UB Social

CS50AI with Python

6.Heredity

Mateus Schwede

Problemática

- Escreva IA para avaliar probabilidade de pessoa ter característica genética específica.

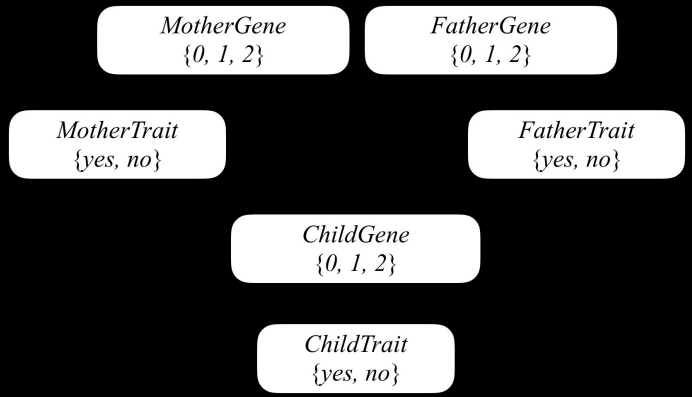
```
$ python heredity.py data/family0.csv
Harry:
  Gene:
    2: 0.0092
    1: 0.4557
    0: 0.5351
  Trait:
    True: 0.2665
    False: 0.7335
James:
  Gene:
    2: 0.1976
    1: 0.5106
    0: 0.2918
  Trait:
    True: 1.0000
    False: 0.0000
Lily:
  Gene:
    2: 0.0036
    1: 0.0136
    0: 0.9827
  Trait:
    True: 0.0000
    False: 1.0000
```

Problemática

- Versões mutadas do gene GJB2 são uma das principais causas de deficiência auditiva em recém-nascidos. Cada pessoa carrega 2 versões do gene, então cada pessoa tem potencial de possuir 0, 1 ou 2 cópias da versão de deficiência auditiva GJB2. A menos que uma pessoa passe por testes genéticos, no entanto, não é tão fácil saber quantas cópias de GJB2 mutado a pessoa tem. Este é "estado oculto": informação que tem efeito que pode-se observar (deficiência auditiva), mas que não necessariamente sabe-se diretamente. Afinal, algumas pessoas podem ter 1 ou 2 cópias de GJB2 mutado, mas não apresentam deficiência auditiva, enquanto outras podem não ter cópias de GJB2 mutado, mas ainda apresentam deficiência auditiva;
- Cada criança herda cópia do gene GJB2 de cada um dos pais. Se um dos pais tiver 2 cópias do gene mutado, ele passará o gene mutado para o filho; se um dos pais não tiver cópias do gene mutado, ele não passará o gene mutado para o filho; e se um dos pais tiver uma cópia do gene mutado, o gene será passado para filho com probabilidade de 0,5. Após gene é passado, porém, ele tem alguma probabilidade de sofrer mutação adicional: mudar de versão do gene que causa deficiência auditiva para versão que não causa, ou vice-versa;
- Pode-se entar modelar todas relações formando Rede Bayesiana de todas variáveis relevantes, como imagem, que considera família de 2 pais e 1 único filho.

Problemática

- Cada pessoa na família tem 1 variável aleatória Gene representando quantas cópias de gene específico (exemplo, versão de deficiência auditiva do GJB2) pessoa tem: valor 0, 1 ou 2. Cada pessoa na família também tem variável aleatória Trait, que é sim ou não dependendo se pessoa expressa traço (exemplo, deficiência auditiva) com base naquele gene. Há seta da variável Gene de cada pessoa para sua variável Trait para codificar ideia de que genes de pessoa afetam probabilidade de que ela tenha traço específico. Enquanto isso, há também seta da variável aleatória Gene da mãe e pai para variável aleatória Gene do filho: genes da criança são dependentes dos genes de seus pais;



Problemática

- Tarefa no projeto é usar modelo para fazer inferências sobre população. Dadas informações sobre pessoas, quem são seus pais e se elas têm característica observável específica (exemplo, perda auditiva) causada por determinado gene, IA inferirá distribuição de probabilidade para genes de cada pessoa, além da distribuição de probabilidade para se qualquer pessoa exibirá característica em questão.

Instruções

- Baixe código de https://cdn.cs50.net/ai/2023/x/projects/2/heredity.zip e descompacte-o.

