Programação II – Atividade Opcional

Tema: Similaridade do Cosseno

Todas as definições aqui apresentadas são informais, porém, precisas o suficiente para a realização das tarefas propostas.

Leia o material apresentado, esclareça dúvidas via fórum e pesquisas na internet, faça os exercícios de fixação, confira as respostas, construa a API (módulo) da forma como especificada. Por último, resolva a atividade pedida ao final.

I. Introdução

I.1. Espaço Vetorial

Um vetor é um objeto matemático dotado de direção, sentido e módulo (tamanho do vetor enquanto segmento de reta orientado). Vetores são especificados por meio das suas componentes. Por sua vez, as componentes são definidas a partir das projeções do vetor sobre cada um dos eixos do sistema de coordenadas utilizado para abrigar o vetor. Cada eixo desse sistema de coordenadas é chamado de dimensão.

Posto dessa forma, fica estabelecido que um espaço vetorial é um conjunto de vetores que compartilham entre si um mesmo conjunto, ou sistema, de coordenadas (um mesmo conjunto de dimensões). Espaços vetoriais existem nas mais variadas configurações e quantidades de dimensões.

Para saber mais:

http://www.somatematica.com.br/emedio/vetores/vetores.php
http://www.somatematica.com.br/emedio/vetores/vetores5.php

I.2. Representação de Vetores e Espaços Vetoriais

Espaços vetoriais não precisam possuir uma representação geométrica. A figura 1 oferece mais de uma representação para espaços vetoriais.

Em 1.a temos um espaço vetorial bidimensional (2D) contendo dois eixos coordenados (dimensões), X e Y. Ali estão representados dois vetores, denominados *S* e *T*. Em 1.b temos um espaço vetorial tridimensional (3D) contendo três eixos coordenados (dimensões), X, Y e Z. Nesse espaço estão representados as versões tridimensionais de outros dois vetores, nomeados de *U* e *V*.

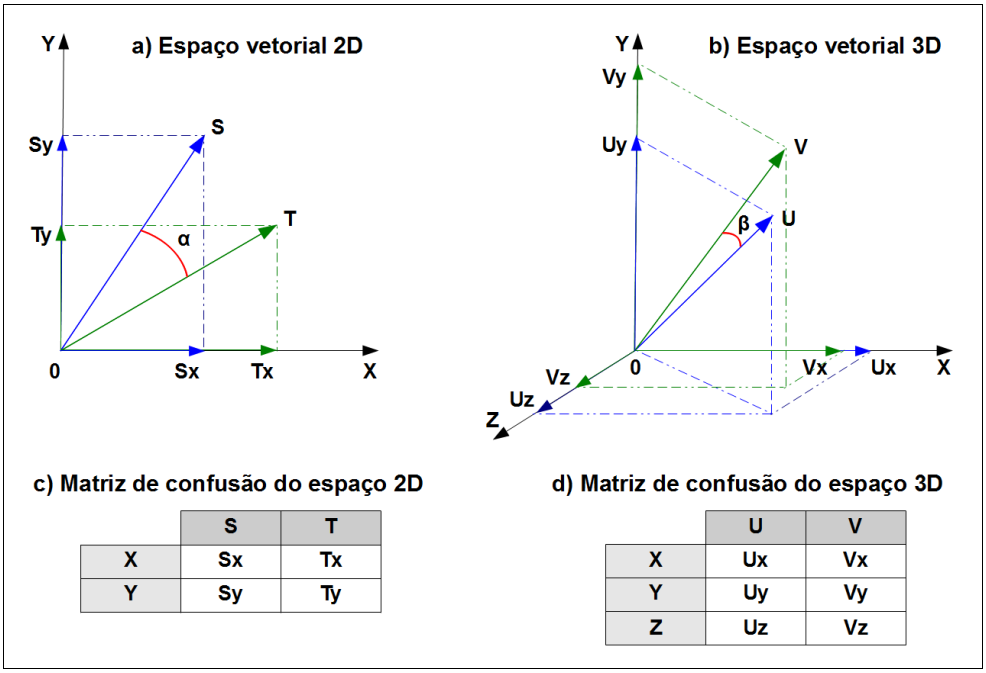
Cada vetor possui suas componentes (projeções) nas respectivas dimensões (eixos) de cada espaço vetorial:

Sx, Sy, Tx, Ty, em 1.a;
Ux, Uy, Uz, Vx, Vy, Vz em 1.b.

Como mencionado anteriormente, vetores não precisam possuir uma representação geométrica. Não é possível, por exemplo, representar geometricamente (visualmente) um espaço vetorial com um número de dimensões maior do que 3, apesar de ser totalmente razoável a sua existência. Esta é a vantagem da representação tabelar mostrada nas figuras 1.c e 1b.

1.c e 1.d mostram o equivalente tabelar/matricial dos espaços vetoriais representados graficamente em 1.a e 1.b.

Nesse tipo de representação, as colunas representam os vetores que povoam o espaço vetorial, e as linhas representam as dimensões, ou eixos, que estruturam o espaço vetorial. Na literatura, esse tipo de leiaute de dados é algumas vezes chamado de matriz de confusão.



	S	T
X	Sx	Tx
Y	Sy	Ty

	U	V
X	Ux	Vx
Y	Uy	Vy
Z	Uz	Vz

Figura 1

Por meio da construção tabelar, pode-se representar espaços vetoriais de quaisquer quantidade de dimensões. A figura 2, por exemplo, ilustra um espaço vetorial de 5 dimensões e 3 vetores .

Ainda na figura 2, por uma questão de praticidade, as dimensões foram nomeadas como 1, 2, 3, 4 e 5, ao invés de x, y, z, etc. Observe que, para esse espaço vetorial, não é possível construir uma representação geométrica utilizando um sistema de eixos ortogonais.

