In [1]:

```
from collections import defaultdict
from sklearn import preprocessing
import matplotlib.pyplot as plt
import numpy as np
from numpy import linalg
from pylab import *
from decimal import Decimal
from sklearn import cross_validation
from sklearn.cross_validation import KFold
from sklearn.metrics import mean_squared_error
from sklearn import linear_model
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
from decimal import Decimal
from scipy.misc import comb
from cvxopt import matrix,solvers
from sklearn import svm
# from cvxopt import solvers

%matplotlib inline
```

In [2]:

```
def load_data(filename,delim,datatype):
    data = np.loadtxt(filename,delimiter=delim,dtype=datatype)
    X = data[:,:-1]
    Y = data[:,-1]
    return X,Y
```

In [29]:

```
def preprocess(x,y):
    X = []
    for i in x:
        row = []
        for feature in i:
            row.append(eval(feature))
        X.append(row)
    X = np.array(X)

    le = preprocessing.LabelEncoder()
    Y = le.fit_transform(y)
    Y = np.array(y)
    return X,Y
```

In [30]:

```
def changey(y):
    labels = np.unique(y)
    Y = []
    for i in y:
        if i == labels[0]:
            Y.append(1)
        elif i == labels[1]:
            Y.append(-1)
    Y = np.array(Y)
    return Y
```

In [31]:

```
def kernel_function(x,y,degree,sigma):

    #gaussian
    if sigma:
        return np.exp(-1.*((linalg.norm(x-y.T))**2)/(2*(sigma**2)))
    # linear equation
    elif degree > 1:
        return math.pow((np.dot(x,y.T) + 1),degree)
    #polynomial
    elif degree == 1:
        return np.dot(x,y.T)
```

In [32]:

```
mean1 = np.array([0])
mean2 = np.array([10])
print (np.concatenate((mean1,mean2),axis=0))
```

```
[ 0 10]
```

In [33]:

```
def random_sep_data():
#     np.random.multivariate_normal(mean, cov, (3, 3))
    mean1 = np.array([0, 10])
    mean2 = np.array([10, 0])
    cov = np.array([[1, 0], [0, 1]])
    X1 = np.random.multivariate_normal(mean1, cov, 100)
    y1 = np.ones(len(X1))
    X2 = np.random.multivariate_normal(mean2, cov, 100)
    y2 = np.ones(len(X2)) * -1
    return X1, y1, X2, y2
```

In [34]:

```
def random_nonsep_data():
    mean1 = np.array([0, 1])
    mean2 = np.array([6, 0])
    cov = np.array([[1, 0], [0, 1]])
    X1 = np.random.multivariate_normal(mean1, cov, 100)
    y1 = np.ones(len(X1))
    X2 = np.random.multivariate_normal(mean2, cov, 100)
    y2 = np.ones(len(X2)) * -1
    return X1, y1, X2, y2
```

In [61]:

```
def lagrange_multi(X,Y,degree,sigma,c,soft=False):
    gram_matrix = []
    y = np.outer(Y,Y.T)
    for i,xi in enumerate(X):
        temp = []
        for j,xj in enumerate(X):
            temp.append(kernel_function(xi,xj,degree,sigma))
        gram_matrix.append(temp)

    pos1 = np.ones(X.shape[0])
    neg1 = np.negative(pos1)
    zeros = np.zeros(X.shape[0])
    gram_matrix = np.array(gram_matrix)

#     .astype(np.double)
    p = matrix(np.multiply(y,gram_matrix))
    q = (matrix(neg1))
    A = (matrix((Y.astype(np.double)))).T
    b = matrix(0.0)

    if soft:
        c_array = pos1*c
        h = matrix(np.row_stack((zeros,c_array)))
        G = matrix(np.row_stack(((np.diag(neg1),np.diag(pos1)))))
    else:
        h = matrix(zeros.astype(np.double))
        G = matrix(np.diag(neg1))

    solvers.options['show_progress'] = False
    sol = solvers.qp(p,q,G,h,A,b)
    alpha = np.ravel(sol['x'])
    return alpha
```

In [36]:

```
def find_vector_set(X_train,alpha,threshold):
    sv = []
    for i,xi in enumerate(X_train):
        if alpha[i] > math.pow(10,-threshold):
            sv.append(xi)
    sv = np.array(sv)
    return sv
```

In [38]:

```
def plot(w0,w,x,svm):
    x1 = 1/(w) - w0*(1/(w))
    x2 = -1/(w) - w0*(1/(w))


    plt.scatter(x[:,0],x[:,1],c='r',marker='o')
    plt.scatter(svm[:,0],svm[:,1],c='b',marker='o')
    plt.show()
```

In [12]:

```
# %history 27
```

In [39]:

```
def calc_parameters(x,y,alpha,supprt_vec_set):
    supprt_vec_size = supprt_vec_set.shape[0]
    w = np.zeros(x.shape[1])
    w0 = 0.0
    for i,a in enumerate(x):
        const = alpha[i]*y[i]
        w += (const*a)

    for i,a in enumerate(supprt_vec_set):
        w0 += y[i] - dot(w,a.T)
    w0 = w0/supprt_vec_size

    return w,w0
```

In [43]:

```
def do_cross_validation(X,Y,k,threshold,degree,sigma,c,soft=False,verbose=True):
    accuracy = list()
    skl_accuracy = list()
    f_measure = list()
    precision = list()

    for train_ind,test_ind in cross_validation.KFold(len(Y),k,shuffle=True,random_st
```

```
    fold = 1
    X_train = X[train_ind]
    X_test = X[test_ind]
    Y_train = Y[train_ind]
    Y_test = Y[test_ind]

    y_predict = []
    if soft:
        alpha = lagrange_multi(X_train,Y_train,degree,sigma,c,soft)
    else:
        alpha = lagrange_multi(X_train,Y_train,degree,sigma,c)

    supprt_vec_set = find_vector_set(X_train,alpha,threshold)
    w,w0 = calc_parameters(X_train,Y_train,alpha,supprt_vec_set)

    clf = svm.SVC()
    clf.fit(X_train,Y_train)


    #linear hard margin case
    if degree == 1 and not sigma:
        for i,x in enumerate(X_test):
            a = dot(w,x.T) + w0
            if a > 0.0:
                y_predict.append(1)
            else:
                y_predict.append(-1)

    #kernel function(Gaussian or polynomial)
    if sigma or (degree > 1):
        for i,x in enumerate(X_test):
            s = 0
            for j,v in enumerate(supprt_vec_set):
                kernel = kernel_function(v,x,degree,sigma)
                s += alpha[j]*Y_train[j]*(kernel)
            if (s+w0) > 0.0:
                y_predict.append(1)
            else:
                y_predict.append(-1)


    y_predict = np.array(y_predict)
    skl_predict = clf.predict(X_test)

    acc = accuracy_score(Y_test,y_predict, sample_weight=None)
    prec = precision_score(Y_test, y_predict, sample_weight=None)
    fm = f1_score(Y_test,y_predict)
    skl_acc = accuracy_score(Y_test,skl_predict, sample_weight=None)
    c_matrix = confusion_matrix(Y_test, y_predict)

    accuracy.append(acc)
    skl_accuracy.append(skl_acc)
    f_measure.append(fm)
    precision.append(prec)
```

```
        if verbose:
            print 'fold:', fold
            print 'accuracy', acc
            print 'precision', prec
            print 'f_measure',fm
            print 'c_matrix'
            print c_matrix
            print ''

        fold += 1
        plot(w0,w,X,supprt_vec_set)

    avg_acc = sum(accuracy)/len(accuracy)
    skl_avg_acc = sum(skl_accuracy)/len(skl_accuracy)
    avg_precision = sum(precision)/len(precision)
    print 'avg. accuracy',avg_acc
    print 'sklearn avg. accuracy', skl_avg_acc
    return
```

In [41]:

```
def iris_data_svm(filename,delim,datatype,label1,label2,random,threshold,c,degree,si
    if not random:
        x,y = load_data(filename,delim,datatype)
        if label1 == 1 and label2 == 2:
            X = x[:100,:]
            Y = y[:100]
        elif label1 == 2 and label2 == 3:
            X = x[50:,:]
            Y = y[50:]
        elif label1 == 1 and label2 == 3:
            X1 = x[:50,:]
            X2 = x[100:,:]
            Y1 = y[:50]
            Y2 = y[100:]
            X = np.row_stack((X1,X2))
            Y = np.append(Y1,Y2)
        X,Y = preprocess(X,changey(Y))
    else:
        if random == "sep":
            X1,y1,X2,y2 = random_sep_data()
            X = np.concatenate((X1,X2))
            Y = np.concatenate((y1,y2))
        elif random == "nonsep":
            X1,y1,X2,y2 = random_nonsep_data()
            X = np.concatenate((X1,X2))
            Y = np.concatenate((y1,y2))


    X = preprocessing.scale(X,axis=0)
    do_cross_validation(X,Y,k,threshold,degree,sigma,c,soft,verbose)
```

Hard Margin

In [46]:

```
#linear
iris_data_svm('bezdekIris.data.txt',',','string',1,3,c=1,degree=0,sigma=None,k=2)
```

```
[ -2.93973563e-04  -9.04429020e-05  -8.59288988e-04  -3.61805120e-04]
1.00559820302
[[ 0 22]
 [ 0 28]]

[-0.00016055  0.00013379 -0.00090976 -0.00042812]
1.00330323684
[[ 0 28]
 [ 0 22]]

0.5
1.0
```

In [ ]:

```
#  from cvxopt import matrix, solvers
# Q = 2*matrix([ [2, .5], [.5, 1] ])
# p = matrix([1.0, 1.0])
# G = matrix([[-1.0,0.0],[0.0,-1.0]])
# h = matrix([0.0,0.0])
# A = matrix([1.0, 1.0], (1,2))
# b = matrix(1.0)
# # do_cross_validation(X_train,Y_train,X_test,k,c,verbose=True)
# sol=solvers.qp(Q, p, G, h, A, b)
```

In [43]:

```
#
iris_data_svm('bezdekIris.data.txt',',','string',1,2,c=None,degree=0,sigma=None,k=2
```

```
[ -1.48972142e-04  -6.26621184e-09  -5.95841478e-04  -2.08558509e-04]
1.00221363052
[[ 0 22]
 [ 0 28]]

[-0.00020879  0.00029734 -0.00071641 -0.00026893]
1.00202314785
[[ 0 28]
 [ 0 22]]

0.5
1.0
```

In [44]:

```
iris_data_svm('bezdekIris.data.txt',',','string',2,3,c=None,degree=0,sigma=None,k=2
```

```
[ -1.64033444e-04  -5.46774501e-05  -3.28061221e-04  -1.64030025e-04]
1.00278306885
[[ 0 22]
 [ 0 28]]

[  3.78851254e-05  -1.15057004e-05  -3.02242553e-04  -1.31329646e-04]
1.00109299565
[[ 0 28]
 [ 0 22]]

0.5
0.94
```

In [45]:

```
iris_data_svm('bezdekIris.data.txt',',','string',1,3,c=None,degree=0,sigma=None,k=2
```

```
[ -2.93973563e-04  -9.04429020e-05  -8.59288988e-04  -3.61805120e-04]
1.00559820302
[[ 0 22]
 [ 0 28]]

[-0.00016055  0.00013379 -0.00090976 -0.00042812]
1.00330323684
[[ 0 28]
 [ 0 22]]

0.5
1.0
```

In [47]:

```
iris_data_svm('bezdekIris.data.txt',',','string',1,3,c=1,degree=2,sigma=None,k=2)
```

```
---------------------------------------------------------------------
-----
ZeroDivisionError                         Traceback (most recent call
last)
<ipython-input-47-56c6bef716ae> in <module>()
----> 1 iris_data_svm('bezdekIris.data.txt',',','string',1,3,c=1,degre
e=2,sigma=None,k=2)

<ipython-input-32-1d72e7c5bef6> in iris_data_svm(filename, delim, data
type, label1, label2, c, degree, sigma, soft, k, verbose)
     20 #     print Y
     21
---> 22     do_cross_validation(X,Y,k,degree,sigma,c,soft,verbose)
     23
     24     Y = changey(Y)

<ipython-input-31-41ccfa550eb9> in do_cross_validation(X, Y, k, degree
, sigma, c, soft, verbose)
     16
     17         supprt_vec_set = find_vector_set(X_train,alpha)
---> 18         w,w0 = calc_parameters(X_train,Y_train,alpha,supprt_ve
c_set)
     19
     20         clf = svm.SVC()

<ipython-input-26-e30040a9e305> in calc_parameters(x, y, alpha, supprt
_vec_set)
      9     for i,a in enumerate(supprt_vec_set):
     10         w0 += y[i] - dot(w,a.T)
---> 11     w0 = w0/supprt_vec_size
     12
     13     return w,w0

ZeroDivisionError: float division by zero
```

In [18]:

```
#gaussian kernel function hard margin
iris_data_svm('bezdekIris.data.txt',',','string',2,3,c=1,degree=None,sigma=0.2,k=2)
```

```
[-0.22046731 -0.06702282 -0.40786054 -0.19774438]
3.92850564485
[[ 0 22]
 [ 0 28]]

[-0.23744244 -0.07714416 -0.46988989 -0.2537859 ]
4.50839581424
[[ 0 28]
 [ 0 22]]

0.5
0.94
```

In [19]:

```
iris_data_svm('bezdekIris.data.txt',',','string',1,3,c=1,degree=None,sigma=0.2,k=2)
```

```
[-0.42389184  0.11350081 -1.09261643 -0.44860738]
6.72316122804
[[ 0 22]
 [ 0 28]]

[-0.42738723  0.16174073 -1.14620483 -0.50421737]
6.79987407216
[[ 0 28]
 [ 0 22]]

0.5
1.0
```

In [20]:

```
iris_data_svm('bezdekIris.data.txt',',','string',1,2,c=1,degree=None,sigma=0.2,k=2)
```

```
[-0.19201013  0.19276656 -0.69251761 -0.25550174]
2.72249584883
[[ 0 22]
 [ 0 28]]

[-0.25545034  0.19281846 -0.76779774 -0.30389116]
3.49797318605
[[ 0 28]
 [ 0 22]]

0.5
1.0
```

Soft Margin

Soft Margin

In [35]:

```
iris_data_svm('bezdekIris.data.txt',',','string',1,2,c=1,degree=0,sigma=None,k=2)
```

```
[-0.19201013  0.19276656 -0.69251761 -0.25550174]
2.72249584883
[[ 0 22]
 [ 0 28]]

[-0.25545034  0.19281846 -0.76779774 -0.30389116]
3.49797318605
[[ 0 28]
 [ 0 22]]

0.5
1.0
```

In [39]:

```
iris_data_svm('bezdekIris.data.txt',',','string',1,3,c=1,degree=0,sigma=None,k=2)
```

```
[-0.01583442  0.00398052 -0.04089935 -0.01758766]
0.136449662312
[[22  0]
 [ 5 23]]

[-0.01569805  0.00507143 -0.04089286 -0.01800649]
0.344585324652
[[15 13]
 [ 0 22]]

0.82
1.0
```

In [40]:

```
iris_data_svm('bezdekIris.data.txt',',','string',2,3,c=1,degree=0,sigma=None,k=2)
```

```
[-0.00583442 -0.00165584 -0.0123539  -0.00676948]
-0.00455653247871
[[22  0]
 [28  0]]

[-0.00694805 -0.00225    -0.01335714 -0.00722078]
0.245148792197
[[ 0 28]
 [ 0 22]]

0.44
0.94
```

In [41]:

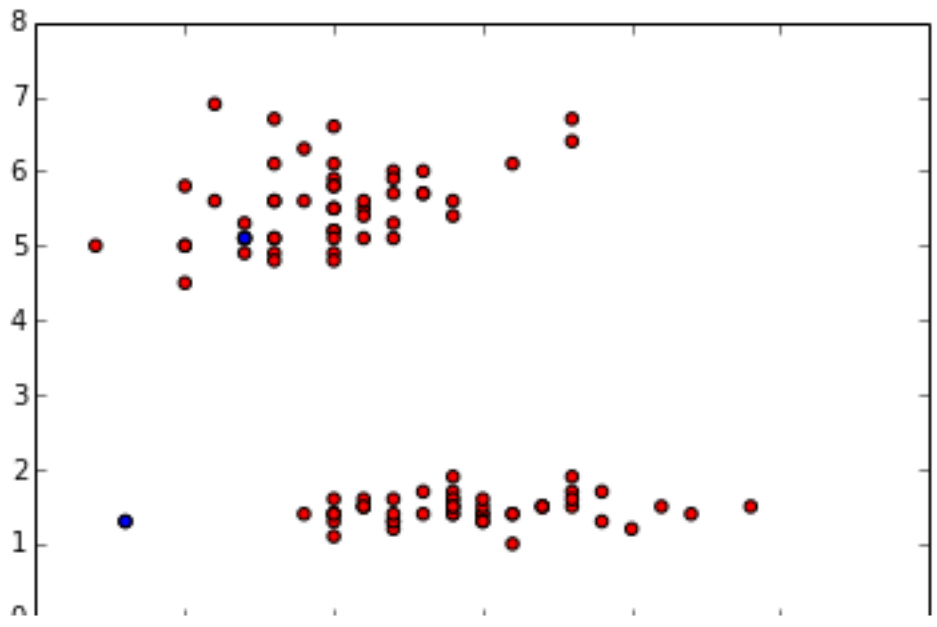
```
#polynomial
iris_data_svm('bezdekIris.data.txt',',','string',2,3,c=1,degree=2,sigma=None,k=2)
```

```
[ -1.64033444e-04  -5.46774501e-05  -3.28061221e-04  -1.64030025e-04]
1.00278306885
[[ 0 22]
 [ 0 28]]

[  3.78851254e-05  -1.15057004e-05  -3.02242553e-04  -1.31329646e-04]
1.00109299565
[[ 0 28]
 [ 0 22]]

0.5
0.94
```

In [92]:

```
iris_data_svm('bezdekIris.data.txt',',','string',1,3,6,c=1,degree=2,sigma=None,k=2)
```

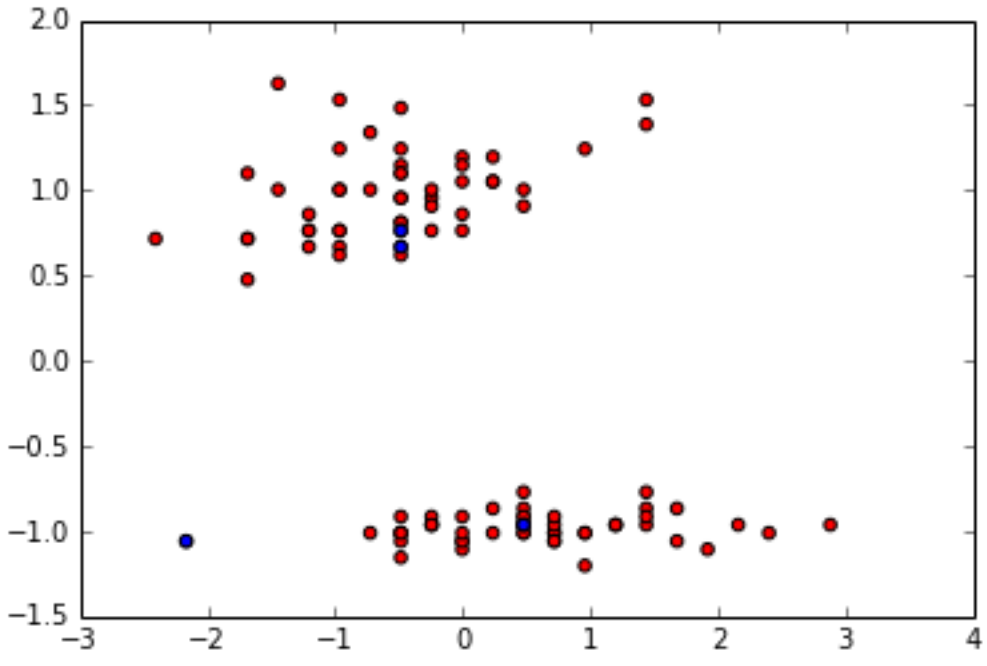
```
[ -5.65050193e-06  -1.64895501e-06  -1.64329281e-05  -6.92292271e-06]
1.0001070109
[[ 0 22]
 [ 0 28]]
```



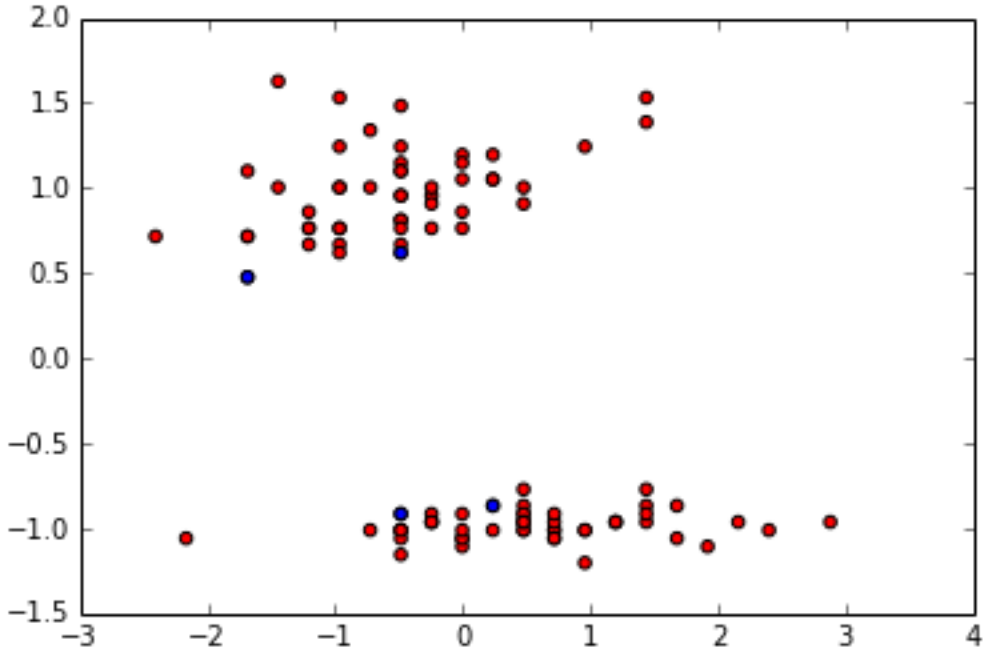
In [26]:

```
iris_data_svm('bezdekIris.data.txt',',','string',1,3,6,c=1,degree=0,sigma=None,k=2,v
```

```
[[20  2]
 [ 0 28]]
```




```
[[28  0]
 [ 0 22]]
```

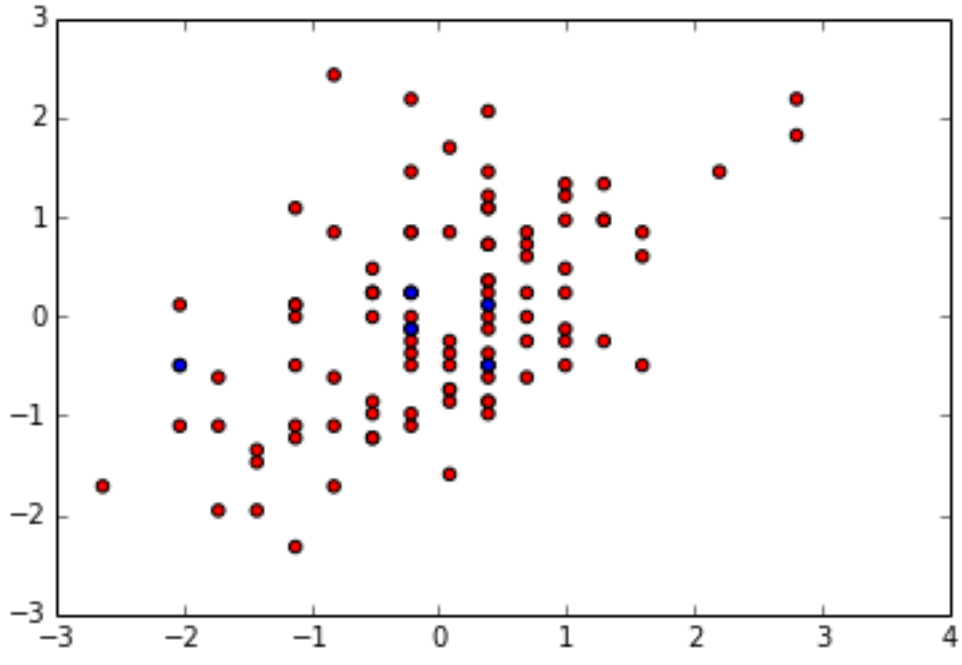



```
0.98
1.0
```

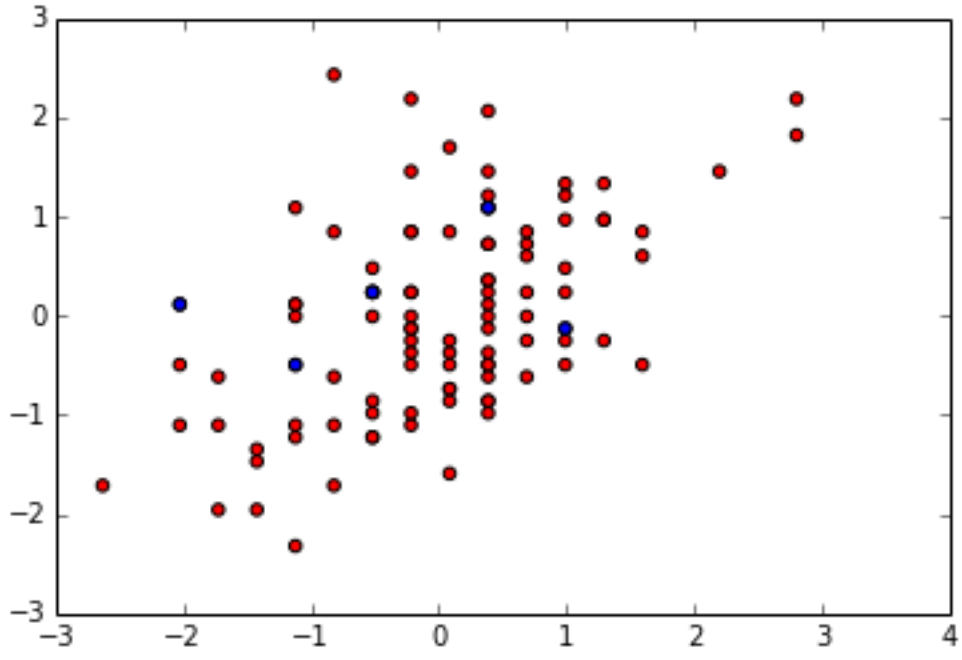
In [27]:

```
iris_data_svm('bezdekIris.data.txt',',','string',2,3,6,c=1,degree=0,sigma=None,k=2,v
```

```
[[22  0]
 [ 2 26]]
```




```
[[26  2]
 [ 0 22]]
```

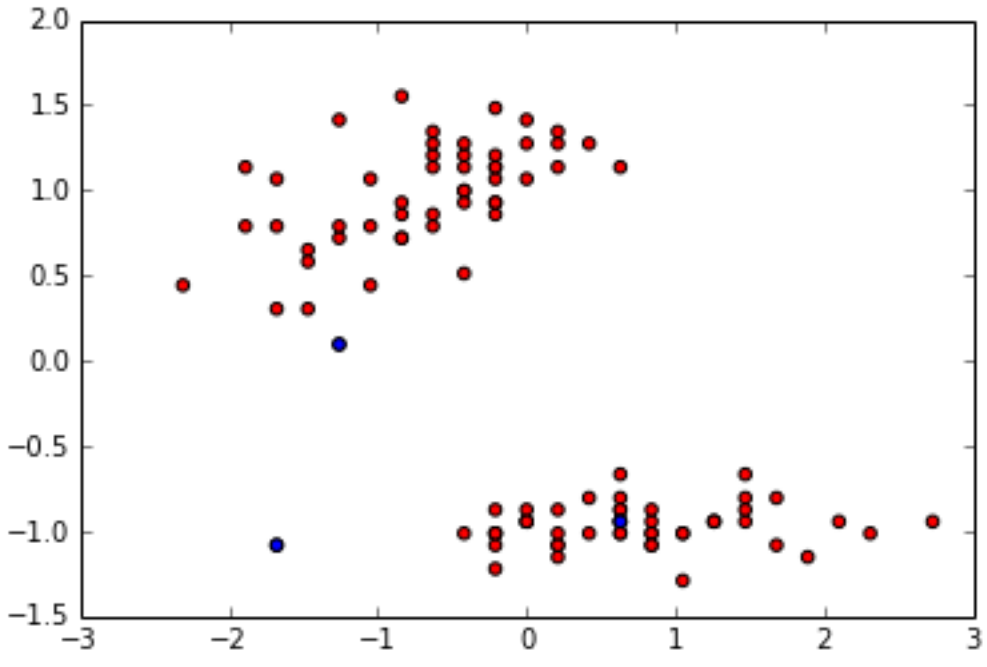



```
0.96
0.93
```

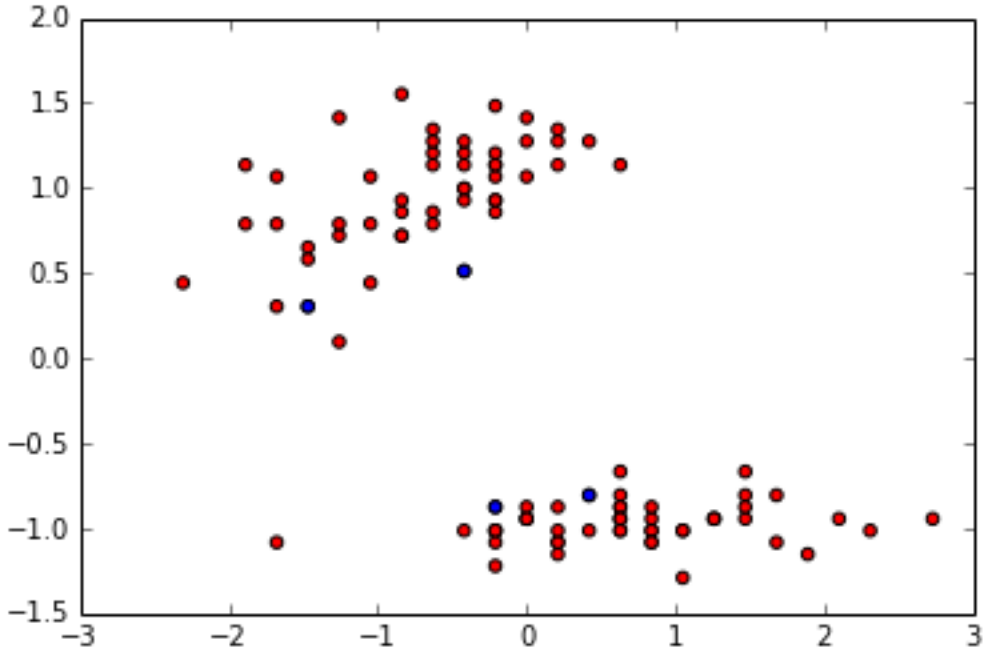
In [28]:

```
iris_data_svm('bezdekIris.data.txt',',','string',1,2,6,c=1,degree=0,sigma=None,k=2,v
```

```
[[22  0]
 [ 0 28]]
```




```
[[27  1]
 [ 0 22]]
```

