4
Uma abordagem por aprendizado por reforço

Nesse capítulo apresentamos uma abordagem do problema de manutenção da atualidade de uma base de dados através de um algoritmo de aprendizado por reforço. Conforme foi explicado no capítulo 3, uma dificuldade desse método é capturar corretamente as características do problema que melhor representam os estados que o ambiente pode assumir, de maneira a construir uma função que seja ao mesmo tempo precisa o suficiente para representar bem os estados visualizados no conjunto de exemplos, mas também capaz de generalizar os resultados para os estados não contemplados no conjunto de exemplos.

Serão apresentados conceitos e características de páginas web que foram consideradas ou desconsideradas para a modelagem do problema e da solução.

4.1.
Limitações de politeness

O termo *politeness*, que pode ser traduzido como *cortesia* ou *boas maneiras*, no problema abordado se relaciona a quão frequente pode ser realizada a atualização de uma página na base de dados. Se imaginarmos que uma página sofra muitas edições em curtos intervalos de tempo, pode ser necessário que a cópia precise ser atualizada com muita frequencia, a fim de manter seu *freshness* instantâneo igual a 1. Entretanto, para saber se a cópia precisa ou não ser atualizada, é necessário verificar se a versão original foi editada. Considerando que várias páginas estão armazenadas em um mesmo servidor, pode ser que o algoritmo que busca manter as cópias atualizadas precise ficar constantemente consultando o servidor, o que pode ser mal interpretado, inclusive como tentativa de ataque. Nessa situação, o limite de *politeness* representa uma taxa máxima de consultas que o algoritmo pode fazer a um determinado servidor, sem que isso seja considerado um problema.

Em um ambiente real, uma mesma página pode estar armazenada em um entre vários servidores, então a frequencia de consulta pode ser distribuída entre estes. Acontece ainda de diferentes páginas estarem armazenadas em diferentes servidores, e cada um destes pode admitir diferentes taxas de consulta, apresentando diferentes limitações de *politeness*. Nesse estudo, no entanto, considera-se que todas as páginas estão armazenadas em um mesmo e único servidor, com uma limitação de *politeness* bem estabelecida, e controlável no algoritmo.

Em um estudo relacionado, [ECK] apresenta um conjunto de algoritmos que usam programação dinâmica para definir os melhores instantes de consulta a um repositório, com o mesmo objetivo de manter um alto valor de freshness médio, respeitando um limite máximo de visitações por unidade de tempo. Nesse mesmo estudo, também é considerada a possibilidade de restringir as consultas a serem espaçadas por um determinado intervalo de tempo. Esse seria mais um limite a ser obedecido pelo algoritmo.

4.2. Dinâmica do sistema

O problema de manter o conjunto de cópias de páginas atualizado é um problema contínuo no tempo, já que a definição de *freshness* que foi apresentada é contínua. Entretanto, a abordagem que este trabalho realiza se vale de uma discretização do problema.

No problema contínuo, seria necessário determinar o instante exato no tempo no qual cada página deve ser novamente copiada da versão original, com o objetivo de manter a cópia atualizada. No sistema discreto aqui proposto, os instantes de tempo nos quais as cópias podem ser realizadas são bem definidos, e espaçados entre si de um valor de $\Delta\tau$ segundos. De acordo com a capacidade de processamento do sistema que mantém as cópias, bem como da banda disponível para tráfego das páginas na rede, existe uma limitação da quantidade de páginas que pode ser atualizada por unidade de tempo. Nessa abordagem do problema, a cada instante de atualização, uma quantidade definida (N) de páginas será

atualizada, quantidade esta determinada pela relação N / $\Delta\tau$, que representa a limitação física do sistema.

Nosso problema passa a ser, então, determinar quais as N páginas que, naquele instante de tempo, se forem atualizadas, promoverão maior ganho no freshness médio do conjunto de cópias.

Assim, nos instantes $t_1 = \Delta\tau$, $t_2 = 2.\Delta\tau$, $t_3 = 3.\Delta\tau$, ..., $t_k = k.\Delta\tau$, o algoritmo tem a oportunidade de revisitar uma quantidade N de páginas. Como hipótese simplificadora, as páginas são visitadas instantaneamente, e na oportunidade do acesso à página é visitada também a página que contém os instantes em que a página original sofreu as últimas edições. Assim como a visita à página da Wikipédia, o acesso às informações sobre as edições também é considerado instantâneo, não consumidor de recursos. Para o corpo de testes da Wikipédia, parte da hipótese simplificadora é, de fato, viável – obter os instantes das últimas edições no ato da revisitação, pois cada artigo da Wikipédia oferece uma página com a lista das últimas edições.