Espaço Vetorial de 5 Dimensões e 3 Vetores

	R	S	T
1	R_1	S_1	T_1
2	R_2	S_2	T_2
3	R_3	S_3	T_3
4	R_4	S_4	T_4
5	R_5	S_5	T_5

Figura 2

Além das componentes de cada vetor, as figuras 1.a e 1.b ilustram uma outra informação também importante para o entendimento do tema similaridade do cosseno, o ângulo formado entre cada par de vetores. Assim, em 1.a temos o ângulo α, formado entre os vetores *S* e *T* (espaço vetorial 2D). E em 1.b temos o ângulo β, formado entre os vetores *U* e *V* (espaço vetorial 3D).

O ângulo entre dois vetores aqui é aquele definido pelo menor percurso angular entre os segmentos de reta que estão representando os vetores na figura.

Esse ângulo define uma relação entre os dois vetores envolvidos na sua delimitação. Essa relação é explorada nas seções seguintes.

I.3. Ângulo entre Vetores e Medida de Similaridade

O valor do ângulo entre dois vetores pode ser determinado a partir do cálculo do cosseno do ângulo. Uma das estratégias mais utilizadas é a determinação do cosseno do ângulo a partir do produto escalar entre dois vetores. Da expressão do produto escalar deriva-se a fórmula (expressão) mostrada na figura 3.

Para o cálculo, são necessários os valores de todas as componentes dos vetores envolvidos. Ainda na figura 3, o exemplo mostra a aplicação direta da fórmula no cálculo do ângulo entre os vetores hipotéticos *R* e *S*.

Na fórmula, U_i e V_i são as componentes vetoriais dos vetores genéricos *U* e *V*. O índice *i* aqui substitui a identificação literal padrão dos eixos coordenados dimensionais. Assim, U_1 representa U_x, U_2 representa U_y e assim por diante. Ver figura 2 para a identificação das dimensões via índices numéricos.

$$\cos(\alpha) = \frac{\sum_{i=1}^{n} U_i V_i}{\sqrt{\sum_{i=1}^{n} U_i^2}\sqrt{\sum_{i=1}^{n} V_i^2}}$$

, onde *Ui* e *Vi* são as componentes dos vetores *U* e *V*, respectivamente.

Exemplo:

Seja o seguinte espaço vetorial de 3 dimensões contendo os vetores R e S.

	R	S
X	1	5
Y	2	6
Z	3	1

Logo,

$$\cos(\alpha) = \frac{(1x5) + (2x6) + (3x1)}{\sqrt{1^2 + 2^2 + 3^2} \text{ x } \sqrt{5^2 + 6^2 + 1^2}}$$

$$\cos(\alpha) = \frac{20}{\sqrt{14} \text{ x } \sqrt{62}} \approx 0.678844233$$

$$\alpha = \arccos(0.678844233) \approx 47.246\,°$$

Figura 3

O ângulo entre dois vetores pode ser calculado desde que os vetores em questão estejam no mesmo espaço vetorial. Além disso, é perfeitamente razoável calcular o ângulo em espaços vetoriais de dimensão qualquer. Lembre-se que vetores não necessitam ter uma representação geométrica. Assim, não é nenhum absurdo calcular o cosseno do ângulo entre dois vetores de dimensão 5, por exemplo (figura 2).

O ângulo entre dois vetores estabelece uma relação simétrica entre os vetores. Ou seja, o ângulo entre o vetor *R* e *S* é o mesmo entre os vetores *S* e *R* (figura 1).

O valor do cosseno entre vetores pode ser tomado como uma medida de similaridade. Ou seja, dois vetores são tão semelhantes entre si quanto maior o valor do cosseno do ângulo entre eles.

Observe que cossenos variam o seu valor no intervalo fechado de 0 a 1. O valor zero é obtido para ângulo de 90 graus e seus múltiplos. Já o valor 1 é obtido para o ângulo de zero graus. Ou seja, quando os vetores estão sobrepostos e apontam para a mesma direção.

No caso do ângulo entre vetores ser de zero grau, eles podem, ou não, possuir os mesmos valores de componentes em cada respectiva dimensão.

Diz se de vetores cujo ângulo entre si produz cosseno igual a 1 que eles são 100% semelhantes (idênticos), e vetores cujo ângulo produz cosseno igual a zero que eles 0% semelhantes (totalmente diferentes). Valores entre 0 e 1 produzem diferentes percentuais de similaridade.

I.4. Compondo Espaços Vetoriais

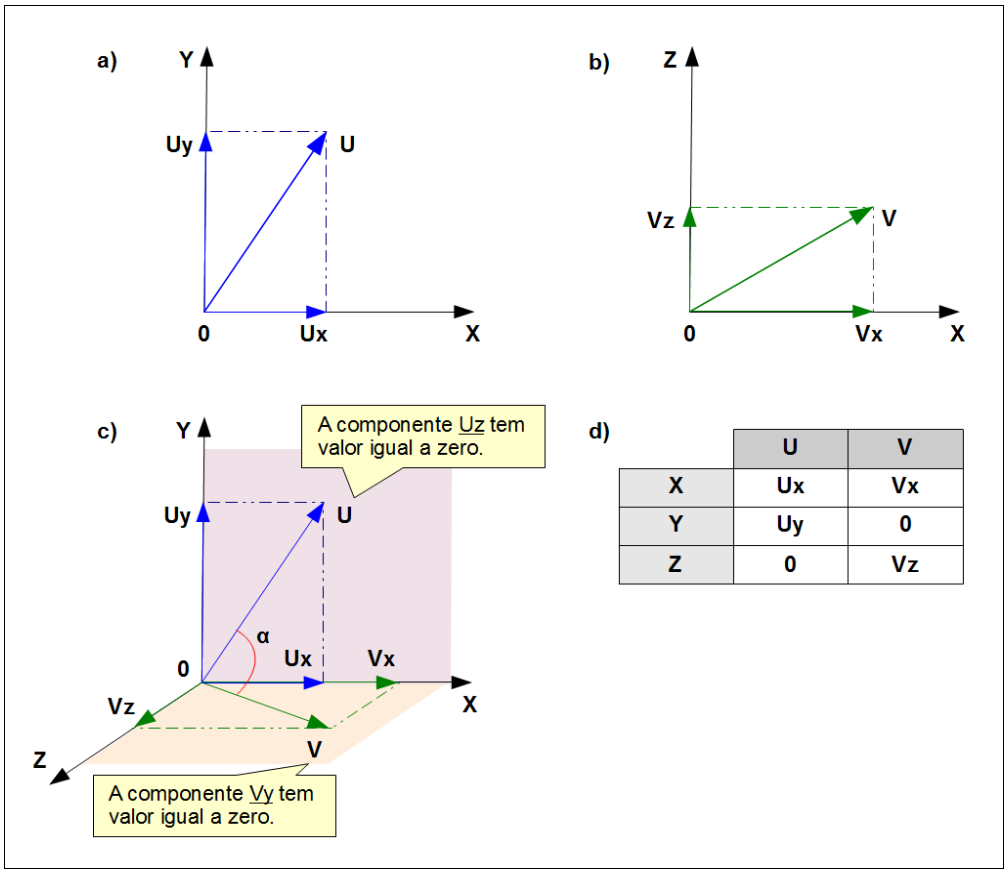
A figura 4.a e 4.b mostra 2 espaços vetoriais distintos. O espaço vetorial XY (plano x/y) abriga o vetor *U*. Já o espaço vetorial XZ abriga o vetor *V*.