Funcionamento

- Em data/family0.csv, 1ª linha define colunas: nome, mãe, pai e traço. Próxima linha indica que Harry tem Lily como mãe, James como pai, e célula vazia para traço significa que não sabemos se Harry tem traço ou não. James não tem pais listados no conjunto de dados (conforme indicado pelas células vazias para mãe e pai) e exibe traço (conforme indicado pelo 1 na célula do traço). Lily também não tem pais listados nos dados, mas não exibe traço (conforme indicado pelo 0 na célula do traço);
- heredity.py possui PROBS, dicionário que contém série de constantes que representam probabilidades de vários eventos diferentes. Todos eventos têm a ver com quantas cópias de gene específico a pessoa tem (doravante denominado simplesmente "o gene") e se pessoa exibe característica específica (doravante denominada "a característica") com base no gene. Os dados são vagamente baseados nas probabilidades para versão de deficiência auditiva do gene GJB2 e característica de deficiência auditiva, mas ao alterar esses valores, pode-se usar IA para tirar inferências sobre outros genes e características também;
- PROBS["gene"] representa distribuição de probabilidade incondicional sobre gene (ou seja, probabilidade se não souber nada sobre pais dessa pessoa). Conforme dados no código de distribuição, na população, há chance de 1% de ter 2 cópias do gene, chance de 3% de ter 1 cópia do gene e chance de 96% de ter 0 cópias do gene;

Funcionamento

- PROBS["trait"] representa probabilidade condicional de que pessoa exiba traço (como deficiência auditiva). Na verdade, são 3 distribuições de probabilidade diferentes: 1 para cada valor possível para gene. PROBS["trait"][2] é distribuição de probabilidade de que pessoa tenha traço, dado que ela tenha 2 versões do gene: neste caso, ela tem 65% de chance de exibir traço e 35% de chance de não exibir traço. Se pessoa tem 0 cópias do gene, tem 1% de chance de exibir traço e 99% de chance de não exibir traço;
- PROBS["mutation"] é probabilidade de gene sofrer mutação, desse gene em questão, para não ser aquele gene, e vice-versa. Se mãe tem 2 versões do gene, por exemplo, e, portanto, passa 1 para filho, ha 1% de chance de ele sofrer mutação para não ser mais gene alvo. Se mãe não tem versões do gene e, portanto, não passa para filho, ha 1% de chance de ele sofrer mutação para ser gene alvo. Então, é possível que, mesmo que nenhum dos pais tenha cópias do gene em questão, filho possa ter 1 ou até 2 cópias do gene. Probabilidades a serem calculadas serão baseadas nesses valores em PROBS;
- Função main carrega dados de arquivo para dicionário people. people mapeia nome de cada pessoa para outro dicionário contendo informações sobre elas: nome, mãe (se houver lista no conjunto de dados), pai (se houver lista no conjunto de dados) e se eles são observados como tendo característica em questão (Verdadeiro se ter, Falso se não ter e None se não souber);

Funcionamento

- main define dicionário de probabilidades, com todas probabilidades inicialmente definidas como 0. Para cada pessoa, IA calculará distribuição de probabilidade sobre quantas cópias do gene ela tem, bem como se ela tem característica ou não. probabilities["Harry"]["gene"][1], por exemplo, será probabilidade de Harry ter 1 cópia do gene, e probabilities["Lily"]["trait"][False] será probabilidade de Lily não exibir característica;
- Dicionário de probabilidades é criado via compreensão de dicionário Python, criando par chave/valor para cada pessoa no dicionário de pessoas;
- Por fim, procura-se calcular probabilidades com base em evidência: sabe-se que certas pessoas exibem ou não traço, espera-se determinar tais probabilidades. Probabilidade condicional somando todas probabilidades conjuntas que satisfazem evidência e, em seguida, normaliza-se tais probabilidades para que cada uma delas some 1. Precisa-se implementar 3 funções isso: joint_probability para calcular probabilidade conjunta, update para adicionar probabilidade conjunta recém-calculada à distribuição de probabilidade existente, e normalize para garantir que todas distribuições de probabilidade somem 1 no fim.

