FinalProject

November 15, 2018

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.cluster import SpectralClustering
from sklearn import mixture
from sklearn.cluster import AgglomerativeClustering
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori
from scipy.cluster.hierarchy import dendrogram, linkage
from scipy.spatial.distance import cdist
from sklearn.cluster import DBSCAN
from sklearn.model_selection import train_test_split
#from sklearn.cross_validation import train_test_split
from sklearn.cluster import MiniBatchKMeans
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import Imputer
from missingpy import KNNImputer
from sklearn.preprocessing import MinMaxScaler
```

In [2]:
```
project = pd.read_csv("/Users/capio/Dropbox/SCHOOL/FALL 2018/STAT 517/FINAL PROJECT/Da
project.head()
```

Out[2]:

	Class	gene_0	gene_1	gene_2	gene_3	gene_4	gene_5	gene_6	\
0	PRAD	0.0	2.017209	3.265527	5.478487	10.431999	0	7.175175	
1	LUAD	0.0	0.592732	1.588421	7.586157	9.623011	0	6.816049	
2	PRAD	0.0	3.511759	4.327199	6.881787	9.870730	0	6.972130	
3	PRAD	0.0	3.663618	4.507649	6.659068	10.196184	0	7.843375	
4	BRCA	0.0	2.655741	2.821547	6.539454	9.738265	0	6.566967	

	gene_7	gene_8	...	gene_16373	gene_16374	gene_16375	\
0	0.591871	0.0	...	8.750533	7.421257	4.692126	
1	0.000000	0.0	...	6.638879	7.991732	5.709045	
2	0.452595	0.0	...	8.205754	10.375778	1.839758	
3	0.434882	0.0	...	8.093185	8.424771	5.502251	
4	0.360982	0.0	...	7.522228	12.176650	10.305423	

```
   gene_16376  gene_16377  gene_16378  gene_16379  gene_16380  gene_16381  \
0    1.334282    1.650856    3.973382    0.000000    9.291933    0.840926
1    0.811142    2.717824    3.255773    1.590818    3.467410    1.178045
2    0.000000    3.004897    0.796598    0.000000    5.533710    0.225892
3    0.434882    2.207862    1.039419    0.000000    8.358278    0.377401
4    0.360982    1.099497    0.360982    0.649386    9.315607    1.052833

   gene_16382
0    7.886642
1    8.864223
2    9.843799
3    7.701320
4   10.646325

[5 rows x 16384 columns]
```

```
In [57]: project.shape
```

```
Out[57]: (801, 16384)
```

```
In [58]: project.rename( columns={'Unnamed: 0':'Labels'}, inplace=True )
         project.head()
```

```
Out[58]:   Class  gene_0    gene_1    gene_2    gene_3     gene_4  gene_5    gene_6  \
         0  PRAD     0.0  2.017209  3.265527  5.478487  10.431999       0  7.175175
         1  LUAD     0.0  0.592732  1.588421  7.586157   9.623011       0  6.816049
         2  PRAD     0.0  3.511759  4.327199  6.881787   9.870730       0  6.972130
         3  PRAD     0.0  3.663618  4.507649  6.659068  10.196184       0  7.843375
         4  BRCA     0.0  2.655741  2.821547  6.539454   9.738265       0  6.566967

              gene_7  gene_8     ...      gene_16373  gene_16374  gene_16375  \
         0  0.591871     0.0     ...        8.750533    7.421257    4.692126
         1  0.000000     0.0     ...        6.638879    7.991732    5.709045
         2  0.452595     0.0     ...        8.205754   10.375778    1.839758
         3  0.434882     0.0     ...        8.093185    8.424771    5.502251
         4  0.360982     0.0     ...        7.522228   12.176650   10.305423

            gene_16376  gene_16377  gene_16378  gene_16379  gene_16380  gene_16381  \
         0    1.334282    1.650856    3.973382    0.000000    9.291933    0.840926
         1    0.811142    2.717824    3.255773    1.590818    3.467410    1.178045
         2    0.000000    3.004897    0.796598    0.000000    5.533710    0.225892
         3    0.434882    2.207862    1.039419    0.000000    8.358278    0.377401
         4    0.360982    1.099497    0.360982    0.649386    9.315607    1.052833

            gene_16382
         0    7.886642
         1    8.864223
         2    9.843799
         3    7.701320
```