Com o conhecimento dos instantes em que página sofreu as últimas edições, é possível estimar a taxa com que ela recebe edições, e essa informação será útil no cálculo do próximo instante em que ela deve ser revisitada

4.3. Estados e ações

Quando modelamos um problema para ser tratado com aprendizado por reforço utilizando métodos de aproximação por gradiente, é fundamental que as características utilizadas para representar um estado do ambiente sejam boas o bastante para generalizar os estados que não sejam contemplados pelos exemplos visitados no treinamento.

Nesse estudo, esperamos validar que a hipótese de que a utilização das informações do tempo decorrido desde a última visita a cada uma das páginas,

associada ao conhecimento da frequência com que essas sofrem edições servem como boas características de modelagem de estado.

Mesmo considerando que as páginas sigam um processo de Poisson, e tendo o tempo entre edições distribuído exponencialmente em função da taxa de edição da página, a simples mensuração de poucos valores de intervalo entre edições não permite afirmar, com certeza, o valor da taxa de edição. Esse valor é, portanto, *estimado* em função do intervalo entre as últimas edições sofridas pela página. É fato que manter as informações de tempo decorrido desde a última visita e frequência (estimada) de edição para cada página e utilizar essas informações para determinar que páginas atualizar a cada instante exige grande capacidade computacional. Além disso, a quantidade de variáveis utilizadas como características para modelar o estado, nesse caso, seria muito grande, e o aprendizado poderia se tornar impraticável.

Este estudo procura, então, uma forma de generalizar o modelo para torná-lo computacionalmente mais prático. Como no conjunto de páginas certamente teremos várias que possuem valores próximos entre si de taxa de edição ou de tempo decorrido desde a última visita, a modelagem do *estado* do sistema passa ser feita da seguinte maneira: ao invés de utilizar as informações relativas a cada uma das páginas, podemos agrupar as páginas em função dos valores que apresentem de taxa de edição e tempo decorrido, e considerar como estado do sistema a quantidade de páginas em cada grupo.

Figura 2 – Exemplo da modelagem de um conjunto de páginas em um estado do sistema

A Figura 2 apresenta um exemplo de como seria utilizado esse modelo para um conjunto de páginas que apresentasse valores parecidos de taxa de edição estimada (λ) e intervalo de tempo desde a última visita (Δt).

No exemplo, existem quatro grupos, um para cada par (λ,Δt). O grupo onde (λ,Δt) = (λ_1,Δt_1) existem 3 páginas, pois 3 páginas possuem essas características. O grupo com (λ_1,Δt_2) contém 2 páginas, o (λ_2,Δt_1) tem apenas 1 página e nenhuma página possui simultaneamente λ_2 e Δt_2 como características.

A quantidade de grupos nos quais as páginas serão reunidas dependerá da capacidade computacional disponível e do grau de refinamento desejado da solução. Foram estudadas duas maneiras de se estabelecer os agrupamentos em função do intervalo de tempo decorrido. Nos primeiros treinamentos realizados, para determinar os intervalos de tempo que definem os grupos foram consideradas a quantidade de páginas utilizadas no treinamento e a quantidade de páginas atualizáveis a cada intervalo de revisitação. Com o treinamento realizado sobre 10.000 páginas, revisitando-se 250 a cada intervalo $\Delta\tau$, tem-se uma média de $40\Delta\tau$ entre duas revisitações de uma mesma página. Então, se desejamos agrupar as páginas em quatro valores de intervalo de tempo desde a última revisitação (Δt), estes podem ser:

- $\Delta t_1 \leq 15.\Delta\tau$
- $15.\Delta\tau < \Delta t_2 \leq 30.\Delta\tau$
- $30.\Delta\tau < \Delta t_3 \leq 45.\Delta\tau$
- $\Delta t_2 > 45.\Delta\tau$

Outra opção de organização das páginas em grupos foi feita, para comparação de resultados. Como a determinação dos intervalos em função de $\Delta\tau$ pode ser considerada subjetiva, a definição dos limites dos grupos foi feita em função dos intervalos de tempo decorrido desde a última visita para cada uma das páginas. Antes de considerar a taxa de edição das páginas, considera-se que, havendo *m* possíveis agrupamentos baseados em tempo decorrido desde a última visita, um algoritmo deveria distribuir *1/m* das páginas em cada agrupamento, a

cada oportunidade de reorganização dos grupos (no quadro da Figura 2, seria como colocar *1/m* das páginas em cada coluna).

