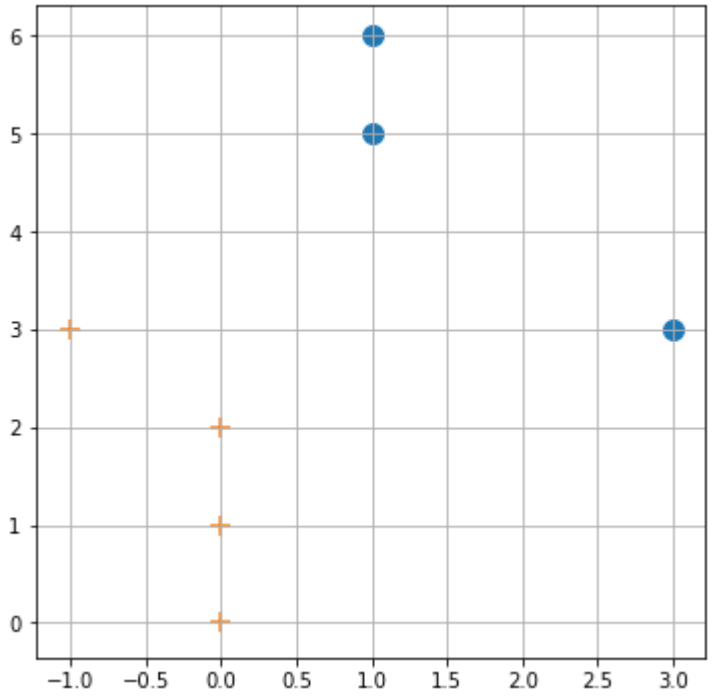
In [13]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import sklearn
from sklearn import svm
from sklearn import metrics
from sklearn.model_selection import train_test_split
```

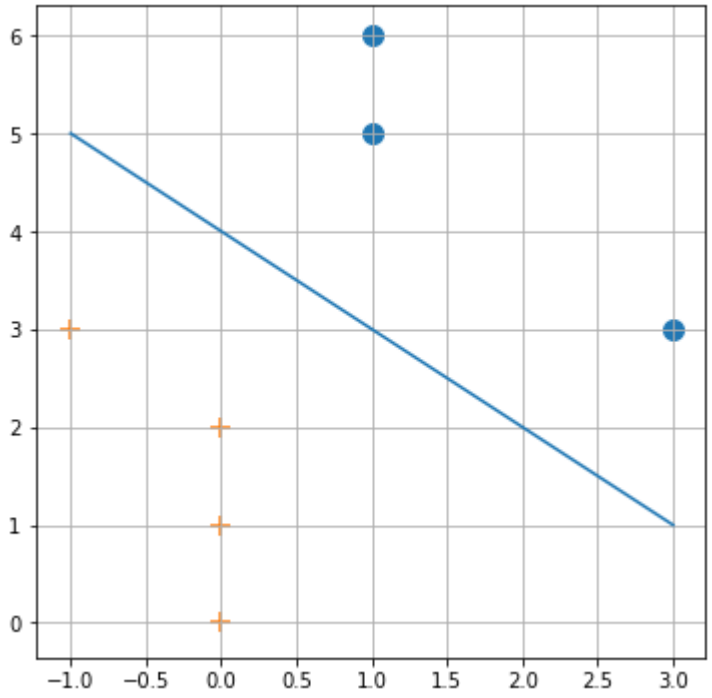
In [14]:
```
#Q1
neg_class = np.array([[1,5],[1,6],[3,3]])
pos_class = np.array([[-1,3],[0,2],[0,1],[0,0]])

#print(neg_class)
#print(pos_class)
```

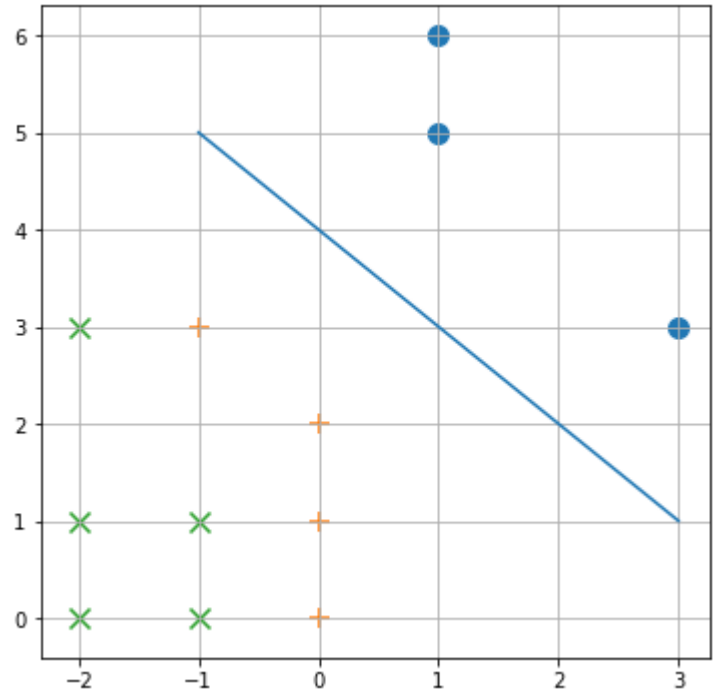
In [15]:
```
fig, ax = plt.subplots(1, 1, figsize=(6, 6))
ax.scatter(neg_class[:, 0], neg_class[:, 1], s = 100, marker = 'o')
ax.scatter(pos_class[:, 0], pos_class[:, 1], s = 100, marker = '+')
plt.grid()
```



In [16]:
```
fig, ax = plt.subplots(1, 1, figsize=(6, 6))
ax.scatter(neg_class[:, 0], neg_class[:, 1], s = 100, marker = 'o')
ax.scatter(pos_class[:, 0], pos_class[:, 1], s = 100, marker = '+')
ax.plot([-1,3],[5,1]) #by inspection, this is the linear classifier that separate
plt.grid()
```



In [17]:
```
new_data = np.array([[-2,0],[-2,1],[-2,3],[-1,0],[-1,1]])
fig, ax = plt.subplots(1, 1, figsize=(6, 6))
ax.scatter(neg_class[:, 0], neg_class[:, 1], s = 100, marker='o')
ax.scatter(pos_class[:, 0], pos_class[:, 1], s = 100, marker = '+')
ax.scatter(new_data[:, 0], new_data[:, 1], s = 100, marker = 'x')
ax.plot([-1,3],[5,1])
ax.grid()
```



In [18]:
```
clf = svm.SVC(kernel = 'linear') # Linear Kernel
X_train = np.concatenate((neg_class, pos_class),axis=0)
classes = [-1, -1, -1, 1, 1, 1, 1]

#Train the model using the training sets
clf.fit(X_train, classes)
print(clf.support_vectors_) #double check
print(clf.intercept_) #double check
print(clf.coef_) #double check
```

```
[[1. 5.]
 [3. 3.]
 [0. 2.]]
[1.99925333]
[[-0.49984 -0.49984]]
```

In [19]:
```
new_clf = svm.SVC(kernel = 'linear') # Linear Kernel
new_X_train = np.concatenate((neg_class, pos_class, new_data),axis=0)
new_classes = [-1, -1, -1, 1, 1, 1, 1, 1, 1, 1, 1, 1]

#Train the model using the training sets
new_clf.fit(new_X_train, new_classes)
print(new_clf.support_vectors_) #double check
print(new_clf.intercept_) #double check
print(new_clf.coef_) #double check
```

```
[[1. 5.]
 [3. 3.]
 [0. 2.]]
[1.99925333]
[[-0.49984 -0.49984]]
```

In [20]:
```
#Q3
#reading the data file into df
df = pd.read_csv("wdbc.data", delimiter = ",", header = None)
display(df)
```

	0	1	2	3	4	5	6	7	8	9	...	22	
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	...	25.380	17
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	...	24.990	23
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	...	23.570	25
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	...	14.910	26
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	...	22.540	16
...	...	...	...	...	...	...	...	...	...	...	...	...	
564	926424	M	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	...	25.450	26
565	926682	M	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	...	23.690	38
566	926954	M	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	...	18.980	34
567	927241	M	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	...	25.740	39
568	92751	B	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	...	9.456	30

569 rows × 32 columns

In [21]:
```
y = df[1].values
x = df.iloc[:, 2:32]
x = np.array(x)

#print(y)
#print(type(y))
#print(x)
#print(type(x))

rs_cols = ["train accuracy", "test accuracy", "precision", "recall"] #column name
rs_rows = ["SVM1", "SVM2", "SVM3"] #row names for the result table
rs_table = pd.DataFrame(columns=rs_cols, index=rs_rows) #creating the result tabl

display(rs_table)
```

	train accuracy	test accuracy	precision	recall
SVM1	NaN	NaN	NaN	NaN
SVM2	NaN	NaN	NaN	NaN
SVM3	NaN	NaN	NaN	NaN

In [22]:
```
print("Calculating for SVM1....")

clf = svm.SVC(kernel = 'linear') # Linear Kernel

train_acc = []
test_acc = []
prec = []
recall = []

scaler = sklearn.preprocessing.StandardScaler()
x_sc = scaler.fit_transform(x)


for i in range(20):
    (x_train, x_test, y_train, y_test) = train_test_split(x_sc, y, test_size = 0.
    clf = clf.fit(x_train, y_train)