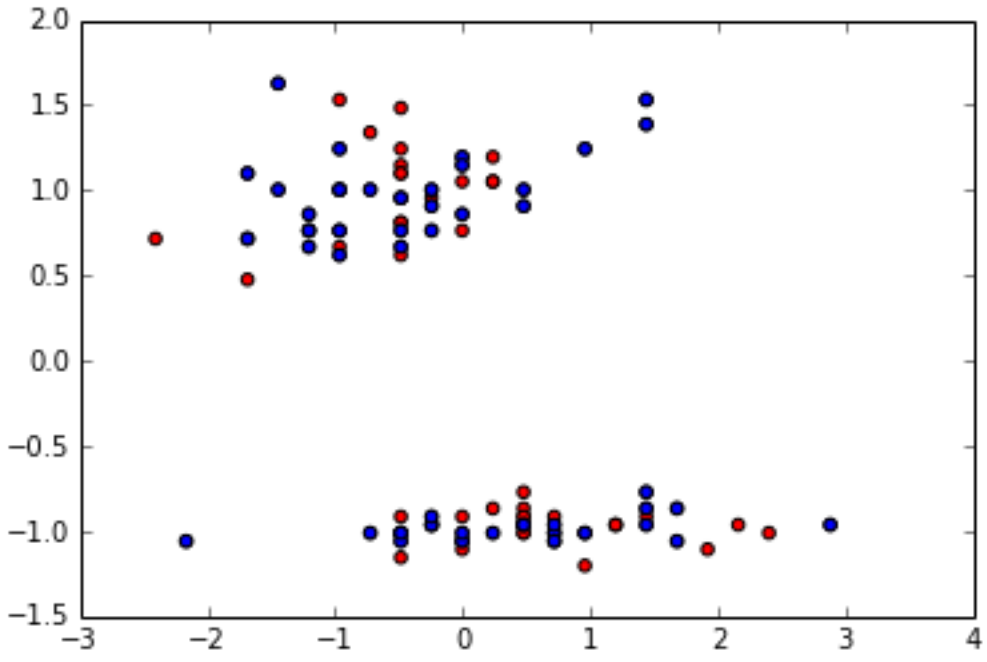



```
0.99
1.0
```

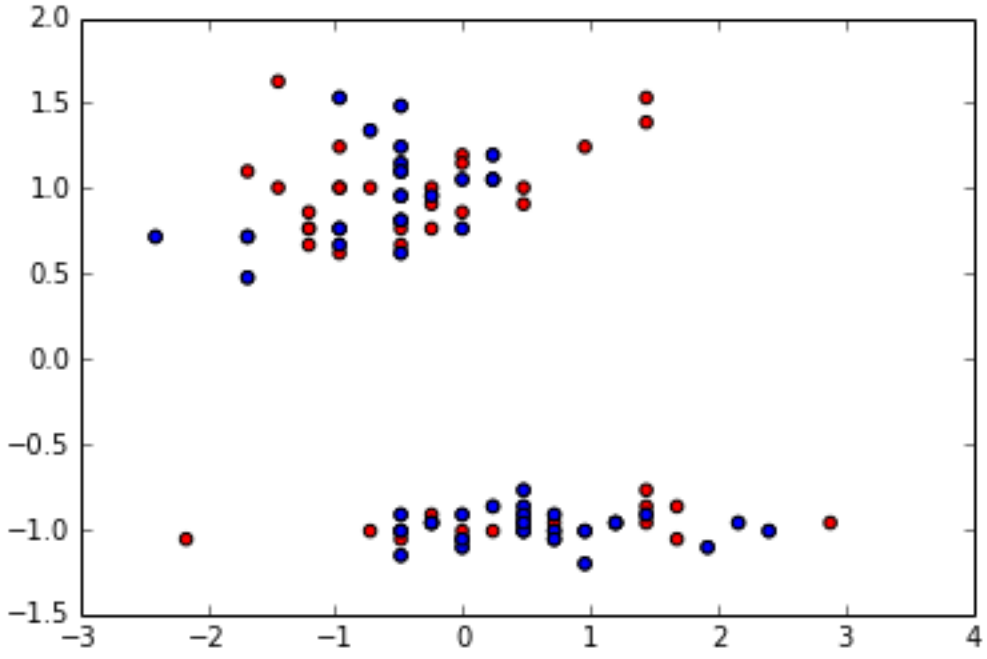
In [35]:

```
iris_data_svm('bezdekIris.data.txt',',','string',1,3,6,c=1,degree=None,sigma=0.2,k=
```

```
accuracy 0.56
```




```
accuracy 0.56
```



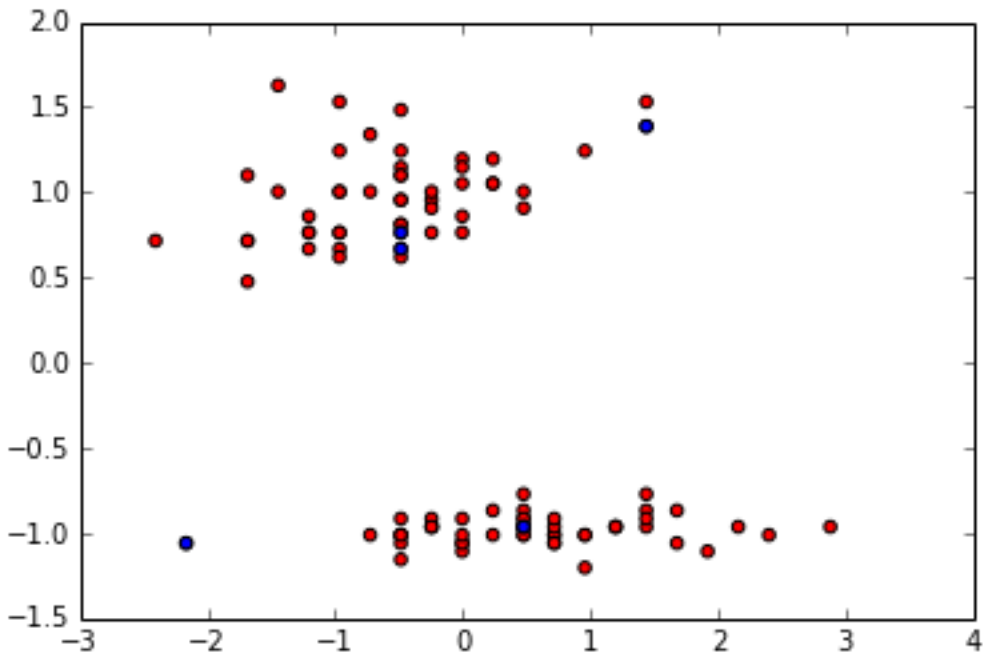

```
0.56
1.0
```

In [40]:

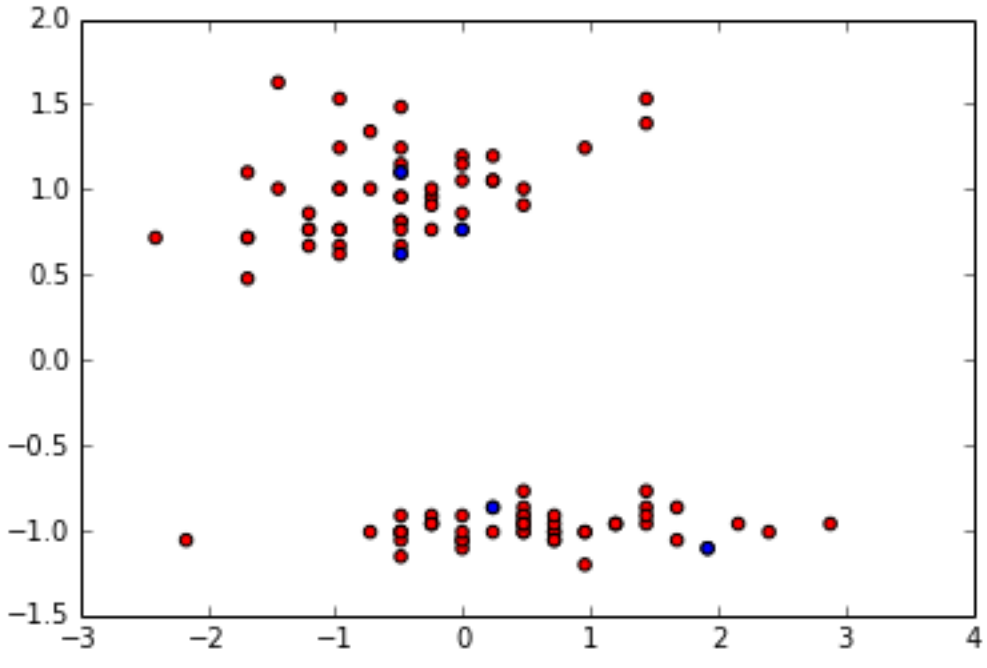
```
1 iris_data_svm('bezdekIris.data.txt',',','string',1,3,6,c=1,degree=2,sigma=None,
```

```
fold: 1
accuracy 0.56
f_measure 0.717948717949
c_matrix
[[ 0 22]
 [ 0 28]]
```

```
/Library/Python/2.7/site-packages/sklearn/metrics/classification.py:93
1: DeprecationWarning: From version 0.18, binary input will not be han
dled specially when using averaged precision/recall/F-score. Please us
e average='binary' to report only the positive class performance.
  'positive class performance.', DeprecationWarning)
```




```
fold: 1
accuracy 0.44
f_measure 0.611111111111
c_matrix
[[ 0 28]
 [ 0 22]]


/Library/Python/2.7/site-packages/sklearn/metrics/classification.py:93
1: DeprecationWarning: From version 0.18, binary input will not be han
dled specially when using averaged precision/recall/F-score. Please us
e average='binary' to report only the positive class performance.
  'positive class performance.', DeprecationWarning)
```




```
0.5
```

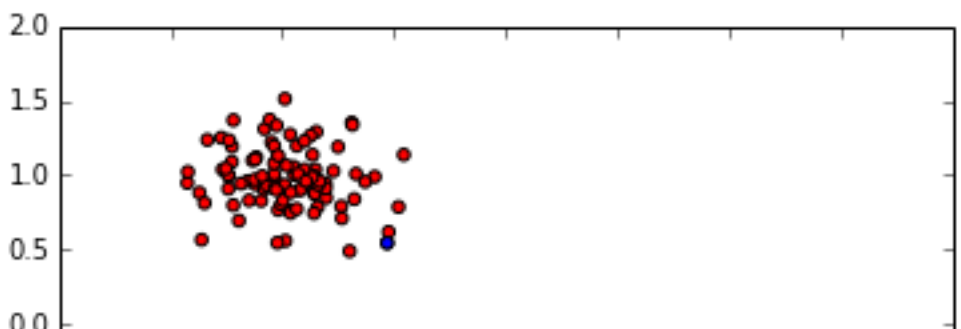
```
1.0
```

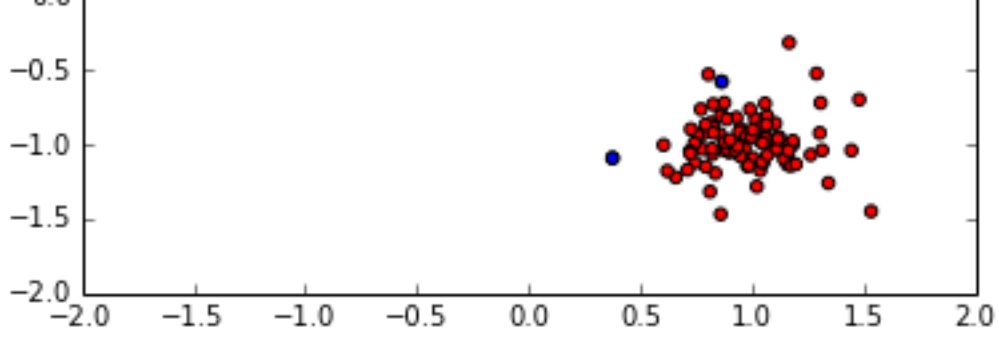
Randomly Generated Separable data (RGSD) : Linear function

In [17]:

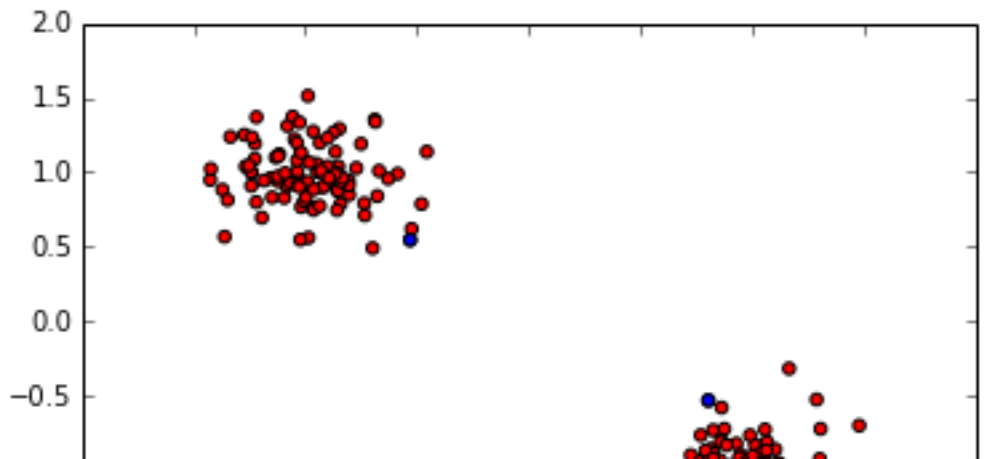
```
    iris_data_svm("",delim=None,datatype=None,label1=None,label2=None,random="sep",t
```

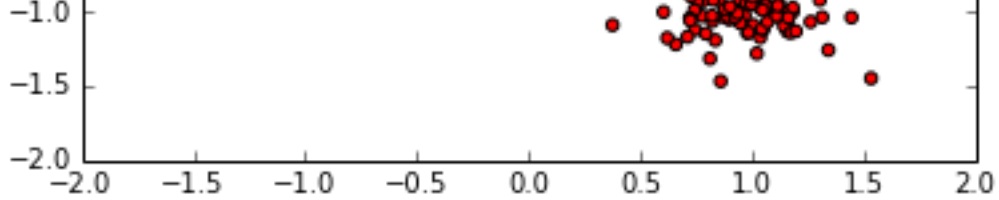
```
gram
[[ 1.62238201  1.74679794  1.74709356 ..., -1.8847827  -1.83693197
  -1.55068918]
 [ 1.74679794  1.90756076  1.88050091 ..., -2.063663   -1.88547354
  -1.60220942]
 [ 1.74709356  1.88050091  1.88140384 ..., -2.02893157 -1.98010727
  -1.67132886]
 ...,
 [-1.8847827  -2.063663   -2.02893157 ...,  2.23361946  2.01575024
   1.71514939]
 [-1.83693197 -1.88547354 -1.98010727 ...,  2.01575024  2.39786025
   1.98789743]
 [-1.55068918 -1.60220942 -1.67132886 ...,  1.71514939  1.98789743
   1.65162262]]
Y
[[ 1.  1.  1. ..., -1. -1. -1.]
 [ 1.  1.  1. ..., -1. -1. -1.]
 ...,
 [-1. -1. -1. ...,  1.  1.  1.]
 [-1. -1. -1. ...,  1.  1.  1.]
 [-1. -1. -1. ...,  1.  1.  1.]]
fold: 1
accuracy 0.75
f_measure 0.8
c_matrix
[[ 5  5]
 [ 0 10]]


/Library/Python/2.7/site-packages/sklearn/metrics/classification.py:93
1: DeprecationWarning: From version 0.18, binary input will not be han
dled specially when using averaged precision/recall/F-score. Please us
e average='binary' to report only the positive class performance.
  'positive class performance.', DeprecationWarning)
```



