Experiment 3 Implementation of SVM classifier

Part A: Generation of support vectors for a dataset

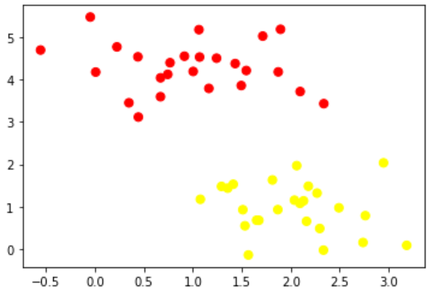
```
import numpy as np
import matplotlib.pyplot as plt
```

```
#Generate a dataset
from sklearn.datasets.samples_generator import make_blobs

X, y = make_blobs(n_samples=50, centers= 2, random_state=0, cluster_std=0.60)
```

```
plt.scatter(X[:,0], X[:,1], c=y, s=50, cmap='autumn')
```

```
<matplotlib.collections.PathCollection at 0x7fa70c7e81d0>
```

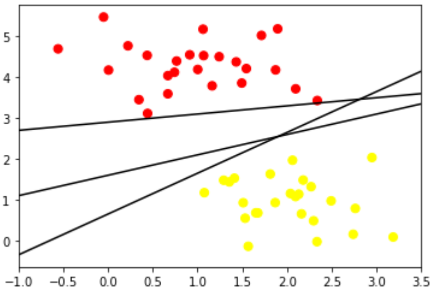


```
xfit= np.linspace(-1,3.5)
```

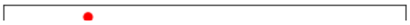
```
xfit
```

```
array([-1.        , -0.90816327, -0.81632653, -0.7244898 , -0.63265306,
       -0.54081633, -0.44897959, -0.35714286, -0.26530612, -0.17346939,
       -0.08163265,  0.01020408,  0.10204082,  0.19387755,  0.28571429,
        0.37755102,  0.46938776,  0.56122449,  0.65306122,  0.74489796,
        0.83673469,  0.92857143,  1.02040816,  1.1122449 ,  1.20408163,
        1.29591837,  1.3877551 ,  1.47959184,  1.57142857,  1.66326531,
        1.75510204,  1.84693878,  1.93877551,  2.03061224,  2.12244898,
        2.21428571,  2.30612245,  2.39795918,  2.48979592,  2.58163265,
        2.67346939,  2.76530612,  2.85714286,  2.94897959,  3.04081633,
        3.13265306,  3.2244898 ,  3.31632653,  3.40816327,  3.5       ])
```

```
#Plotting lines
plt.scatter(X[:,0], X[:,1], c=y, s=50, cmap='autumn')
for m,b in [(1,0.65), (0.5,1.6), (0.2, 2.9)]:
  #print(m,b)
  #print(xfit, m*xfit+b)
  plt.plot(xfit, m*xfit+b, '-k')
  plt.xlim(-1,3.5)
```



```
#Plotting lines
plt.scatter(X[:,0], X[:,1], c=y, s=50, cmap='autumn')
for m,b,d in [(1,0.65,0.33), (0.5,1.6,0.55), (0.2, 2.9, 0.2)]:
  #print(m,b)
  #print(xfit, m*xfit+b)
  yfit = m*xfit+b
  plt.plot(xfit,yfit,'-k')
  plt.fill_between(xfit, yfit-d, yfit+d, edgecolor= 'none', color='#AAAA', alpha=0.4)
  plt.xlim(-1,3.5)
```



```
#Create a SVM model
from sklearn.svm import SVC
model = SVC(kernel = 'linear', C=1E10)
model.fit(X,y)
```

```
     SVC(C=10000000000.0, break_ties=False, cache_size=200, class_weight=None,
         coef0=0.0, decision_function_shape='ovr', degree=3, gamma='scale',
         kernel='linear', max_iter=-1, probability=False, random_state=None,
         shrinking=True, tol=0.001, verbose=False)
```