Note que não poderemos utilizar a expressão da figura 3 para calcular o cosseno do ângulo entre *U* e *V* pois esses vetores não habitam o mesmo espaço vetorial.

Uma forma de resolver o problema é compor um único espaço vetorial a partir dos espaços XY e XZ. Esse novo espaço vetorial possuirá como dimensões a <u>união</u> das dimensões dos espaços vetoriais envolvidos. A figura 4.c mostra o novo espeço vetorial, identificado como XYZ.

A figura 4.c também mostra os vetores *U* e *V* mapeados para esse novo espaço, assim como as suas respectivas representações em forma de tabela. Observe que todos os vetores agora possuem 3 dimensões (componentes) e que a nova dimensão acrescentada ao espaço vetorial de origem de cada vetor possui valor de componente igual a zero.

O vetor *U* ganhou uma nova componente Z, com valor igual a zero. Já o vetor *V* ganhou uma nova componente Y, de valor igual a zero. Agora todos os vetores habitam o mesmo espaço vetorial e o uso da expressão da figura 3 para o cálculo do cosseno é perfeitamente viável.



	U	V
X	Ux	Vx
Y	Uy	0
Z	0	Vz

Figura 4

Qualquer operação entre vetores só é possível se eles compartilharem o mesmo espaço vetorial. Essa restrição não se aplica apenas ao cálculo da similaridade.

I.5. Contextualizando Espaços Vetoriais

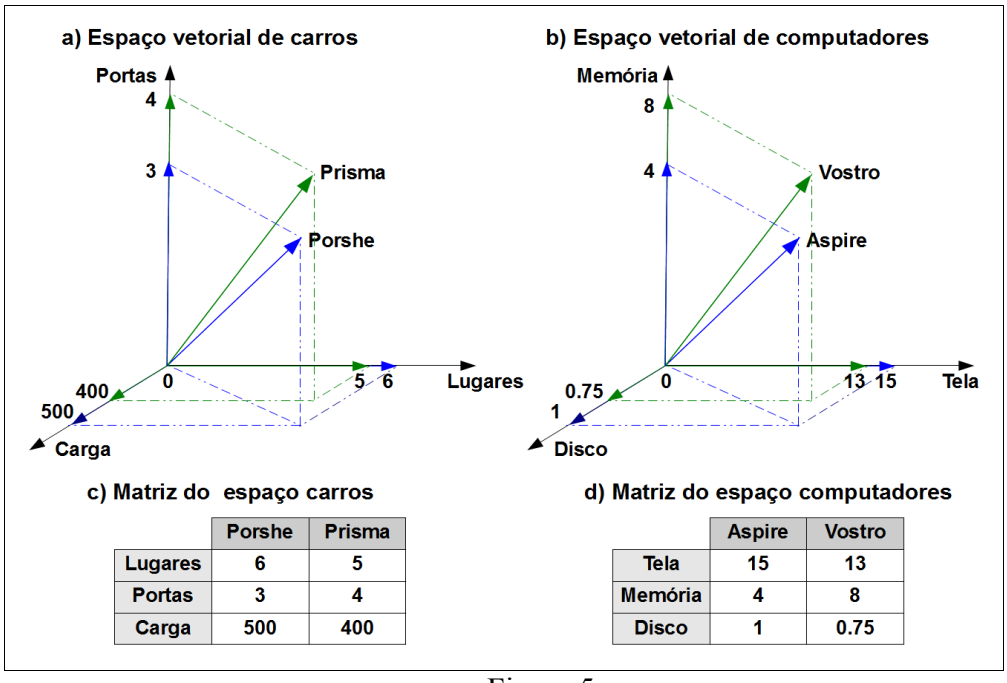
É possível fornecer um tema para os vetores que habitam um espaço vetorial. Pode-se utilizar vetores para representar documentos em uma coleção de documentos, modelos de carros em uma frota, perfis profissionais de uma empresa e etc. Em cada caso, cada vetor representaria um objeto do universo do domínio.

No caso em que vetores desempenham o papel de objetos de um domínio, o que os eixos coordenados, as dimensões do espaço vetorial, representariam ? A resposta é, as dimensões fariam as vezes de características de cada objeto/vetor representado. Cada dimensão representaria um atributo, ou qualidade, do objeto do domínio de interesse. Nesse caso, chamamos as dimensões de dimensões de qualidade, ou dimensões de características.

Em um espaço vetorial como os vistos nas seções anteriores, as dimensões não possuem um significado além do algébrico abstrato (X, Y, Z, etc.). Quando damos um significado a essas dimensões estamos dando um contexto de aplicação para o espaço vetorial.

