Universidad Nacional Autónoma de México

Inteligencia Artificial

Mediciones de Distancia

Grupo 3

Nombre:
Barreiro Valdez Alejandro

Práctica 3

Profesor: Dr. Guillermo Gilberto Molero Castillo

10 de marzo de 2022

Introducción

En esta práctica se utilizará el módulo cdist para calcular los diferentes tipos de métricas de distancia utilizados en el Machine Learning. Para cada una de las métricas de distancia se calculará la matriz distancia entre toda la población, un sector de la población y la distancia entre dos objetos. Se compararán dichas distancias para reforzar los conocimientos vistos en teoría sobre el cálculo que cada una de las métricas realiza.

Objetivo

Obtener las matrices de distancia (Euclidiana, Chebyshev, Manhattan, Minkowski) a partir de una matriz de datos.

Desarrollo

La primera matriz de distancia que se genera es Euclidiana y esto se genera con el módulo de cdist. De parámetros se pasan los datos y la opción de 'euclidean'. A partir de estas distancias se genera un Data Frame. Dicha matriz es de dimensión de 202 columnas por 202 filas justo como se esperaba.

```
DstEuclidiana = cdist(Hipoteca, Hipoteca, metric='euclidean')
MEuclidiana = pd.DataFrame(DstEuclidiana)
```

```
print(MEuclidiana)
#MEuclidiana
```

```
                0              1              2              3    \
0        0.000000  236994.701964   78577.840350  260974.591407
1    236994.701964       0.000000  315439.176808   26550.527773
2     78577.840350  315439.176808       0.000000  339168.030097
3    260974.591407   26550.527773  339168.030097       0.000000
4     51769.581416  287970.807817   31494.808048  312273.311450
..             ...            ...            ...            ...
197   53923.596347  275716.907131   62456.926862  301032.758046
198  122858.123985  357126.266127   54616.719848  381921.267013
199   18967.999420  249015.957900   69848.439181  273593.155481
200   37975.571227  261065.405879   66722.600009  286156.617026
201  147421.532182  380612.957023   78717.767975  405600.560294
```

A partir de aquí se generó la distancia Euclidiana entre objetos. Se puede realizar el cálculo de toda la población, de un segmento o de dos objetos. Primero, se realizó la distancia Euclidiana de toda la población y se imprimió el resultado. El resultado

completo no se puede ver porque son 202 filas y 202 columnas. Esto genera la distancia que tienen todos los vectores entre sí.

```
DstEuclidiana = cdist(Hipoteca, Hipoteca, metric='euclidean')
MEuclidiana = pd.DataFrame(DstEuclidiana)
```

```
print(MEuclidiana)
#MEuclidiana
```

```
                0              1              2              3    \
0        0.000000  236994.701964   78577.840350  260974.591407
1   236994.701964       0.000000  315439.176808   26550.527773
2    78577.840350  315439.176808       0.000000  339168.030097
3   260974.591407   26550.527773  339168.030097       0.000000
4    51769.581416  287970.807817   31494.808048  312273.311450
..            ...            ...            ...            ...
197  53923.596347  275716.907131   62456.926862  301032.758046
198 122858.123985  357126.266127   54616.719848  381921.267013
199  18967.999420  249015.957900   69848.439181  273593.155481
200  37975.571227  261065.405879   66722.600009  286156.617026
201 147421.532182  380612.957023   78717.767975  405600.560294

                4              5              6              7    \
0    51769.581416   39149.060512   30003.797860  206425.706195
1   287970.807817  276141.622437  207115.404780   33742.472390
2    31494.808048   39645.760694  108564.128321  284512.007926
3   312273.311450  300095.494243  231251.131504   54727.355591
4        0.000000   15176.476831   81054.392885  257851.794851
..            ...            ...            ...            ...
197  33981.853349   40260.802178   75868.017471  247632.363897
198  71302.672762   85526.168533  151476.305814  327814.188465
199  39655.591686   31283.565350   43417.279394  219384.112141
200  35401.101452   34884.353785   59413.858484  232457.935674
201  96032.256950  110377.422284  175732.010638  351670.615456
```