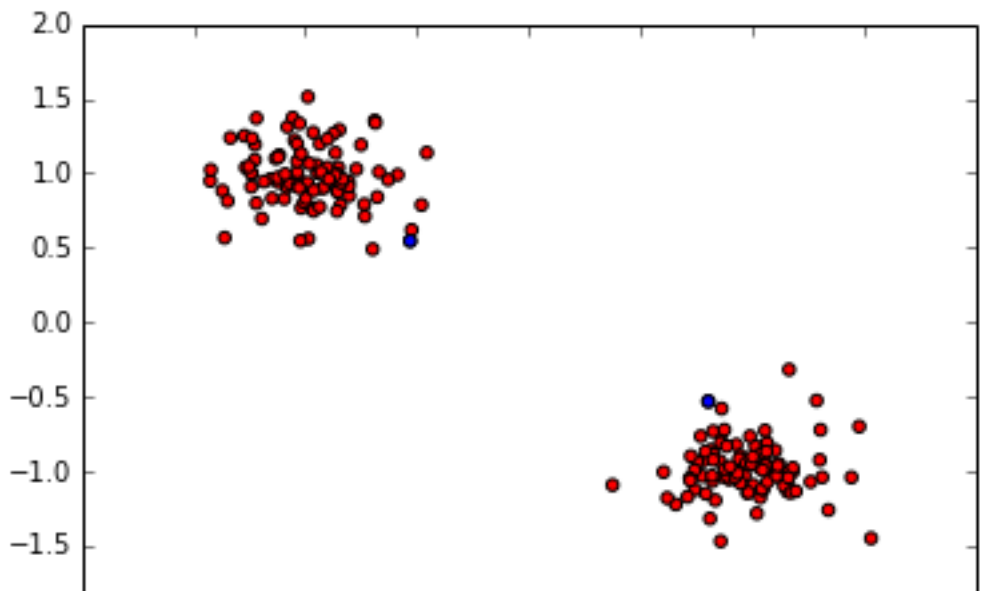


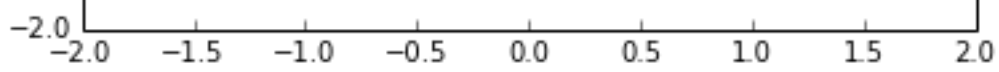

```
gram
[[ 1.62238201  1.74679794  1.74709356 ..., -1.8847827  -1.83693197
  -1.55068918]
 [ 1.74679794  1.90756076  1.88050091 ..., -2.063663   -1.88547354
  -1.60220942]
 [ 1.74709356  1.88050091  1.88140384 ..., -2.02893157 -1.98010727
  -1.67132886]
 ...,
 [-1.8847827  -2.063663   -2.02893157 ...,  2.23361946  2.01575024
   1.71514939]
 [-1.83693197 -1.88547354 -1.98010727 ...,  2.01575024  2.39786025
   1.98789743]
 [-1.55068918 -1.60220942 -1.67132886 ...,  1.71514939  1.98789743
   1.65162262]]
Y
[[ 1.  1.  1. ..., -1. -1. -1.]
 [ 1.  1.  1. ..., -1. -1. -1.]
 [ 1.  1.  1. ..., -1. -1. -1.]
 ...,
 [-1. -1. -1. ...,  1.  1.  1.]
 [-1. -1. -1. ...,  1.  1.  1.]
 [-1. -1. -1. ...,  1.  1.  1.]]
fold: 1
accuracy 1.0
f_measure 1.0
c_matrix
[[ 8  0]
 [ 0 12]]


/Library/Python/2.7/site-packages/sklearn/metrics/classification.py:93
1: DeprecationWarning: From version 0.18, binary input will not be han
dled specially when using averaged precision/recall/F-score. Please us
e average='binary' to report only the positive class performance.
  'positive class performance.', DeprecationWarning)
```



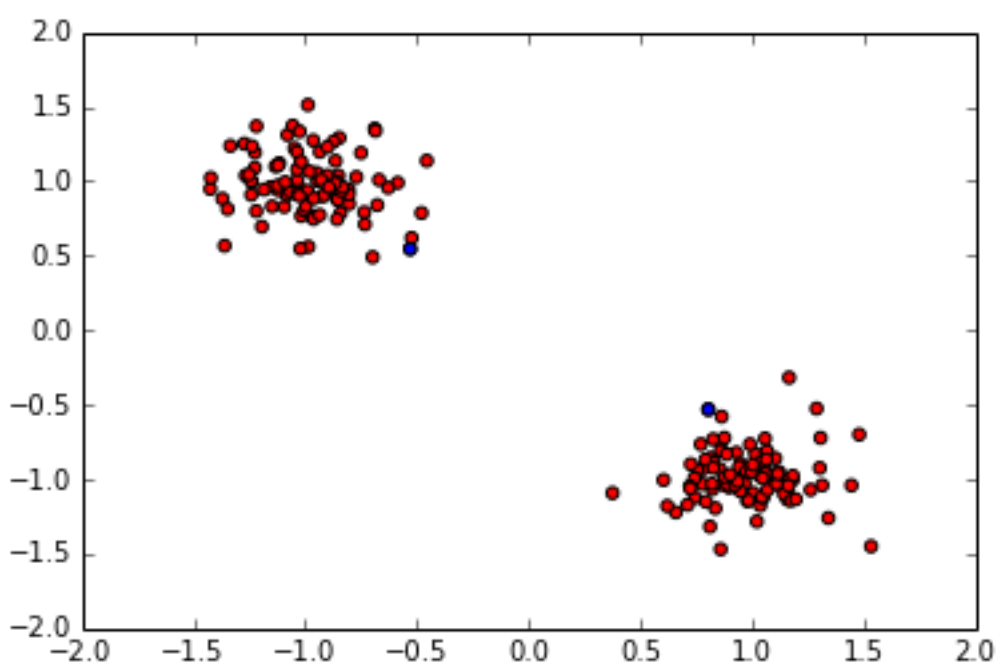



```
gram
[[ 1.62238201  1.74709356  1.70560044 ..., -1.8847827  -1.83693197
  -1.55068918]
 [ 1.74709356  1.88140384  1.83715769 ..., -2.02893157 -1.98010727
  -1.67132886]
 [ 1.70560044  1.83715769  1.80956376 ..., -1.95453696 -2.00354023
  -1.68306996]
 ...,
 [-1.8847827  -2.02893157 -1.95453696 ...,  2.23361946  2.01575024
   1.71514939]
 [-1.83693197 -1.98010727 -2.00354023 ...,  2.01575024  2.39786025
   1.98789743]
 [-1.55068918 -1.67132886 -1.68306996 ...,  1.71514939  1.98789743
   1.65162262]]
Y
[[ 1.  1.  1. ..., -1. -1. -1.]
 [ 1.  1.  1. ..., -1. -1. -1.]
 [ 1.  1.  1. ..., -1. -1. -1.]
 ...,
 [-1. -1. -1. ...,  1.  1.  1.]
 [-1. -1. -1. ...,  1.  1.  1.]
 [-1. -1. -1. ...,  1.  1.  1.]]
fold: 1
accuracy 1.0
f_measure 1.0
c_matrix
[[12  0]
 [ 0  8]]


/Library/Python/2.7/site-packages/sklearn/metrics/classification.py:93
1: DeprecationWarning: From version 0.18, binary input will not be han
dled specially when using averaged precision/recall/F-score. Please us
e average='binary' to report only the positive class performance.
  'positive class performance.', DeprecationWarning)
```

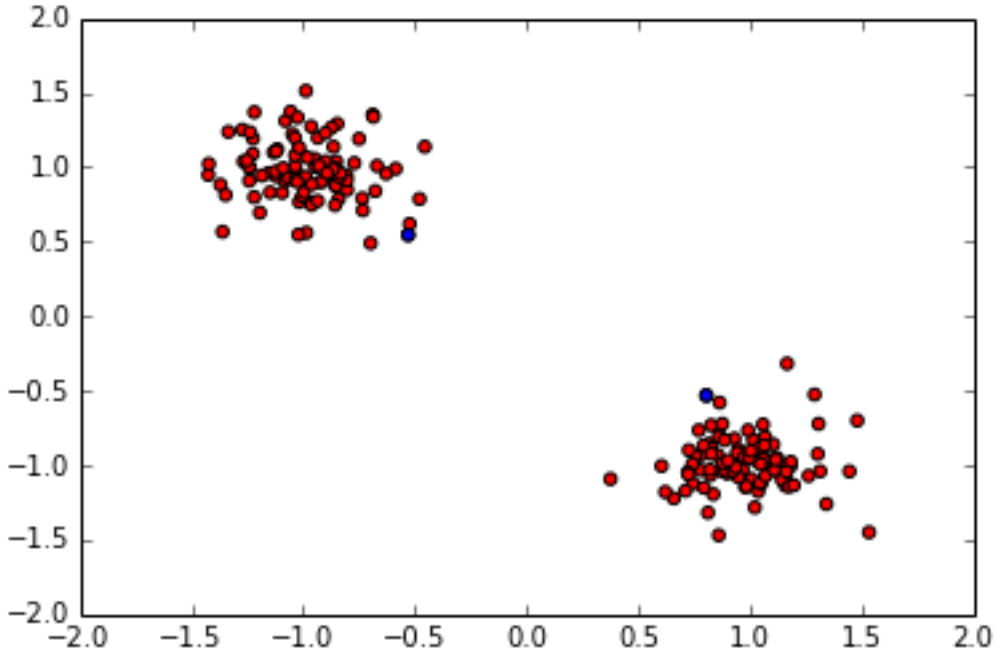





```
gram
[[ 1.62238201  1.74679794  1.74709356 ..., -1.8847827  -1.83693197
  -1.55068918]
 [ 1.74679794  1.90756076  1.88050091 ..., -2.063663   -1.88547354
  -1.60220942]
 [ 1.74709356  1.88050091  1.88140384 ..., -2.02893157 -1.98010727
  -1.67132886]
 ...,
 [-1.8847827  -2.063663   -2.02893157 ...,  2.23361946  2.01575024
   1.71514939]
 [-1.83693197 -1.88547354 -1.98010727 ...,  2.01575024  2.39786025
   1.98789743]
 [-1.55068918 -1.60220942 -1.67132886 ...,  1.71514939  1.98789743
   1.65162262]]
Y
[[ 1.  1.  1. ..., -1. -1. -1.]
 [ 1.  1.  1. ..., -1. -1. -1.]
 [ 1.  1.  1. ..., -1. -1. -1.]
 ...,
 [-1. -1. -1. ...,  1.  1.  1.]
 [-1. -1. -1. ...,  1.  1.  1.]
 [-1. -1. -1. ...,  1.  1.  1.]]
fold: 1
accuracy 1.0
f_measure 1.0
c_matrix
[[ 9  0]
 [ 0 11]]


/Library/Python/2.7/site-packages/sklearn/metrics/classification.py:93
1: DeprecationWarning: From version 0.18, binary input will not be han
dled specially when using averaged precision/recall/F-score. Please us
e average='binary' to report only the positive class performance.
  'positive class performance.', DeprecationWarning)
```

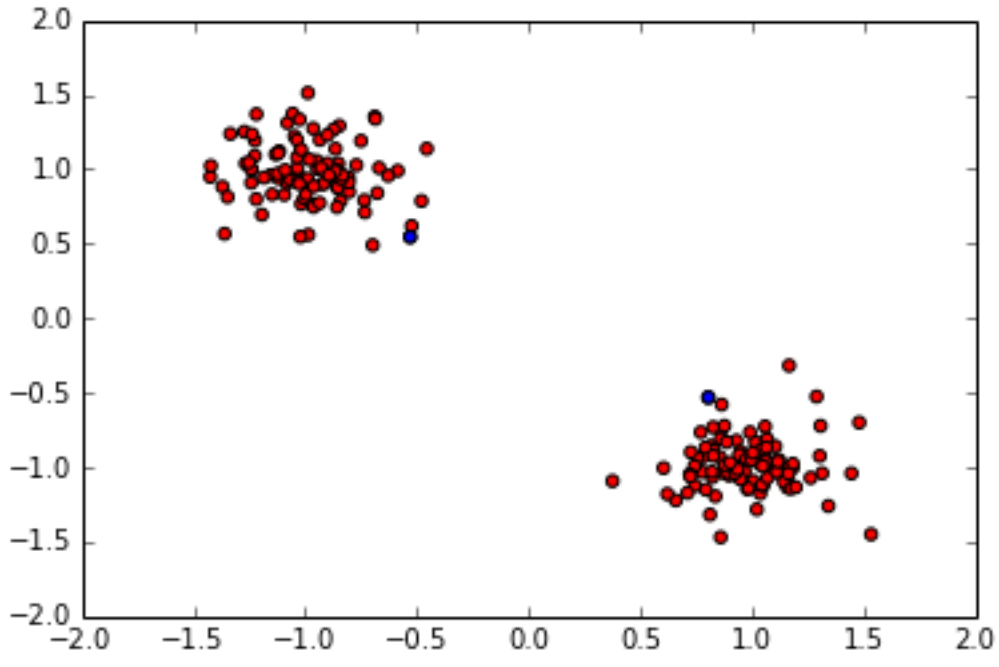


```
gram
[[ 1.62238201  1.74679794  1.74709356 ..., -2.67224858 -1.8847827
  -1.83693197]
 [ 1.74679794  1.90756076  1.88050091 ..., -2.83863279 -2.063663
  -1.88547354]
 [ 1.74709356  1.88050091  1.88140384 ..., -2.87848575 -2.02893157
  -1.98010727]
 ...,
 [-2.67224858 -2.83863279 -2.87848575 ...,  4.45691844  3.05507401
   3.15839133]
 [-1.8847827  -2.063663   -2.02893157 ...,  3.05507401  2.23361946
   2.01575024]
 [-1.83693197 -1.88547354 -1.98010727 ...,  3.15839133  2.01575024
   2.39786025]]
Y
[[ 1.  1.  1. ..., -1. -1. -1.]
 [ 1.  1.  1. ..., -1. -1. -1.]
 [ 1.  1.  1. ..., -1. -1. -1.]
 ...,
 [-1. -1. -1. ...,  1.  1.  1.]
 [-1. -1. -1. ...,  1.  1.  1.]
 [-1. -1. -1. ...,  1.  1.  1.]]
fold: 1
accuracy 1.0
f_measure 1.0
c_matrix
[[ 9  0]
 [ 0 11]]


/Library/Python/2.7/site-packages/sklearn/metrics/classification.py:93
1: DeprecationWarning: From version 0.18, binary input will not be han
dled specially when using averaged precision/recall/F-score. Please us
e average='binary' to report only the positive class performance.
  'positive class performance.', DeprecationWarning)
```