A figura 5 fornece dois exemplos de espaços vetoriais contextualizados. O primeiro exemplo diz respeito a uma frota de carros. Já o segundo exemplo mapeia para o espaço vetorial um catálogo de computadores. Observe o papel das dimensões de qualidade (ou características). Cada componente de cada vetor define o valor de determinada característica (atributo) para o seu respectivo vetor objeto de domínio.

Talvez a representação tabelar/matricial (matriz de confusão), em 5.c e 5.d, seja mais familiar a esse tipo de modelagem do que a representação geométrica em si.



c) Matriz do espaço carros

	Porshe	Prisma
Lugares	6	5
Portas	3	4
Carga	500	400

d) Matriz do espaço computadores

	Aspire	Vostro
Tela	15	13
Memória	4	8
Disco	1	0.75

Figura 5

Tanto em 5.a quanto em 5.b, temos um espaço vetorial composto de 2 vetores e 3 dimensões de qualidades (características). Os valores ali expressos são todos fictícios.

No espaço vetorial de carros foram consideradas relevantes as características número de portas, quantidade de lugares e capacidade de carga (em Kg).

A partir dessa perspectiva, são mapeados dois vetores/carros, o *Prisma* e o *Porsche*.

O vetor *Prisma* especifica que esse modelo de carro possui 5 lugares, 4 portas e é capaz de transportar até 400 Kg. Já o vetor Porshe atesta que esse modelo de automóvel possui 6 lugares, 3 portas e é capaz de transportar até 500 Kg.

No espaço vetorial de computadores, foram consideradas relevantes as características tamanho da tela (em polegadas), quantidade de memória (em Gigabytes) e capacidade de armazenamento de disco (em Terabytes). A partir dessa perspectiva, são mapeados dois vetores/computadores, o *Vostro* (Dell) e o *Aspire* (Acer).

O vetor *Vostro* especifica que esse modelo de computador possui um monitor de 13 polegadas, uma quantidade de memória de 8 GB e um disco rígido de 750 MB (0.75 T). Já o vetor *Aspire* especifica que esse modelo de computador possui um monitor de 15 polegadas, uma quantidade de memória de 4 GB e um disco rígido de 1 T.

Uma vez representados no espaço vetorial, os objetos do domínio estarão sujeitos à diversas operações vetoriais, como soma e cálculo de similaridade, por exemplo. O significado do resultado dessas operações no universo do domínio é determinado pelo especialista/analista do domínio. Por exemplo, qual o significado no mundo real de um vetor carro que é a soma de dois vetores carro ?

Observe que as dimensões de características trabalham sempre com valores numéricos. Assim, características essencialmente não numéricas, como valores SIM/NÃO, devem ser sempre convertidas para numéricos. Por exemplo, no espaço vetorial de carros, a caraterística *Presença de ABS*, de valor SIM/NÃO, deve ser mapeada para um espaço numérico a fim de representar SIM, possui ABS (o valor 1, por exemplo) e NÃO, não possui ABS (o valor zero, por exemplo).

Os exemplos da figura 5 são apenas ilustrativos. Um espaço vetorial mais efetivo para um sistema de informação possuiria mais do que 3 características/qualidades/dimensões.

O espaço vetorial de carros, por exemplo, poderia ser n-dimensional, contendo características como: quantidades de airbags, quantidade de freio a disco (2 ou 4), quantidade de combustíveis aceitos pelo motor, cilindrada do motor e etc.

Nesse ponto do estudo deve ser óbvio que um espaço vetorial constituído de mais de 3 dimensões só poderá ser representado via tabela/matriz de confusão.

II. Exercícios de Fixação

Construa as funções pedidas segundo as especificações fornecidas. Confira o resultado da execução da sua implementação com os resultados fornecidos nas figuras exemplos. Utilize esses resultados como gabarito. Teste suas funções fazendo uso dos arquivos fornecidos no pacote da atividade.

II.1. Função cosseno

Construa a função *cosseno(lstU, lstV)* que calcula e retorna o cosseno entre dois vetores representados pelas listas parâmetros de entrada, *lstU* e *lstV*. Lembre-se que vetores só podem ter o seu cosseno calculado se pertencerem ao mesmo espaço vetorial (I.3). Caso os vetores não pertençam ao mesmo espaço vetorial (não possuam a mesma quantidade de dimensões), a função deve retornar **None**.

Utilize os vetores exemplo *R* e *S* da figura 3. Confira se o valor retornado pela sua função é o mesmo que consta da figura. Apenas para teste, a função que calcula o ângulo, em radianos, é a função arco-cosseno, *acos(<valor cosseno>)*, módulo *math*. Para converter o ângulo em graus, utilize *math.degrees(<ângulo em radianos>)*.

II.2. Função matrizCos

Construa a função *matrizCos(lstVet)* . A função recebe como entrada uma lista de vetores (lista de listas, cada lista interna representando um vetor), representando um espaço vetorial qualquer. A função devolve como saída uma matriz de cossenos de ângulos entre todos os pares de vetores do espaço vetorial. Ver figura 6 abaixo.

a) Matriz de confusão do espaço vetorial

	R	S	T	U	V	W
X	1	5	7	8	2	2
Y	2	6	3	1	6	1
Z	3	1	0	2	1	5

b) Matriz de cossenos do espaço vetorial

	R	S	T	U	V	W
R	1.0	0.7	0.5	0.5	0.7	0.9
S	0.7	1.0	0.9	0.7	0.9	0.5
T	0.5	0.9	1.0	0.9	0.7	0.4
U	0.5	0.7	0.9	1.0	0.5	0.6
V	0.7	0.9	0.7	0.5	1.0	0.4
W	0.9	0.5	0.4	0.6	0.4	1.0

Figura 6

Os valores constantes na figura são apenas exemplos. A função deve funcionar para qualquer espaço vetorial.

A matriz retornada conterá o cosseno do ângulo que cada vetor faz com cada outro vetor, inclusive com ele mesmo. Desnecessário dizer que a diagonal da matriz conterá apenas 1.

Observe que a matriz é simétrica: os valores acima da diagonal principal são os mesmos abaixo da diagonal. Resumindo, o ângulo/cosseno entre os vetores *R* e *S* é o mesmo que o ângulo/cosseno entre *S* e *R*.

A figura 6 mostra uma lista de vetores exemplos e sua respectiva matriz de cossenos de ângulos. Construa uma aplicação de testes para avaliar se a sua função está corretamente implementada. Utilize os vetores constantes em 6.a como dados de entrada de testes. Empregue a matriz em 6.b como um gabarito para avaliar a saída das sua função.

Importante: A função não deve se ater aos títulos das colunas e linhas da matriz (letras nas colunas e linhas representando os nomes dos vetores). A função trabalha e produz apenas o conteúdo numérico da matriz. A função não exibe a matriz.

Para exibir o conteúdo mostrado na figura 6.b, construa uma função com o único propósito de imprimir a matriz no formato desejado. A função recebe como entrada a matriz já preenchida e produz como resultado a exibição formatada que aparece em 6.b.

II.3. Funções compoeEV, salvaEV, carregaEV

Construa a função *compoeEV(<nome pasta com coleção vetores>)*. A teoria por trás do objetivo da função está descrita no item I.4. A função recebe como parâmetro de entrada o nome de uma pasta contendo uma coleção de arquivos representando vetores pertencentes a um determinado domínio contextualizado (I.5). Dentro dessa pasta, sempre existirá um arquivo chamado *bdev.txt*.

O arquivo *bdev.txt* possui como conteúdo os nomes de todos os arquivo contidos na pasta, um nome por linha. Por sua vez, os arquivos de vetor contém, em cada linha, o nome de uma dimensão do espaço vetorial e o respectivo valor da componente na dimensão. Veja a figura 6 para entender o conteúdo desses dois tipos de arquivo.

A figura 7 mostra exemplos de arquivos contextualizados para representar um espaço vetorial de uma coleção de modelos de notebooks. Cada vetor representa um modelo de notebook (aqui identificados por *ntb01.txt* a *ntb10.txt*). A figura 7.a mostra um conteúdo exemplo de um típico arquivo *bdev.txt*. Já a figura 7.b mostra dois típicos conteúdos de arquivos contendo vetores representando modelos de notebooks.

a) Conteúdo exemplo do arquivo de vetores.

```
ntb01.txt
ntb02.txt
ntb03.txt
ntb04.txt
ntb05.txt
ntb06.txt
ntb07.txt
ntb08.txt
ntb09.txt
ntb10.txt
```

b) Conteúdo exemplo de dois arquivos vetores notebooks.

ntb01.txt	ntb01.txt
bateria,4	bateria,8
cpu,8	cache,4
disco,2	cpu,4
pl-video,4	disco,2
ssd,256	hdmi,1
usb,2	pl-video,2
	ssd,128
	tela,15

c) Espaço vetorial normalizado (composto) da base *vetnotebooks*.

```
['bateria','cache','cpu','disco','hdmi','peso','pl-video','ssd','tela','usb']
```

Figura 7

Observe que os arquivos de vetores não pertencem todos ao mesmo espaço vetorial. A quantidade e o tipo das dimensões varia levemente de um vetor para outro. É justamente esse o trabalho da função *compoeEV*. A função deve processar todos os arquivos vetores e devolver como saída uma lista contendo todas as dimensões encontradas nos arquivos de vetores, sem repetição. Ou seja, uma lista representando um único espaço vetorial normalizado (ou composto) formado a partir da união dos espaços vetoriais de cada vetor da coleção de vetores.

Faz parte do material disponibilizado pelo professor uma base de dados para o teste da função *compoeEV*. Localize a pasta *vetnotebooks* e, dentro dela, o arquivo de vetores *bdev.txt*. Construa uma aplicação que utilize a função *compoeEV* para obter e exibir na tela o espaço vetorial normalizado dessa coleção de vetores. Confira sua resposta com o gabarito mostrado na figura 7.c.

A base de dados exemplo contém um arquivo com a lista de vetores da base (arquivo *bdev.txt*) mais 20 arquivos representando 20 vetores mapeando modelos de notebooks para o espaço vetorial.

Construa a função *salvaEV(<lista dimensões espaço vetorial>)*. A função recebe como parâmetro de entrada uma lista contendo as dimensões de um espaço vetorial qualquer (ver figura 7.c). A função deve salvar o conteúdo da lista em um arquivo chamado *dimev.txt*. O conteúdo deve ser salvo da seguinte forma: cada elemento da lista em uma linha do arquivo texto. Se a lista passada estiver vazia, a função deve retornar o valor **None**.

Construa a função *carregaEV(<nome do arquivo>)*. A função recebe como parâmetro de entrada o nome de um arquivo texto contendo as dimensões de um espaço vetorial qualquer. O arquivo texto armazena o espaço vetorial na forma de uma dimensão por linha, como descrito no parágrafo anterior para a função *salvaEV(..)*. A função deve retornar uma lista contendo o espaço vetorial carregado (ver figura 7.c). Caso o arquivo fornecido não exista, a função deve retornar o valor **None**.

II.4. Funções matrizEV, impMatrizEV

Construa a função *matrizEV(<nome pasta com coleção vetores>)*. A teoria por trás do objetivo da função está descrita na seção I. Assim como *compoeEV(..)*, esta função recebe como parâmetro de entrada o nome de uma pasta contendo uma coleção de vetores pertencentes a um domínio contextualizado. O contexto dos vetores é o mesmo descrito para a função *compoeEV*, seção II.3. Ou seja, os vetores pertencem a diferentes espaços vetoriais. Como em II.3, a pasta também abriga um arquivo *bdev.txt* contendo a relação de todos os arquivos de vetores da pasta.

A função *matrizEV* deve normalizar os espaços vetoriais da pasta via chamada a *compoeEV*. Em seguida, após ler o arquivo *bdvet.txt*, construir a matriz de confusão do espaço vetorial normalizado (composto). A figura 8 exibe uma matriz de confusão exemplo para o espaço vetorial normalizado da pasta *vetnotebooks*. Por uma questão de espaço disponível, a matriz da figura mostra apenas os valores dos 10 primeiros vetores da coleção.