```
4   10.646325

[5 rows x 16384 columns]
```

```
In [59]: proj=project.loc[:, ~project.columns.str.contains('Class')]
         proj.head()
```

Out[59]:

	gene_0	gene_1	gene_2	gene_3	gene_4	gene_5	gene_6	\
0	0.0	2.017209	3.265527	5.478487	10.431999	0	7.175175	
1	0.0	0.592732	1.588421	7.586157	9.623011	0	6.816049	
2	0.0	3.511759	4.327199	6.881787	9.870730	0	6.972130	
3	0.0	3.663618	4.507649	6.659068	10.196184	0	7.843375	
4	0.0	2.655741	2.821547	6.539454	9.738265	0	6.566967	

	gene_7	gene_8	gene_9	...	gene_16373	gene_16374	gene_16375	\
0	0.591871	0.0	0.0	...	8.750533	7.421257	4.692126	
1	0.000000	0.0	0.0	...	6.638879	7.991732	5.709045	
2	0.452595	0.0	0.0	...	8.205754	10.375778	1.839758	
3	0.434882	0.0	0.0	...	8.093185	8.424771	5.502251	
4	0.360982	0.0	0.0	...	7.522228	12.176650	10.305423	

	gene_16376	gene_16377	gene_16378	gene_16379	gene_16380	gene_16381	\
0	1.334282	1.650856	3.973382	0.000000	9.291933	0.840926	
1	0.811142	2.717824	3.255773	1.590818	3.467410	1.178045	
2	0.000000	3.004897	0.796598	0.000000	5.533710	0.225892	
3	0.434882	2.207862	1.039419	0.000000	8.358278	0.377401	
4	0.360982	1.099497	0.360982	0.649386	9.315607	1.052833	

	gene_16382
0	7.886642
1	8.864223
2	9.843799
3	7.701320
4	10.646325

[5 rows x 16383 columns]

```
In [60]: proj.isnull().sum() #no missing values
```

```
Out[60]: gene_0        0
         gene_1        0
         gene_2        0
         gene_3        0
         gene_4        0
         gene_5        0
         gene_6        0
         gene_7        0
         gene_8        0
         gene_9        0
```