Especificações

- Conclua implementações de joint_probability, update e normalize;
- Função joint_probability recebe entrada dicionário de pessoas, junto com dados sobre quem tem quantas cópias de cada um dos genes e quem exibe característica. Retornando probabilidade conjunta de todos eventos ocorrerem;
    - Função aceita 4 valores como entrada: people, one_gene, two_genes e have_trait:
        - people é dicionário de pessoas. onde chaves representam nomes, e valores são dicionários que contêm chaves mãe e pai. Presume-se que mãe e pai estão em branco (nenhuma informação parental no conjunto de dados), ou mãe e pai se referirão a outras pessoas no dicionário de pessoas;
        - one_gene é conjunto de todas pessoas para calcular probabilidade de ter cópia do gene;
        - two_genes é conjunto de todas pessoas para calcular probabilidade de ter 2 cópias do gene;
        - have_trait é um conjunto de todas as pessoas para as quais queremos calcular a probabilidade de que tenham a característica;
        - Para qualquer pessoa que não esteja em one_gene ou two_genes, calcular probabilidade de que ela não tenha cópias do gene; e para qualquer pessoa que não esteja em have_trait, calcular probabilidade de que ela não tenha característica.

Especificações

- Exemplo, se família for composta por Harry, James e Lily, chamar função onde one_gene = {"Harry"}, two_genes = {"James"} e trait = {"Harry", "James"} deve calcular probabilidade de Lily ter 0 cópias do gene, Harry ter 1 cópia do gene, James ter 2 cópias do gene, Harry exibir traço, James exibir traço e Lily não exibir traço;
- Para qualquer pessoa sem pais listados no conjunto de dados, usa-se distribuição de probabilidade PROBS["gene"] para determinar probabilidade de que eles tenham nº específico do gene;
- Para qualquer pessoa com pais no conjunto de dados, cada um passará 1 dos 2 genes para filho aleatoriamente, e há chance PROBS["mutation"] de que sofra mutação (passe de ser gene para não ser gene, ou vice-versa);
- Use distribuição de probabilidade PROBS["trait"] para calcular probabilidade de pessoa ter ou não traço específico.

- Função update adiciona nova probabilidade de distribuição conjunta às distribuições de probabilidade existentes em probabilities;
  - Função aceita 5 valores como entrada: probabilities, one_gene, two_genes, have_trait, e p:

Especificações

        - probabilities é dicionário de pessoas conforme descrito na seção “Entendimento”. Cada pessoa é mapeada para uma distribuição de “genes” e uma distribuição de “trait”;
        - one_gene é conjunto de pessoas com cópia do gene na distribuição conjunta atual;
        - two_genes é conjunto de pessoas com 2 cópias do gene na distribuição conjunta atual;
        - have_trait é conjunto de pessoas com característica na distribuição conjunta atual;
        - p é probabilidade da distribuição conjunta.
    - Para cada pessoa em probabilities, função deve atualizar distribuição probabilities[person]["gene"] e distribuição probabilities[person]["trait"] adicionando p ao valor apropriado em cada distribuição. Todos outros valores devem ser deixados inalterados;
    - Exemplo, se "Harry" estivesse em two_genes e have_trait, então p seria adicionado a probabilities["Harry"]["gene"][2] e probabilities["Harry"]["trait"][True];
    - Função não deve retornar nenhum valor: ela só precisa atualizar dicionário de probabilidades.
- Função normalize atualiza dicionário de probabilidades para que cada distribuição de probabilidade seja normalizada (ou seja, soma 1, com proporções relativas iguais);

Especificações

- Função aceita único valor: probabilities:
  - probabilities é dicionário de people, onde cada pessoa é mapeada para distribuição de “genes” e distribuição de “trait”.
- Para ambas distribuições para cada pessoa em probabilidades, função deve normalizar distribuição para que valores na distribuição somem 1 e valores relativos na distribuição sejam os mesmos;
- Exemplo, se probabilities["Harry"]["trait"][True] fosse 0,1 e probabilities["Harry"]["trait"][False] fosse 0,3, então deve-se atualizar 1º valor para 0,25 e último valor para 0,75: valores agora somam 1, e último valor ainda é 3 vezes maior que 1º valor;
- Função não deve retornar nenhum valor: ela só precisa atualizar dicionário de probabilidades;

- Não pode-se modificar nada mais em heredity.py além das 3 funções que especificação solicita, embora pode-se escrever funções adicionais e/ou importar outros módulos da biblioteca padrão do Python. Também pode-se importar numpy ou pandas, mas não usar nenhum outro módulo Python de terceiros.

Joint Probability

- Para auxiliar como calcular probabilidades conjuntas, tem-se exemplo. Considere seguinte valor para people:

```
{
  'Harry': {'name': 'Harry', 'mother': 'Lily', 'father': 'James', 'trait': None},
  'James': {'name': 'James', 'mother': None, 'father': None, 'trait': True},
  'Lily': {'name': 'Lily', 'mother': None, 'father': None, 'trait': False}
}
```

- Cálculo de joint_probability(people, {"Harry"}, {"James"}, {"James"}). one_gene é {"Harry"}, two_genes é {"James"} e have_trait é {"James"}, representando probabilidade: Lily tenha 0 cópias do gene e não tenha característica, Harry tenha 1 cópia do gene e não tenha característica, e James tenha 2 cópias do gene e tenha característica;

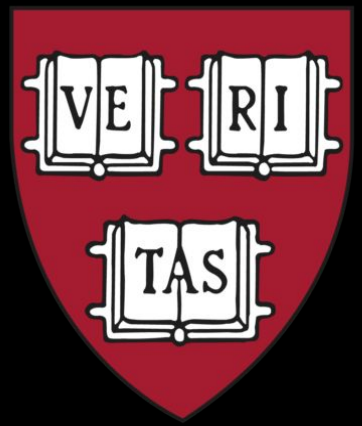
- Lily (ordem das pessoas não importa, desde que multipliquem-se valores corretos, já que multiplicação é comutativa). Lily tem 0 cópias do gene com probabilidade 0,96 (este é PROBS["gene"][0]). Dado que ela tem 0 cópias do gene, ela não tem característica com probabilidade 0,99 (este é PROBS["trait"][0][False]). Assim, probabilidade de que ela tenha 0 cópias do gene e não tenha característica é 0,96 * 0,99 = 0,9504;

Joint Probability

- James tem 2 cópias do gene com probabilidade 0,01 (este é PROBS["gene"][2]). Dado que ele tem 2 cópias do gene, probabilidade de que ele tenha característica é 0,65. Assim, probabilidade de que ele tenha 2 cópias do gene e tenha característica é 0,01 * 0,65 = 0,0065;
- Qual a probabilidade de Harry ter 1 cópia do gene? Há 2 formas disso acontecer. Ou ele obtém gene da mãe e não do pai, ou ele obtém gene do pai e não da mãe. Mãe Lily tem 0 cópias do gene, então Harry obterá gene da mãe com probabilidade 0,01 (isso é PROBS["mutação"]), já que única maneira de obter o gene da mãe é se ele sofrer mutação; inversamente, Harry não obterá gene da mãe com probabilidade 0,99. Pai James tem 2 cópias do gene, então Harry obterá gene do pai com probabilidade 0,99 (isso é 1 - PROBS["mutação"]), mas obterá gene da mãe com probabilidade 0,01 (chance de mutação). Ambos casos podem ser somados para obter 0,99 * 0,99 + 0,01 * 0,01 = 0,9802, probabilidade de Harry ter 1 cópia do gene;
- Harry tem 1 cópia do gene, probabilidade de que ele não tenha característica é 0,44 (isto é PROBS["trait"][1][False]). Então, probabilidade de que Harry tenha 1 cópia do gene e não tenha característica é 0,9802 * 0,44 = 0,431288;
- Portanto, toda probabilidade conjunta é apenas resultado da multiplicação de todos valores para cada 1 das 3 pessoas: 0,9504 * 0,0065 * 0,431288 = 0,0026643247488.

Submissão

- Visual Studio Code online: https://cs50.dev
- Testar precisão da lógica do algoritmo: check50 ai50/projects/2024/x/heredity
- Testar estilização do código: style50 heredity.py
- Para submissão:
  - Em https://submit.cs50.io/invites/d03c31aef1984c29b5e7b268c3a87b7b, entre com GitHub e autorize CS50;
  - Instale pacote Git, Python 3 (e pip), instalando pacotes: pip3 install style50 check50 submit50
  - Submeta o projeto: submit50 ai50/projects/2024/x/heredity
- Verificar avaliação: https://cs50.me/cs50ai.



UB Social

Mateus Schwede
HBS ID 202400167108 - DCE ID @00963203