Matriz de confusão do espaço vetorial da pasta *vetnotebooks*.

	1	2	3	4	5	6	7	8	9	10
bate	4.00	8.00	6.00	8.00	4.00	4.00	2.00	4.00	0.00	2.00
cach	3.00	3.00	2.00	4.00	0.00	2.00	0.00	4.00	0.00	2.00
cpu	4.00	6.00	4.00	0.00	2.00	6.00	2.00	8.00	2.00	4.00
disc	0.25	1.00	1.00	0.75	0.75	0.50	0.50	0.00	0.00	0.25
hdmi	0.00	0.00	2.00	1.00	0.00	2.00	2.00	1.00	1.00	2.00
peso	1.50	2.00	1.00	1.00	1.50	1.50	1.00	2.50	1.00	1.50
pl-v	1.00	1.00	1.00	0.00	4.00	0.00	2.00	0.00	1.00	0.00
ssd	0.25	0.50	0.25	0.12	0.06	0.50	0.06	0.06	0.25	0.06
tela	13.00	17.00	0.00	13.00	15.00	15.00	0.00	15.00	11.00	17.00
usb	2.00	3.00	3.00	2.00	2.00	2.00	1.00	2.00	2.00	0.00

Obs.: o título das colunas referem-se aos arquivos ntb01,txt, ntb02.txt, etc.

Figura 8

Obviamente, a matriz na figura é apenas um exemplo. A função *matrizEV* deve ser capaz de gerar a matriz de qualquer pasta contendo arquivos de vetores e um arquivo *bdev.txt*.

A função deve retornar como saída uma tupla contendo a matriz de confusão e a lista com as dimensões do espaço vetorial (a saída da chamada a *compoeEV*).

Não é papel da função imprimir a matriz na tela. Logo, os títulos de colunas e linhas mostrados na figura não são preocupação da função *matrizEV*. A função trata apenas do conteúdo numérico da matriz. A exibição da matriz é um problema a parte, tratado pela função a seguir.

Construa a função *impMatrizEV(<matriz de confusão>,<lista dimensões espaço vetorial>)* que tem como objetivo exibir a matriz num formato semelhante ao mostrado na figura 8. A função recebe como parâmetros de entrada a matriz de confusão e a lista com os nomes das dimensões do espaço vetorial, justamente as saídas produzidas pela função *matrizEV*. Observe que os conteúdos numéricos estão formatados para exibir 2 casas decimais.

II.5. Função matrizCos2

Construa a função *matrizCos2(<matriz de confusão>)* a partir da função *matrizCos*, em II.2. A função *matrizCos2* recebe como parâmetro de entrada a matriz de confusão do espaço vetorial. A saída da função tem o mesmo significado da saída da função *matrizCos*.

Lembre-se que a função *matrizCos* recebe como parâmetro uma lista de listas, com cada lista interna representando um vetor. Ou seja, essa função recebe uma matriz onde as colunas são as dimensões do espaço vetorial, e as linhas, os vetores. Contudo, a matriz de confusão organiza esses mesmos dados de forma oposta, as colunas são os vetores e as linhas as dimensões do espaço vetorial.

Dica: altere o código da função *matrizCos* e insira no inicio da função um algoritmo para fazer a transposta da matriz de confusão fornecida como parâmetro de entrada. Feito isso, o código para gerar a matriz de cossenos em *matrizCos2* será o mesmo já construído para *matrizCos*.

A figura 9 exibe a matriz de cossenos para o espaço vetorial da pasta *vetnotebooks*. Por uma questão de espaço na figura, apenas uma parcela (15 x 15) da matriz completa (20 x 20) é exibida .

Matriz de cossenos do espaço vetorial da pasta *vetnotebooks*

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	1.00	0.99	0.44	0.93	0.95	0.98	0.34	0.98	0.93	0.96	0.95	0.99	0.90	0.95	0.93
2	0.99	1.00	0.53	0.94	0.94	0.98	0.41	0.97	0.89	0.93	0.94	0.99	0.92	0.91	0.92
3	0.44	0.53	1.00	0.48	0.32	0.44	0.84	0.48	0.19	0.25	0.36	0.43	0.47	0.22	0.43
4	0.93	0.94	0.48	1.00	0.89	0.89	0.31	0.86	0.81	0.87	0.89	0.94	0.93	0.91	0.76
5	0.95	0.94	0.32	0.89	1.00	0.93	0.34	0.89	0.95	0.94	0.99	0.97	0.94	0.95	0.90
6	0.98	0.98	0.44	0.89	0.93	1.00	0.38	0.99	0.94	0.97	0.93	0.97	0.86	0.92	0.97
7	0.34	0.41	0.84	0.31	0.34	0.38	1.00	0.39	0.22	0.23	0.39	0.36	0.40	0.18	0.46
8	0.98	0.97	0.48	0.86	0.89	0.99	0.39	1.00	0.90	0.95	0.88	0.96	0.81	0.89	0.96
9	0.93	0.89	0.19	0.81	0.95	0.94	0.22	0.90	1.00	0.97	0.95	0.93	0.85	0.96	0.92
10	0.96	0.93	0.25	0.87	0.94	0.97	0.23	0.95	0.97	1.00	0.92	0.96	0.83	0.96	0.93
11	0.95	0.94	0.36	0.89	0.99	0.93	0.39	0.88	0.95	0.92	1.00	0.96	0.94	0.94	0.90
12	0.99	0.99	0.43	0.94	0.97	0.97	0.36	0.96	0.93	0.96	0.96	1.00	0.93	0.95	0.91
13	0.90	0.92	0.47	0.93	0.94	0.86	0.40	0.81	0.85	0.83	0.94	0.93	1.00	0.89	0.79
14	0.95	0.91	0.22	0.91	0.95	0.92	0.18	0.89	0.96	0.96	0.94	0.95	0.89	1.00	0.84
15	0.93	0.92	0.43	0.76	0.90	0.97	0.46	0.96	0.92	0.93	0.90	0.91	0.79	0.84	1.00

Obs.: o espaço vetorial completo gera uma matriz 20 x 20.

Figura 9

Construa uma aplicação para testar a função construída. Utilize a função de impressão da matriz de cossenos desenvolvida em II.2.

A aplicação deve fazer:

i) calcular a matriz de confusão usando a função *matrizEV*,
ii) passar os valores retornador para *matrizCos2*,
iii) imprimir a matriz de cossenos na tela.

Utilize a pasta *vetnotebooks* como massa de dados de teste. Confira visualmente a sua saída com os dados mostrados na figura 9. Utilize a figura como um gabarito para avaliar a sua função.

II.6. Função salvaMat(<matriz>,<nome do arquivo>)

Construa a função *salvaMat* que recebe como parâmetros de entrada uma matriz qualquer (matriz de cossenos, matriz de confusão) e o nome de um arquivo texto de saída. A função salva a matriz no arquivo texto da seguinte forma, cada linha da matriz é uma linha no arquivo texto. Apenas o conteúdo numérico da matriz é salvo. Seus rótulos com títulos de linhas e colunas não fazem parte da matriz.

II.7. Função carregaMatConfu(<nome do arquivo>)

Construa a função *carregaMatConfu* que recebe como parâmetros de entrada o nome de um arquivo texto de entrada. O arquivo texto de entrada deve ter o formato descrito na seção anterior para a função salvaMat. *carregaMatConfu* deve retornar a matriz de confusão gerada a partir do conteúdo lido do arquivo.

II.8. Módulo libsimcoss

Reúna todas as funções construídas na seção II em um arquivo chamado *libsimcoss.py*. Esse arquivo será um módulo biblioteca a ser utilizado na seção III a seguir. Esteja certo de ter testado todas as funções utilizando as bases de dados fornecidas e os gabaritos constantes nas figuras exemplos.

III. Atividade Proposta

Utilize a base de dados *vetfuncionarios* (pasta de mesmo nome) para construir o que é pedido nas seções abaixo. As construções devem fazer uso do módulo *libsimcoss.py.*

A base de dados vetfuncionarios é um espaço vetorial que representa as respostas de cada um dos 100 funcionários de uma empresa a um questionário de 50 perguntas. A ideia é traçar uma matriz de afinidades entre funcionários da empresa para aprimorar políticas de relações humanas no trabalho.

Cada uma das perguntas define uma dimensão preferência de cada funcionário. Cada dimensão possui como domínio números de 0 a 5 representando uma escala de simpatia/gosto do funcionário ao tema da pergunta. A interpretação da escala é a seguinte: 0 (não gosto), 1 (gosto muito pouco), 2 (indiferente), 3 (gosto), 4 (gosto muito), 5 (não vivo sem).

A relação completa das dimensões está listada abaixo. Cada dimensão ocupa a lacuna na pergunta '*O quanto você gosta de _____ ?*'

(Obs.: remova os acentos durante a implementação em código)

```
aventura, axé, bar, baralho, bicicleta, cachorro, caminhar, cantar, casa,
cerveja, cidade, cinema, comedia, comida, contos, cor, desenhar, dirigir, drama,
drink, escrever, feiras, filmar, flor, forró, fotografar, fruta, funk, futebol,
galeria, games, gato, ler, montanha, museu, nadar, novela, pássaro, pintar,
poesia, praia, romance, salão, show, sobremesa, suco, teatro, tocar, vinho,
xadrez
```

Por exemplo, a dimensão '*teatro*' corresponde a pergunta '*O quanto você gosta de teatro ?*'. A exceção é a dimensão/característica '*cor*'. Essa dimensão diz respeito à cor preferida do funcionário. Os valores possíveis vão de 0 a 9, cada um representando uma cor do espectro de cores (0-preto, 1-branco, 2-vermelho, etc.). Aventura, comédia, romance, drama dizem respeito a gêneros de filmes. Salão se refere a danças de salão.

Da mesma forma que a base *vetnotebooks* nas seções II.3 a II.5, os vetores da base *vetfuncionarios* não pertencem todos ao mesmo espaço vetorial. Para processá-los é necessário antes renuí-los em um único espaço vetorial.

III.1. Função matAfinidade(<matriz cosseno>, <valor limite afinidade>)

Acrescente ao módulo *libsimcoss* a função *matAfinidade*. A função recebe como entradas uma matriz de cossenos (II.2) e um número real. A função deve alterar o conteúdo da matriz de cosseno a fim de transformá-la em uma matriz de afinidade entre vetores. Ou seja, a função recebe uma matriz vetores x vetores e retorna uma matriz vetores x vetores.

A alteração se dará da seguinte forma: se o conteúdo do elemento linha/coluna for menor que o <*valor limite de afinidade*> (também chamado coeficiente de afinidade) então esse valor será modificado para receber zero. Se o conteúdo for maior ou igual ao limite então o seu novo valor será 1.

A matriz de afinidade é uma matriz quadrada de conteúdo binário, 0s e 1s. Não confundir as matrizes de cossenos e afinidades com a matriz de confusão. Esta última pode ou não ser quadrada (se a quantidade de dimensões for igual à quantidade de vetores), e não é binária.

No contexto de funcionários, a matriz de afinidade representará os funcionários que possuem alto potencial de fazerem parcerias entre si.

A figura 9 abaixo mostra um trecho da matriz de afinidade do espaço vetorial *vetfuncionarios*. A matriz real é muito extensa (100 x 100) para caber na figura.

Dica: modifique o código da função *matrizCos* em II.2.

III.2. Aplicação procevfunc.py

Construa a aplicação *procevfunc.py* com o perfil de aplicação baseada em linha de comando. A aplicação deve receber como parâmetro de entrada (durante a chamada do programa) o nome de uma base de vetores, pasta, e a partir daí gerar os seguintes produtos (fazendo uso da *libsimcoss*):

i. Matriz de cossenos;
ii. Matriz de confusão;
iii. Matriz de afinidade;
iv. Lista com as dimensões do espaço vetorial normalizado;
v. Arquivo *bdmatcos.txt* contendo a matriz de cossenos salva.;
vi. Arquivo *bdmatconfu.txt* contendo a matriz de confusão salva;
vii. Arquivo *bdmatafinidade.txt* contendo a matriz de afinidade entre vetores;
viii. Gráfico de rede de conexões (ver seção II.4 a seguir).

