Cholesky

June 19, 2019

In [0]:
```
import numpy as np
import pandas as pd

def Cholesky(n, A):

    L = np.zeros((n,n))

    L[0][0] = np.sqrt(A[0][0]) #definindo primeiro elemento da matriz


    for i in range(1, n): #definindo primeira coluna da matriz
      L[i][0]=A[i][0]/L[0][0]


    for i in range(1, n-1): #definindo as demais colunas da matriz

      L[i][i] = A[i][i] #definindo cada um dos elementos da diagonal
      for k in range (i):
        L[i][i]= L[i][i] - (L[i][k])**2
      L[i][i] = np.sqrt(L[i][i])

      for j in range (i+1, n): #definindo os elementos abaixo da diagonal
        L[j][i]=A[j][i]
        for k in range (i):
          L[j][i]= L[j][i] - L[j][k]*L[i][k]
        L[j][i]= L[j][i]/L[i][i]


    L[n-1][n-1] = A[n-1][n-1] #definindo o último elemento da matriz
    for k in range (0, n-1):
      L[n-1][n-1] = L[n-1][n-1] - (L[n-1][k])**2


    L[n-1][n-1] = np.sqrt(L[n-1][n-1])


    T = np.transpose(L)
```

```
            print(L)
            print(T)

            return L, T
```

```
In [0]: def invCholesky(n,L, T):


          Y = np.zeros((n,n))
          V = np.zeros((n,n)) #inversa do A
          I = np.identity(n)


          #Resolver o sistema AV=I -> LTV=I -> LX=I e TV=X

          for i in range (0,n): #percorrer colunas do Y e I
                Y[i] = np.linalg.solve(L,I[i])


          for i in range (0,n):   #percorrer colunas do V e Y
                V[i] = np.linalg.solve(T,Y[i])


          print (V)

          return V
```

```
In [0]: def getHilbert(n):

            A = np.zeros([n,n])

            for i in range(n):
                for j in range(n):
                    A[i][j] = 1/(i+j+1)

            return A
```

```
In [0]: A = [[1,1,2],
             [1,3,3],
             [2,3,6]] #matrizinha pra testes
```

```
In [0]: L,T = Cholesky(3,A)
```

```
[[1.         0.         0.        ]
 [1.         1.41421356 0.        ]
 [2.         0.70710678 1.22474487]]
[[1.         1.         2.        ]
 [0.         1.41421356 0.70710678]
```

```
 [0.         0.         1.22474487]]
```

```
In [0]: A = np.dot(L,T)
```

```
In [0]: A = getHilbert(3)
        L,T = Cholesky(3,A)
        A=np.dot(L,T)
        print (A)
```

```
[[1.         0.         0.        ]
 [0.5        0.28867513 0.        ]
 [0.33333333 0.28867513 0.0745356 ]]
[[1.         0.5        0.33333333]
 [0.         0.28867513 0.28867513]
 [0.         0.         0.0745356 ]]
[[1.         0.5        0.33333333]
 [0.5        0.33333333 0.25      ]
 [0.33333333 0.25       0.2       ]]
```

```
In [0]:   V = invCholesky(3,L, T)
```

```
[[   9.  -36.   30.]
 [ -36.  192. -180.]
 [  30. -180.  180.]]
```

```
In [0]: P = np.dot(V,A)
        print (P)
```

```
[[ 1.00000000e+00  0.00000000e+00  3.33066907e-16]
 [-1.40628250e-15  1.00000000e+00  2.26485497e-15]
 [-3.33066907e-15  0.00000000e+00  1.00000000e+00]]
```

```
In [0]: def norma_mat1(A):

            A = np.array(A)



            if((A.shape == (int(A.shape[0]),)) | (A.shape[0] == 1)):

                return sum([np.abs(v) for v in A])

            c = []
```

```
            for i in range(A.shape[0]):
                c.append(sum([np.abs(v) for v in A[:,i]]))


            return np.max(c)
```

```
In [0]: K1 = norma_mat1(V)*norma_mat1(A)
        K1

        n=3
        for k in range(n):
            for j in range(n):
                if(k!=j):
                    ni = norma_mat1(A[:,k])
                    nj = norma_mat1(A[:,j])

                    cond = ni/nj

                    if(K1 > cond):
                        print(f"\nK1 é maior que a coluna {k} pela coluna {j}, K1 vale {K1.rou
                    else:
                        print("nop")
```

```
K1 é maior que a coluna 0 pela coluna 1, K1 vale 748.0 e cond vale 1.6923


K1 é maior que a coluna 0 pela coluna 2, K1 vale 748.0 e cond vale 2.3404


K1 é maior que a coluna 1 pela coluna 0, K1 vale 748.0 e cond vale 0.5909


K1 é maior que a coluna 1 pela coluna 2, K1 vale 748.0 e cond vale 1.383


K1 é maior que a coluna 2 pela coluna 0, K1 vale 748.0 e cond vale 0.4273


K1 é maior que a coluna 2 pela coluna 1, K1 vale 748.0 e cond vale 0.7231
```

```
In [0]:
```