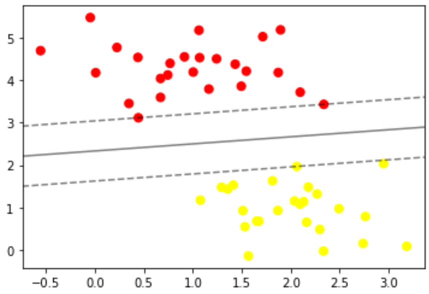
```
#define a function paralled plot_svc_decision_function

def plot_svc_decision_function(model, ax=None, plot_support=True):
  if ax is None:
    ax = plt.gca()                         #get coordinate axis
  xlim = ax.get_xlim()
  ylim = ax.get_ylim()
  #create grid to evaluate model
  x= np.linspace(xlim[0], xlim[1],30)
  y= np.linspace(ylim[0], ylim[1],30)
  Y, X = np.meshgrid(y,x)
  xy = np.vstack([X.ravel(),Y.ravel()]).T
  P = model.decision_function(xy).reshape(X.shape)

  ax.contour(X,Y,P, colors='k', levels = [-1,0,1], alpha = 0.5, linestyles = ['--', '-', '--'])

  #plot support vectors
  if plot_support:
    ax.scatter(model.support_vectors_[:,0],
               model.support_vectors_[:,1],
               s=300, linewidth=1, facecolors = 'none');
  ax.set_xlim(xlim)
  ax.set_ylim(ylim)
```

```
plt.scatter(X[:,0], X[:,1], c=y, s=50, cmap='autumn')
plot_svc_decision_function(model)
```



```
model.support_vectors_
```

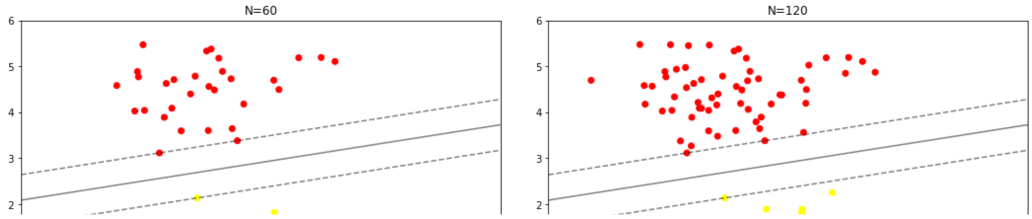
```
     array([[0.44359863, 3.11530945],
            [2.33812285, 3.43116792],
            [2.06156753, 1.96918596]])
```

```
def plot_svm(N=10, ax= None):
  X,y = make_blobs(n_samples=200, n_features=2, centers=2, cluster_std=0.60, random_state=0)

  X=X[:N]
  y=y[:N]
  model = SVC(kernel='linear', C=1E10)
  model.fit(X,y)

  ax=ax or plt.gca()
  ax.scatter(X[:,0], X[:,1], c=y, cmap='autumn')
  ax.set_xlim(-1,4)
  ax.set_ylim(-1,6)
  plot_svc_decision_function(model,ax)
```

```
fig, ax =plt.subplots(1,2,figsize=(16,6))
fig.subplots_adjust(left=0.0625 , right=0.95, wspace=0.1)
for axi, N in zip(ax,[60,120]):
  plot_svm(N,axi)
  axi.set_title('N={0}'.format(N))
```