Após a definição dos grupos em função do tempo (as *colunas*, na Figura 2), são definidos os agrupamentos em função da taxa de edição. Os valores de taxa de edição que definem os grupos podem ser obtidos através de um algoritmo de agrupamento. Nos treinamentos, um algoritmo *k-Means* [WOL] é executado no início do processo, de modo a agrupar as páginas que tenham taxas de edição semelhantes. O "*k*", do algoritmo *k-Means*, representa a quantidade de grupos que se deseja criar. Se, no decorrer do treinamento, for percebido que uma página teve sua taxa de edição variada, de forma que sua nova taxa se aproxime mais do valor médio das taxas de páginas de outro grupo, essa página é então reposicionada no novo grupo, e o processo continua.

O estado do sistema, capturado pela quantidade de páginas pertencentes a cada grupo, varia no tempo: como consequência da ação do algoritmo (revisitação da página), o tempo decorrido desde a última visita vai a zero, o que a faz mudar de grupo na mesma "linha" da matriz da Figura 2. Ao revisitar a página, é possível verificar se sua taxa de edição estimada continua a mesma. Caso ela tenha sido alterada, o que em se tratando de um processo de Poisson caracterizaria um comportamento não-homogêneo, sua coluna na matriz da Figura 2 seria alterada. Nesse caso, a mudança de estado do sistema aconteceu independente da ação do algoritmo, apenas foi percebida por ele.

	Δt_1	Δt_2
λ_1	2	2
λ_2	2	0

Figura 3 – Exemplo de estado do sistema

Como exemplo, voltemos à Figura 2: se, ao revisitarmos a página representada pela esfera mais à esquerda na primeira linha, for percebido que seu valor de taxa de edição mudou de λ_1 para λ_2, então o estado agora estaria como na

Figura 3, com duas páginas apresentando valores (λ_1,Δt_1) e outras duas com valores (λ_2,Δt_1).

Ainda nesse exemplo, se $\Delta t_2 > \Delta t_1$, e muito tempo se passasse sem que nenhuma dessas páginas fosse revisitada, o sistema mudaria para a situação da Figura 4:

	Δt_1	Δt_2
λ_1	0	4
λ_2	0	2

Figura 4 – Exemplo de estado do sistema

Nesse caso, não seria possível perceber mudança de taxa de edição (λ), mas a informação de tempo decorrido desde a última visita seria atualizada para um valor maior, e todas as páginas teriam essa característica como Δt_2.

Com esta modelagem, a dinâmica do sistema, que elege N páginas para atualizar a cada intervalo de tempo $\Delta\tau$, pode ser encarada de uma nova maneira, que é eleger um *grupo* para ter N de suas páginas atualizadas.

Definem-se, assim, as *ações* que o algoritmo pode tomar: para um sistema com *k* grupos, existem *k* ações, e cada uma delas refere-se a atualizar, de um dos grupos do sistema, uma quantidade equivalente a até N páginas. Se o tamanho dos grupos for significativamente maior que N, esta política não precisa ser revista. Caso os grupos assumam frequentemente tamanhos menores que N, a capacidade de atualização de páginas estaria sendo desperdiçada. Quando o algoritmo escolhe um grupo com menos de N páginas, digamos n < N, pode-se escolher, aleatoriamente ou através de nova utilização do algoritmo, um segundo grupo para ter (N-n) páginas visitadas. Essa função de aproveitamento de capacidade ociosa é implementada recursivamente, aumentando a eficiência, mas também a complexidade do algoritmo.

4.4. Função de Aprendizado

Considerando-se os conceitos apresentados neste capítulo (*politeness*, modelagem do estado em grupos de páginas e das ações na atualização de N páginas de um determinado grupo), propõe-se uma implementação que utilize um mecanismo de aprendizagem por reforço, valendo-se de um método de diferença temporal atualizado por gradiente.

O "estado" (s) do ambiente é definido pelo valor das características (quantidade de páginas em cada grupo) no momento anterior à definição do grupo que terá uma parcela de suas páginas atualizada. A "ação" é definida como a escolha do grupo, e a consequente atualização de até N páginas deste e, após um intervalo de tempo de duração $\Delta\tau$ (tempo necessário para atualizar as N páginas, dada a limitação física do sistema), atinge-se um novo estado (s') do sistema. A "recompensa" é definida como o valor do freshness médio esperado para o sistema, pelas próximas T unidades de tempo. Na Figura 5 vemos uma representação do eixo do tempo, com as marcações dos instantes em que o sistema se encontra nos estados s e s', o tempo $\Delta\tau$ decorrido entre eles e o intervalo T para o qual se deseja manter o melhor freshness médio possível.