```
gene_10       0
gene_11       0
gene_12       0
gene_13       0
gene_14       0
gene_15       0
gene_16       0
gene_17       0
gene_18       0
gene_19       0
gene_20       0
gene_21       0
gene_22       0
gene_23       0
gene_24       0
gene_25       0
gene_26       0
gene_27       0
gene_28       0
gene_29       0
             ..
gene_16353    0
gene_16354    0
gene_16355    0
gene_16356    0
gene_16357    0
gene_16358    0
gene_16359    0
gene_16360    0
gene_16361    0
gene_16362    0
gene_16363    0
gene_16364    0
gene_16365    0
gene_16366    0
gene_16367    0
gene_16368    0
gene_16369    0
gene_16370    0
gene_16371    0
gene_16372    0
gene_16373    0
gene_16374    0
gene_16375    0
gene_16376    0
gene_16377    0
gene_16378    0
gene_16379    0
```

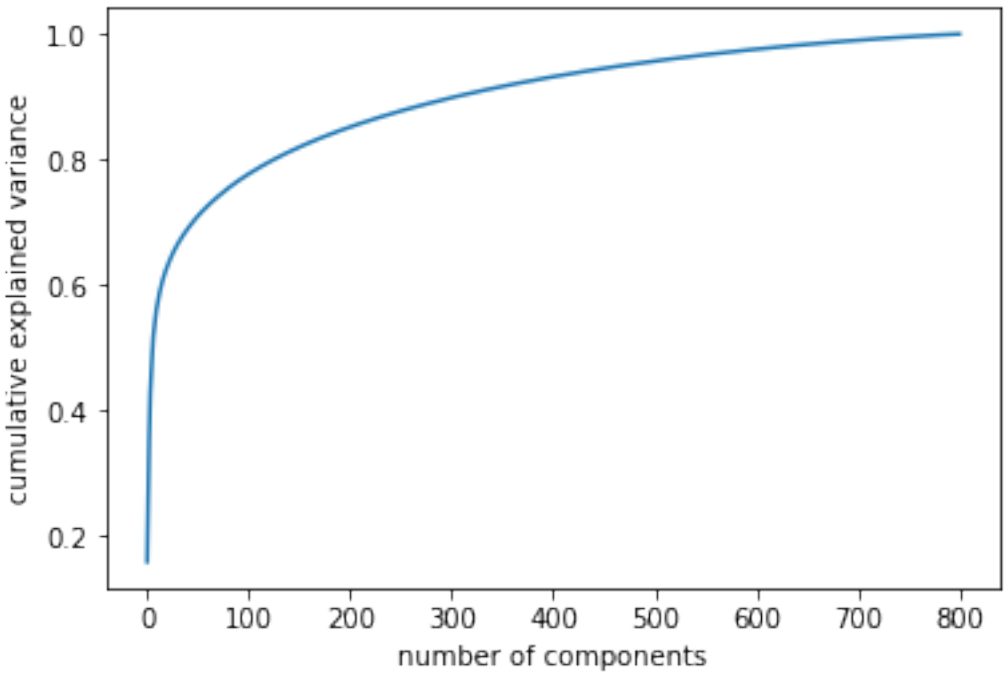
```
gene_16380    0
gene_16381    0
gene_16382    0
Length: 16383, dtype: int64
```

In [62]:
```
#apriori(proj, min_support=0.001, use_colnames=True)
```

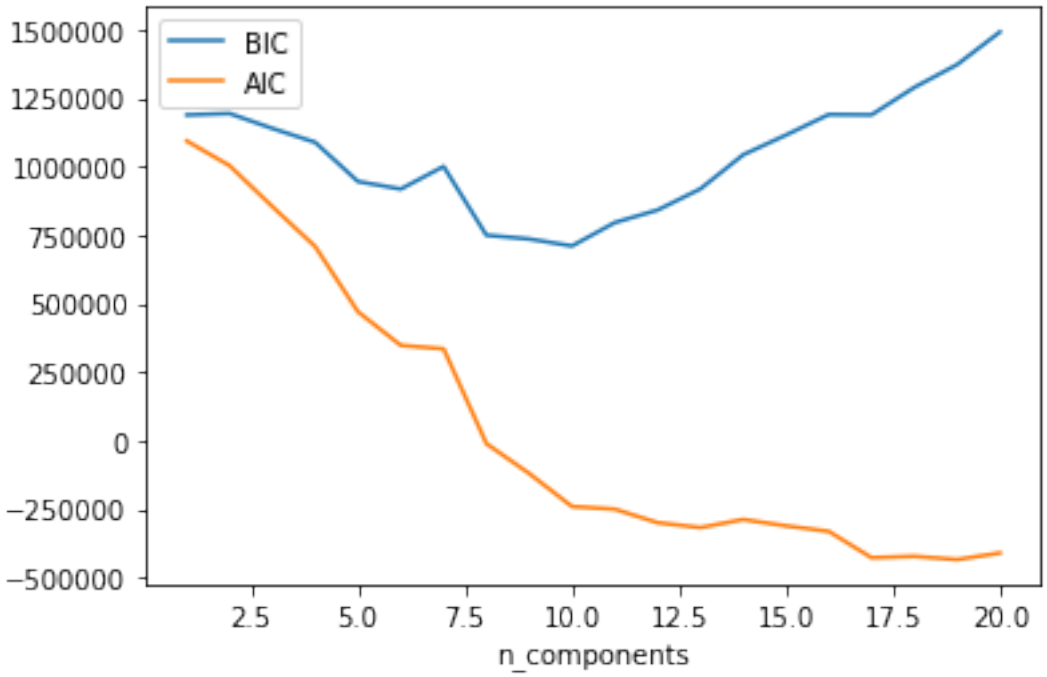
In [63]:
```
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
scaler = StandardScaler()
scaler.fit(proj)
X_scaled = scaler.transform(proj)
```

