarrow{\theta_t}$. Esse vetor de derivadas é, então, o gradiente de f em relação a $\overrightarrow{\theta_t}$. O método é chamado de "atualização por gradiente" porque a variação de $\overrightarrow{\theta_t}$ é proporcional ao gradiente negativo do erro quadrático do exemplo verificado, que é a direção na qual o erro diminui mais rapidamente.

Na mesma referência [SUT], há exemplos de aplicações nas quais a técnica foi aplicada com sucesso.