Después, se seleccionó una sola porción de los datos, 10 filas, para generar las distancias que existen entre estas filas. Para ello también se imprime una matriz y se utiliza cdist.

```
DstEuclidiana = cdist(Hipoteca.iloc[0:10], Hipoteca.iloc[0:10], metric='euclidean')
MEuclidiana = pd.DataFrame(DstEuclidiana)
print(MEuclidiana.round(3))
```

```
         0          1          2          3          4          5  \
0    0.000  236994.702   78577.840  260974.591   51769.581   39149.061
1  236994.702      0.000  315439.177   26550.528  287970.808  276141.622
2   78577.840  315439.177      0.000  339168.030   31494.808   39645.761
3  260974.591   26550.528  339168.030      0.000  312273.311  300095.494
4   51769.581  287970.808   31494.808  312273.311      0.000   15176.477
5   39149.061  276141.622   39645.761  300095.494   15176.477       0.000
6   30003.798  207115.405  108564.128  231251.132   81054.393   69082.894
7  206425.706   33742.472  284512.008   54727.356  257851.795  245517.453
8  108991.941  345963.774   31548.759  369945.815   58617.026   69857.764
9   76488.543  312810.380   17030.195  337121.576   24868.540   38195.246

         6          7          8          9
0   30003.798  206425.706  108991.941   76488.543
1  207115.405   33742.472  345963.774  312810.380
2  108564.128  284512.008   31548.759   17030.195
3  231251.132   54727.356  369945.815  337121.576
4   81054.393  257851.795   58617.026   24868.540
5   69082.894  245517.453   69857.764   38195.246
6       0.000  176803.205  138853.961  105892.924
7  176803.205       0.000  315357.551  282695.457
8  138853.961  315357.551       0.000   34544.425
9  105892.924  282695.457   34544.425       0.000
```

Por último, se obtuvo la distancia entre dos objetos utilizando cdist.

```
Objeto1 = Hipoteca.iloc[0]
Objeto2 = Hipoteca.iloc[1]
dstEuclidiana = distance.euclidean(Objeto1,Objeto2)
dstEuclidiana
```

```
236994.70196398906
```

Para Chebyshev se tiene un proceso similar, solo cambia un parámetro en cdist para indicar que se quiere este tipo de distancia. Primero se obtiene la distancia de toda la población.

```
DstChebyshev = cdist(Hipoteca, Hipoteca, metric='chebyshev')
MChebyshev = pd.DataFrame(DstChebyshev)
```

```
print(MChebyshev)
```

```
          0         1         2         3         4         5         6    \
0         0.0  236897.0   78221.0  260933.0   51068.0   39137.0   29812.0
1    236897.0       0.0  315118.0   24036.0  287965.0  276034.0  207085.0
2     78221.0  315118.0       0.0  339154.0   27153.0   39084.0  108033.0
3    260933.0   24036.0  339154.0       0.0  312001.0  300070.0  231121.0
4     51068.0  287965.0   27153.0  312001.0       0.0   11931.0   80880.0
..        ...       ...       ...       ...       ...       ...       ...
197   39277.0  273777.0   46740.0  297813.0   30789.0   40152.0   66692.0
198  119579.0  356476.0   41358.0  380512.0   68511.0   80442.0  149391.0
199   14435.0  248872.0   66246.0  272908.0   39093.0   27162.0   41787.0
200   30015.0  260005.0   55113.0  284041.0   27960.0   30890.0   52920.0
201  142420.0  379317.0   64199.0  403353.0   91352.0  103283.0  172232.0

          7         8         9    ...       192       193       194  \
0    206291.0  108990.0   75902.0  ...   33048.0   95888.0  110795.0
1     30606.0  345887.0  312799.0  ...  243940.0  332785.0  347692.0
2    284512.0   30769.0   16852.0  ...   71178.0   22862.0   32574.0
3     54642.0  369923.0  336835.0  ...  267976.0  356821.0  371728.0
4    257359.0   57922.0   24834.0  ...   44025.0   44820.0   59727.0
..        ...       ...       ...  ...       ...       ...       ...
197  243171.0   72110.0   39022.0  ...   29837.0   59008.0   73915.0
198  325870.0   28623.0   43677.0  ...  112536.0   23691.0    8784.0
199  218266.0   97015.0   63927.0  ...   18613.0   83913.0   98820.0
200  229399.0   85882.0   52794.0  ...   16065.0   72780.0   87687.0
201  348711.0   38523.0   66518.0  ...  135377.0   46532.0   31625.0
```

