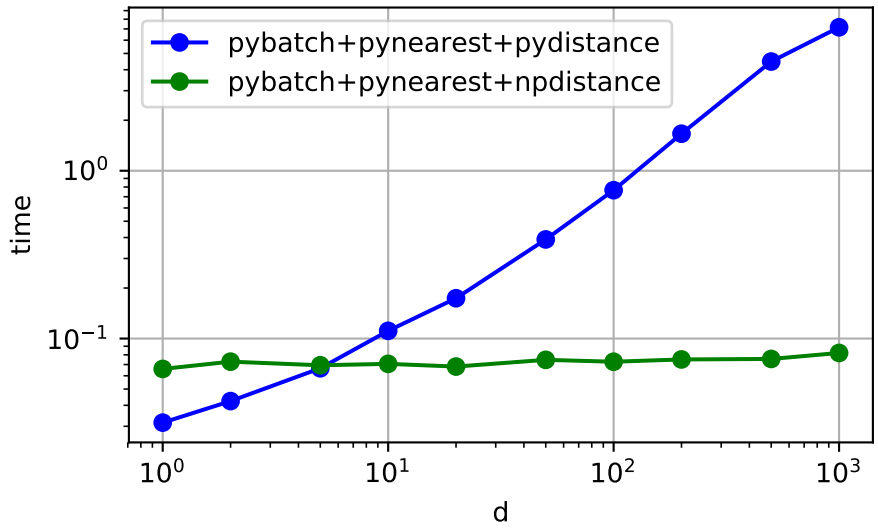
sheet2_fangyu

October 27, 2017

In [4]:
```
import matplotlib
from matplotlib import pyplot as plt
import time
import data
import ex2

def pydistance(x1,x2):
        return sum([(x1d-x2d)**2 for x1d,x2d in zip(x1,x2)])

def pynearest(u,X,Y,distance=pydistance):
        xbest = None
        ybest = None
        dbest = float('inf')
        for x,y in zip(X,Y):
                d = distance(u,x)
                if d < dbest:
                        ybest = y
                        xbest = x
                        dbest = d
        return ybest


def pybatch(U,X,Y,nearest=pynearest,distance=pydistance):
        return [nearest(u,X,Y,distance=distance) for u in U]

U,X,Y = data.toy(20,100,50)
```

In [5]:
```
U,X,Y = data.toy(20,100,50)
print(\
"----------------------------------------\n\
comparison to baseline implementation\n\
('pybatch+pynearest+pydistance', {})\n\
(pybatch+pynearest+npdistance, {})\n\
----------------------------------------".format(pybatch(U,X,Y), pybatch(U,X,Y, distanc
#print(pybatch(U,X,Y, nearest=ex2.npnearest))

%matplotlib inline
```

```
        from IPython.display import set_matplotlib_formats
        set_matplotlib_formats('pdf', 'png')
        plt.rcParams['savefig.dpi'] = 90

        # Values for the number of dimensions d to test
        dlist = [1, 2, 5, 10, 20, 50, 100, 200, 500, 1000]
        # Measure the computation time for each choice of number of dimensions d
        tlist = []
        for d in dlist:
                U,X,Y = data.toy(100,100,d)
                a = time.clock()
                pybatch(U,X,Y)
                b = time.clock()
                tlist += [b-a]

        nptlist = []
        for d in dlist:
                U,X,Y = data.toy(100,100,d)
                a = time.clock()
                pybatch(U,X,Y,distance=ex2.npdistance)
                b = time.clock()
                nptlist += [b-a]

        # Plot the results in a graph
        plt.figure(figsize=(5,3))
        plt.plot(dlist, tlist,'-o', color='blue', label='pybatch+pynearest+pydistance')
        plt.plot(dlist, nptlist,'-o', color='green', label='pybatch+pynearest+npdistance')
        plt.xscale('log');plt.yscale('log'); plt.xlabel('d'); plt.ylabel('time'); plt.grid(Tru
        plt.legend()

----------------------------------------
comparison to baseline implementation
('pybatch+pynearest+pydistance', [1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0])
(pybatch+pynearest+npdistance, [1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0])
----------------------------------------


Out[5]: <matplotlib.legend.Legend at 0x117221550>
```