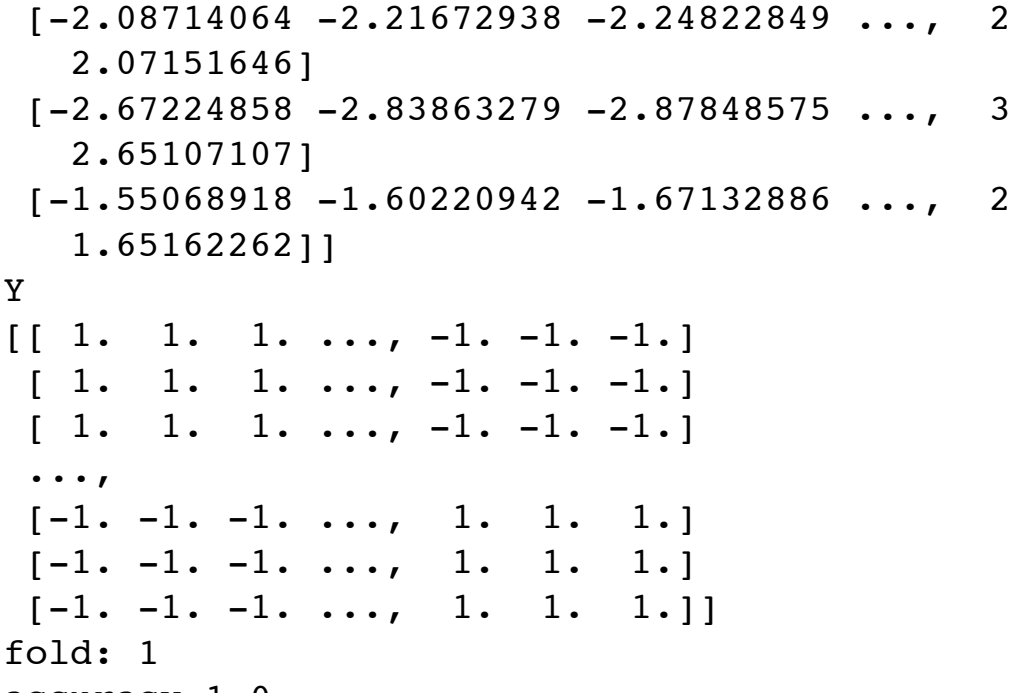

```
gram
[[ 1.62238201  1.74679794  1.74709356 ..., -1.8847827  -1.83693197
  -1.55068918]
```

```
 [ 1.74679794  1.90756076  1.88050091 ..., -2.063663   -1.88547354
  -1.60220942]
 [ 1.74709356  1.88050091  1.88140384 ..., -2.02893157 -1.98010727
  -1.67132886]
 ...,
 [-1.8847827  -2.063663   -2.02893157 ...,  2.23361946  2.01575024
   1.71514939]
 [-1.83693197 -1.88547354 -1.98010727 ...,  2.01575024  2.39786025
   1.98789743]
 [-1.55068918 -1.60220942 -1.67132886 ...,  1.71514939  1.98789743
   1.65162262]]
Y
[[ 1.  1.  1. ..., -1. -1. -1.]
 [ 1.  1.  1. ..., -1. -1. -1.]
 [ 1.  1.  1. ..., -1. -1. -1.]
 ...,
 [-1. -1. -1. ...,  1.  1.  1.]
 [-1. -1. -1. ...,  1.  1.  1.]
 [-1. -1. -1. ...,  1.  1.  1.]]
fold: 1
accuracy 1.0
f_measure 1.0
c_matrix
[[ 6  0]
 [ 0 14]]
/Library/Python/2.7/site-packages/sklearn/metrics/classification.py:93
1: DeprecationWarning: From version 0.18, binary input will not be han
dled specially when using averaged precision/recall/F-score. Please us
e average='binary' to report only the positive class performance.
  'positive class performance.', DeprecationWarning)
```



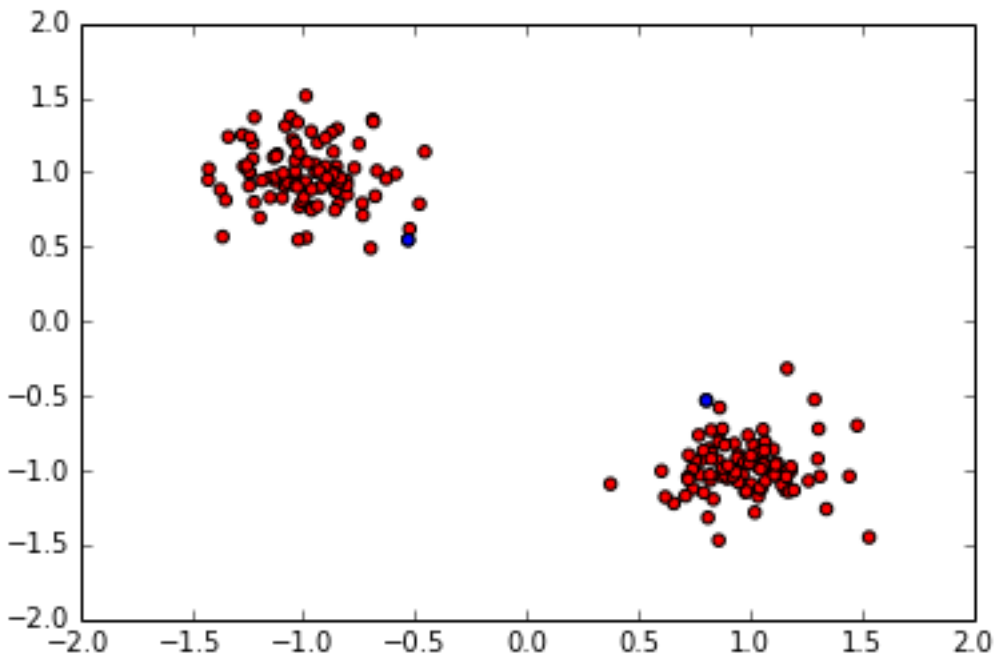

```
gram
[[ 1.62238201  1.74679794  1.74709356 ..., -2.08714064 -2.67224858
  -1.55068918]
 [ 1.74679794  1.90756076  1.88050091 ..., -2.21672938 -2.83863279
  -1.60220942]
```

```
 [ 1.74709356  1.88050091  1.88140384 ..., -2.24822849 -2.87848575
  -1.67132886]
 ...,
 [-2.08714064 -2.21672938 -2.24822849 ...,  2.71966829  3.48156832
   2.07151646]
 [-2.67224858 -2.83863279 -2.87848575 ...,  3.48156832  4.45691844
   2.65107107]
 [-1.55068918 -1.60220942 -1.67132886 ...,  2.07151646  2.65107107
   1.65162262]]
Y
[[ 1.  1.  1. ..., -1. -1. -1.]
 [ 1.  1.  1. ..., -1. -1. -1.]
 [ 1.  1.  1. ..., -1. -1. -1.]
 ...,
 [-1. -1. -1. ...,  1.  1.  1.]
 [-1. -1. -1. ...,  1.  1.  1.]
 [-1. -1. -1. ...,  1.  1.  1.]]
fold: 1
accuracy 1.0
f_measure 1.0
c_matrix
[[12  0]
 [ 0  8]]


/Library/Python/2.7/site-packages/sklearn/metrics/classification.py:93
1: DeprecationWarning: From version 0.18, binary input will not be han
dled specially when using averaged precision/recall/F-score. Please us
e average='binary' to report only the positive class performance.
  'positive class performance.', DeprecationWarning)
```



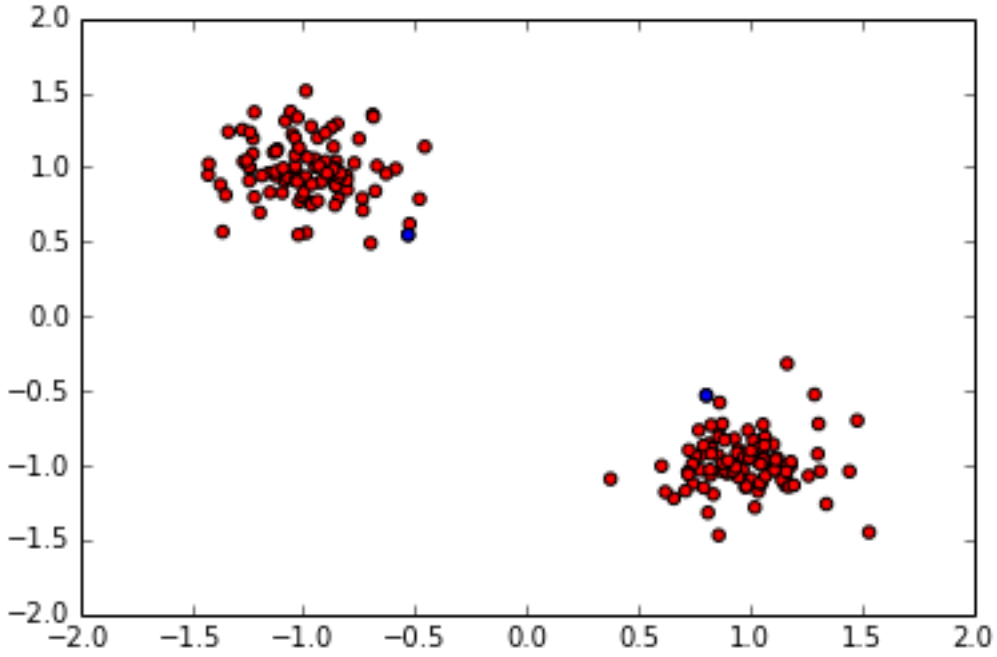

```
gram
[[ 1.62238201  1.74679794  1.74709356 ..., -1.8847827  -1.83693197
  -1.55068918]
 [ 1.74679794  1.90756076  1.88050091 ..., -2.063663   -1.88547354
  -1.60220942]
 [ 1.74709356  1.88050091  1.88140384 ..., -2.02893157 -1.98010727
  -1.67132886]
```

```
 ...,
 [-1.8847827  -2.063663   -2.02893157 ...,  2.23361946  2.01575024
   1.71514939]
 [-1.83693197 -1.88547354 -1.98010727 ...,  2.01575024  2.39786025
   1.98789743]
 [-1.55068918 -1.60220942 -1.67132886 ...,  1.71514939  1.98789743
   1.65162262]]
Y
[[ 1.  1.  1. ..., -1. -1. -1.]
 [ 1.  1.  1. ..., -1. -1. -1.]
 [ 1.  1.  1. ..., -1. -1. -1.]
 ...,
 [-1. -1. -1. ...,  1.  1.  1.]
 [-1. -1. -1. ...,  1.  1.  1.]
 [-1. -1. -1. ...,  1.  1.  1.]]
fold: 1
accuracy 1.0
f_measure 1.0
c_matrix
[[10  0]
 [ 0 10]]


/Library/Python/2.7/site-packages/sklearn/metrics/classification.py:93
1: DeprecationWarning: From version 0.18, binary input will not be han
dled specially when using averaged precision/recall/F-score. Please us
e average='binary' to report only the positive class performance.
  'positive class performance.', DeprecationWarning)
```



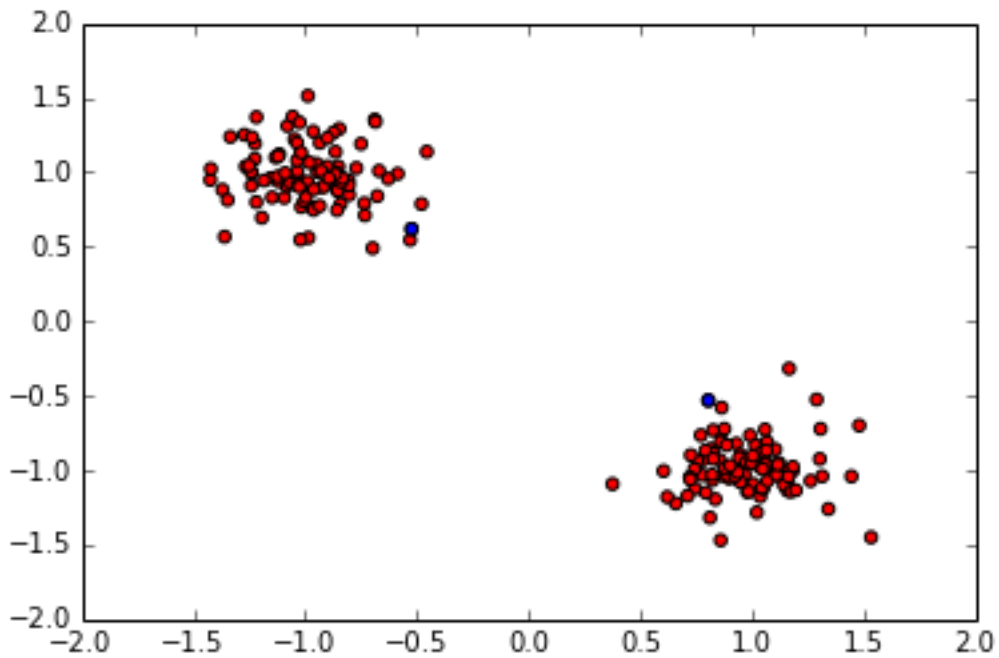

```
gram
[[ 1.90756076  1.81538246  2.24819667 ..., -2.063663   -1.88547354
  -1.60220942]
 [ 1.81538246  1.80956376  2.16056211 ..., -1.95453696 -2.00354023
  -1.68306996]
 [ 2.24819667  2.16056211  2.65504682 ..., -2.42976259 -2.27580772
  -1.92890919]
 ...,
 [-2.063663   -1.95453696 -2.42976259 ...,  2.23361946  2.01575024
   1.71514939]
```

```
 [-1.88547354 -2.00354023 -2.27580772 ...,  2.01575024  2.39786025
   1.98789743]
 [-1.60220942 -1.68306996 -1.92890919 ...,  1.71514939  1.98789743
   1.65162262]]
Y
[[ 1.  1.  1. ..., -1. -1. -1.]
 [ 1.  1.  1. ..., -1. -1. -1.]
 [ 1.  1.  1. ..., -1. -1. -1.]
 ...,
 [-1. -1. -1. ...,  1.  1.  1.]
 [-1. -1. -1. ...,  1.  1.  1.]
 [-1. -1. -1. ...,  1.  1.  1.]]
fold: 1
accuracy 1.0
f_measure 1.0
c_matrix
[[13  0]
 [ 0  7]]


/Library/Python/2.7/site-packages/sklearn/metrics/classification.py:93
1: DeprecationWarning: From version 0.18, binary input will not be han
dled specially when using averaged precision/recall/F-score. Please us
e average='binary' to report only the positive class performance.
  'positive class performance.', DeprecationWarning)
```




```
gram
[[ 1.62238201  1.74679794  1.74709356 ..., -1.8847827  -1.83693197
  -1.55068918]
 [ 1.74679794  1.90756076  1.88050091 ..., -2.063663   -1.88547354
  -1.60220942]
 [ 1.74709356  1.88050091  1.88140384 ..., -2.02893157 -1.98010727
  -1.67132886]
 ...,
 [-1.8847827  -2.063663   -2.02893157 ...,  2.23361946  2.01575024
   1.71514939]
 [-1.83693197 -1.88547354 -1.98010727 ...,  2.01575024  2.39786025
   1.98789743]
```

```
 [-1.55068918 -1.60220942 -1.67132886 ...,  1.71514939  1.98789743
   1.65162262]]
Y
[[ 1.  1.  1. ..., -1. -1. -1.]
 [ 1.  1.  1. ..., -1. -1. -1.]
 [ 1.  1.  1. ..., -1. -1. -1.]
 ...,
 [-1. -1. -1. ...,  1.  1.  1.]
 [-1. -1. -1. ...,  1.  1.  1.]
 [-1. -1. -1. ...,  1.  1.  1.]]
fold: 1
accuracy 1.0
f_measure 1.0
c_matrix
[[11  0]
 [ 0  9]]


/Library/Python/2.7/site-packages/sklearn/metrics/classification.py:93
1: DeprecationWarning: From version 0.18, binary input will not be han
dled specially when using averaged precision/recall/F-score. Please us
e average='binary' to report only the positive class performance.
  'positive class performance.', DeprecationWarning)
```




```
0.975
1.0
```