Matriz de afinidade do espaço vetorial da pasta *vetfuncionarios*

	1	2	3	4	5	6	7	8	9	10
1	0.00	0.00	1.00	0.00	0.00	0.00	1.00	0.00	0.00	1.00
2	0.00	0.00	0.00	1.00	0.00	1.00	0.00	1.00	1.00	0.00
3	1.00	0.00	0.00	1.00	0.00	1.00	1.00	0.00	1.00	1.00
4	0.00	1.00	1.00	0.00	0.00	1.00	1.00	0.00	1.00	0.00
5	0.00	0.00	0.00	0.00	0.00	1.00	1.00	0.00	0.00	0.00
6	0.00	1.00	1.00	1.00	1.00	0.00	1.00	0.00	1.00	1.00
7	1.00	0.00	1.00	1.00	1.00	1.00	0.00	1.00	1.00	1.00
8	0.00	1.00	0.00	0.00	0.00	0.00	1.00	0.00	1.00	0.00
9	0.00	1.00	1.00	1.00	0.00	1.00	1.00	1.00	0.00	0.00
10	1.00	0.00	1.00	0.00	0.00	1.00	1.00	0.00	0.00	0.00

Obs.: o espaço vetorial completo gera uma matriz 50 x 100.

Figura 10

Na matriz da figura 10, o valor utilizado para o coeficiente de afinidade foi de 0.7. Faça testes com outros valores. Observe como as afinidades se distribuem pela matriz à medida que os valores desse coeficiente são gerados.

III.4. Função exibeRedeAfin(<matriz de afinidade>)

Acrescente ao arquivo de aplicação *procevfunc.py* a função *exibeRedeAfin*. Para construir essa função é necessário pesquisar e aprender a usar o módulo python *NetworkX*.

O módulo *NetworkX* exibe graficamente redes de conexões entre elementos de um arranjo tipo grafo. A matriz de afinidade pode ser considerada um grafo na medida em que descreve a interconexão entre vetores (vetores que tem afinidade com outros vetores).

A figura 10 fornece um exemplo de uma rede criada pela biblioteca *NetworkX*. No diagrama, os círculos coloridos representam os vetores 1, 2, 3, etc. As linhas conectando os vetores informam que esses vetores possuem uma afinidade entre eles.

No caso do contexto de funcionários, as linhas indicam um valor igual a 1 no elemento linha/coluna da matriz de afinidades. Além disso, nesse mesmo contexto, o diagrama pode ser interpretado como uma representação visual dos potenciais agrupamentos de colaboradores na empresa.

Diagramas como esse são indicados quando se quer visualizar e pesquisar dados inter-relacionados e detectar possíveis aglomerações e concentrações desses dados. Assim como investigar e compreender os processos e dinâmicas responsáveis por sua formação.

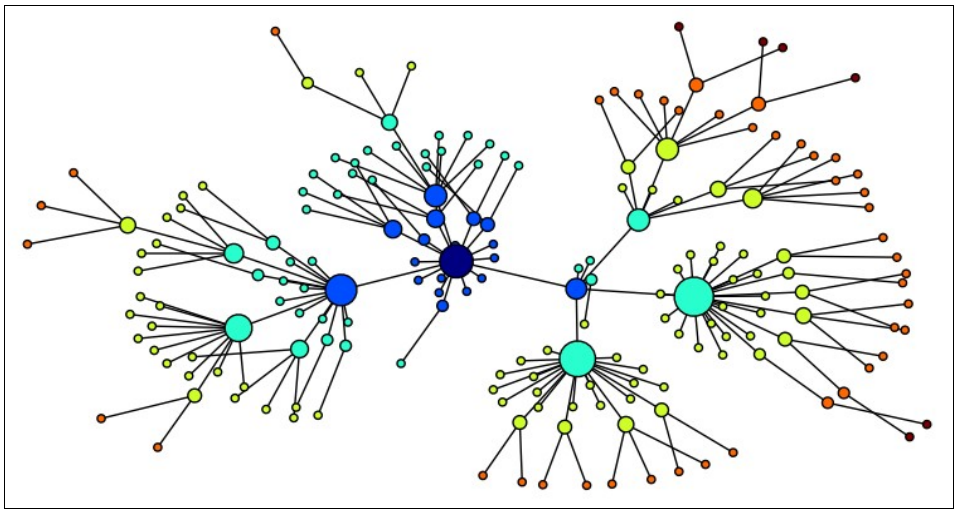


Figura 10

Utilize o material fornecido nos links abaixo para estudar e fazer funcionar o módulo *NetworkX*:

1. http://networkx.github.io/
2. http://www.python-course.eu/networkx.php
3. http://snap.stanford.edu/class/cs224w-2011/nx_tutorial/nx_tutorial.pdf
4. https://www.udacity.com/wiki/creating-network-graphs-with-python

Se achar necessário, pesquise outras fontes e compartilhe no fórum de troca de ideias sobre a atividade.

III.5. Observações Finais

Lembre-se que essa atividade é a referência para a última avaliação. O conteúdo da terceira prova será exatamente as funções que povoam o módulo libsimcoss e a aplicação *procevfunc.py*.

Apesar da atividade ser opcional, construí-la é um investimento interessante pois auxilia no estudo para a última avaliação. Todas as funções pedidas são de construção simples e o resultado final (figura 10) é muito interessante. Além de ser um conhecimento novo interessante para se implementar em um sistema de informação.

Utilize o fórum da atividade para se comunicar com o professor e trocar ideias com os colegas.

Siga a risca as especificações desse documento quando for construir as funções pedidas.

Contacte o professor via fórum para comunicar algum erro de digitação nesse documento, assim como eventuais falhas nas bases de dados fornecidas pelo professor.

Bom trabalho!

Fim