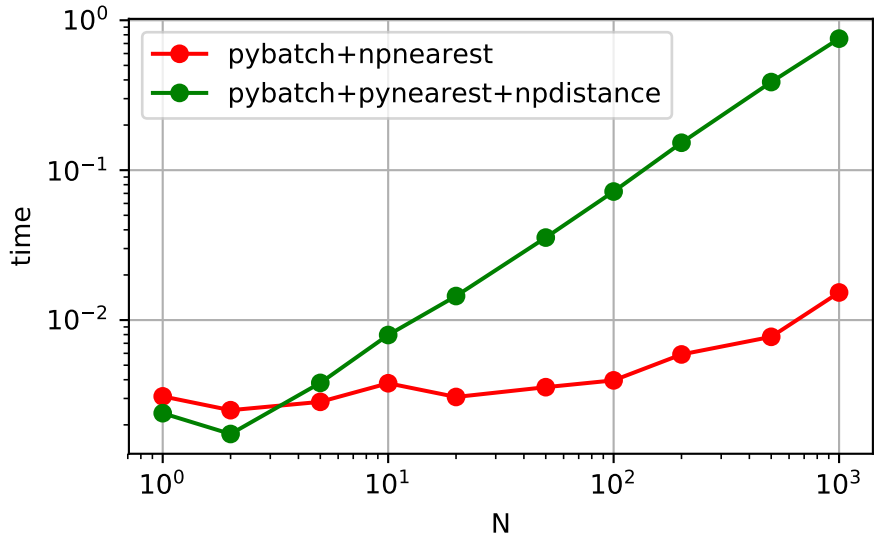



```
In [6]: U,X,Y = data.toy(20,100,50)
        print(\
        "----------------------------------------\n\
        comparison to baseline implementation\n\
        ('pybatch+pynearest+npdistance', {})\n\
        (pybatch+npnearest, {})\n\
        ----------------------------------------"\
              .format(pybatch(U,X,Y, distance=ex2.npdistance), pybatch(U,X,Y,nearest=ex2.npnear

        nptlist = []
        for n in dlist:
                U,X,Y = data.toy(100,n,100)
                a = time.clock()
                pybatch(U,X,Y,distance=ex2.npdistance)
                b = time.clock()
                nptlist += [b-a]

        npnn_tlist = []
        for n in dlist:
                U,X,Y = data.toy(100,n,100)
                a = time.clock()
                pybatch(U,X,Y,nearest=ex2.npnearest, distance=ex2.npdistance)
                b = time.clock()
                npnn_tlist += [b-a]

        # Plot the results in a graph
        plt.figure(figsize=(5,3))
        #plt.plot(dlist, tlist,'-o', color='blue', label='pybatch+pynearest+pydistance')
```

```
        plt.plot(dlist, npnn_tlist,'-o', color='red', label='pybatch+npnearest')
        plt.plot(dlist, nptlist,'-o', color='green', label='pybatch+pynearest+npdistance')
        plt.xscale('log');plt.yscale('log'); plt.xlabel('N'); plt.ylabel('time'); plt.grid(Tru
        plt.legend()
```

```
----------------------------------------
comparison to baseline implementation
('pybatch+pynearest+npdistance', [1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0])
(pybatch+npnearest, [1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0])
----------------------------------------
```

