In [3]:

```
import numpy as np
from matplotlib import pyplot as plt
import random
```

In [38]:

```
def generate(N):
    x1 = np.random.uniform(1, 5, size=(N,))
    x2 = np.random.uniform(10,100, size=(N,))
    X = np.column_stack((x1, x2))
    i = random.randint(0,49)
    j = random.randint(50,99)
    slope = (x2[j]-x2[i])/(x1[j]-x1[i])
    intercept = x2[j]-x1[j]*slope
    return X, slope, intercept

def target_f(slope, intercept, x):
    return slope*x + intercept

def label_y(slope, intcpt, x1, x2):
    y = []
    for i, j in zip(x1, x2):
        if target_f(slope, intercept, i)>j:
            y.append(1)
        else:
            y.append(-1)
    return y

def flipper(y):
    for j in range(10):
        i = random.choice(range(len(y)))
        if y[i] == 1:
            y[i] = -1
        else:
            y[i] = 1
    return y
```

In [19]:

```
class Perceptron:
    def __init__(self, X):
        self.weights = np.zeros(shape=len(X[0])+1)

    def fit(self, X, y):
        T = 10
        while T>0:
            for i, x in enumerate(X):
                if self.predict(x, self.weights)!=y[i]:
                    x = np.insert(x, 0, 1)
                    self.weights = self.weights + y[i]*x
            T -= 1
        return self.weights

    def predict(self, x, w):
        return np.sign(np.dot(np.transpose(w[1:]),x) + w[0])
```

In [51]:

```
class Pocket:
    def __init__(self, X, y):
        self.N = len(y)

    def fit(self, X, y, X_test, y_test):
        weights_old = np.zeros(shape=len(X[0])+1)
        error1_list = []
        error2_list = []
        error2_test_list = []
        error1_test_list = []
        p = Perceptron(X)
        for t in range(1000):
            error_1 = 0
            error_2 = 0
            error1_test = 0
            error2_test = 0
            weights_new = p.fit(X, y)
            for i, x in enumerate(X):
                if predict(x, weights_new)!=y[i]:
                    error_1 = error_1 + 1
                if predict(x, weights_old)!=y[i]:
                    error_2 = error_2 + 1
            error2_list.append(error_2/self.N)
            error1_list.append(error_1/self.N)
            for i, x in enumerate(X_test):
                if predict(x, weights_new)!=y_test[i]:
                    error1_test = error1_test + 1
                if predict(x, weights_old)!=y_test[i]:
                    error2_test = error2_test + 1
            error2_test_list.append(error2_test/self.N)
            error1_test_list.append(error1_test/self.N)
            if error_1 <= error_2:
                weights_old = weights_new

        return error1_list, error2_list, error1_test_list, error2_test_list

def predict(x, w):
    return np.sign(np.dot(np.transpose(w[1:]),x) + w[0])
```

