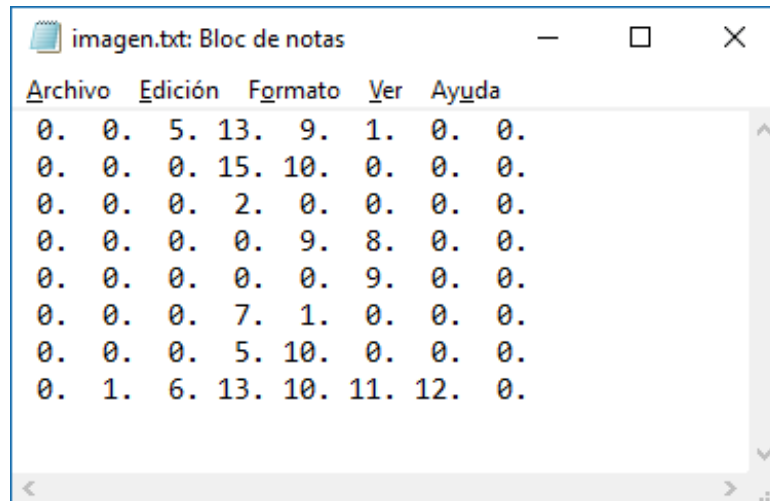


# Ejercicio - máquinas de soporte vectorial

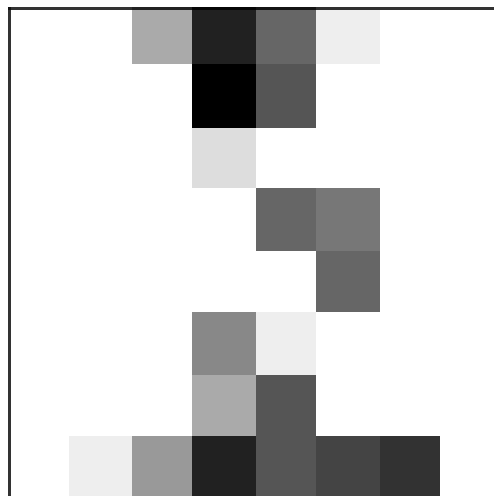
## Clasificación de una imagen

Considera una matriz como la siguiente, que se encuentra en el archivo "matriz.txt":



A	r	c	h	i	v	o	
0.	0.	5.	13.	9.	1.	0.	0.
0.	0.	0.	15.	10.	0.	0.	0.
0.	0.	0.	2.	0.	0.	0.	0.
0.	0.	0.	0.	9.	8.	0.	0.
0.	0.	0.	0.	0.	9.	0.	0.
0.	0.	0.	7.	1.	0.	0.	0.
0.	0.	0.	5.	10.	0.	0.	0.
0.	1.	6.	13.	10.	11.	12.	0.

Si interpretamos sus valores como intensidades de gris, esta imagen podría representar uno de los dígitos que tenemos en nuestra librería:



Quizás un 1 o un 2...

Lo que se pide en este ejercicio es, simplemente, usar un algoritmo de tipo svm para clasificarla, contrastándola con los dígitos de la librería de dígitos de sklearn.

Para ello, puedes completar el siguiente script:

```
import numpy as np

# Cargar la imagen del archivo (y comprobar la carga):
# ...

# Visualizarla, con ayuda de matplotlib.pyplot:
# ...

# Visualizarla, con ayuda de matplotlib.pyplot:
# ...

# Cargar la librería de imágenes de dígitos:

from sklearn import datasets
digits = datasets.load_digits()
# ...

# Cargar y el clasificador svm:
# ...

# y ajustarlo para precedir con todos los ejemplares como de entrenamiento:
# ...

# Efectuar la predicción:
# ...
```

Eso es lo que te pido.

In [1]:



```
# Cargar la imagen del archivo (y comprobar la carga):

import numpy as np

archivo_imagen = open("matriz.txt", "r")
mi_imagen = np.array([[float(e) for e in linea.split()] for linea in archivo_imagen])
archivo_imagen.close()

"""
La misma imagen, creada con un archivo:

mi_imagen = \
    np.array([[ 0.,  0.,  5., 13.,  9.,  1.,  0.,  0.],
               [ 0.,  0.,  0., 15., 10.,  0.,  0.,  0.],
               [ 0.,  0.,  0.,  2.,  0.,  0.,  0.,  0.],
               [ 0.,  0.,  0.,  0.,  9.,  8.,  0.,  0.],
               [ 0.,  0.,  0.,  0.,  0.,  9.,  0.,  0.],
               [ 0.,  0.,  0.,  7.,  1.,  0.,  0.,  0.],
               [ 0.,  0.,  0.,  5., 10.,  0.,  0.,  0.],
               [ 0.,  1.,  6., 13., 10., 11., 12.,  0.]])

"""

print(mi_imagen)

# Visualizar la imagen, con ayuda de matplotlib.pyplot:

import matplotlib.pyplot as plt
plt.figure(1, figsize=(3, 3))
plt.imshow(mi_imagen, cmap=plt.cm.gray_r, interpolation='nearest')
plt.show()
```

```
[[ 0.  0.  5. 13.  9.  1.  0.  0.]
 [ 0.  0.  0. 15. 10.  0.  0.  0.]
 [ 0.  0.  0.  2.  0.  0.  0.  0.]
 [ 0.  0.  0.  0.  9.  8.  0.  0.]
 [ 0.  0.  0.  0.  0.  9.  0.  0.]
 [ 0.  0.  0.  7.  1.  0.  0.  0.]
 [ 0.  0.  0.  5. 10.  0.  0.  0.]
 [ 0.  1.  6. 13. 10. 11. 12.  0.]
```

<Figure size 300x300 with 1 Axes>

In [2]:



```
# Cargar la librería de imágenes de dígitos:

from sklearn import datasets
digits = datasets.load_digits()

# Por si se desea comprobar la lectura:
# print(digits.data)
# print(digits.target)

# Cargar y el clasificador svm:

from sklearn import svm
clf = svm.SVC(gamma=0.001, C=100.)

# y ajustarlo para precedir con todos los ejemplares como de entrenamiento:
clf.fit(digits.data, digits.target)
```

Out[2]:

```
SVC(C=100.0, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

In [3]:



```
# Realizar la predicción:

mi_imagen_plana = mi_imagen.reshape(1, 64)
clf.predict(mi_imagen_plana)
```

Out[3]:

```
array([2])
```