Out[6]: <matplotlib.legend.Legend at 0x1a1d42e358>

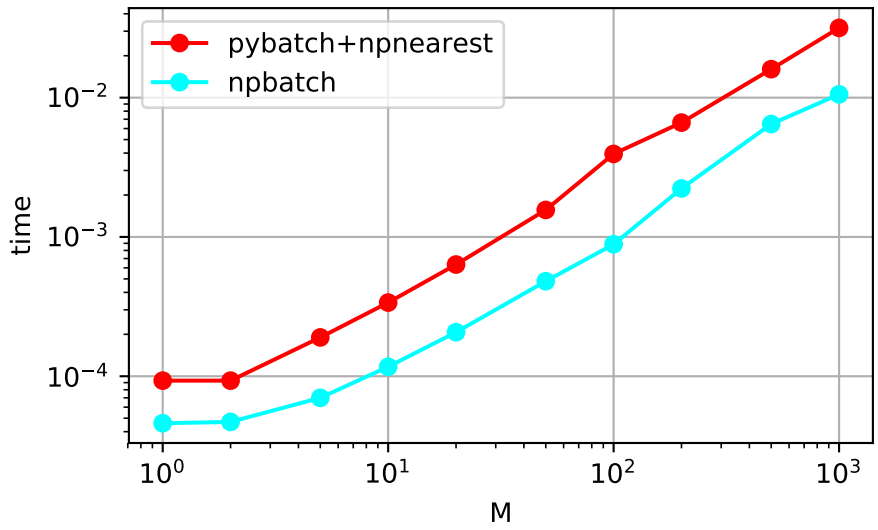


In [7]:
```
U,X,Y = data.toy(20,100,50)
print(\
"----------------------------------------\n\
comparison to baseline implementation\n\
('pybatch+pynearest+npdistance', {})\n\
(npbatch, {})\n\
----------------------------------------"\
      .format(pybatch(U,X,Y,nearest=ex2.npnearest, distance=ex2.npdistance), ex2.npbat

npnn_tlist = []
for m in dlist:
        U,X,Y = data.toy(m,100,100)
        a = time.clock()
        pybatch(U,X,Y,nearest=ex2.npnearest, distance=ex2.npdistance)
        b = time.clock()
```

```
            npnn_tlist += [b-a]

        npbatch_tlist = []
        for m in dlist:
            U,X,Y = data.toy(m,100,100)
            a = time.clock()
            ex2.npbatch(U,X,Y)
            b = time.clock()
            npbatch_tlist += [b-a]

        # Plot the results in a graph
        plt.figure(figsize=(5,3))
        #plt.plot(dlist, tlist,'-o', color='blue', label='pybatch+pynearest+pydistance')
        plt.plot(dlist, npnn_tlist,'-o', color='red', label='pybatch+npnearest')
        plt.plot(dlist, npbatch_tlist,'-o', color='cyan', label='npbatch')
        plt.xscale('log');plt.yscale('log'); plt.xlabel('M'); plt.ylabel('time'); plt.grid(Tru
        plt.legend()
```