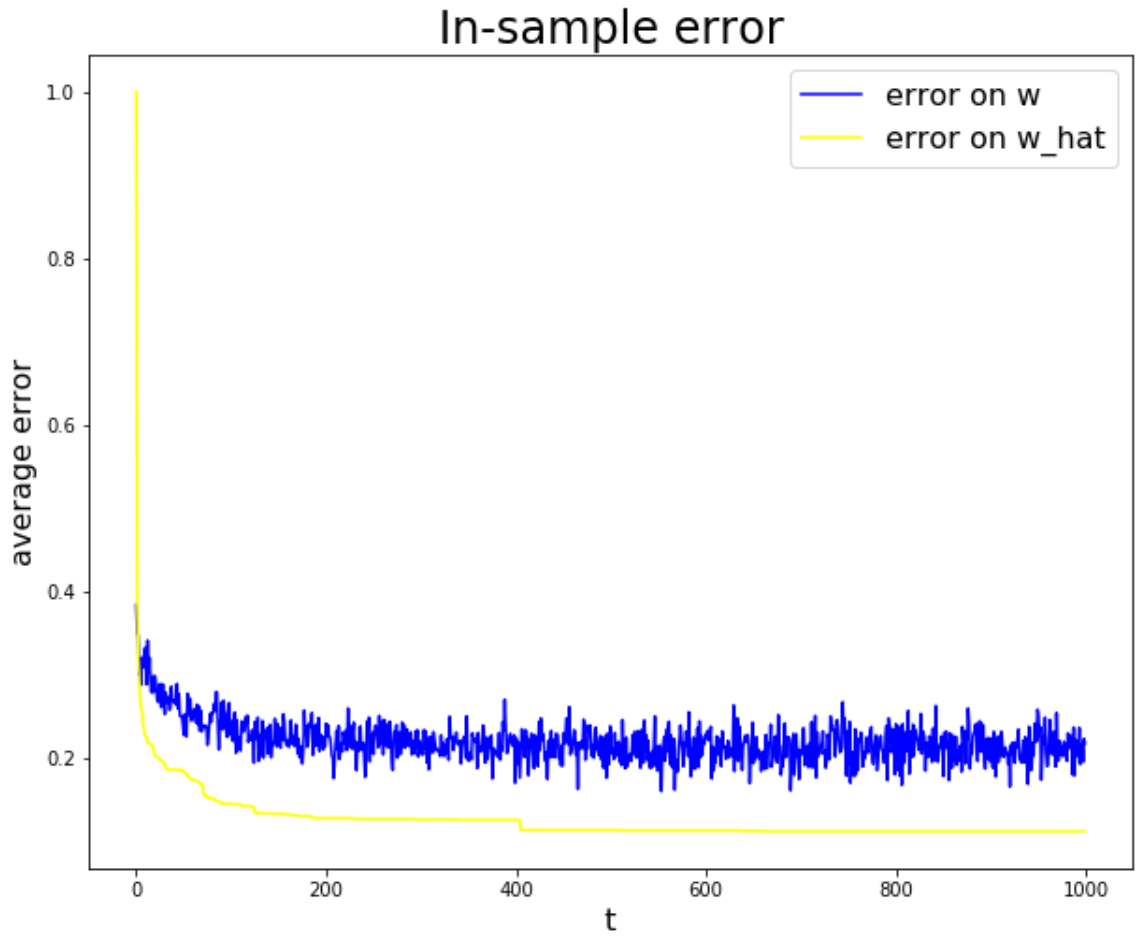
In [58]:

```
error1_train_old = np.zeros(shape=1000)
error2_train_old = np.zeros(shape=1000)
error1_test_old = np.zeros(shape=1000)
error2_test_old = np.zeros(shape=1000)
for i in range(20):
    train = generate(100)
    x1 = train[0][:,0]
    x2 = train[0][:,1]
    X = train[0]
    slope = train[1]
    intercept = train[2]
    y = np.asarray(label_y(slope, intercept, x1, x2))
    y_new = np.asarray(flipper(y))
    test = generate(1000)
    test_x1 = test[0][:,0]
    test_x2 = test[0][:,1]
    test_X = test[0]
    slope_test = test[1]
    intercept_test = test[2]
    y_test = np.asarray(label_y(slope_test, intercept_test, test_x1, test_x2))
    y_new_test = np.asarray(flipper(y_test))
    pocket = Pocket(X, y_new)
    error1_train, error2_train, error1_test, error2_test = pocket.fit(X, y_new, test_X,
y_new_test)
    error1_train_old += error1_train
    error2_train_old += error2_train
    error1_test_old += error1_test
    error2_test_old += error2_test

error1_train_old = error1_train_old/20
error2_train_old = error2_train_old/20
error1_test_old = error1_test_old/20
error2_test_old = error2_test_old/20
```

In [59]:

```
plt.figure(figsize=(10, 8));
plt.title('In-sample error', fontsize = 24)
plt.plot(np.arange(0, 1000), error1_train_old, color='blue', label = 'error on w');
plt.plot(np.arange(0, 1000), error2_train_old, color='yellow', label = 'error on w_hat'
);
plt.xlabel(xlabel='t', fontsize=16)
plt.ylabel(ylabel='average error', fontsize=16)
plt.legend(fontsize=16)
plt.show()
```



In [60]:

```
plt.figure(figsize=(10, 8));
plt.title('Out-of-sample error', fontsize = 24)
plt.plot(np.arange(0, 1000), error1_test_old, color='green', label = 'error on w');
plt.plot(np.arange(0, 1000), error2_test_old, color='red', label = 'error on w_hat');
plt.xlabel(xlabel='t', fontsize=16)
plt.ylabel(ylabel='average error', fontsize=16)
plt.legend(fontsize=16)
plt.show()
```