Después se obtuvo la distancia Chebyshev de diez filas de la población.

```
DstChebyshev = cdist(Hipoteca.iloc[0:10], Hipoteca.iloc[0:10], metric='chebyshev')
MChebyshev = pd.DataFrame(DstChebyshev)
print(MChebyshev)
```

```
          0         1         2         3         4         5         6  \
0       0.0  236897.0   78221.0  260933.0   51068.0   39137.0   29812.0
1  236897.0       0.0  315118.0   24036.0  287965.0  276034.0  207085.0
2   78221.0  315118.0       0.0  339154.0   27153.0   39084.0  108033.0
3  260933.0   24036.0  339154.0       0.0  312001.0  300070.0  231121.0
4   51068.0  287965.0   27153.0  312001.0       0.0   11931.0   80880.0
5   39137.0  276034.0   39084.0  300070.0   11931.0       0.0   68949.0
6   29812.0  207085.0  108033.0  231121.0   80880.0   68949.0       0.0
7  206291.0   30606.0  284512.0   54642.0  257359.0  245428.0  176479.0
8  108990.0  345887.0   30769.0  369923.0   57922.0   69853.0  138802.0
9   75902.0  312799.0   16852.0  336835.0   24834.0   36765.0  105714.0

          7         8         9
0  206291.0  108990.0   75902.0
1   30606.0  345887.0  312799.0
2  284512.0   30769.0   16852.0
3   54642.0  369923.0  336835.0
4  257359.0   57922.0   24834.0
5  245428.0   69853.0   36765.0
6  176479.0  138802.0  105714.0
7       0.0  315281.0  282193.0
8  315281.0       0.0   33088.0
9  282193.0   33088.0       0.0
```

Por último se obtuvo la distancia Chebyshev entre dos objetos.

```
Objeto1 = Hipoteca.iloc[0]
Objeto2 = Hipoteca.iloc[1]
dstChebyshev = distance.chebyshev(Objeto1,Objeto2)
dstChebyshev
```

```
236897
```