    #for train dataset
    y_train_pred = clf.predict(x_train)
    train_acc.append(metrics.accuracy_score(y_train, y_train_pred))

    #for test dataset
    y_test_pred = clf.predict(x_test)

    test_acc.append(metrics.accuracy_score(y_test, y_test_pred))
    prec.append(metrics.precision_score(y_test, y_test_pred, pos_label = 'M'))
    recall.append(metrics.recall_score(y_test, y_test_pred, pos_label = 'M'))

#compute average of 20 performance
mean_train_acc = np.mean(train_acc)
mean_test_acc = np.mean(test_acc)
mean_prec = np.mean(prec)
mean_recall = np.mean(recall)

rs_table.loc['SVM1']['train accuracy'] = mean_train_acc
rs_table.loc['SVM1']['test accuracy'] = mean_test_acc
rs_table.loc['SVM1']['precision'] = mean_prec
rs_table.loc['SVM1']['recall'] = mean_recall

print(rs_table)
```

```
Calculating for SVM1....
     train accuracy test accuracy precision    recall
SVM1        0.98907      0.971053  0.978561  0.943474
SVM2            NaN           NaN       NaN       NaN
SVM3            NaN           NaN       NaN       NaN
```

In [23]:
```
print("Calculating for SVM2....")

clf = svm.SVC(kernel = 'rbf') # RBF Kernel

train_acc = []
test_acc = []
prec = []
recall = []

scaler = sklearn.preprocessing.StandardScaler()
x_sc = scaler.fit_transform(x)

for i in range(20):
    (x_train, x_test, y_train, y_test) = train_test_split(x_sc, y, test_size = 0.
    clf = clf.fit(x_train, y_train)

    #for train dataset
    y_train_pred = clf.predict(x_train)
    train_acc.append(metrics.accuracy_score(y_train, y_train_pred))

    #for test dataset
    y_test_pred = clf.predict(x_test)

    test_acc.append(metrics.accuracy_score(y_test, y_test_pred))
    prec.append(metrics.precision_score(y_test, y_test_pred, pos_label = 'M'))
    recall.append(metrics.recall_score(y_test, y_test_pred, pos_label = 'M'))

#compute average of 20 performance
mean_train_acc = np.mean(train_acc)
mean_test_acc = np.mean(test_acc)
mean_prec = np.mean(prec)
mean_recall = np.mean(recall)

rs_table.loc['SVM2']['train accuracy'] = mean_train_acc
rs_table.loc['SVM2']['test accuracy'] = mean_test_acc
rs_table.loc['SVM2']['precision'] = mean_prec
rs_table.loc['SVM2']['recall'] = mean_recall

print(rs_table)
```

```
Calculating for SVM2....
     train accuracy test accuracy precision    recall
SVM1        0.98907      0.971053  0.978561  0.943474
SVM2       0.986683      0.968421  0.967682  0.948719
SVM3            NaN           NaN       NaN       NaN
```

In [24]:
```
print("Calculating for SVM3....")

clf = svm.SVC(C = 10, kernel = 'rbf') # RBF Kernel with varying C value

train_acc = []
test_acc = []
prec = []
recall = []

scaler = sklearn.preprocessing.StandardScaler()
x_sc = scaler.fit_transform(x)

for i in range(20):
    (x_train, x_test, y_train, y_test) = train_test_split(x_sc, y, test_size = 0.
    clf = clf.fit(x_train, y_train)

    #for train dataset
    y_train_pred = clf.predict(x_train)
    train_acc.append(metrics.accuracy_score(y_train, y_train_pred))

    #for test dataset
    y_test_pred = clf.predict(x_test)

    test_acc.append(metrics.accuracy_score(y_test, y_test_pred))
    prec.append(metrics.precision_score(y_test, y_test_pred, pos_label = 'M'))
    recall.append(metrics.recall_score(y_test, y_test_pred, pos_label = 'M'))

#compute average of 20 performance
mean_train_acc = np.mean(train_acc)
mean_test_acc = np.mean(test_acc)
mean_prec = np.mean(prec)
mean_recall = np.mean(recall)

rs_table.loc['SVM3']['train accuracy'] = mean_train_acc
rs_table.loc['SVM3']['test accuracy'] = mean_test_acc
rs_table.loc['SVM3']['precision'] = mean_prec
rs_table.loc['SVM3']['recall'] = mean_recall

print(rs_table)
```

```
Calculating for SVM3....
     train accuracy test accuracy precision    recall
SVM1        0.98907      0.971053  0.978561  0.943474
SVM2       0.986683      0.968421  0.967682  0.948719
SVM3       0.993467      0.972515  0.971899  0.955727
```