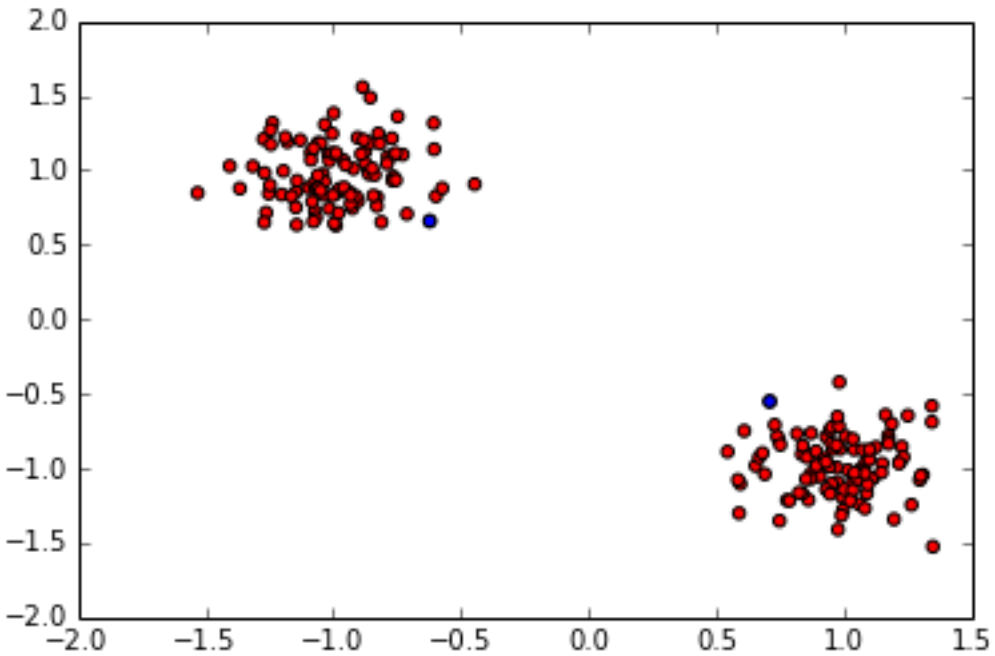
RGSD: Polynomial degree = 2

In [46]:

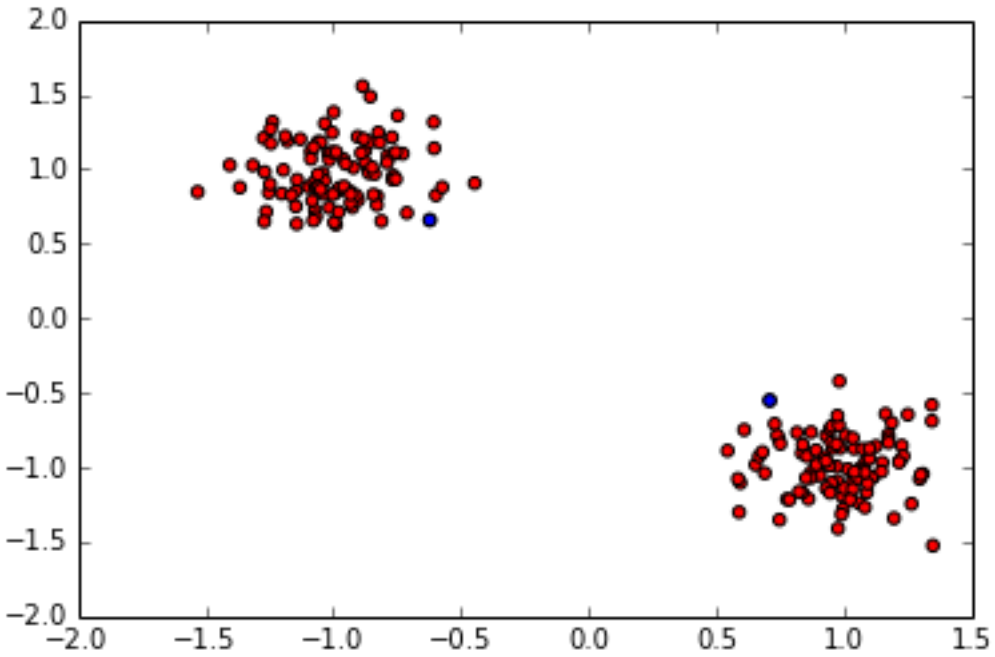
```
iris_data_svm("",delim=None,datatype=None,label1=None,label2=None,random="sep",thres
```

```
fold: 1
accuracy 0.5
precision 0.5
f_measure 0.666666666667
```

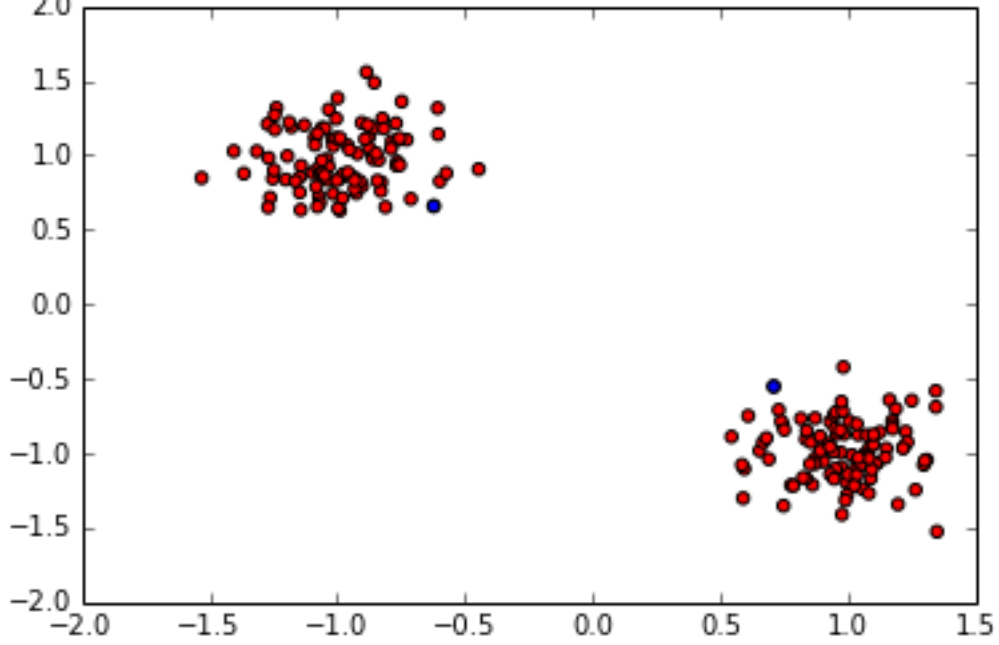
```
c_matrix
[[ 0 10]
 [ 0 10]]
```



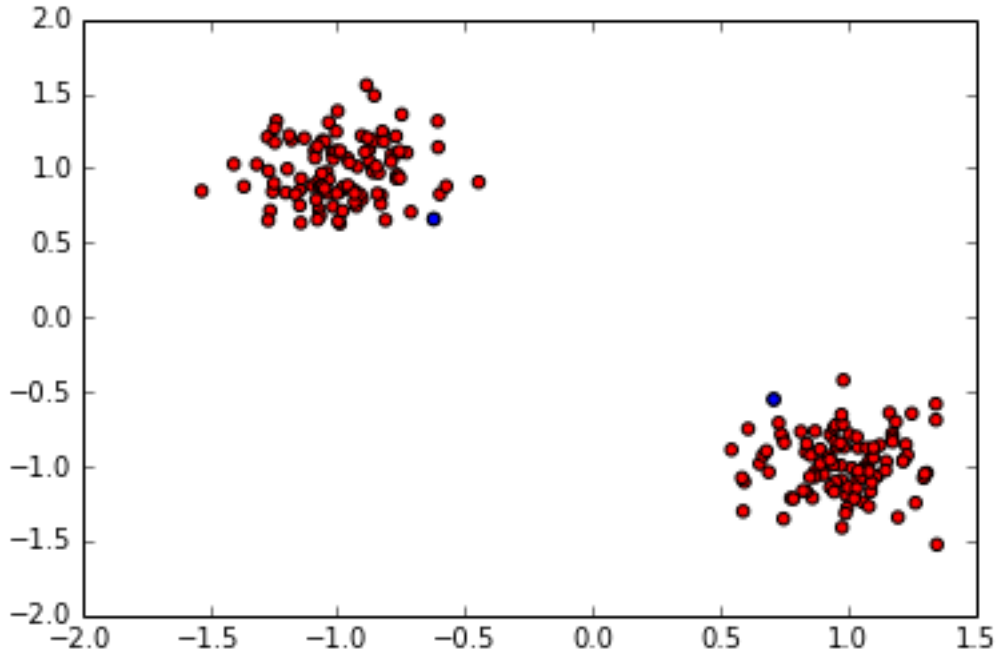

```
fold: 1
accuracy 0.6
precision 0.6
f_measure 0.75
c_matrix
[[ 0  8]
 [ 0 12]]
```



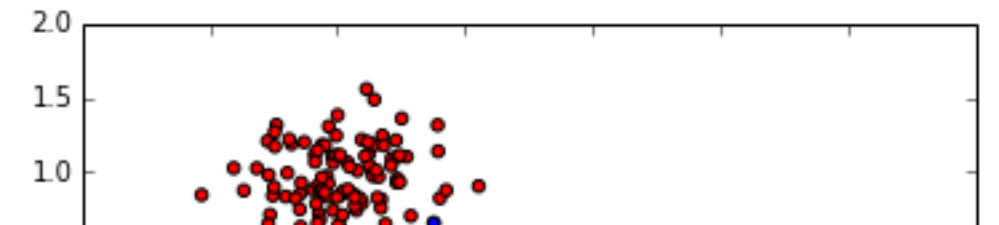

```
fold: 1
accuracy 0.4
precision 0.4
f_measure 0.571428571429
c_matrix
[[ 0 12]
 [ 0  8]]
```

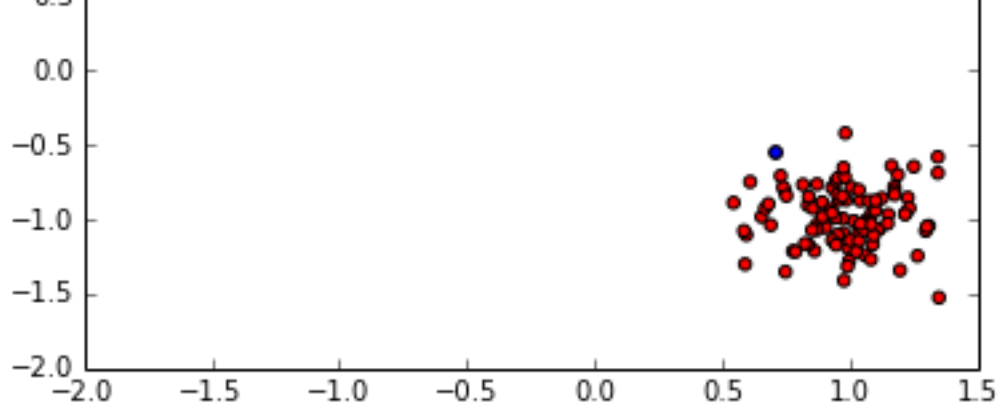



```
fold: 1
accuracy 0.55
precision 0.55
f_measure 0.709677419355
c_matrix
[[ 0  9]
 [ 0 11]]
```

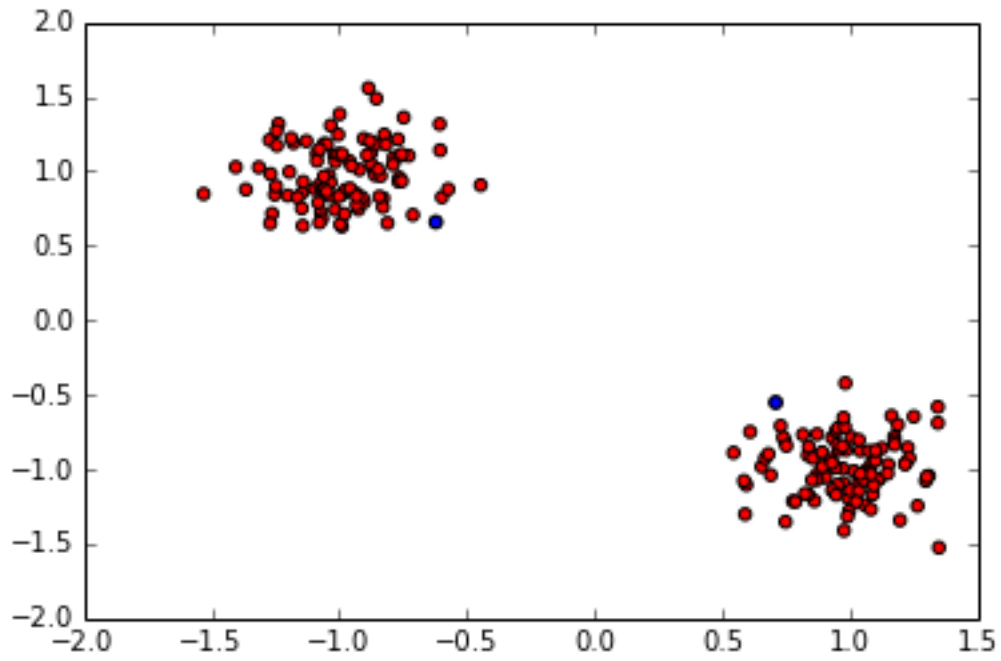



```
fold: 1
accuracy 0.55
precision 0.55
f_measure 0.709677419355
c_matrix
[[ 0  9]
 [ 0 11]]
```

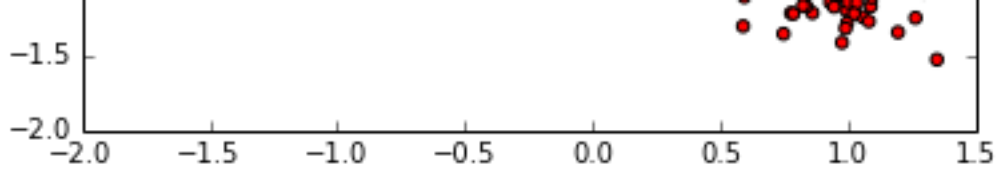





```
fold: 1
accuracy 0.7
precision 0.7
f_measure 0.823529411765
c_matrix
[[ 0  6]
 [ 0 14]]
```



