In [75]:

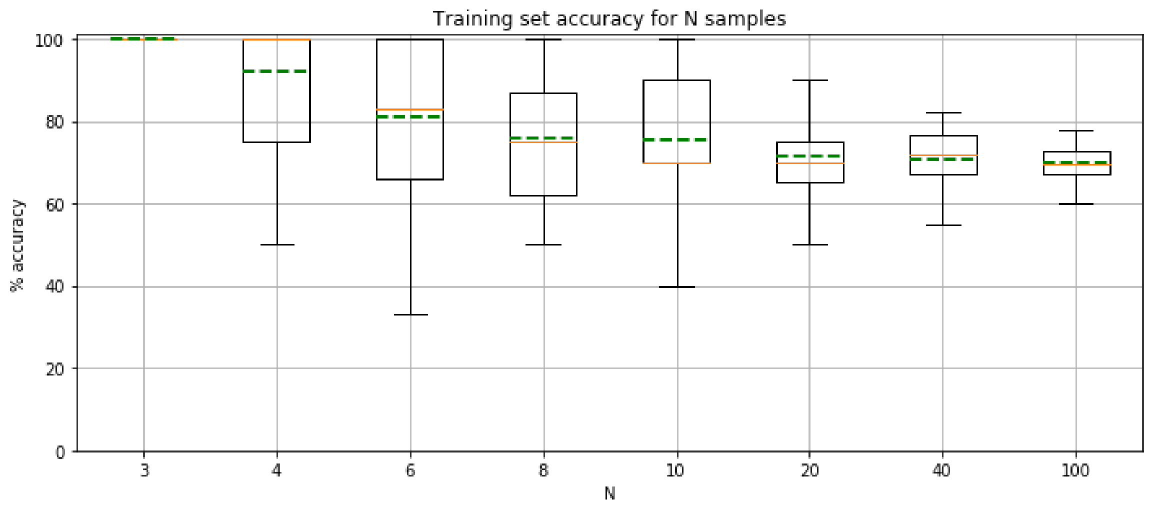
```
# Exercise H8.2: Variability of classification
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

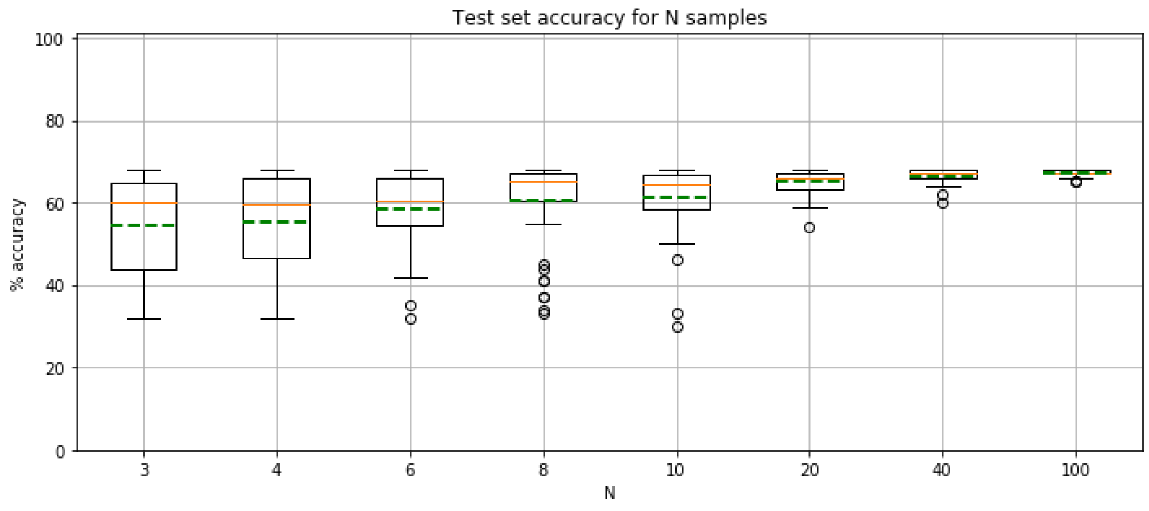
In [76]:

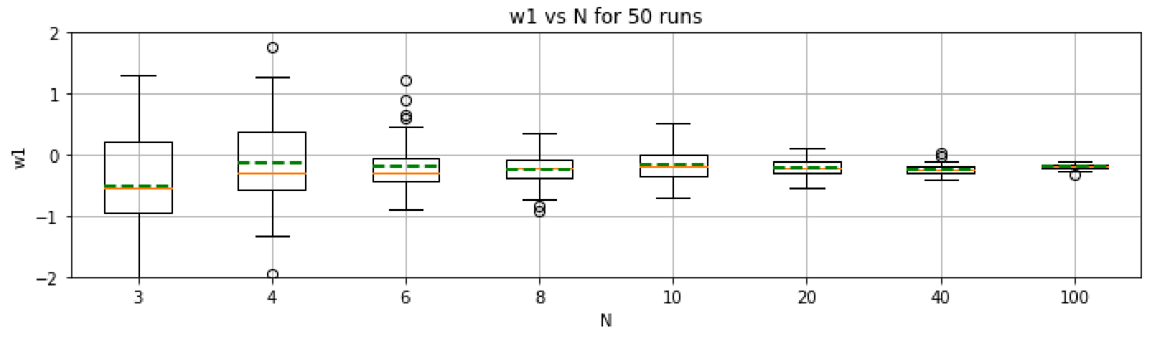
```
m1 = [0,1]
m2 = [1,0]
sigma = 2
testSampleSize = 1000
repeatTimes = 50