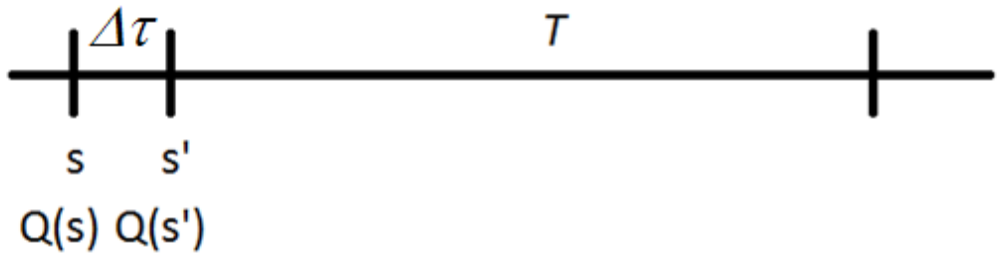

Figura 5 – Definição de estados, função de aprendizado e recompensa

Para possibilitar essa modelagem, a função de aprendizado, *Q(s)*, é criada para representar o valor do freshness médio esperado para o sistema pelos próximos $T+\Delta\tau$ instantes de tempo. O objetivo é que através do aprendizado a função Q convirja para o valor ótimo esperado pelo sistema que age de acordo com a política ótima, definido na equação 10, $V^*(s)$.

Como é difícil mensurar o valor do freshness do sistema a cada instante de tempo, considera-se que, durante o intervalo $\Delta\tau$,o valor do freshness do sistema como um todo permanecerá com o valor de freshness igual ao obtido imediatamente após a tomada da ação escolhida (a atualização das páginas do grupo eleito). Assim, a cada passo do algoritmo de aprendizado, a função Q(s), utilizada para determinar qual a melhor ação a ser tomada, tem seu valor atualizado de acordo com uma ponderação entre o valor já aprendido até o momento e um valor médio construído entre o valor do freshness instantâneo, obtido analisando-se o estado instantâneo das páginas, que considera-se que será mantido pelos próximos $\Delta\tau$ instantes de tempo e o valor previsto para o próximo estado, ***s'***, pelas próximas *T* unidades de tempo.

$$novoQ(s) := (1-\alpha)Q(s) + \alpha\left(\frac{\Delta\tau.Fresh + (T-\Delta\tau)Q(s')}{\mathrm{T}}\right) \qquad \text{(Eq 15)}$$

O fator α representa uma taxa de aprendizado: o peso da nova experiência no processo de aprendizado é mais importante que o valor aprendido até o momento.

4.4.1. Modelagem em função de regressão logística e seus resultados

A primeira modelagem vislumbrada para a função de aprendizado foi uma combinação linear das características utilizadas para representar o estado (quantidade de páginas em cada grupo). Sendo G o conjunto de grupos nos quais as páginas foram organizadas, g_i identificando o i-ésimo grupo, e $n(g_i)$ representando a quantidade de páginas no grupo, o valor de Q(s) foi estimado por

$$Q(s) = \sum_{i=1}^{|G|} \theta_i n(g_i) \qquad \text{(Eq 16)}$$

onde θ representa o coeficiente que deve ser aprendido pelo algoritmo.

Como a função Q(s) representa o freshness médio esperado do sistema, após um determinado intervalo de tempo, seu contradomínio é definido em [0,1]. Ao limitar os valores assumidos pela função no intervalo [0,1], diversas ações sobre

estados do sistema acabavam ultrapassando esses limites, e acabavam por assumir resultados iguais (em 0 ou 1). Em face desses problemas, a modelagem em função linear foi descartada.

Por apresentar contradomínio equivalente ao definido para a função Q(s), [0,1], a função de regressão logística foi escolhida para modelá-la.

Essa função é caracterizada por:

$$f(x) = \frac{1}{1 + e^{-x}} \qquad \text{(Eq 17)}$$

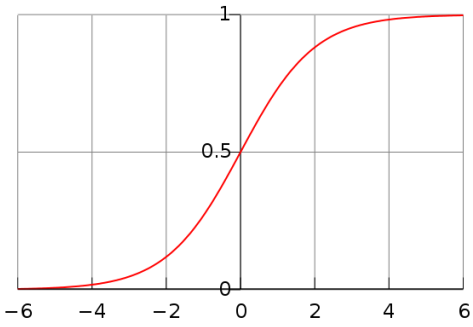

Figura 6 – Gráfico da função de regressão logística

Quando a função de regressão logística é utilizada em problemas de aprendizado por reforço, geralmente é utilizada na forma