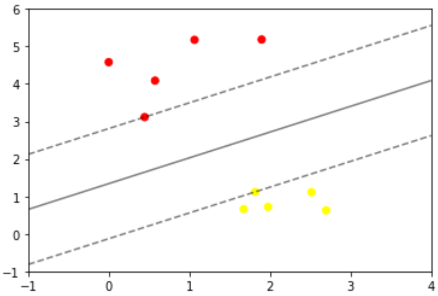


```
pip install ipywidgets
```

```
Requirement already satisfied: ipywidgets in /usr/local/lib/python3.7/dist-packages (7.6.3)
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in /usr/local/lib/python3.7/dist-packages (from ipywidgets) (1.0.0)
Requirement already satisfied: ipykernel>=4.5.1 in /usr/local/lib/python3.7/dist-packages (from ipywidgets) (4.10.1)
Requirement already satisfied: nbformat>=4.2.0 in /usr/local/lib/python3.7/dist-packages (from ipywidgets) (5.1.3)
Requirement already satisfied: traitlets>=4.3.1 in /usr/local/lib/python3.7/dist-packages (from ipywidgets) (5.0.5)
Requirement already satisfied: widgetsnbextension~=3.5.0 in /usr/local/lib/python3.7/dist-packages (from ipywidgets) (3.5.1)
Requirement already satisfied: ipython>=4.0.0 in /usr/local/lib/python3.7/dist-packages (from ipywidgets) (5.5.0)
Requirement already satisfied: jupyter-client in /usr/local/lib/python3.7/dist-packages (from ipykernel>=4.5.1->ipywidgets) (5.3.5)
Requirement already satisfied: tornado>=4.0 in /usr/local/lib/python3.7/dist-packages (from ipykernel>=4.5.1->ipywidgets) (5.1.1)
Requirement already satisfied: pickleshare in /usr/local/lib/python3.7/dist-packages (from ipython>=4.0.0->ipywidgets) (0.7.5)
Requirement already satisfied: pygments in /usr/local/lib/python3.7/dist-packages (from ipython>=4.0.0->ipywidgets) (2.6.1)
Requirement already satisfied: prompt-toolkit<2.0.0,>=1.0.4 in /usr/local/lib/python3.7/dist-packages (from ipython>=4.0.0->ipywidgets) (1.0.18)
Requirement already satisfied: simplegeneric>0.8 in /usr/local/lib/python3.7/dist-packages (from ipython>=4.0.0->ipywidgets) (0.8.1)
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.7/dist-packages (from ipython>=4.0.0->ipywidgets) (57.4.0)
Requirement already satisfied: pexpect in /usr/local/lib/python3.7/dist-packages (from ipython>=4.0.0->ipywidgets) (4.8.0)
Requirement already satisfied: decorator in /usr/local/lib/python3.7/dist-packages (from ipython>=4.0.0->ipywidgets) (4.4.2)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in /usr/local/lib/python3.7/dist-packages (from nbformat>=4.2.0->ipywidgets) (2.6.0)
Requirement already satisfied: ipython-genutils in /usr/local/lib/python3.7/dist-packages (from nbformat>=4.2.0->ipywidgets) (0.2.0)
Requirement already satisfied: jupyter-core in /usr/local/lib/python3.7/dist-packages (from nbformat>=4.2.0->ipywidgets) (4.7.1)
Requirement already satisfied: wcwidth in /usr/local/lib/python3.7/dist-packages (from prompt-toolkit<2.0.0,>=1.0.4->ipython>=4.0.0->ipywidgets) (0.2.5)
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.7/dist-packages (from prompt-toolkit<2.0.0,>=1.0.4->ipython>=4.0.0->ipywidgets) (1.15.0)
Requirement already satisfied: notebook>=4.4.1 in /usr/local/lib/python3.7/dist-packages (from widgetsnbextension~=3.5.0->ipywidgets) (5.3.1)
Requirement already satisfied: Send2Trash in /usr/local/lib/python3.7/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets) (1.8.0)
Requirement already satisfied: nbconvert in /usr/local/lib/python3.7/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets) (5.6.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.7/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets) (2.11.3)
Requirement already satisfied: terminado>=0.8.1 in /usr/local/lib/python3.7/dist-packages (from notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets) (0.11.0)
Requirement already satisfied: pyzmq>=13 in /usr/local/lib/python3.7/dist-packages (from jupyter-client->ipykernel>=4.5.1->ipywidgets) (22.2.1)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from jupyter-client->ipykernel>=4.5.1->ipywidgets) (2.8.2)
Requirement already satisfied: ptyprocess in /usr/local/lib/python3.7/dist-packages (from terminado>=0.8.1->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets)
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7/dist-packages (from jinja2->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets) (2.
Requirement already satisfied: pandocfilters>=1.4.1 in /usr/local/lib/python3.7/dist-packages (from nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidge
Requirement already satisfied: testpath in /usr/local/lib/python3.7/dist-packages (from nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets) (0.5.0)
Requirement already satisfied: defusedxml in /usr/local/lib/python3.7/dist-packages (from nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets) (0.7.1
Requirement already satisfied: mistune<2,>=0.8.1 in /usr/local/lib/python3.7/dist-packages (from nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets)
Requirement already satisfied: bleach in /usr/local/lib/python3.7/dist-packages (from nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets) (4.0.0)
Requirement already satisfied: entrypoints>=0.2.2 in /usr/local/lib/python3.7/dist-packages (from nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets
Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-packages (from bleach->nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidgets)
Requirement already satisfied: webencodings in /usr/local/lib/python3.7/dist-packages (from bleach->nbconvert->notebook>=4.4.1->widgetsnbextension~=3.5.0->ipywidge
Requirement already satisfied: pyparsing>=2.0.2 in /usr/local/lib/python3.7/dist-packages (from packaging->bleach->nbconvert->notebook>=4.4.1->widgetsnbextension~=
```

```
from ipywidgets import interact, fixed
interact(plot_svm, N=[10,50,100,150,200], ax=fixed(None));
```

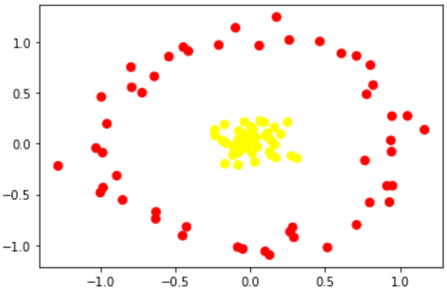
N 10



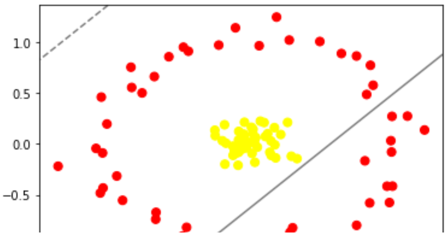
```
from sklearn.datasets.samples_generator import make_circles
X, y= make_circles(100, factor=.1, noise=.1)
```

```
plt.scatter(X[:,0], X[:,1], c=y, s=50, cmap='autumn')
```

```
<matplotlib.collections.PathCollection at 0x7fa70b9e6050>
```



```
clf= SVC(kernel='linear').fit(X,y)
plt.scatter(X[:,0], X[:,1], c=y, s=50, cmap='autumn')
plot_svc_decision_function(clf,plot_support = False)
```



```
r= np.exp(-(X**2).sum(1))
```

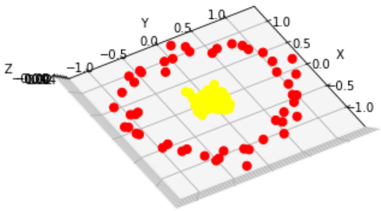
```
from mpl_toolkits import mplot3d


def plot_3D(elev=30, azim=30, X=X, y=y):
  ax=plt.subplot(projection='3d')
  ax.scatter3D(X[:,0], X[:,1], c=y, s=50, cmap='autumn')
  ax.view_init(elev=elev, azim=azim)
  ax.set_xlabel('X')
  ax.set_ylabel('Y')
  ax.set_zlabel('Z')


interact(plot_3D, elev=[-90,90], azip=(-180,180), X=fixed(X), y=fixed(y));
```