```
/Users/capio/anaconda2/lib/python2.7/site-packages/sklearn/preprocessing/data.py:617: DataConve
  return self.partial_fit(X, y)
/Users/capio/anaconda2/lib/python2.7/site-packages/ipykernel_launcher.py:6: DataConversionWarni
```

In [64]:
```
pca = PCA().fit(proj)
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('number of components')
plt.ylabel('cumulative explained variance');
```

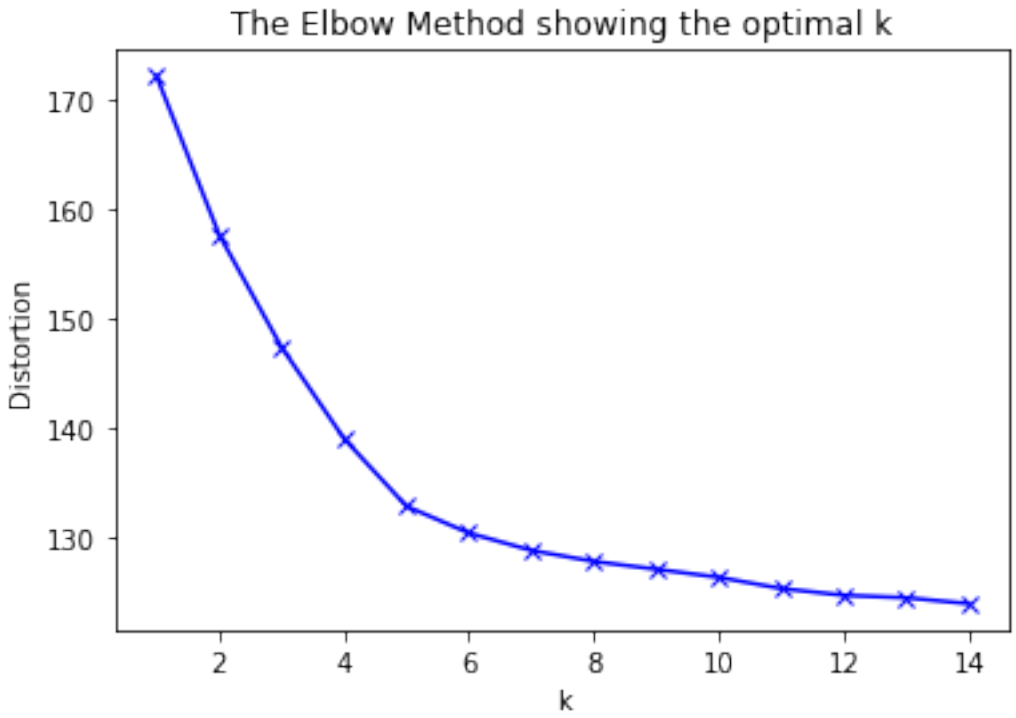


```
In [65]: pca = PCA(n_components=200)
         pca.fit(proj)
         proj_200PCA = pca.transform(proj)

In [66]: # Using BIC to determine optimal number of clusters

In [67]: n_components = np.arange(1, 21)
         models = [mixture.GaussianMixture(n, covariance_type='full', random_state=0).fit(proj
                   for n in n_components]

         plt.plot(n_components, [m.bic(proj_200PCA) for m in models], label='BIC')
         plt.plot(n_components, [m.aic(proj_200PCA) for m in models], label='AIC')
         plt.legend(loc='best')
         plt.xlabel('n_components');
```



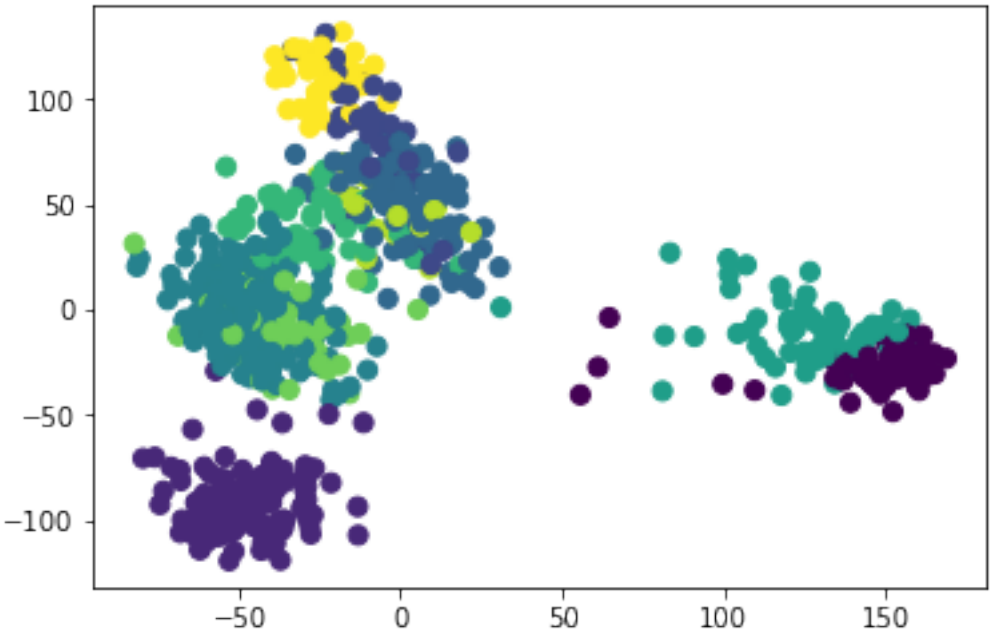