def calc_weights(train_set):
    y = train_set[:, 2]
    A = np.column_stack((np.ones(train_set.shape[0]), train_set[:, 0], train_set
[:, 1]))
    wb = np.linalg.lstsq(A, y)[0]
    return wb[0], wb[1:]

def generateSample(N):
    if(N%2 == 0):
        numSamples1 = int(N/2)
        numSamples2 = numSamples1
    else:
        numSamples1 = int(N/2)
        numSamples2 = numSamples1 + 1
    p_c1 = np.zeros([numSamples1, 2])
    p_c2 = np.zeros([numSamples2, 2])
    p_c1 = np.random.normal(m1,np.sqrt(sigma),[numSamples1,2])
    p_c2 = np.random.normal(m2,np.sqrt(sigma),[numSamples2,2])
    p1_c1 = np.concatenate([p_c1.T, np.ones((numSamples1,1)).T]).T
    p2_c2 = np.concatenate([p_c2.T, -np.ones((numSamples2,1)).T]).T
    return np.concatenate([p1_c1, p2_c2])

test_set = generateSample(testSampleSize)
N = [3, 4, 6, 8, 10, 20, 40, 100]
train_w1s, train_w2s, train_bs, train_percentages, test_percentages = np.zeros((
5, len(N), repeatTimes))
for i, n in enumerate(N):
    train_w1s_runs, train_w2s_runs, train_bs_runs, train_percentages_runs, test_
percentages_runs = np.zeros((5, repeatTimes))
    for j in range(repeatTimes):
        train_set = generateSample(n)
        b, w = calc_weights(train_set)
        train_yT = np.sign(w.T.dot(train_set.T[:2]) + b)
        train_percentages_runs[j] = 100 * np.sum(train_yT == train_set[:, 2]) /
train_set.shape[0]
        train_w1s_runs[j], train_w2s_runs[j] = w
        train_bs_runs[j] = b
        test_yT = np.sign(w.T.dot(test_set.T[:2]) + b)
        test_percentages_runs[j] = 100 * np.sum(test_yT == test_set[:, 2]) / tes
t_set.shape[0]
    train_w1s[i], train_w2s[i], train_bs[i] = train_w1s_runs, train_w2s_runs, tr
ain_bs_runs
    train_percentages[i], test_percentages[i] = train_percentages_runs, test_per