elev -90

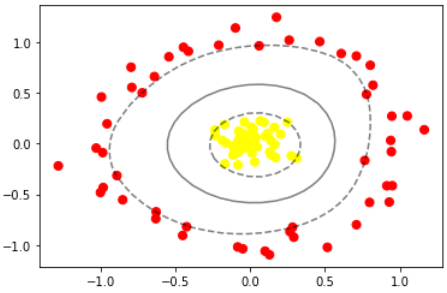
azim 30



```
clf=SVC(kernel='rbf', C=1E6)
clf.fit(X,y)
```

```
    SVC(C=1000000.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
        decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
        max_iter=-1, probability=False, random_state=None, shrinking=True,
        tol=0.001, verbose=False)
```

```
plt.scatter(X[:,0], X[:,1], c=y, s=50, cmap='autumn')
plot_svc_decision_function(clf)
plt.scatter(clf.support_vectors_[:,0], clf.support_vectors_[:,1], lw=1, facecolors='none');
```



```
#PART B: Apllication of Svm for face recognition


from sklearn.datasets import fetch_lfw_people
faces = fetch_lfw_people(min_faces_per_person=80)
```

```
    Downloading LFW metadata: https://ndownloader.figshare.com/files/5976012
    Downloading LFW metadata: https://ndownloader.figshare.com/files/5976009
    Downloading LFW metadata: https://ndownloader.figshare.com/files/5976006
    Downloading LFW data (~200MB): https://ndownloader.figshare.com/files/5976015
```

```
print(faces.target_names)
```

```
    ['Colin Powell' 'Donald Rumsfeld' 'George W Bush' 'Gerhard Schroeder'
     'Tony Blair']
```

```
print(faces.images.shape)
```

```
    (1140, 62, 47)
```

```
from sklearn.decomposition import PCA as RandomizedPCA
from sklearn.pipeline import make_pipeline



pca = RandomizedPCA(n_components=100, whiten=True, random_state=1)
svc=SVC(kernel='rbf', class_weight='balanced')
```

```
svc=SVC(kernel='rbf', class_weight='balanced')
model = make_pipeline(pca, svc)
```

```
from sklearn.model_selection import train_test_split
Xtrain,Xtest,ytrain,ytest = train_test_split(faces.data,faces.target,random_state=1)
```

```
model.fit(Xtrain,ytrain)
```

```
    Pipeline(memory=None,
             steps=[('pca',
                     PCA(copy=True, iterated_power='auto', n_components=100,
                         random_state=1, svd_solver='auto', tol=0.0, whiten=True)),
                    ('svc',
                     SVC(C=1.0, break_ties=False, cache_size=200,
                         class_weight='balanced', coef0=0.0,
                         decision_function_shape='ovr', degree=3, gamma='scale',
                         kernel='rbf', max_iter=-1, probability=False,
                         random_state=None, shrinking=True, tol=0.001,
                         verbose=False))],
             verbose=False)
```

```
y_pred =model.predict(Xtest)
```

```
from sklearn.metrics import classification_report
print(classification_report(ytest, y_pred, target_names=faces.target_names))
```

```
                   precision    recall  f1-score   support

      Colin Powell       0.88      0.92      0.90        53
   Donald Rumsfeld       0.76      0.76      0.76        21
     George W Bush       0.88      0.94      0.91       139
 Gerhard Schroeder       0.93      0.74      0.83        35
        Tony Blair       0.88      0.76      0.81        37

          accuracy                           0.87       285
         macro avg       0.86      0.82      0.84       285
      weighted avg       0.87      0.87      0.87       285
```

```
target_names = faces.target_names
_, h, w = faces.images.shape
#Visualization
import matplotlib.pyplot as plt

def plot_gallery(images, titles, h, w, rows=3, cols=4):
  plt.figure(figsize=(10,10))
  for i in range(rows*cols):
    plt.subplot(rows,cols,i+1)
    plt.imshow(images[i].reshape((h,w)),cmap=plt.cm.gray)
    plt.title(titles[i])
    plt.xticks(())
    plt.yticks(())
def titles(y_pred, ytest, target_names):
  for i in range(y_pred.shape[0]):
    pred_name = target_names[y_pred[i]].split(' ')[-1]
    true_name = target_names[ytest[i]].split(' ')[-1]
    yield 'predicted: {0}\ntrue: {1}'.format(pred_name, true_name)

prediction_titles = list(titles(y_pred, ytest, target_names))
plot_gallery(Xtest, prediction_titles, h, w)
```