In [1]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from matplotlib.colors import ListedColormap
```

In [202]:

```
class Neuron:
    def __init__(self,eta=0.01):
        self.eta=eta

    def init_weights(self,X):
        return np.random.random(1+X.shape[1])

    def train(self,X,outputs,e_max):
        self.w_ =np.random.random(1+X.shape[1])
        self.error_ = []
        epoch=1
        done = False
        while not done:
            print("Epoch : ",epoch)
            err=0
            for x,d in zip(X,outputs):
                out = self.predict(x)
                err += 0.5*(d-out)**2
                print("For input pattern : ",x)
                self.w_[1:] = self.w_[1:] + self.eta*(d-out)*self.gradient(x
                self.w_[0] = self.w_[0] + self.eta*(d-out)*self.gradient(x)*
                print("Weights : ",self.w_)
            if err<e_max:
                done = True
                print("Training done")
            else:
                print("$")
                epoch+=1
                self.error_.append(err)
            print("Error : ",err)
        print("No of epochs required for training are : ",epoch)
        return self

    def get_weights(self):
        return self.w_

    def net_input(self,X):
        return np.dot(X,self.w_[1:])+self.w_[0]

    def activation(self,X):
        net = self.net_input(X)
        return (1-np.exp(-net))/(1+np.exp(-net))

    def gradient(self,X):
        return 0.5*(1-self.predict(X)**2)

    def predict(self,X):
        return self.activation(X)
```

In [203]:

```
#OR dataset
X = np.array([[0,0],[0,1],[1,0],[1,1]])
d = np.array([-1,1,1,1])
```

In [204]:

```
neuron = Neuron()
```

In [205]:

```
neuron.init_weights(X)
```

```
Out[205]: array([ 0.26549065,  0.90864179,  0.44052682])
```

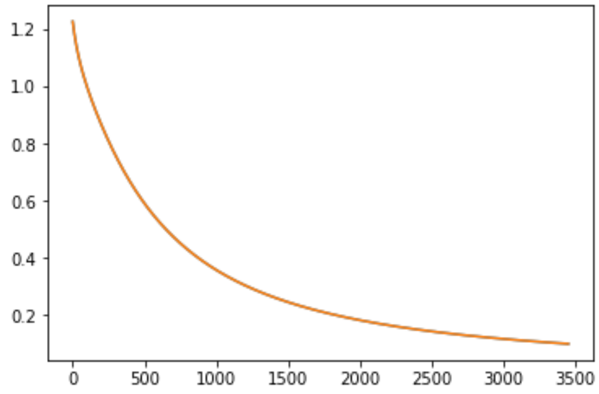
```
In [207]: neuron.train(X,d,0.01)
```

```
Epoch :  1
For input pattern :  [0 0]
Weights :  [ 0.93986753  0.41262672  0.66013043]
For input pattern :  [0 1]
Weights :  [ 0.94080606  0.41262672  0.66106954]
For input pattern :  [1 0]
Weights :  [ 0.94214491  0.41396663  0.66106954]
For input pattern :  [1 1]
Weights :  [ 0.94263122  0.4144533   0.66155621]
$
Error :  1.20581356182
Epoch :  2
For input pattern :  [0 0]
Weights :  [ 0.93682344  0.4144533   0.66155621]
For input pattern :  [0 1]
Weights :  [ 0.93776425  0.4144533   0.6624976 ]
For input pattern :  [1 0]
Weights :  [ 0.93910535  0.41579547  0.6624976 ]
For input pattern :  [1 1]
Weights :  [ 0.93959149  0.41628197  0.66298409]
```

```
In [208]: neuron.train(X,d,0.1)
```

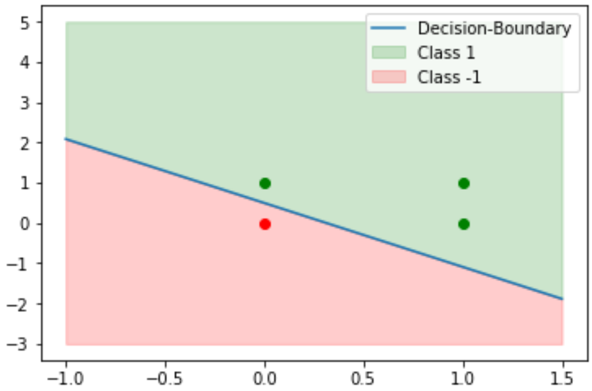
```
Epoch :  1
For input pattern :  [0 0]
Weights :  [ 0.38773046  0.60311068  0.18736527]
For input pattern :  [0 1]
Weights :  [ 0.39104595  0.60311068  0.19068385]
For input pattern :  [1 0]
Weights :  [ 0.39317372  0.60524055  0.19068385]
For input pattern :  [1 1]
Weights :  [ 0.39484143  0.60691123  0.19235454]
$
Error :  1.22726120177
Epoch :  2
For input pattern :  [0 0]
Weights :  [ 0.38909389  0.60691123  0.19235454]
For input pattern :  [0 1]
Weights :  [ 0.39239     0.60691123  0.19565374]
For input pattern :  [1 0]
Weights :  [ 0.39450478  0.60902808  0.19565374]
For input pattern :  [1 1]
Weights :  [ 0.39615073  0.61067695  0.19730261]
```

```
In [210]: plt.plot(neuron.error_)
          plt.show()
```



In [237]:

```
x = np.arange(-1,2,0.5)
a,b,c = neuron.w_[1],neuron.w_[2],neuron.w_[0]
y = (-1*c*-1*a*x)/b
colors = ['blue','green','red']
plt.plot(x,y,label='Decision-Boundary')
for i,j in zip(X,outputs):
    plt.scatter(i[0],i[1],c='green' if j==1 else 'red')
plt.fill_between(x,y,5,color='green',alpha=0.2,label='Class 1')
plt.fill_between(x,y,-3,color='red',alpha=0.2,label='Class -1')
plt.legend()
plt.show()
```



In [239]:

```
neuron.predict(np.array([0,0]))
```

Out[239]:

```
-0.66005394506759 64
```

In [240]:

```
#AND dataset
X = np.array([[0,0],[0,1],[1,0],[1,1]])
d = np.array([-1,-1,-1,1])
```

In [241]:

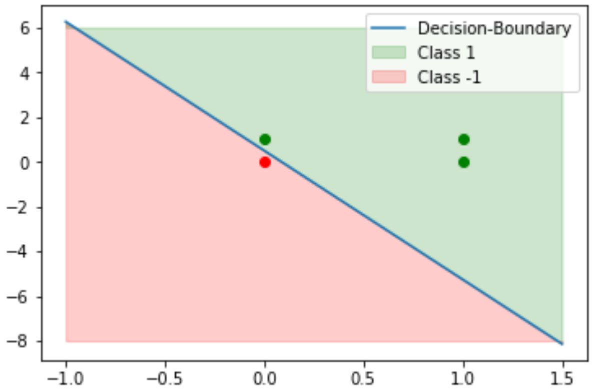
```
neuron2 = Neuron()
```

In [242]:

```
neuron2.train(X,d,0.1)
```

```
Epoch :  1
For input pattern :  [0 0]
Weights :  [ 0.11385472  0.41859781  0.70210733]
For input pattern :  [0 1]
Weights :  [ 0.10794468  0.41859781  0.6962107 ]
For input pattern :  [1 0]
Weights :  [ 0.10206546  0.41272742  0.6962107 ]
For input pattern :  [1 1]
Weights :  [ 0.10368619  0.414351    0.69783428]
$
Error :  2.41864381799
Epoch :  2
For input pattern :  [0 0]
Weights :  [ 0.09844132  0.414351    0.69783428]
For input pattern :  [0 1]
Weights :  [ 0.09252295  0.414351    0.69192908]
For input pattern :  [1 0]
Weights :  [ 0.08665803  0.40849455  0.69192908]
For input pattern :  [1 1]
Weights :  [ 0.08833011  0.41016962  0.69360415]
```

In [245]:
```
x = np.arange(-1,2,0.5)
a,b,c = neuron2.w_[1],neuron2.w_[2],neuron2.w_[0]
y = (-1*c*-1*a*x)/b
colors = ['blue','green','red']
plt.plot(x,y,label='Decision-Boundary')
for i,j in zip(X,outputs):
    plt.scatter(i[0],i[1],c='green' if j==1 else 'red')
plt.fill_between(x,y,6,color='green',alpha=0.2,label='Class 1')
plt.fill_between(x,y,-8,color='red',alpha=0.2,label='Class -1')
plt.legend()
plt.show()
```



In [246]:
```
#XOR dataset
X = np.array([[0,0],[0,1],[1,0],[1,1]])
d = np.array([-1,1,1,-1])
```

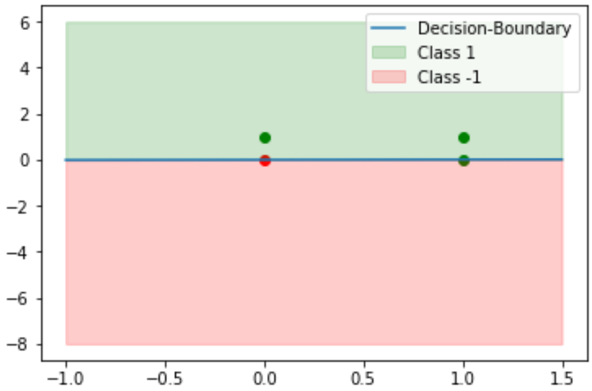
In [247]:
```
neuron3 = Neuron()
```

In [250]:
```
neuron3.train(X,d,0.5)
```

```
Epoch :  1
For input pattern :  [0 0]
Weights :  [ 0.33655288  0.8327487   0.54506474]
For input pattern :  [0 1]
Weights :  [ 0.33897618  0.8327487   0.54749048]
For input pattern :  [1 0]
Weights :  [ 0.34068335  0.8344574   0.54749048]
For input pattern :  [1 1]
Weights :  [ 0.33629325  0.83009389  0.54312697]
$
Error :  2.4070377773
Epoch :  2
For input pattern :  [0 0]
Weights :  [ 0.33062221  0.83009389  0.54312697]
For input pattern :  [0 1]
Weights :  [ 0.33306694  0.83009389  0.54557417]
For input pattern :  [1 0]
Weights :  [ 0.33479303  0.83182154  0.54557417]
For input pattern :  [1 1]
Weights :  [ 0.33037791  0.8274332   0.54118582]
```

In [251]:

```
x = np.arange(-1,2,0.5)
a,b,c = neuron3.w_[1],neuron3.w_[2],neuron3.w_[0]
y = (-1*c*-1*a*x)/b
colors = ['blue','green','red']
plt.plot(x,y,label='Decision-Boundary')
for i,j in zip(X,outputs):
    plt.scatter(i[0],i[1],c='green' if j==1 else 'red')
plt.fill_between(x,y,6,color='green',alpha=0.2,label='Class 1')
plt.fill_between(x,y,-8,color='red',alpha=0.2,label='Class -1')
plt.legend()
plt.show()
```



In [ ]: