In [3]:
```
# -*- coding: utf-8 -*-
"""
Created on Fri Oct 25 15:14:54 2019

@author: jorge
"""

import sympy as sym
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.ticker as mtick

height = 10
width = 10

mpl.rcParams['figure.figsize'] = (width, height)
mpl.rcParams['font.size'] = 20
mpl.rcParams['figure.titlesize'] = 'small'
mpl.rcParams['legend.fontsize'] = 'small'
mpl.rcParams['xtick.major.size'] = 12
mpl.rcParams['xtick.minor.size'] = 8
mpl.rcParams['xtick.labelsize'] = 18
mpl.rcParams['ytick.major.size'] = 12
mpl.rcParams['ytick.minor.size'] = 8
mpl.rcParams['ytick.labelsize'] = 18

#get_ipython().run_line_magic('matplotlib', 'inline')
#https://scipy-lectures.org/packages/sympy.html
# ^^ how to use sympy ^^

HX ,HY  = 50,50 #number of x,y  points for countour
xmin,xmax = -15,15
ymin,ymax = -12,12
x1 = np.linspace(xmin,xmax,HX)
x2 = np.linspace(ymin,ymax,HY)
X1,X2 = np.meshgrid(x1,x2) # genertate mesh grid
w1=sym.Symbol('w1') # define symbols
w2=sym.Symbol('w2')

j=(w1**2+w1*w2+3*w2**2)# define equation

#compute gradient
j_grad1=sym.diff(j,w1)
j_grad2=sym.diff(j,w2)
#compute hessian
hess11=sym.diff(j_grad1,w1)
hess12=sym.diff(j_grad1,w2)
hess21=sym.diff(j_grad2,w1)
hess22=sym.diff(j_grad2,w2)

# Routines for problems
ew_thresh = 0.5
jw_thresh = 0.5
np.random.seed(seed=1)
rand_w = np.random.normal(0, 3**2, 2)
w0s = [[-5, 5], rand_w, rand_w]
#prob_title = ('a', 'b', 'c')
funcs = ('E', 'E', 'J')
labels = ['{}({:.2f},{:.2f})'.format(funcs[i], w0s[i][0], w0s[i][1]) for i in range(le
n(w0s))]
```

In [4]:
```
fig, axes = plt.subplots(nrows=3, sharex=True)
for n, w0 in enumerate(w0s):
    log_etas = np.arange(-3, 0.6, 0.1, dtype=np.float) # learning rate
    ax = axes[n]
    iters=[] # number of iterations until conveged
    #fig, ax = plt.subplots()
    for le in log_etas:
        eta = 10**le
        w=w0#[-5,5] # starting point
        wStar=[0,0] # ending point
        ew=[]
        jw=[]
        max_iters=100 # number of iterations
        count= 1
        line=[]

        while(True):
            #compute gradient matrix and hessian matrix
            g= np.array([float(j_grad1.subs({w1:w[0],w2:w[1]})),float(j_grad2.subs({w
1:w[0],w2:w[1]}))])
            H= np.array([[float(hess11.subs({w1:w[0],w2:w[1]})),float(hess12.subs({w1:
w[0],w2:w[1]}))],
                         [float(hess21.subs({w1:w[0],w2:w[1]})),float(hess22.subs({w1:w
[0],w2:w[1]}))]])

            wnew =  w-eta*g
            line.append(w)
            #loop check
            if( count>max_iters ):
                iters.append(max_iters)
                break
            elif(np.isnan(g).any()):
                break
            else:
                count=count + 1
                wprev=w.copy()

                w=wnew.copy()

                ew_i = np.linalg.norm(w-wStar)
                jw_i = j.subs({w1:w[0],w2:w[1]})
                ew.append(ew_i)
                jw.append(jw_i)
                if n < 2:
                    if(ew_i <= ew_thresh):
                        iters.append(count)
                        break
                else:
                    if(jw_i <= jw_thresh):
                        iters.append(count)
                        break
        line=np.array(line)

    print('Min Iterations @ log learning rate =', log_etas[np.argmin(iters)], 'Iterati
ons:', iters[np.argmin(iters)])
    ax.plot(log_etas, iters, color='black', label=labels[n])
    ax.set_ylabel('Iterations')
    ax.set_xlabel(r'$\log\eta$')
    ax.set_ylim(0,105)
    ax.set_xlim(-3, 0.5)
    if n < 2:
        ax.get_xaxis().set_visible(False)
    else:
        ax.xaxis.set_major_locator(mtick.MultipleLocator(0.5))
        ax.xaxis.set_minor_locator(mtick.MultipleLocator(0.1))
    ax.yaxis.set_major_locator(mtick.MultipleLocator(20))
    ax.yaxis.set_minor_locator(mtick.MultipleLocator(5))
    ax.legend()
fig.tight_layout(h_pad=0)
```

```
Min Iterations @ log learning rate = -0.5999999999999979 Iterations: 6
Min Iterations @ log learning rate = -0.5999999999999979 Iterations: 7
Min Iterations @ log learning rate = -0.5999999999999979 Iterations: 7
```