02

Definindo a distância euclidiana entre usuários

Transcrição

A gente comentou que existem algumas maneiras diferentes de abordar um problema de recomendação, por exemplo por meio de uma regressão com probabilidades, com uma classificação de qual elemento recomendar, como um problema de ranqueamento, etc.

Nessa etapa, vamos utilizar os dados que viemos trabalhando para criarmos a base de um algoritmo de Machine Learning. No nosso conjunto de dados, temos duas informações muito importantes: as características dos filmes e as notas que os usuários deram a eles.

No último exemplo, o nosso usuário tinha apenas assistido a alguns filmes, mas não tínhamos as notas que ele deu para eles. Ele poderia ter odiado Jumanji, e nesse caso não faria sentido recomendar um filme parecido com esse.

Agora, vamos tentar outra abordagem de recomendação: se o Guilherme gostou do Jumanji, buscaremos outros usuários que gostaram de Jumanji, acreditando que pessoas similares em uma área vão compartilhar outras características similares. Claro, a definição de "similar" é algo que também teremos que abordar.

Vamos imaginar um cenário simples para começarmos a refletir. O João deu nota 4 para o filme Toy Story, a Maria deu nota 5 e o Paulo deu nota 3. Dado esse conjunto de dados, quem é a pessoa mais parecida com a Maria? Bom, eu diria que o João é a pessoa mais parecida, pois ele deu uma nota mais próxima que a dela.

E quem é a pessoa mais próxima do João? Dá na mesma, pois tanto Maria quanto Paulo estão à mesma distância dele. Se se adicionássemos outra usuária, a Joana, que deu 3,5 para Toy Story? Agora, a pessoa mais próxima da Maria ainda é o João, mas a pessoa mais próxima do João é a Joana, a pessoa mais próxima de Paulo também é a Joana, e João e Paulo estão igualmente próximos da Joana.

Nesse cenário, a distância entre João e Maria é 1, e a distância entre João e Joana é 0,5. Por isso, o João está mais próximo da Joana que da Maria. É uma simples conta de subtração (x - y). Mas temos que tomar cuidado, afinal, 5 - 4 (a distância entre Maria e João) é 1, e 4 - 5 (a distância entre João e Maria) é... -1? O comum é fazermos essa conta mantendo sempre os valores positivos, pegando o valor absoluto dessa diferença.

Tudo isso está sendo calculado com base em apenas um filme, mas e se tivéssemos dois? Vamos supor que o João deu as notas 4 e 4,5, e a Maria deu as notas 5 e 5. Os filmes, nesse caso, não importam. Como calcularemos a distância entre eles?

Quando tínhamos uma reta, bastava subtrairmos os dois pontos. Porém, agora não temos mais somente uma dimensão, mas sim duas. Podemos imaginar que cada um desses arranjos ([4, 4,5] e [5, 5]) são pontos compostos por X e Y, então vamos plotá-los.

Importaremos o matplotlib.pyplot como plt, e faremos plt.plot(4, 4.5, "go") para gerarmos um plano cartesiano com uma bolinha verde nessas coordenadas, representando as notas do João. Em seguida, faremos plt.plot(5, 5, "yo"), gerando uma bolinha amarela que representa as notas da Maria.

Agora, adicionaremos as legendas para esses dois pontos com plt.legend(["João, "Maria"]), e um título com plt.title("Calcular a distância entre dois entre dois usuários").

Para calcularmos a distância entre os pontos, vamos imaginar que eles são duas das três pontas de um triângulo. Usaremos, então, o ponto [5, 4.5] para completarmos esse triângulo. Vamos plotá-lo no plano com plt.plot(), utilizando linestyle para

gerar as linhas e color="b" para mantermos somente uma cor nas três linhas.

```
plt.plot([4, 5], [4.5, 4.5], linestyle="-")
plt.plot([4, 5], [4.5, 5], linestyle="-")
plt.plot([5, 5], [5, 4.5], linestyle="-")
```

Temos, então, um triângulo retângulo, e para calcularmos a distância entre João e Maria só precisaremos descobrir a hipotenusa desse triângulo.

Se tentarmos fazer uma operação entre os arrays que representam os pontos de joao e maria, não conseguiremos, pois o Python não suporta operações com listas nativamente. Porém, o Numpy suporta. Portanto, faremos import numpy as np e criaremos arrays do numpy para esses dois usuários. Iremos subtrair essas duas listas, obtendo a distância entre os dois em cada um dos eixos: no eixo x, a distância é -1; e no eixo y, ela é -0,5.

Agora, tiraremos a raiz quadrada da soma dos quadrados dessas duas distância, o que é justamente o teorema de Pitágoras. Para isso, definiremos a função pitagoras(a,b), que receberá dois pontos quaisquer. Nela, faremos a - b, devolvendo dois valores que chamaremos de delta_x e delta_y.

Em seguida, faremos `return sqrt(delta_x * delta_x + delta_y * delta_y)` , de modo a obtermos raiz quadrada (square root) da soma dos quadrados desses valores, que é a nossa hipotenusa. Não se esqueça de importar a função sqrt() do módulo math.

Dessa forma, podemos fazer pitagoras(joao, maria), obtendo como retorno 1.1180. Portanto, a distância entre João e Maria, dadas as notas que eles deram para dois filmes, é 1.1180.

No Numpy, existe a função npt.linealg.norm(), que calcula diretamente isso que estamos chamando de pitágoras. Podemos, inclusive, redefinir a função pitagoras(a,b) como sendo simplesmente o retorno de npt.linealg.norm(a - b), e o resultado continuará o mesmo.

Também podemos calcular as distâncias entre mais usuários. Como exemplo, faremos plt.plot(3.5, 4.5, "bo"), gerando um ponto azul que representará as notas de nossa terceira usuária, a Joaquina. Agora que temos esses três pontos, queremos saber quem é mais próximo do João. Na verdade, é possível inferir a resposta a olho, mas isso seria difícil se tivéssemos muitos usuários.

Matematicamente, podemos calcular o pitagoras() entre João e Maria, e em seguida entre João e Joaquina. Veremos, então, que a Joaquina está muito mais próxima do João (0.5) do que a Maria (1.11). Portanto, se formos recomendar filmes para o João, poderemos nos basear naqueles que a Joaquina gostou.

Essa é uma simplificação do algoritmo que iremos implementar aos poucos nesse curso. A ideia é utilizarmos a função pitagoras() para calcularmos as distâncias entre dois pontos - inclusive, podemos renomeá-la como distancia(). Esses dois pontos podem ser compostos por 2, 3, 4, 5 ou mais filmes, afinal, o numpy nos ajuda a calcular a diferença/norma entre dois vetores, obstante quantas forem as dimensões.

Com base nessas distâncias, saberemos dizer quais usuários são mais similares entre si, o que servirá de base para implementarmos o resto do algoritmo.