La siguiente distancia que se obtuvo de manera análoga a las enteriores fue la de Manhattan. Para esta distancia se define el parámetro de cdist como 'cityblock'. Primero se obtuvo la distancia Manhattan entre toda la población de los datos.

```
DstManhattan = cdist(Hipoteca, Hipoteca, metric='cityblock')
MManhattan = pd.DataFrame(DstManhattan)
```

```
print(MManhattan)
```

```
          0         1         2         3         4         5         6    \
0       0.0  244759.0   86474.0  267180.0   60166.0   40701.0   34820.0
1  244759.0       0.0  330117.0   36279.0  290551.0  284974.0  211391.0
2   86474.0  330117.0       0.0  343632.0   43970.0   47121.0  120112.0
3  267180.0   36279.0  343632.0       0.0  326816.0  305557.0  239544.0
4   60166.0  290551.0   43970.0  326816.0       0.0   22231.0   87564.0
..      ...       ...       ...       ...       ...       ...       ...
197 79103.0  309758.0   91619.0  346023.0   47649.0   45004.0  106529.0
198 150173.0 380902.0   79933.0  417009.0   90619.0  111702.0  177657.0
199  29640.0 260529.0   91786.0  296786.0   48342.0   45653.0   57422.0
200  56348.0 287219.0   96298.0  323494.0   52608.0   50009.0   83992.0
201 182937.0 413618.0  112701.0  449329.0  123615.0  144286.0  210313.0

          7         8         9    ...       192       193       194  \
0   214460.0  110235.0   87151.0  ...   42016.0  114232.0  134639.0
1    45617.0  354186.0  316302.0  ...  273005.0  345071.0  365610.0
2   284636.0   38493.0   20633.0  ...  113950.0   44058.0   64475.0
3    59000.0  375313.0  351115.0  ...  309110.0  381346.0  401785.0
4   274210.0   67449.0   27261.0  ...   70668.0   54626.0   75273.0
..       ...       ...       ...  ...       ...       ...       ...
197 293479.0  115098.0   72986.0  ...   37533.0   83779.0   94204.0
198 364493.0   41890.0   65914.0  ...  118577.0   37321.0   18312.0
199 243852.0  115505.0   73547.0  ...   25282.0   85344.0  105541.0
200 270624.0  119947.0   77993.0  ...   20708.0   87710.0   98305.0
201 397243.0   74604.0   98540.0  ...  141811.0   70323.0   51138.0
```

Se obtuvo la distancia Manhattan entre diez filas de los datos.

```
DstManhattan = cdist(Hipoteca.iloc[0:10], Hipoteca.iloc[0:10], metric='cityblock')
MManhattan = pd.DataFrame(DstManhattan)
print(MManhattan)
```

```
         0         1         2         3         4         5         6  \
0      0.0  244759.0   86474.0  267180.0   60166.0   40701.0   34820.0
1 244759.0       0.0  330117.0   36279.0  290551.0  284974.0  211391.0
2  86474.0  330117.0       0.0  343632.0   43970.0   47121.0  120112.0
3 267180.0   36279.0  343632.0       0.0  326816.0  305557.0  239544.0
4  60166.0  290551.0   43970.0  326816.0       0.0   22231.0   87564.0
5  40701.0  284974.0   47121.0  305557.0   22231.0       0.0   74941.0
6  34820.0  211391.0  120112.0  239544.0   87564.0   74941.0       0.0
7 214460.0   45617.0  284636.0   59000.0  274210.0  253389.0  188566.0
8 110235.0  354186.0   38493.0  375313.0   67449.0   71574.0  143573.0
9  87151.0  316302.0   20633.0  351115.0   27261.0   48998.0  112221.0

         7         8         9
0 214460.0  110235.0   87151.0
1  45617.0  354186.0  316302.0
2 284636.0   38493.0   20633.0
3  59000.0  375313.0  351115.0
4 274210.0   67449.0   27261.0
5 253389.0   71574.0   48998.0
6 188566.0  143573.0  112221.0
7      0.0  323035.0  300521.0
8 323035.0       0.0   44100.0
9 300521.0   44100.0       0.0
```

Por último se calculó la distancia Manhattan entre dos objetos.

```
Objeto1 = Hipoteca.iloc[0]
Objeto2 = Hipoteca.iloc[1]
dstManhattan = distance.cityblock(Objeto1,Objeto2)
dstManhattan
```

```
244759
```

Para Minkowski se define un Lambda como parámetro de lo recto o curvo que debe ser la ruta para la distancia. Para esta práctica se define a Lambda como 1.5 en el algoritmo, este parámetro corresponde a p. Se realizan los mismos pasos que en el cálculo de distancias anteriores. Se calculó la distancia entre todos los datos de la población.

```
DstMinkowski = cdist(Hipoteca, Hipoteca, metric='minkowski', p=1.5)
MMinkowski = pd.DataFrame(DstMinkowski)
```

```
print(MMinkowski)
```

```
                0              1              2              3    \
0        0.000000  237690.995925   79782.466760  261389.573558
1   237690.995925       0.000000  317144.541987   28999.550044
2    79782.466760  317144.541987       0.000000  339376.378955
3   261389.573558   28999.550044  339376.378955       0.000000
4    53372.216100  288074.733923   34836.105302  313818.956903
..            ...            ...            ...            ...
197  60770.233816  281405.644842   70353.660441  309154.836773
198 128687.635109  360119.702102   61510.907561  387055.019657
199  21714.620373  250061.119850   74860.624904  276559.126021
200  42815.775409  264889.398939   74602.554581  292321.617039
201 155395.390030  385435.511309   87986.061870  412690.548292

                4              5              6              7    \
0    53372.216100   39260.690697   30673.683784  207250.873149
1   288074.733923  276926.258979  207408.636739   36799.022688
2    34836.105302   40977.188827  110319.616169  284512.877279
3   313818.956903  300409.654429  232080.294553   55184.304245
4        0.000000   17046.898384   81840.258784  260012.361790
..            ...            ...            ...            ...
197  37318.860168   40811.991276   83771.074585  256837.573004
198  75607.413634   91839.027927  156357.233170  333809.911618
199  41172.682830   34857.378658   46295.299016  223131.869044
200  39959.337646   38607.596589   65536.701429  239632.877207
201 102457.030136  118784.257654  182746.119425  359717.588847
```

Se generó una matriz de distancias Minkowski de 10 filas de los datos.

```
DstMinkowski = cdist(Hipoteca.iloc[0:10], Hipoteca.iloc[0:10], metric='minkowski', p=1.5)
MMinkowski = pd.DataFrame(DstMinkowski)
print(MMinkowski)
```

```
              0             1             2             3             4  \
0      0.000000  237690.995925   79782.466760  261389.573558   53372.216100
1 237690.995925       0.000000  317144.541987   28999.550044  288074.733923
2  79782.466760  317144.541987       0.000000  339376.378955   34836.105302
3 261389.573558   28999.550044  339376.378955       0.000000  313818.956903
4  53372.216100  288074.733923   34836.105302  313818.956903       0.000000
5  39260.690697  276926.258979   40977.188827  300409.654429   17046.898384
6  30673.683784  207408.636739  110319.616169  232080.294553   81840.258784
7 207250.873149   36799.022688  284512.877279   55184.304245  260012.361790
8 109035.213044  346609.614856   32977.126225  370236.872408   60284.016224
9  78197.161473  312975.503513   17574.226078  338719.479124   25100.249754

              5             6             7             8             9
0  39260.690697   30673.683784  207250.873149  109035.213044   78197.161473
1 276926.258979  207408.636739   36799.022688  346609.614856  312975.503513
2  40977.188827  110319.616169  284512.877279   32977.126225   17574.226078
3 300409.654429  232080.294553   55184.304245  370236.872408  338719.479124
4  17046.898384   81840.258784  260012.361790   60284.016224   25100.249754
5      0.000000   69749.100955  246187.491676   69936.944305   40487.806354
6  69749.100955       0.000000  178267.963453  139247.210167  106708.022220
7 246187.491676  178267.963453       0.000000  315980.319533  284964.264428
8  69936.944305  139247.210167  315980.319533       0.000000   36693.205417
9  40487.806354  106708.022220  284964.264428   36693.205417       0.000000
```

Por último, se calculó la distancia Minkowski entre dos objetos.

```
Objeto1 = Hipoteca.iloc[0]
Objeto2 = Hipoteca.iloc[1]
dstMinkowski = distance.minkowski(Objeto1,Objeto2, p=1.5)
dstMinkowski
```

```
237690.9959246789
```

