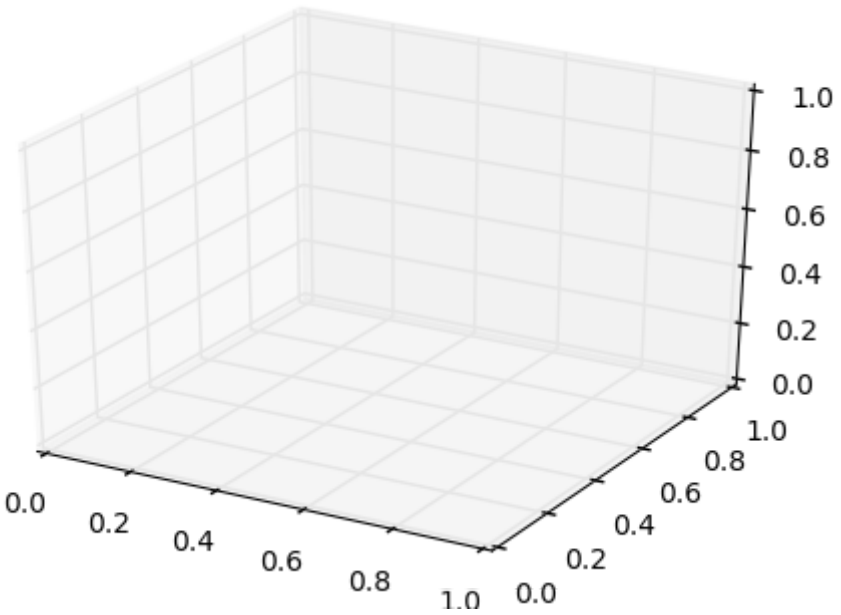
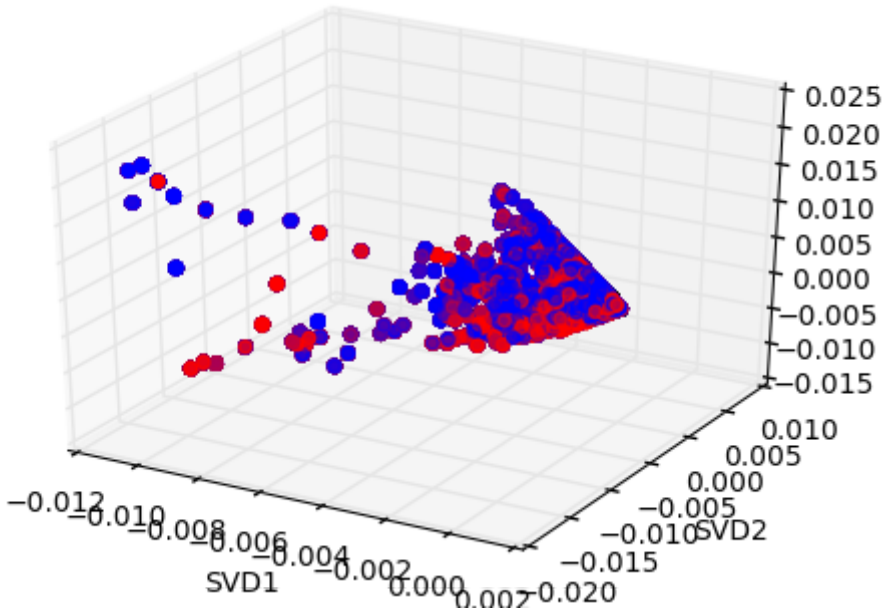
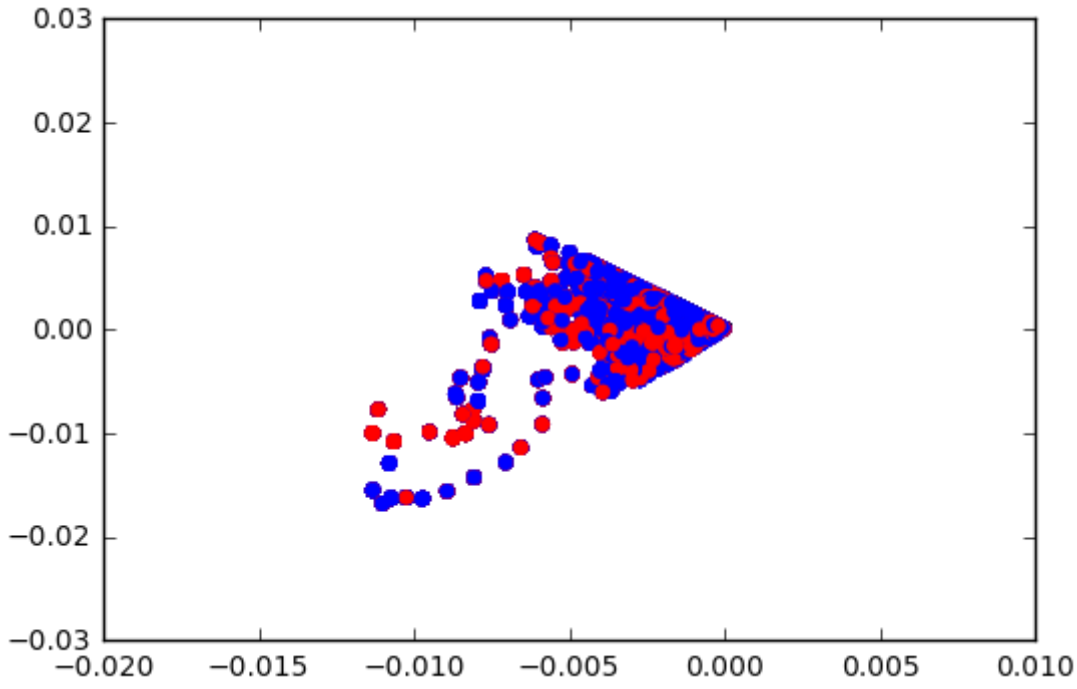
In [1]:
```
%matplotlib inline
import numpy as np
import mltools as ml
import mltools.dtree as mldt
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from scipy import linalg
```