$$y = \frac{1}{1 + e^{-f(X)}} \qquad \text{(Eq 18)}$$

onde *f(X)* é uma função analítica em *X*. Sua derivada contínua, utilizada para ajustes na direção do gradiente, é dada por

$$y' = y(1 - y)\frac{df}{dX} \qquad \text{(Eq 19)}$$

Nos testes com essa modelagem, a função analítica *f(X)* utilizada foi a combinação linear das características do estado (quantidade de páginas em cada grupo). As derivadas parciais são então dadas por

$$\frac{\delta y}{\delta \theta_i} = y(1 - y)n(g_i) \qquad \text{(Eq 20)}$$

$n(g_i)$, como anteriormente, representa a quantidade de páginas no i-ésimo grupo de páginas.

Ao contrário da modelagem com função linear, o algoritmo não teve problemas com valores fora do contradomínio [0,1], pela própria definição da função, e o aprendizado ocorreu com sucesso.

4.5. Processo de Aprendizado

Conforme foi tratado no capítulo 3, algoritmos de aprendizado por reforço utilizam as recompensas percebidas pelo agente ao interagir com o ambiente e, de acordo com o valor das recompensas, considera que a ação tomada naquele estado foi boa ou ruim, devendo ou não ser repetida no futuro. Para armazenar todos os possíveis estados pelos quais o sistema poderia passar, seria necessário armazenar uma quantidade muito grande de valores em memória, o que tornaria o processo de aprendizado lento e desinteressante. Utilizou-se, no lugar disso, uma modelagem dos estados do sistema através de uma aproximação por função, que generaliza os estados admissíveis pelo sistema com a leitura de *features* (características) desses estados. Nesse tipo de modelagem, estados que possuam características parecidas devem reagir de forma semelhante quando submetidos às mesmas ações.

O conjunto de cópias de páginas web que deve ser mantido atualizado passa a ser modelado, então, como um ambiente no qual as páginas estão organizadas em grupos, em função de suas taxas de edição estimadas e tempo desde última visita (conforme Figura 2), e os estados do sistema são representados por um conjunto de características desses grupos, quais sejam: a quantidade de páginas pertencente a cada grupo. O conjunto de ações sobre o sistema está definido como atualizar até N páginas de um dos grupos (cada ação se refere a um grupo).

O aprendizado se dará, então, através de uma função que utilizará como características do estado a quantidade de páginas em cada grupo, e terá

coeficientes reais que devem ser aprendidos através de tentativa e erro, recebendo recompensas positivas ou negativas, de acordo com o resultado obtido.

Q(s) será então representado por uma função Q(f_1, f_2, ...), $\mathbb{N}^n \mapsto [0,1]$, onde os *fi* representam os valores de cada uma das características do estado. O valor assumido pela função deve representar o valor esperado para o freshness médio de todo o conjunto de páginas, pelas próximas *T* unidades de tempo.

A sequência de aprendizado será a seguinte:

- O ambiente encontra-se em um determinado estado **s**
- De acordo com a política aprendida até o momento (representada pela função de aprendizagem), o algoritmo determina qual a melhor ação a ser tomada – a ação **A** que maximiza o valor de **Q(s)** no instante imediatamente posterior à tomada da ação **A**.
    - Com uma determinada taxa ε, o algoritmo escolhe uma ação aleatória – não necessariamente a que maximiza o valor de Q(s). Isso tem por objetivo proporcionar *exploração* ao algoritmo, conforme comentado na seção 3.1.1.
- Após a tomada da ação, uma leitura do ambiente é feita, e o algoritmo recebe uma recompensa. Essa recompensa é definida em função do novo freshness alcançado pelo sistema e quanto esse freshness, mantido durante o intervalo $\Delta\tau$, impacta no freshness médio considerando o intervalo *T*.
- De acordo com o valor da recompensa, os valores dos coeficientes são atualizados, o que gera uma nova política.

$$novoQ(s) := (1-\alpha)Q(s) + \alpha\left(\frac{\Delta\tau.Fresh + (T-\Delta\tau)Q(s')}{\mathrm{T}}\right) \qquad \text{(Eq 21)}$$

- O sistema avança $\Delta\tau$ no tempo
- O ambiente assume um novo estado **s'** e o ciclo se repete

Os estados **s** e **s'** são definidos nos instantes imediatamente anteriores à escolha da melhor ação, e depois do transcurso do intervalo $\Delta\tau$, durante o qual a capacidade de atualização de páginas do sistema está comprometida e nenhuma outra ação pode ser realizada até a chegada ao próximo estado.