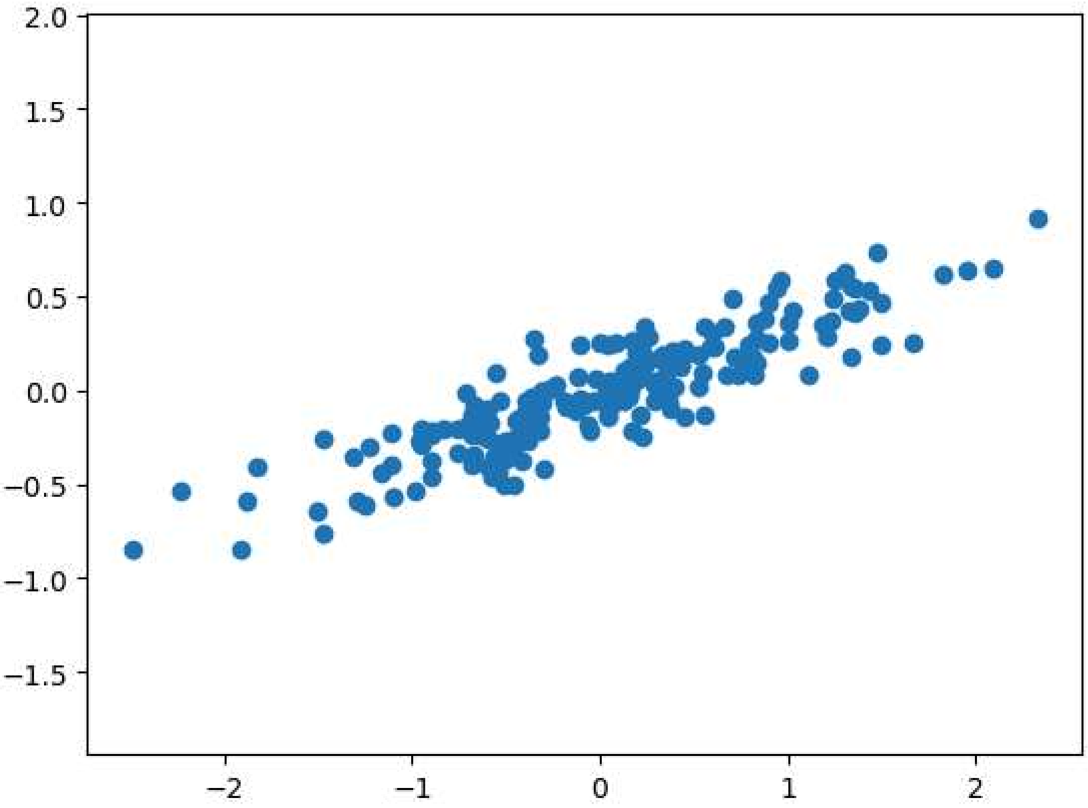
In [1]:

```
import sklearn
import numpy as np
import os
%matplotlib inline
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:

```
rng = np.random.RandomState(1)
X = np.dot(rng.rand(2,2), rng.randn(2,200)).T
plt.scatter(X[:,0],X[:,1])
plt.axis('equal');
```



In [10]:

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca.fit(X)
```

Out[10]:

```
PCA(n_components=2)
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [6]:

```
print (pca.components_)
```

```
[[-0.94446029 -0.32862557]
 [-0.32862557  0.94446029]]
```

In [7]:

```
print (pca.explained_variance_)
```

```
[0.7625315 0.0184779]
```

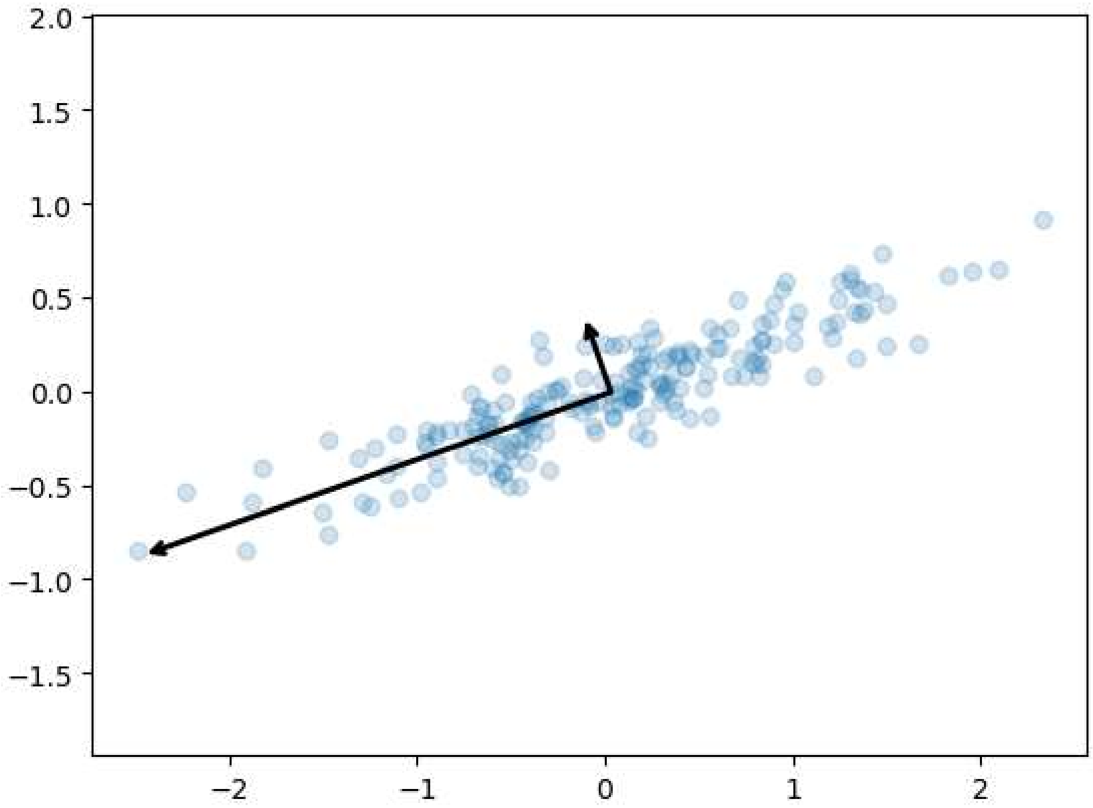
In [9]:

```
def draw_vector(v0, v1, ax=None):
    ax = ax or plt.gca()
    arrowprops=dict(arrowstyle='->',
                    linewidth=2,
                    shrinkA=0, shrinkB=0)
    ax.annotate('',v1, v0, arrowprops=arrowprops)
```

In [14]:

```
#plt data

plt.scatter(X[:,0], X[:,1], alpha=0.2)
for length, vector in zip(pca.explained_variance_, pca.components_):
    v = vector * 3 * np.sqrt(length)
    draw_vector(pca.mean_, pca.mean_ + v)
plt.axis('equal');
```



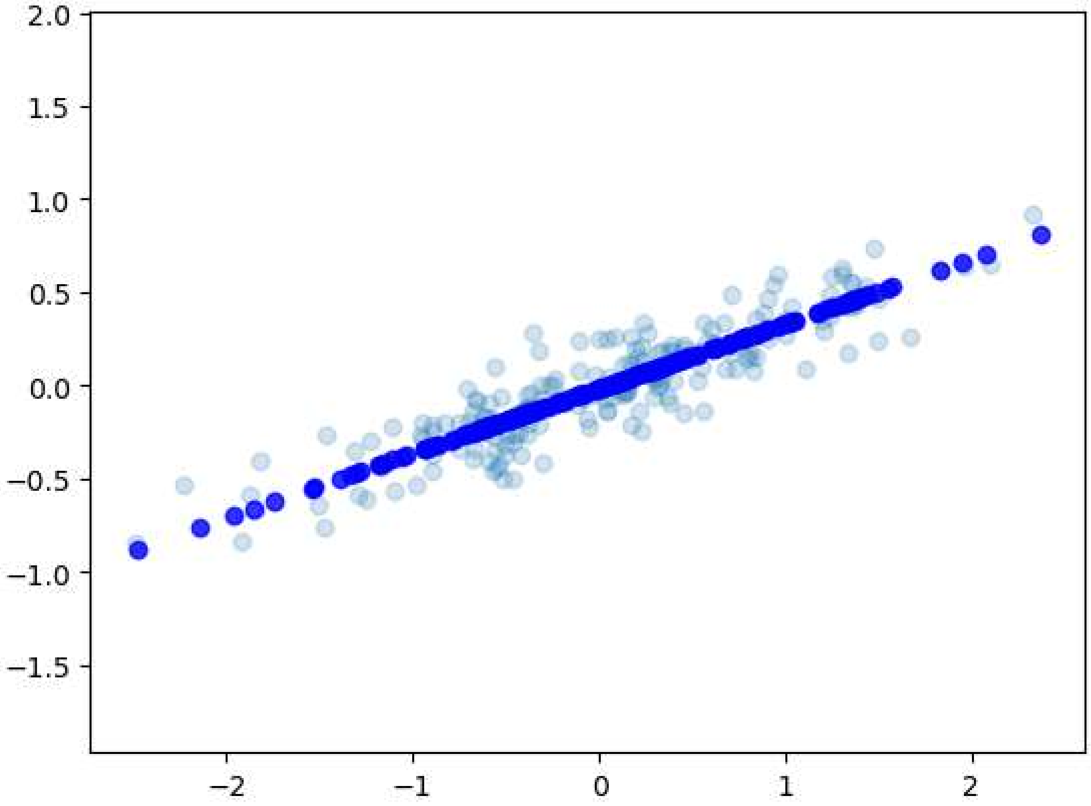
In [16]:

```
#PCA as Dimensionality Reduction Algorithm
pca = PCA(n_components=1)
pca.fit(X)
X_pca = pca.transform(X)
print ("original shape :", X_pca.shape)
print ("transformed shape :", X_pca.shape)
```

```
original shape : (200, 1)
transformed shape : (200, 1)
```

In [17]:

```
X_new = pca.inverse_transform(X_pca)
plt.scatter(X[:,0], X[:,1], alpha=0.2)
plt.scatter(X_new[:,0], X_new[:,1], color = 'b', alpha=0.8)
plt.axis('equal');
```



In [19]:

```
from sklearn.datasets import load_digits
digits= load_digits()
digits.data.shape
```

Out[19]:

```
(1797, 64)
```

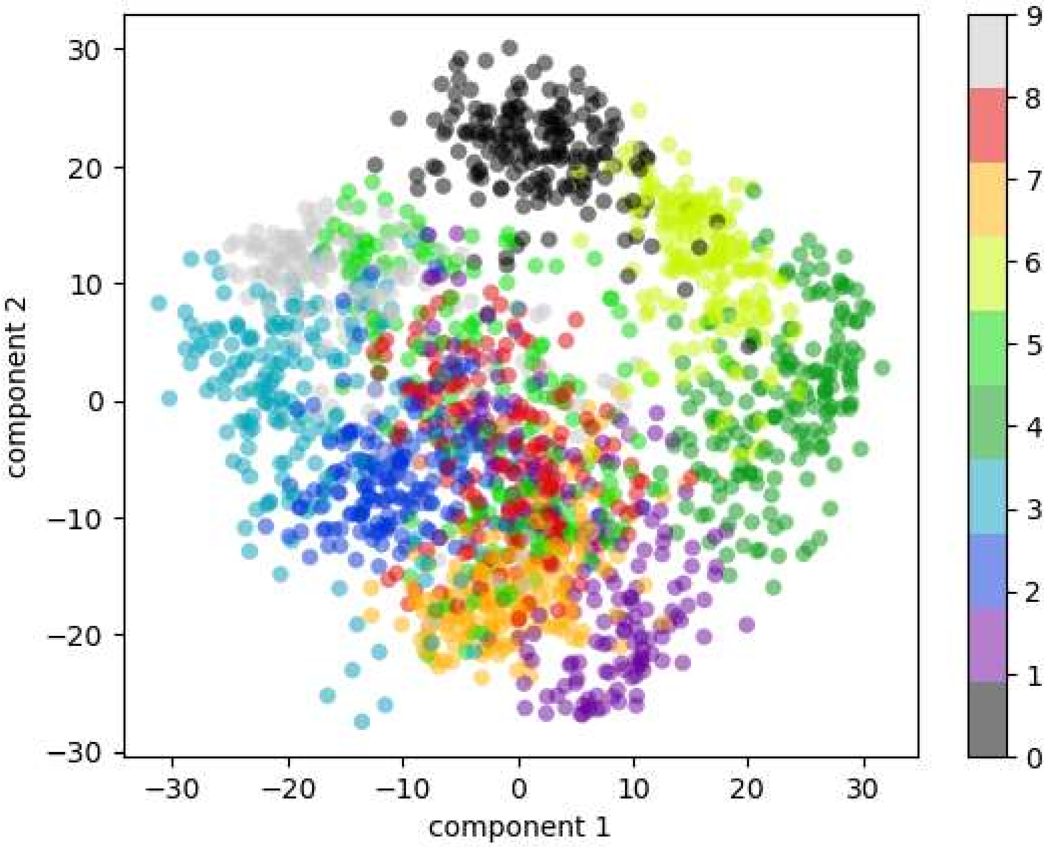
In [20]:

```
pca = PCA(2) #project from 64 to 2 dimensions
pca.fit(digits.data)
projected = pca.transform(digits.data)
print(digits.data.shape)
print(projected.shape)
```

```
(1797, 64)
(1797, 2)
```

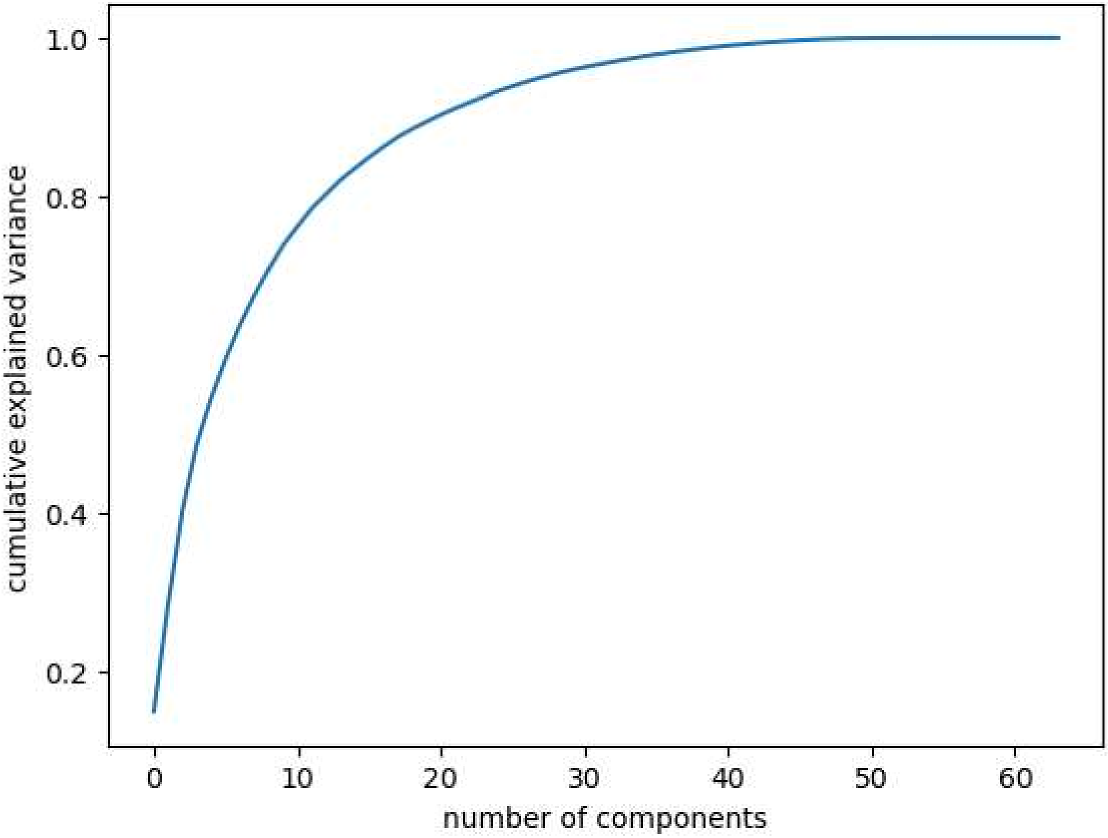
```
In [23]: plt.scatter(projected[:, 0], projected[:, 1],
                     c=digits.target, edgecolor='none', alpha =0.5,
                     cmap=plt.cm.get_cmap('nipy_spectral',10))
         plt.xlabel('component 1')
         plt.ylabel('component 2')
         plt.colorbar();
```

```
C:\Users\User\AppData\Local\Temp\ipykernel_7592\557690667.py:3: MatplotlibDep
recationWarning: The get_cmap function was deprecated in Matplotlib 3.7 and w
ill be removed two minor releases later. Use ``matplotlib.colormaps[name]`` o
r ``matplotlib.colormaps.get_cmap(obj)`` instead.
  cmap=plt.cm.get_cmap('nipy_spectral',10))
```



In [24]:

```
#choosing the number of components
pca= PCA().fit(digits.data)
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance');
```



In [ ]:

```
```