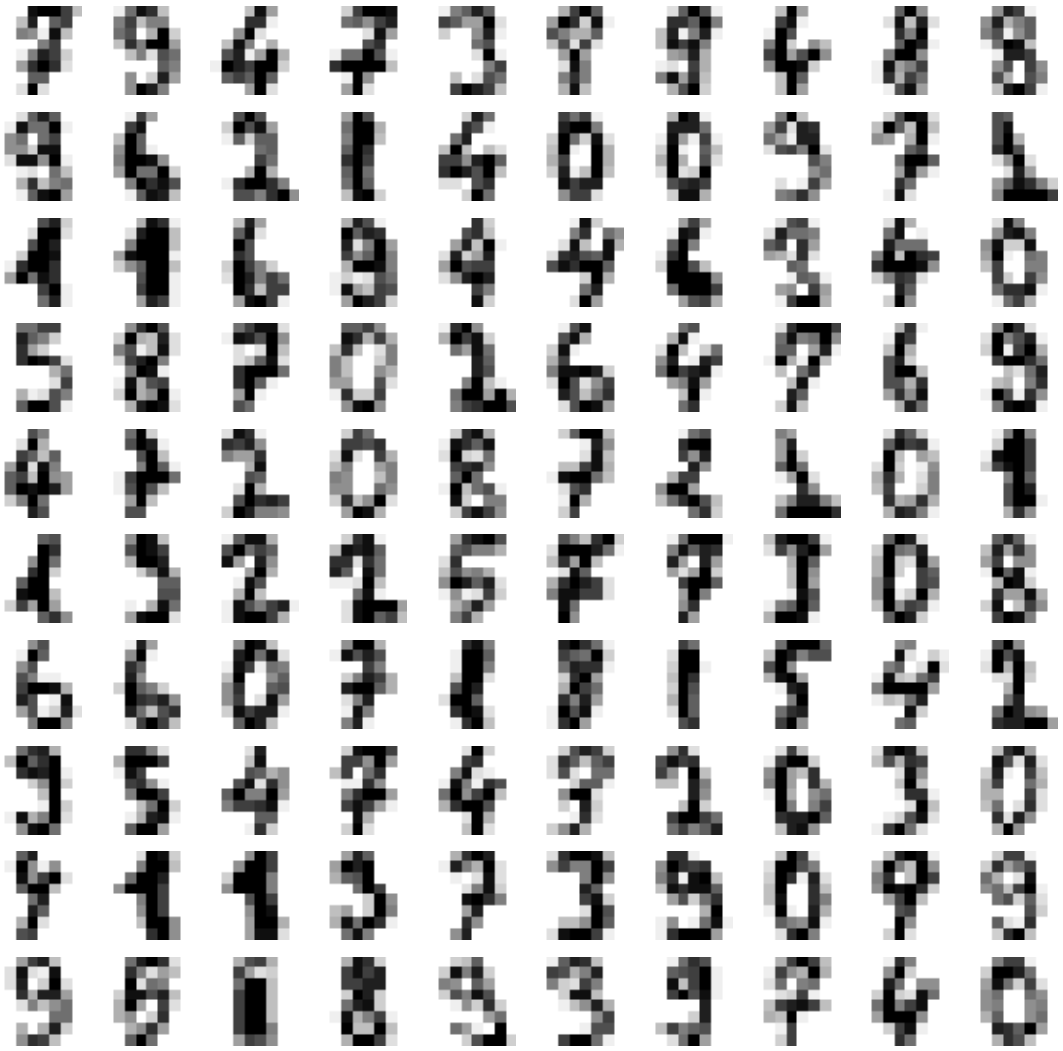
```
----------------------------------------
comparison to baseline implementation
('pybatch+pynearest+npdistance', [1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0])
(npbatch, [1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0])
----------------------------------------
```

Out[7]: <matplotlib.legend.Legend at 0x1a1d7dd2e8>



In [18]:
```
#exercise 4a
X, Y, I = data.digits()
```

```
plt.figure(figsize=(10,10))
for i in range(100):
    #plt.subplot(nrows, ncols, subplot_index<start from 1>: nrows&ncols mean the figu
    plt.subplot(10,10,i+1)
    plt.axis('off')
    plt.imshow(I[i], cmap=plt.cm.gray_r, interpolation='nearest')
```



In [22]:
```
#exercise 4b
training_set = X[0:1000]
training_label = Y[0:1000]
test_set = X[1000:]
test_label_real = Y[1000:] #ground truth
test_label = ex2.npbatch(test_set, training_set, training_label)
print(test_label)
```

```
[0, 7, 3, 5, 9, 4, 7, 2, 5, 6, 1, 2, 7, 0, 0, 6, 2, 2, 4, 4, 3, 4, 0, 2, 7, 9, 1, 4, 4, 4, 9, 4
```

In [28]:
```
#exercise 4c
from functools import reduce
diff = reduce(lambda C,x: C+1 if x[0]!=x[1] else C, zip(test_label_real, test_label),
ans = diff / len(test_label_real)
print(ans)
```

```
0.00878293601003764
```