In [30]:
```
Ytr =  np.genfromtxt('data/Y_train.txt');
Xtr =  np.genfromtxt('data/X_train.txt');
Xtest =  np.genfromtxt('data/X_test.txt')

[Xtr,Xte,Ytr,Yte] = ml.splitData(Xtr,Ytr, .75);
#[Xtr,Xte,Ytr,Yte] = ml.splitData(Xtr,Ytr, .5);
S=3
U, s, V = linalg.svd( Xtr, full_matrices=False )
#Sig = mat(eye(S)*s[:S])
#tak out columns you don't need
newdata = U[:,:S]

# this line is used to retrieve dataset
#~ new = U[:,:2]*Sig*V[:2,:]

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
colors = ['blue','red','black']
c= [colors[int(y)] for y in Ytr ];
ax.scatter(newdata[:,0],newdata[:,1],newdata[:,2], color= c)
#for i in xrange(Xtr.shape[0]):
#    ax.scatter(newdata[i,0],newdata[i,1])
plt.xlabel('SVD1')
plt.ylabel('SVD2')
plt.show()
```

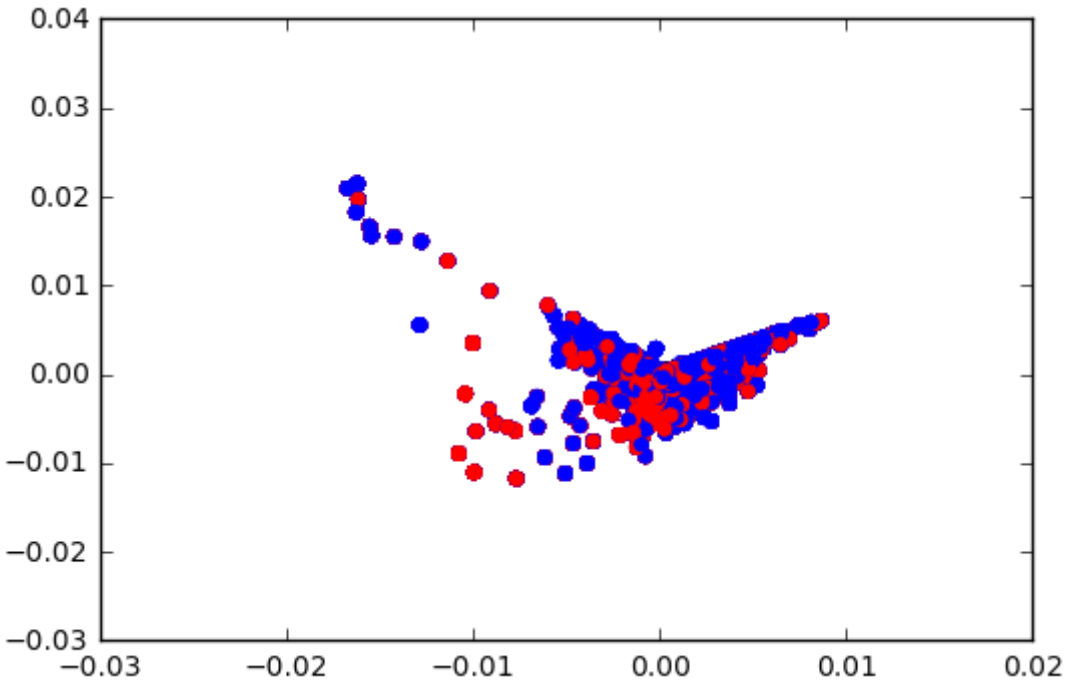




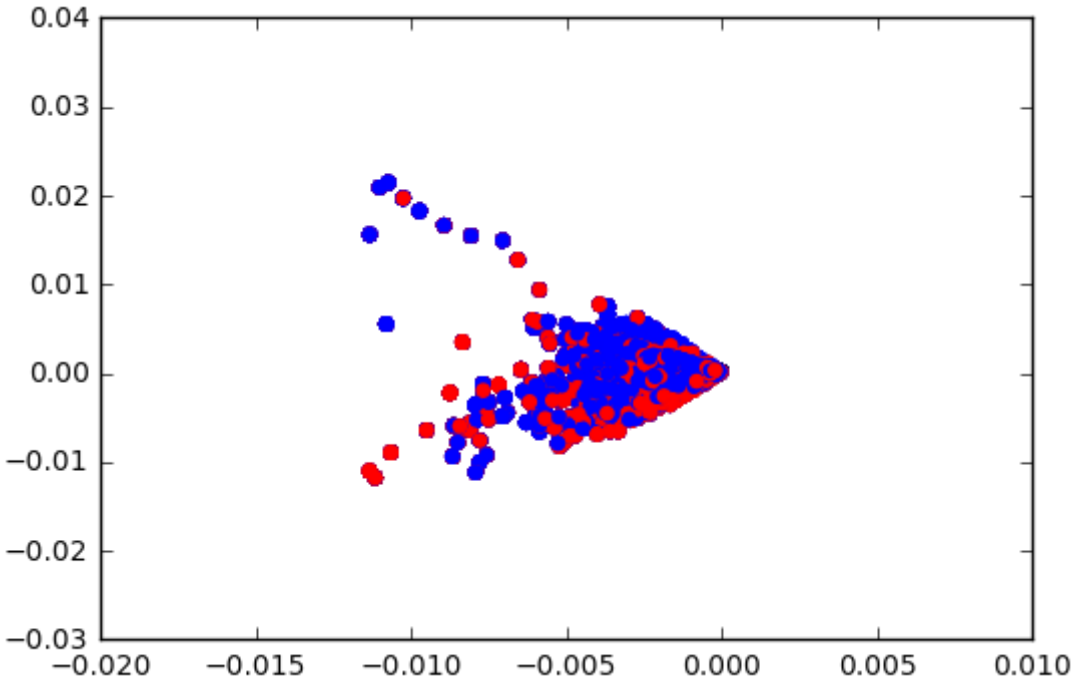

In [32]:
```
plt.scatter(newdata[:,0],newdata[:,1],color= c)
plt.show()
```



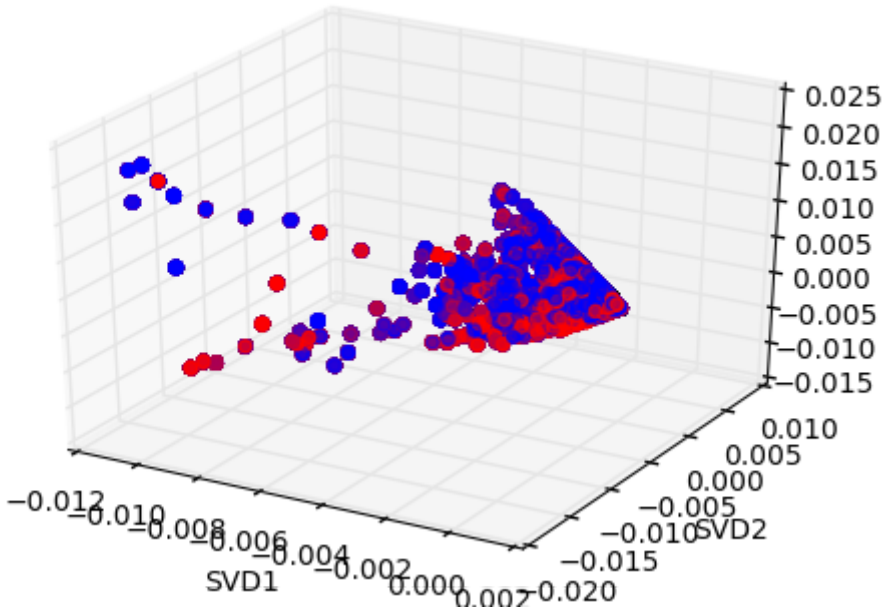
In [33]:
```
plt.scatter(newdata[:,1],newdata[:,2],color= c)
plt.show()
```



