In [55]:
```
import numpy as np
from matplotlib import pyplot as plt
from sklearn.linear_model import LinearRegression
```

In [66]:
```
in_data = []
with open('in.dta','r') as f:
    for line in f:
        in_data.append([float(i) for i in line.split()])
in_data = np.asarray(in_data)
```

In [67]:
```
out_data = []
with open('out.dta','r') as f:
    for line in f:
        out_data.append([float(i) for i in line.split()])
out_data = np.asarray(out_data)
```

In [69]:
```
#print(out_data)
```

In [70]:
```
def nonlinear_transform(X):
    new_X = []
    for x in X:
        x1 = x[0]
        x2 = x[1]
        new_x = [1, x1, x2, x1**2, x2**2, x1*x2, np.abs(x1-x2), np.abs(x1+x2
        new_X.append(new_x)
    return np.asarray(new_X)
```

In [71]:
```
def linear_regression(X, y):
    X_plus = np.linalg.inv(X.transpose().dot(X)).dot(X.transpose())
    w = X_plus.dot(y)
    return(w)
```

In [72]:
```
def linear_regression_weight_decay(X, y, l):
    X_plus = np.linalg.inv(X.transpose().dot(X) + l*np.eye(X.shape[1])).dot(
    w = X_plus.dot(y)
    return(w)
```

In [73]:
```
def get_error(X, w, y):
    err = 0
    count = 0
    for x in X:
        err += (w.dot(x) - y[count])**2
        count += 1
    err = err/float(count)
    return err
```

In [104]:
```
def get_classification_error(X, w, y):
    correct_idx = []
    count = 0
    for x in X:
        if np.sign(w.dot(x)) == y[count]:
            correct_idx.append(count)
        count += 1
    class_err = 1-len(correct_idx)/float(count)
    return class_err
```

In [105]:
```
def plot_target_in_x_space(x,y):
    neg_x = [x[i,0] for i in range(len(x[:,0])) if y[i] < 0]
    pos_x = [x[i,0] for i in range(len(x[:,0])) if y[i] > 0]
    neg_y = [x[i,1] for i in range(len(x[:,1])) if y[i] < 0]
    pos_y = [x[i,1] for i in range(len(x[:,1])) if y[i] > 0]
    plt.plot(neg_x, neg_y,'r*')
    plt.plot(pos_x, pos_y,'b*')
    plt.title('Target data')
    plt.show()
```

In [106]:
```
def plot_disagreement_in_x_space(X,y,Z,w):
    correct_idx = []
    count = 0
    for x in Z:
        if np.sign(w.dot(x)) == y[count]:
            correct_idx.append(count)
        count += 1
    neg_x = [X[i,0] for i in range(len(X[:,0])) if i not in correct_idx]
    pos_x = [X[i,0] for i in range(len(X[:,0])) if i in correct_idx]
    neg_y = [X[i,1] for i in range(len(X[:,1])) if i not in correct_idx]
    pos_y = [X[i,1] for i in range(len(X[:,1])) if i in correct_idx]
    plt.plot(neg_x, neg_y,'g*', label = 'correct')
    plt.plot(pos_x, pos_y,'y*', label = 'incorrect')
    plt.title('Linear Regression misclassified points')
    plt.xlabel('x_1')
    plt.ylabel('x_2')
    plt.legend()
    plt.savefig('Linear_Regression_disagreement.jpg')
    plt.show()
```

In [107]:
```
#using my own implementation
X = in_data[:,:2]
y = in_data[:,2]
#plot_target_in_x_space(X,y)
Z = nonlinear_transform(X)
#print(X.shape, y.shape, X[0,:], y[0])
w = linear_regression(Z,y)
in_sample_err = get_classification_error(Z,w,y)
print(in_sample_err)
#plot_disagreement_in_x_space(X,y,Z,w)
test_X = out_data[:,:2]
test_Z = nonlinear_transform(test_X)
test_y = out_data[:,2]
#plot_target_in_x_space(test_X,test_y)
test_err = get_classification_error(test_Z,w,test_y)
#plot_disagreement_in_x_space(test_X,test_y,test_Z,w)
print(test_err)
```

```
0.0285714285714
0.084
```

In [112]:
```
#using my own implementation, with weight decay
k = -3
l = 10**k
X = in_data[:,:2]
X = nonlinear_transform(X)
y = in_data[:,2]
#print(X.shape, y.shape, X[0,:], y[0])
w = linear_regression_weight_decay(X,y,l)
in_sample_err = get_classification_error(X,w,y)
print(in_sample_err)
test_X = nonlinear_transform(out_data[:,:2])
test_y = out_data[:,2]
test_err = get_classification_error(test_X,w,test_y)
print(test_err)
```

```
0.0285714285714
0.08
```

In [109]:
```
#using my own implementation, with weight decay
k = 3
l = 10**k
X = in_data[:,:2]
X = nonlinear_transform(X)
y = in_data[:,2]
#print(X.shape, y.shape, X[0,:], y[0])
w = linear_regression_weight_decay(X,y,l)
in_sample_err = get_classification_error(X,w,y)
print(in_sample_err)
test_X = nonlinear_transform(out_data[:,:2])
test_y = out_data[:,2]
test_err = get_classification_error(test_X,w,test_y)
print(test_err)
```

```
0.371428571429
0.436
```

In [110]:
```
#using my own implementation, with weight decay
k = -2
for k in [2,1,0,-1,-2]:
    l = 10**k
    X = in_data[:,:2]
    X = nonlinear_transform(X)
    y = in_data[:,2]
    #print(X.shape, y.shape, X[0,:], y[0])
    w = linear_regression_weight_decay(X,y,l)
    in_sample_err = get_classification_error(X,w,y)
    #print(in_sample_err)
    test_X = nonlinear_transform(out_data[:,:2])
    test_y = out_data[:,2]
    test_err = get_classification_error(test_X,w,test_y)
    print(k,test_err)
```

```
(2, 0.22799999999999998)
(1, 0.124)
(0, 0.09199999999999997)
(-1, 0.05600000000000005)
(-2, 0.08399999999999996)
```

In [ ]:
```

```