```
In [68]: #Using Elbow Method to determine the optimal number of clusters

In [69]: distortions = []
         K = range(1,15)
         for k in K:
             kmeanModel = KMeans(n_clusters=k).fit(proj)
             kmeanModel.fit(proj)
             distortions.append(sum(np.min(cdist(proj, kmeanModel.cluster_centers_, 'euclidean

In [70]: plt.plot(K, distortions, 'bx-')
         plt.xlabel('k')
```

```
plt.ylabel('Distortion')
plt.title('The Elbow Method showing the optimal k')
plt.show()
```

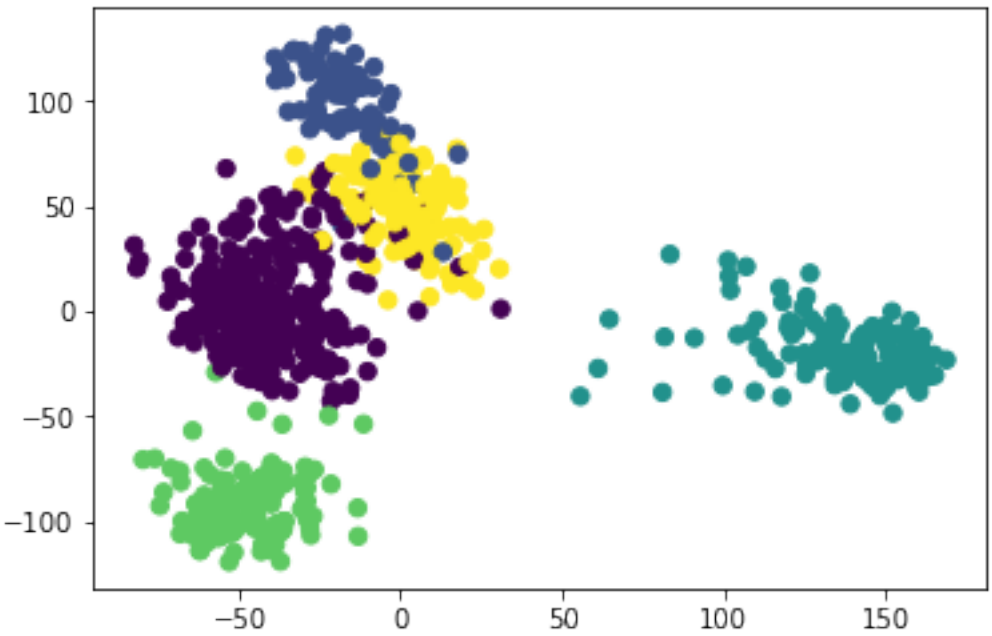


In [71]:
```
#Clustering
```

In [72]:
```
#Clustering with 200 PC's using Spectral Clustering
model = SpectralClustering(n_clusters=10, affinity='nearest_neighbors',
                           assign_labels='kmeans')
labels = model.fit_predict(proj_200PCA)
plt.scatter(proj_200PCA[:, 0], proj_200PCA[:, 1], c=labels,
            s=50, cmap='viridis');
```

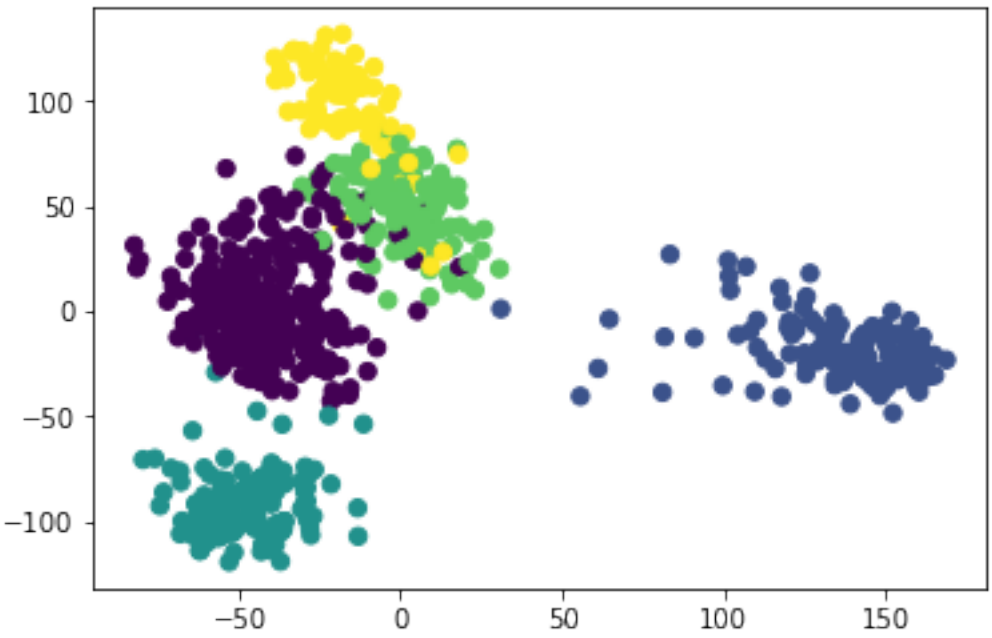


In [73]:
```