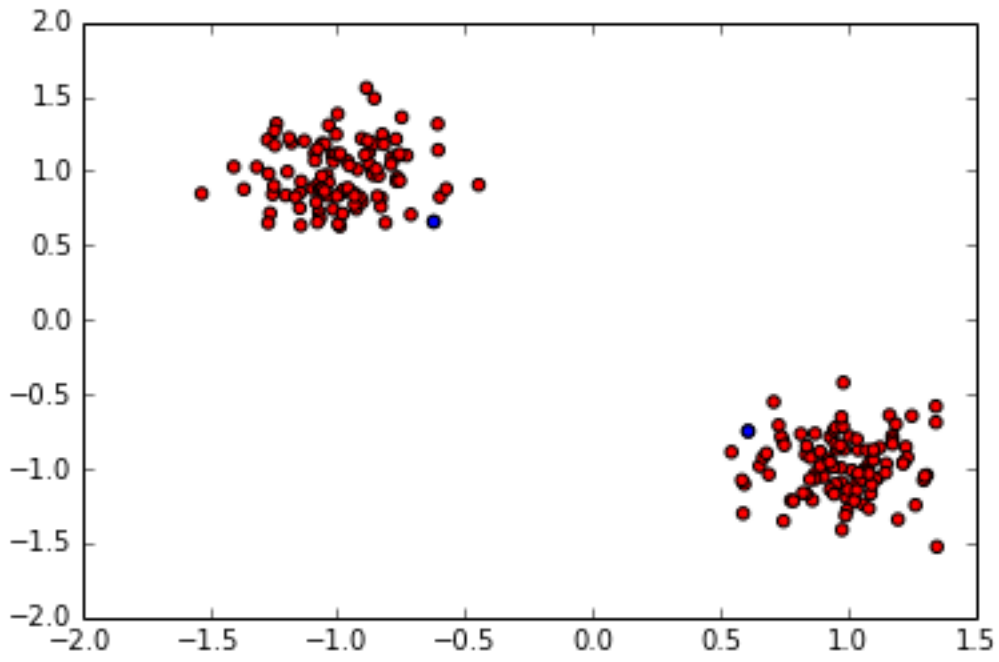

```
fold: 1
accuracy 0.4
precision 0.4
f_measure 0.571428571429
c_matrix
[[ 0 12]
 [ 0  8]]
```



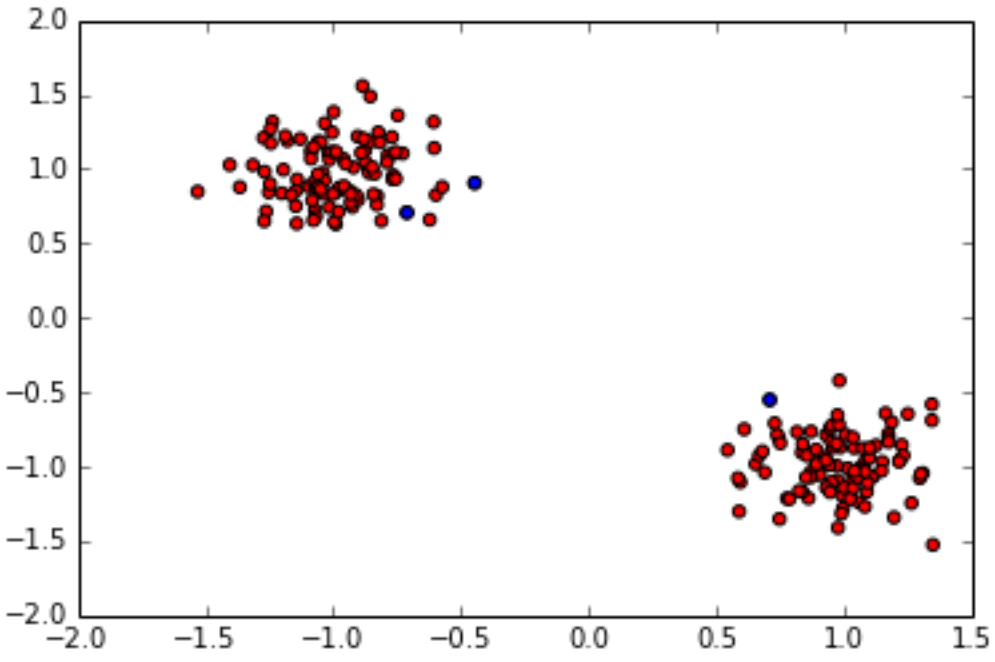

```
fold: 1
accuracy 0.5
precision 0.5
f_measure 0.666666666667
c_matrix
[[ 0 10]
 [ 0 10]]
```



```
fold: 1
accuracy 0.35
precision 0.35
f_measure 0.518518518519
c_matrix
[[ 0 13]
 [ 0  7]]
```



fold: 1
accuracy 0.45
precision 0.45
f_measure 0.620689655172
c_matrix
[[0 11]
 [0 9]]



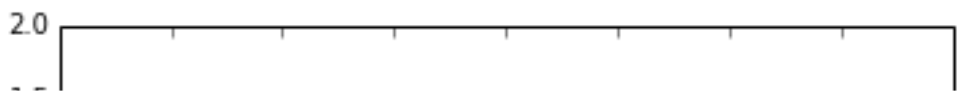
avg. accuracy 0.5
sklearn avg. accuracy 1.0

RGSD : Gaussian function

In [47]:

```
iris_data_svm("",delim=None,datatype=None,label1=None,label2=None,random="sep",thres
```

```
fold: 1
accuracy 0.5
precision 0.0
f_measure 0.0
c_matrix
[[10  0]
 [10  0]]


/Library/Python/2.7/site-packages/sklearn/metrics/classification.py:95
8: UndefinedMetricWarning: Precision is ill-defined and being set to 0
.0 due to no predicted samples.
  'precision', 'predicted', average, warn_for)
/Library/Python/2.7/site-packages/sklearn/metrics/classification.py:95
8: UndefinedMetricWarning: F-score is ill-defined and being set to 0.0
due to no predicted samples.
  'precision', 'predicted', average, warn_for)
```

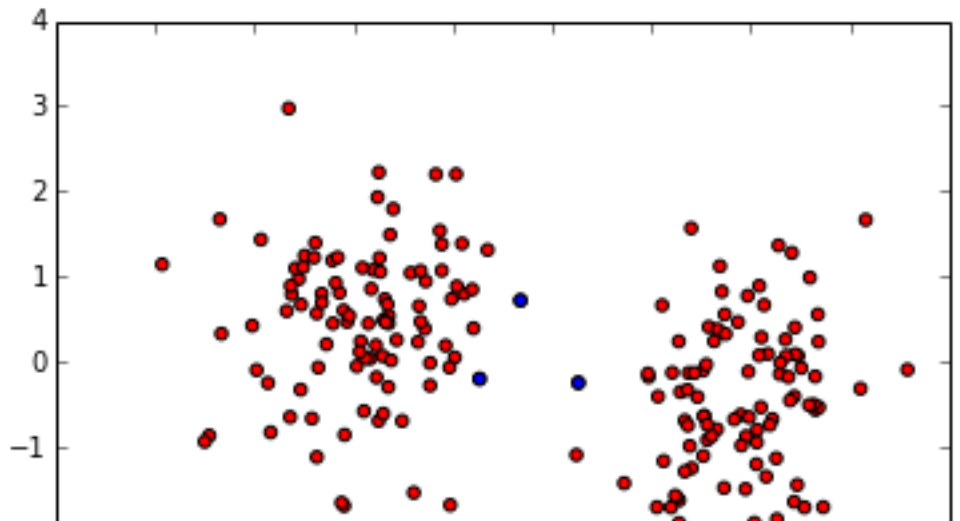


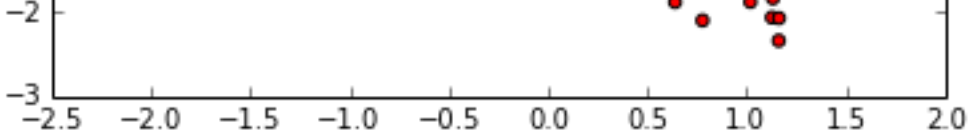
Randomly Generated Non-Separable Data (RGNSD) : Linear function

In [44]:

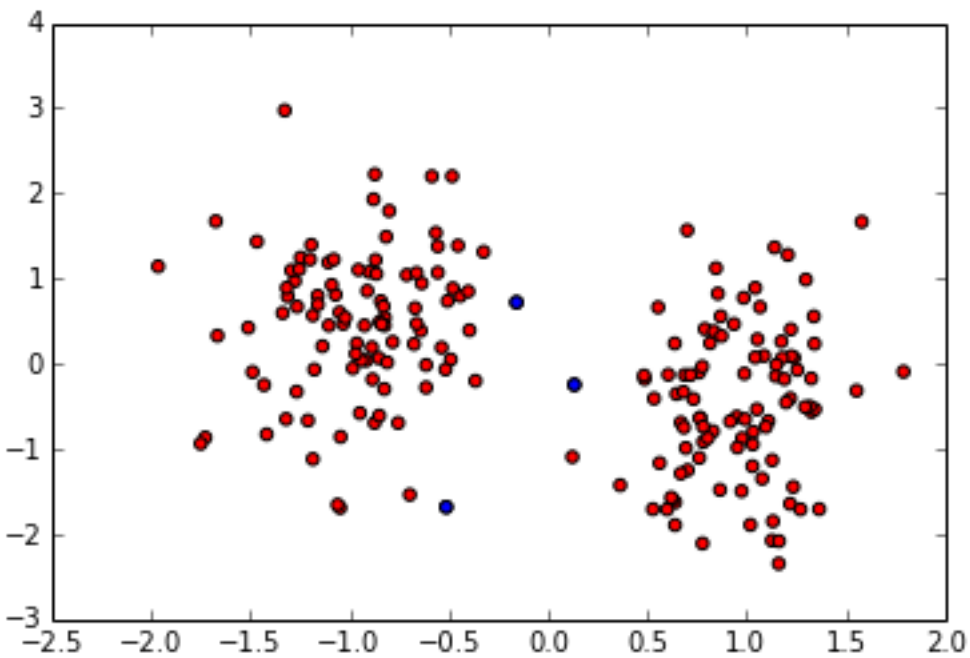
```
iris_data_svm("",delim=None,datatype=None,label1=None,label2=None,random="nonsep",tl
```

```
fold: 1
accuracy 1.0
precision 1.0
f_measure 1.0
c_matrix
[[10  0]
 [ 0 10]]
```

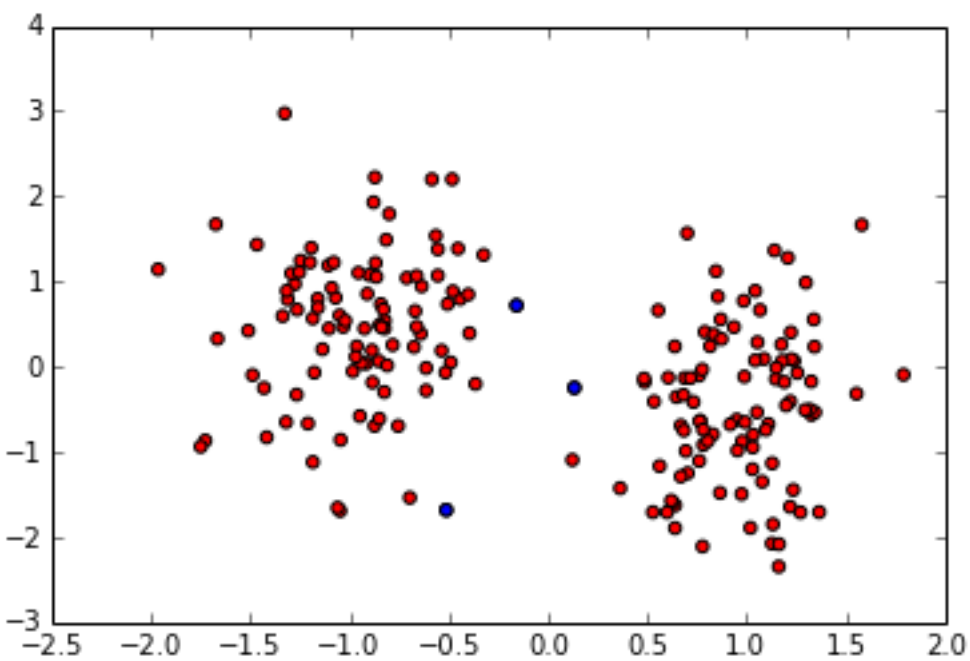




```
fold: 1
accuracy 1.0
precision 1.0
f_measure 1.0
c_matrix
[[ 8  0]
 [ 0 12]]
```

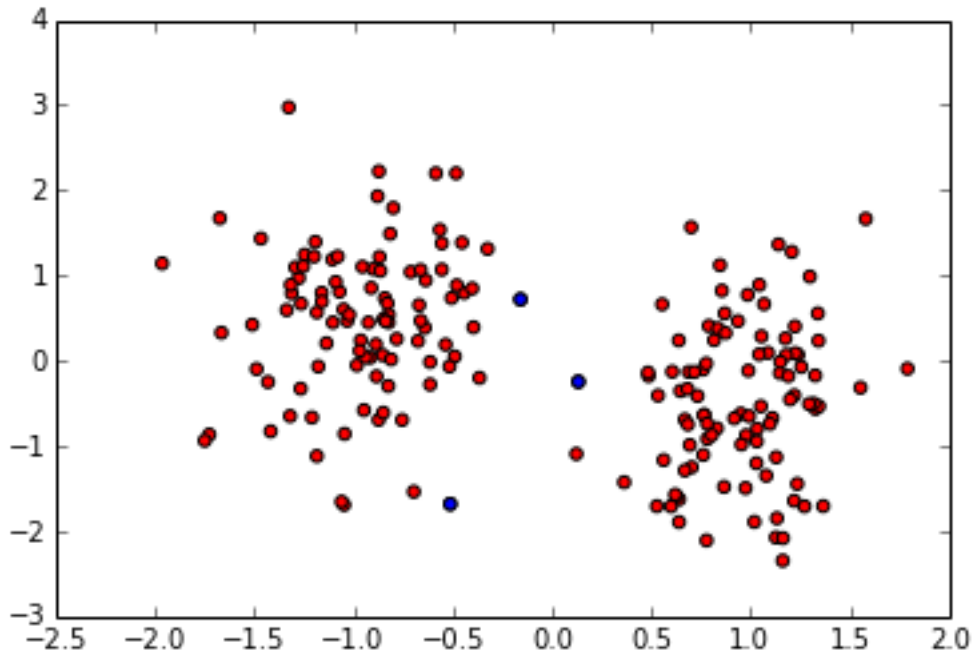


```
fold: 1
accuracy 1.0
precision 1.0
f_measure 1.0
c_matrix
[[12  0]
 [ 0  8]]
```

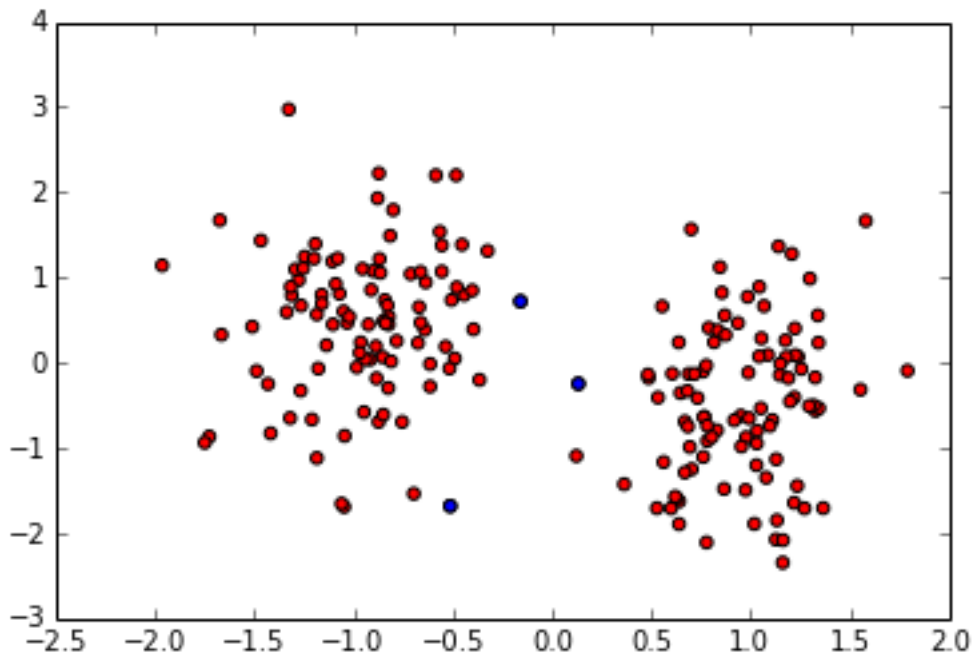


```
fold: 1
```

```
accuracy 1.0
precision 1.0
f_measure 1.0
c_matrix
[[ 9  0]
 [ 0 11]]
```