Se generó una versión normalizada de esta práctica para tener facilidad para manejar los datos. La normalización consiste en estandarizar la escala que tienen los datos. De esta manera, cuando un algoritmo opera sobre dichos datos se tiene una mayor facilidad de las operaciones. Se le da a todas las variables el mismo peso para cuando se deba trabajar con un algoritmo que tome dos o más variables. Se normalizó utilizando la biblioteca de sklearn donde solo se pasan los datos a trabajar y se obtiene una matriz estandarizada.

```
from sklearn.preprocessing import StandardScaler, MinMaxScaler
estandarizar = StandardScaler()                          # Se instancia el objeto StandardScaler o MinMaxScaler
MEstandarizada = estandarizar.fit_transform(Hipoteca)    # Se calculan la media y desviación y se escalan los d
```

```
pd.DataFrame(MEstandarizada)
```

	0	1	2	3	4	5	6	7	8	9
0	0.620129	0.104689	-1.698954	0.504359	0.649475	0.195910	-1.227088	0.562374	-0.984420	1.419481
1	1.063927	-0.101625	-0.712042	-0.515401	0.259224	1.937370	-0.029640	1.295273	0.596915	-0.704483
2	0.891173	0.226266	-0.912634	1.667244	1.080309	-0.379102	1.167809	-0.170526	1.387582	1.419481
3	1.274209	1.128886	-1.578599	-1.559015	0.909604	2.114062	-1.227088	-0.903426	-0.589086	-0.704483
4	0.719611	-0.400042	0.090326	0.027279	0.159468	-0.179497	-1.227088	-0.903426	-0.589086	1.419481
...	...	...	...	...	...	...	...	...	...	...
197	-0.671949	-1.037402	1.125381	-0.163554	-1.617963	-0.075199	-1.227088	-0.903426	-0.984420	-0.704483
198	-0.594508	0.215214	0.467439	-0.241079	-0.973876	-0.683130	1.167809	1.295273	1.387582	-0.704483
199	-1.057368	-0.061099	0.515581	1.005294	-0.183849	0.107880	-0.029640	1.295273	1.387582	-0.704483
200	-0.968013	-0.385305	1.261783	0.814462	-1.083273	0.026040	-0.029640	0.562374	0.201581	-0.704483
201	-0.578424	0.683102	-0.856468	-0.795686	-1.545397	-0.851037	-1.227088	-0.903426	-0.193753	-0.704483

Se compararon las cuatro métricas generadas entre la distancia de dos objetos.

Tipo de distancia	**Distancia Calculada**
Euclidiana	236994.70196398906
Chebyshev	236897
Manhattan	244759
Minkowski	237690.9959246789

La distancia de Manhattan es la mayor de todas y esto tiene mucho sentido ya que genera una distancia calculando el espacio como si fueran cuadras de una ciudad, no se genera una distancia en línea recta como la Euclidiana. Minkowski es mayor que la Euclidiana y menor que la de Manhattan porque es una combinación de ambas. Se genera una curva que hace que no sea directa como la Euclidiana pero tampoco se como una cuadrícula. La distancia de Chebyshev es ligeramente menor que la

Euclidiana y esto se puede explicar gracias a que no se toman todos los valores de las coordenadas que se tienen. Se toma el valor máximo pero esto puede generar que la distancia sea menor o mayor.

Conclusión

En esta práctica se pudo visualizar la manera en que se calculan distancias en un conjunto de datos utilizando el módulo de cdist. También, se pudo visualizar la manera en que se pueden seleccionar datos para generar una matriz de distancias o cómo se puede calcular la distancia entre dos objetos. Esta práctica será de gran utilidad para futuros algoritmos porque se utilizan las métricas de distancia para posteriores algoritmos. Se observaron las variaciones que existen entre las distancias. Para la distancia Euclidiana se calcula la distancia directa hacia el otro objeto. Para la distancia Manhattan se calcula como si fuera la cuadra de una ciudad. Para Minkowski se genera una curva que es el punto medio entre las dos métricas anteriores. Para Chebyshev se seleccionan solo las coordenadas máximas por lo que su tamaño puede variar. Con esta práctica se generó un dominio sobre el cálculo de distancias con diferentes métricas en Python.