#Clustering with 200 PC's using Gaussian Mixture Modeling
gmm = mixture.GaussianMixture(n_components=5).fit(proj_200PCA)
labels = gmm.predict(proj_200PCA)
plt.scatter(proj_200PCA[:, 0], proj_200PCA[:, 1], c=labels, s=40, cmap='viridis');
```



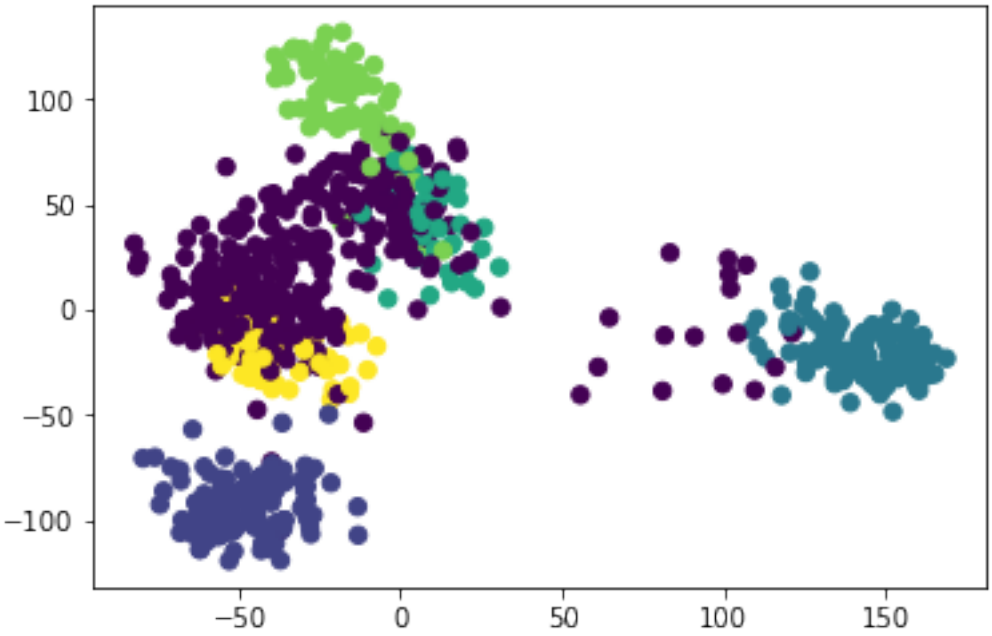
In [74]:
```
#Clustering with 200 PC's using Agglomerative Clustering: Recursively merges
#pair of clusters that minimally increases a given linkage distance
clustering = AgglomerativeClustering(n_clusters=5).fit(proj_200PCA)
labels = clustering.labels_
plt.scatter(proj_200PCA[:, 0], proj_200PCA[:, 1], c=labels, s=40, cmap='viridis')
```

Out[74]:
```
<matplotlib.collections.PathCollection at 0x1a19062850>
```



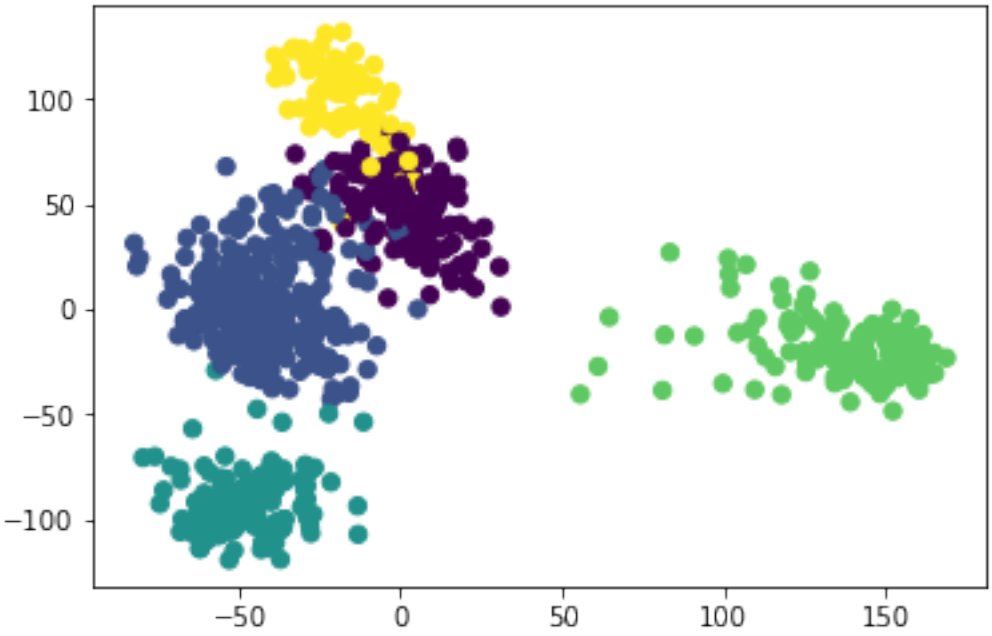
In [75]:
```
#Clustering with 200 PC's using DBSCAN
db = DBSCAN(eps=0.3, min_samples=10,metric='cosine').fit(proj_200PCA)
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_
plt.scatter(proj_200PCA[:, 0], proj_200PCA[:, 1], c=labels, s=40, cmap='viridis')
```

Out[75]:
```
<matplotlib.collections.PathCollection at 0x1a18e20090>
```



In [76]:
```
#Clustering with 200 PC's using MiniBatchKmeans
clustering = MiniBatchKMeans(n_clusters=5).fit(proj_200PCA)
labels = clustering.labels_
plt.scatter(proj_200PCA[:, 0], proj_200PCA[:, 1], c=labels, s=40, cmap='viridis')
```

Out[76]:
```
<matplotlib.collections.PathCollection at 0x1a190131d0>
```



```
In [82]: #Comparing Labels to samples

In [83]: labels = pd.DataFrame(labels)
         labels.columns = ['Labels']

In [84]: y_project = project[u'Class']
         y_project = pd.DataFrame(y_project)

In [85]: bytypes = pd.merge(y_project, labels, left_index=True, right_index=True)
         bytypes.head()

Out[85]:   Class  Labels
         0  PRAD       2
         1  LUAD       0
         2  PRAD       2
         3  PRAD       2
         4  BRCA       1

In [86]: #haplo = bytypes.Group.unique()
         #clusts = bytypes.groupnum.unique()
```