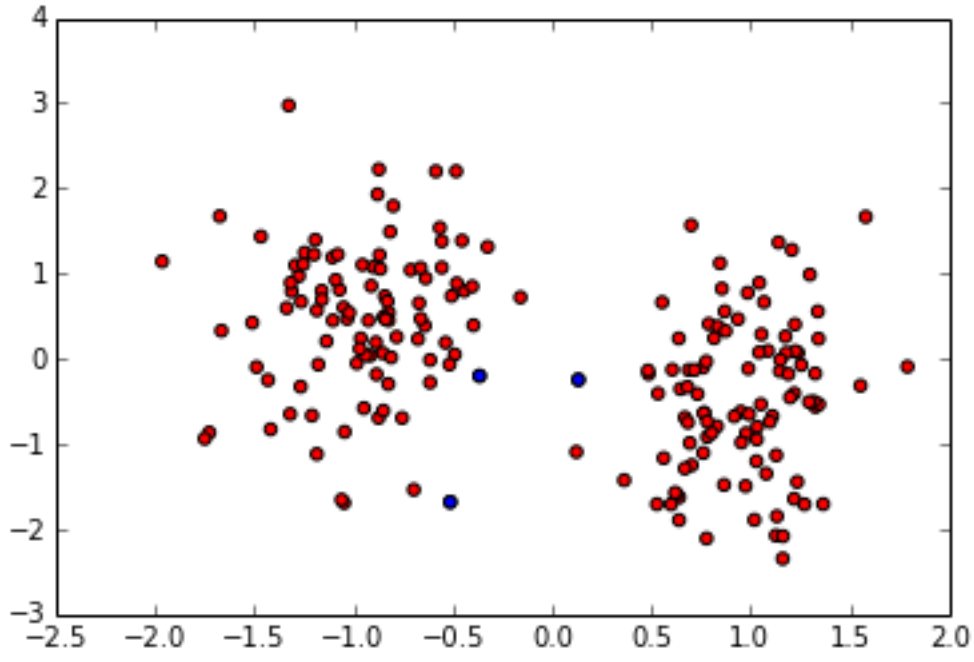



```
fold: 1
accuracy 1.0
precision 1.0
f_measure 1.0
c_matrix
[[ 9  0]
 [ 0 11]]
```

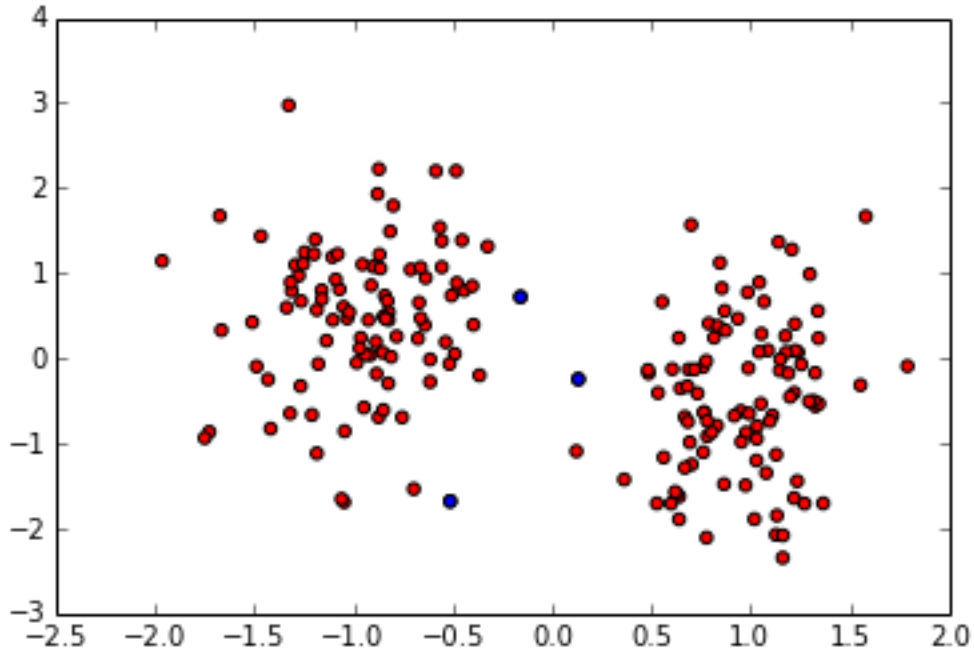



```
fold: 1
accuracy 1.0
precision 1.0
f_measure 1.0
c_matrix
[[ 6  0]
```

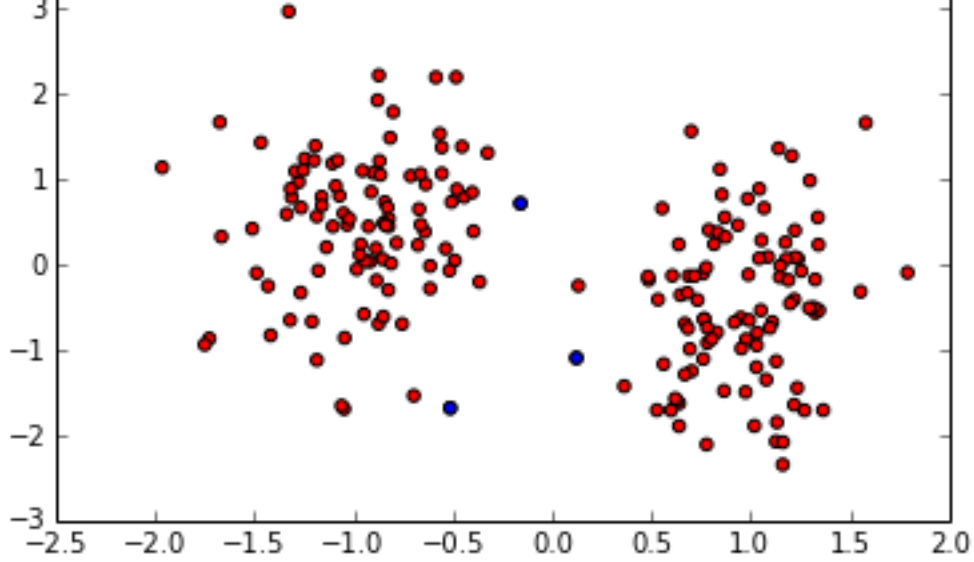
```
 [ 0 14]]
```



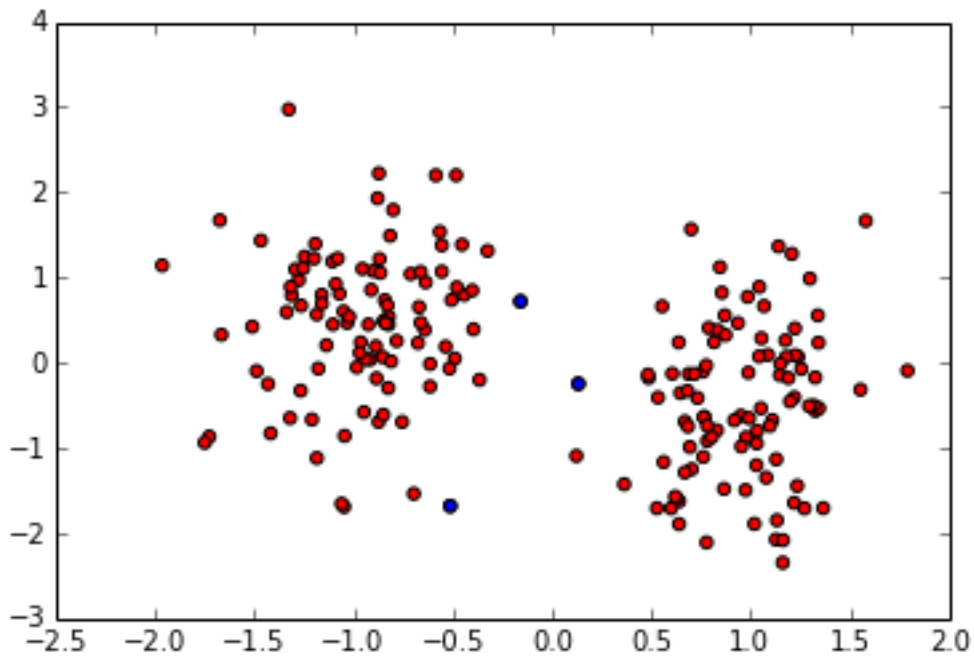

```
fold: 1
accuracy 1.0
precision 1.0
f_measure 1.0
c_matrix
[[12  0]
 [ 0  8]]
```



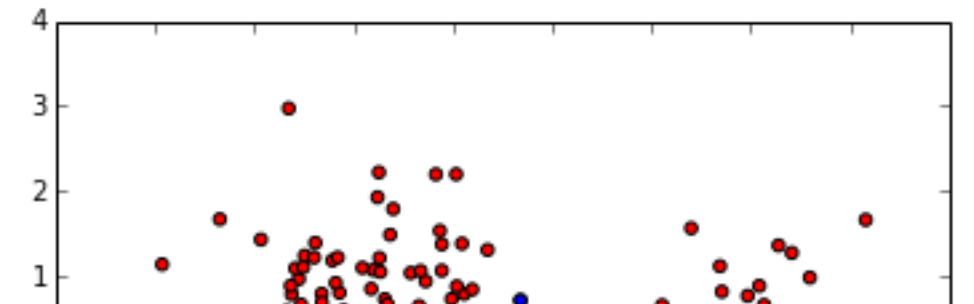

```
fold: 1
accuracy 0.95
precision 0.909090909091
f_measure 0.952380952381
c_matrix
[[ 9  1]
 [ 0 10]]
```

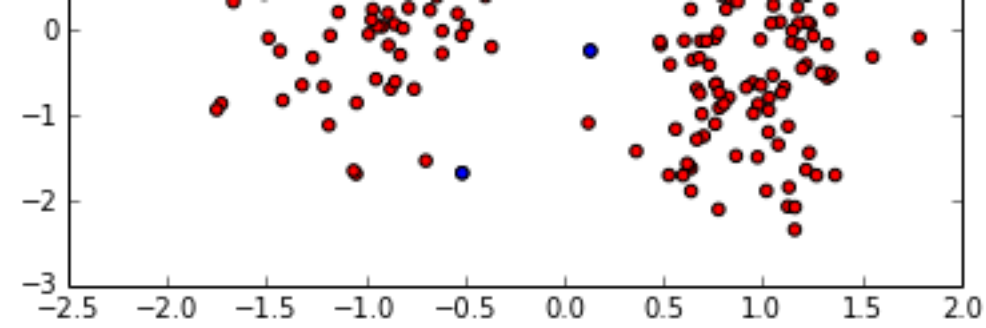



```
fold: 1
accuracy 1.0
precision 1.0
f_measure 1.0
c_matrix
[[13  0]
 [ 0  7]]
```




```
fold: 1
accuracy 1.0
precision 1.0
f_measure 1.0
c_matrix
[[11  0]
 [ 0  9]]
```





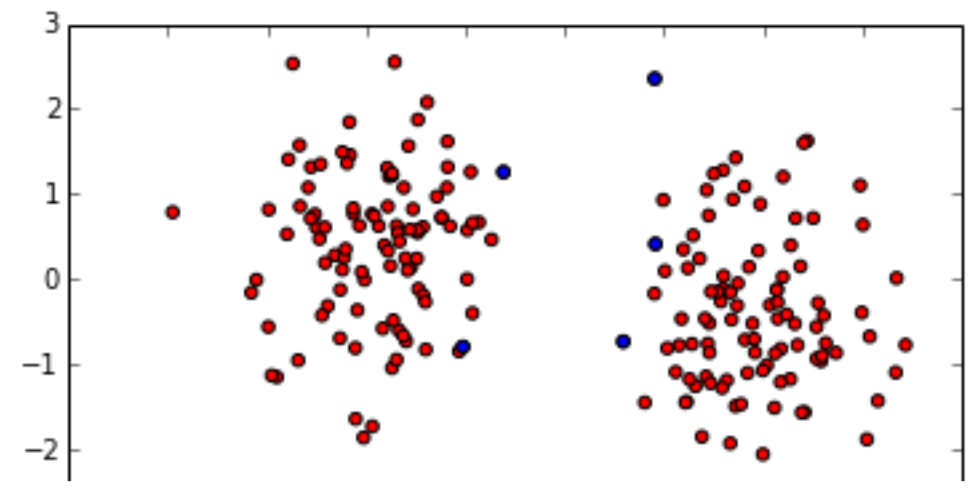

```
avg. accuracy 0.995
sklearn avg. accuracy 1.0
```

RGNSD: Polynomial degree = 2

In [48]:

```
iris_data_svm("",delim=None,datatype=None,label1=None,label2=None,random="nonsep",tl
```

```
fold: 1
accuracy 0.5
precision 0.5
f_measure 0.666666666667
c_matrix
[[ 0 10]
 [ 0 10]]
```

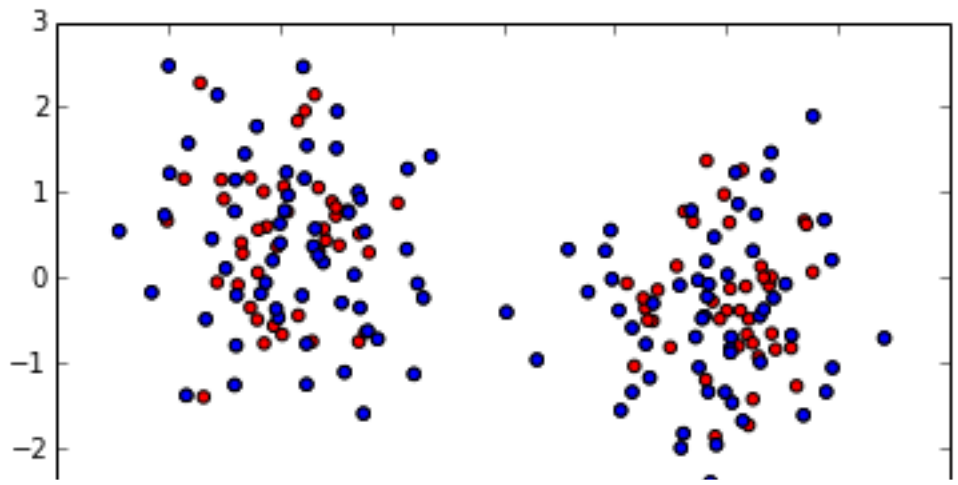


RGNSD: Gaussian function

In [59]:

```
iris_data_svm("",delim=None,datatype=None,label1=None,label2=None,random="nonsep",tl
```

```
fold: 1
accuracy 0.5
precision 0.5
f_measure 0.666666666667
c_matrix
[[ 0 10]
 [ 0 10]]
```



In [63]:

```
# iris_data_svm("",delim=None,datatype=None,label1=None,label2=None,random="nonsep"
```

In [ ]:

```

```