centages_runs
```

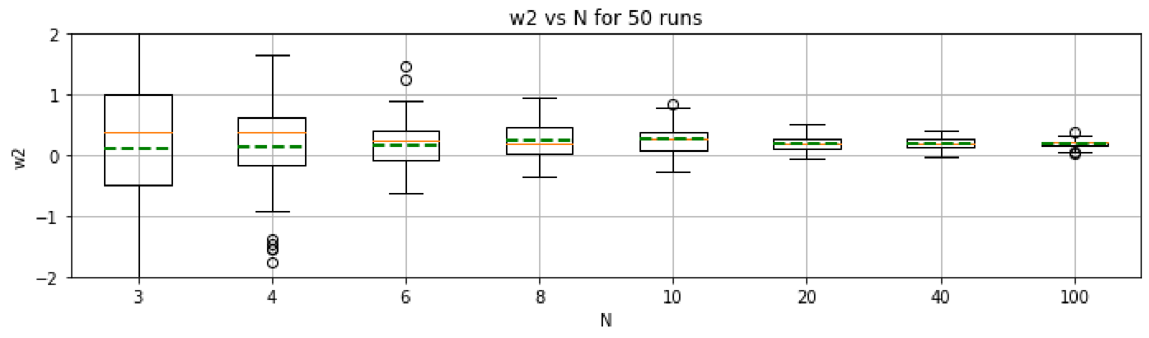
In [77]:

```
def  errorbar_plot(percentage_matrix, ax, xticks, title='', labels=None, ylim=[0
, 101], **kwargs):
    meanlineprops = dict(linewidth=2, color='green')
    ax.boxplot(percentage_matrix.T, meanprops=meanlineprops, meanline=True, **kw
args)
    ax.grid(True)
    ax.set_xticklabels(xticks)
    ax.set_title(title)
    ax.set_ylim(*ylim)
    if labels:
        ax.set_xlabel(labels[0])
        if (len(labels) > 1):
            ax.set_ylabel(labels[1])
    return ax

fig, ax1 = plt.subplots(1, 1, figsize=(10, 4.5))
errorbar_plot(train_percentages, ax1, N, title='Training set accuracy for N samp
les', labels=['N', '% accuracy'], showmeans=True)
fig.tight_layout()

fig, ax2 = plt.subplots(1, 1, figsize=(10, 4.5))
errorbar_plot(test_percentages, ax2, N, title='Test set accuracy for N samples',
 labels=['N', '% accuracy'], showmeans=True)
fig.tight_layout()

fig, ax3 = plt.subplots(1, 1, figsize=(10, 3))
errorbar_plot(train_w1s, ax3, N, title='w1 vs N for 50 runs', labels=['N', 'w1'
], ylim=[-2, 2], showmeans=True)
fig.tight_layout()

fig, ax4 = plt.subplots(1, 1, figsize=(10, 3))
errorbar_plot(train_w2s, ax4, N, title='w2 vs N for 50 runs', labels=['N', 'w2'
], ylim=[-2, 2], showmeans=True)
fig.tight_layout()

fig, ax5 = plt.subplots(1, 1, figsize=(10, 3))
errorbar_plot(train_bs, ax5, N, title='b vs N for 50 runs', labels=['N', 'b'], y
lim=[-2, 2], showmeans=True)
fig.tight_layout()
```


Training set accuracy for N samples
% accuracy
0
20
40
60
80
100
3
4
6
8
10
20
40
100
N


Test set accuracy for N samples
% accuracy
0
20
40
60
80
100
3
4
6
8
10
20
40
100
N


w1 vs N for 50 runs
w1
−2
−1
0
1
2
3
4
6
8
10
20
40
100
N


w2 vs N for 50 runs
w2
−2
−1
0
1
2
3
4
6
8
10
20
40
100
N