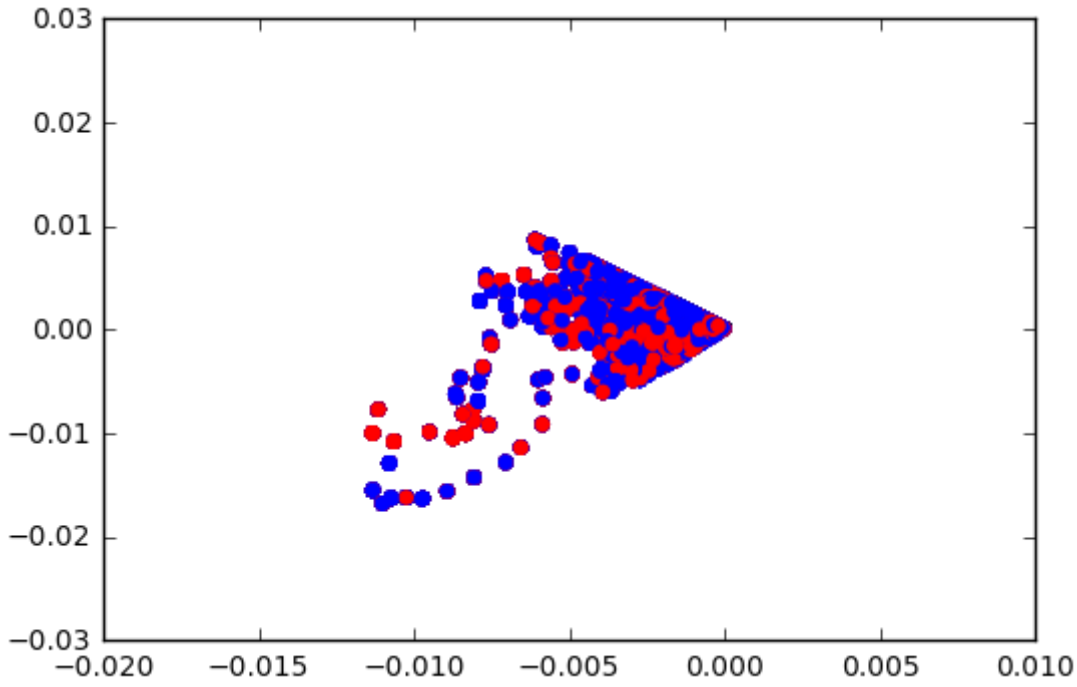

In [34]:
```
plt.scatter(newdata[:,0],newdata[:,2],color= c)
plt.show()
```



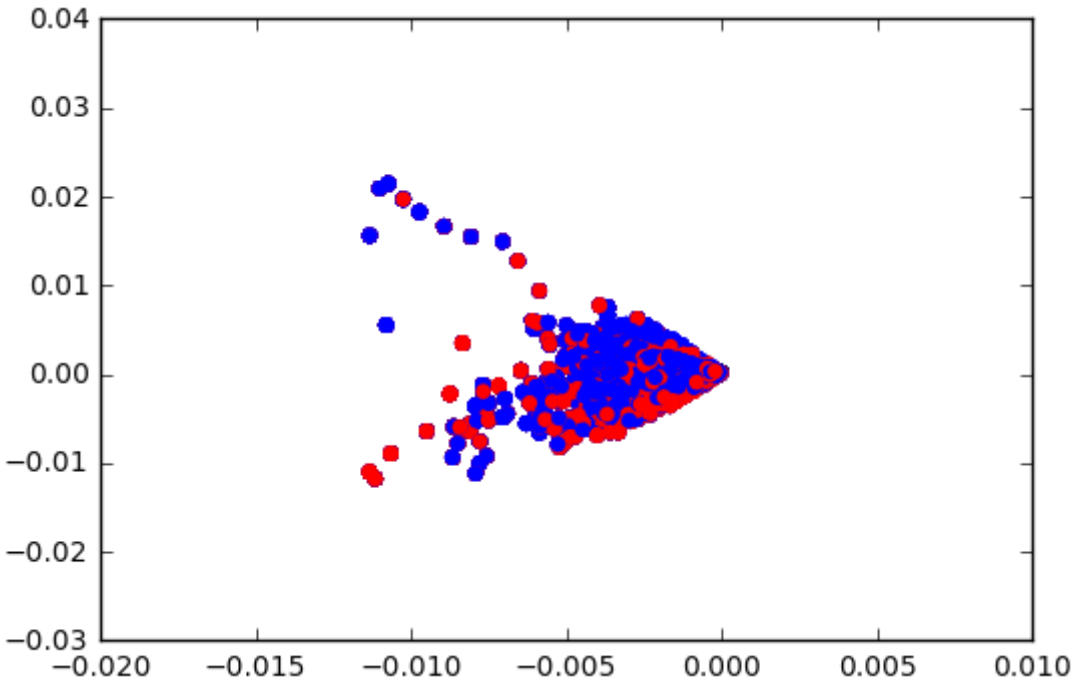
In [35]:
```
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
colors = ['blue','red']
c= [colors[int(y)] for y in Ytr ];
c
ax.scatter(newdata[:,0],newdata[:,1],newdata[:,2], color = c)

plt.xlabel('SVD1')
plt.ylabel('SVD2')
plt.show()
```



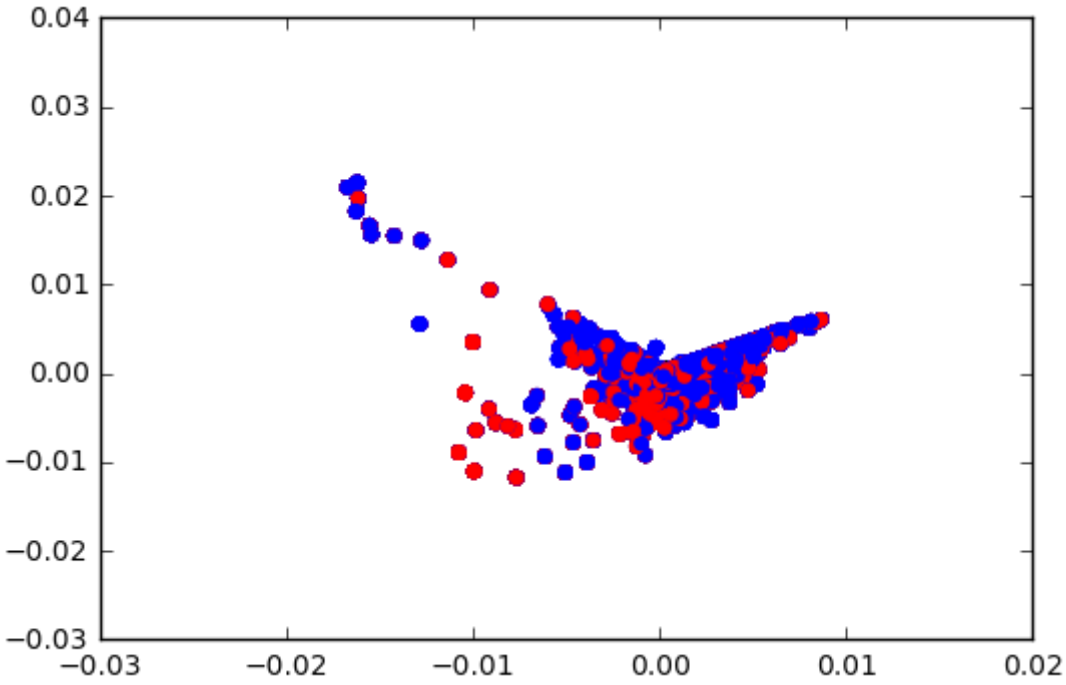

In [36]:
```
plt.scatter(newdata[:,0],newdata[:,1],color = c)
plt.show()
```



In [37]:
```
plt.scatter(newdata[:,0],newdata[:,2],color = c)
plt.show()
```




In [38]:
```
plt.scatter(newdata[:,1],newdata[:,2],color = c)
plt.show()
```



In [39]:

```
import os, sys
def make_views(ax,angles,elevation=None, width=4, height = 3,
               prefix='tmprot_',**kwargs):
    """
    Makes jpeg pictures of the given 3d ax, with different angles.
    Args:
        ax (3D axis): te ax
        angles (list): the list of angles (in degree) under which to
                       take the picture.
        width,height (float): size, in inches, of the output images.
        prefix (str): prefix for the files created.

    Returns: the list of files created (for later removal)
    """

    files = []
    ax.figure.set_size_inches(width,height)

    for i,angle in enumerate(angles):

        ax.view_init(elev = elevation, azim=angle)
        fname = '%s%03d.jpeg'%(prefix,i)
        ax.figure.savefig(fname)
        files.append(fname)

    return files



##### TO TRANSFORM THE SERIES OF PICTURE INTO AN ANIMATION

def make_movie(files,output, fps=10,bitrate=1800,**kwargs):
    """
    Uses mencoder, produces a .mp4/.ogv/... movie from a list of
    picture files.
    """

    output_name, output_ext = os.path.splitext(output)
    command = { '.mp4' : 'mencoder "mf://%s" -mf fps=%d -o %s.mp4 -ovc lavc\
                         -lavcopts vcodec=msmpeg4v2:vbitrate=%d'
                         %(",".join(files),fps,output_name,bitrate)}

    command['.ogv'] = command['.mp4'] + '; ffmpeg -i %s.mp4 -r %d %s'%(outpu

    print command[output_ext]
    output_ext = os.path.splitext(output)[1]
    os.system(command[output_ext])



def make_gif(files,output,delay=100, repeat=True,**kwargs):
    """
    Uses imageMagick to produce an animated .gif from a list of
    picture files.
    """

    loop = -1 if repeat else 0
```

```
    os.system('convert -delay %d -loop %d %s %s'
              %(delay,loop," ".join(files),output))



def make_strip(files,output,**kwargs):
    """
    Uses imageMagick to produce a .jpeg strip from a list of
    picture files.
    """

    os.system('montage -tile 1x -geometry +0+0 %s %s'%(" ".join(files),outpu



##### MAIN FUNCTION

def rotanimate(ax, angles, output, **kwargs):
    """
    Produces an animation (.mp4,.ogv,.gif,.jpeg,.png) from a 3D plot on
    a 3D ax

    Args:
        ax (3D axis): the ax containing the plot of interest
        angles (list): the list of angles (in degree) under which to
                       show the plot.
        output : name of the output file. The extension determines the
                 kind of animation used.
        **kwargs:
            - width : in inches
            - heigth: in inches
            - framerate : frames per second
            - delay : delay between frames in milliseconds
            - repeat : True or False (.gif only)
    """

    output_ext = os.path.splitext(output)[1]

    files = make_views(ax,angles, **kwargs)

    D = { '.mp4' : make_movie,
          '.ogv' : make_movie,
          '.gif': make_gif ,
          '.jpeg': make_strip,
          '.png':make_strip}

    D[output_ext](files,output,**kwargs)

    #for f in files:
    #    os.remove(f)

angles = np.linspace(0,360,21)[:-1]
rotanimate(ax, angles,'movie.gif',delay=5)
```