CMPS 392 - Machine Learning

Assignment 3

Mohamed Nassar, Spring 2020

Ex 1

In [1]:

```
# Explain
import numpy as np
a = np.array([0., np.finfo(np.float32).eps/2 ]).astype('float32')
print (a.argmax())
print ( (a+1).argmax() )
```

```
1
0
```

Ex 2

In [30]:

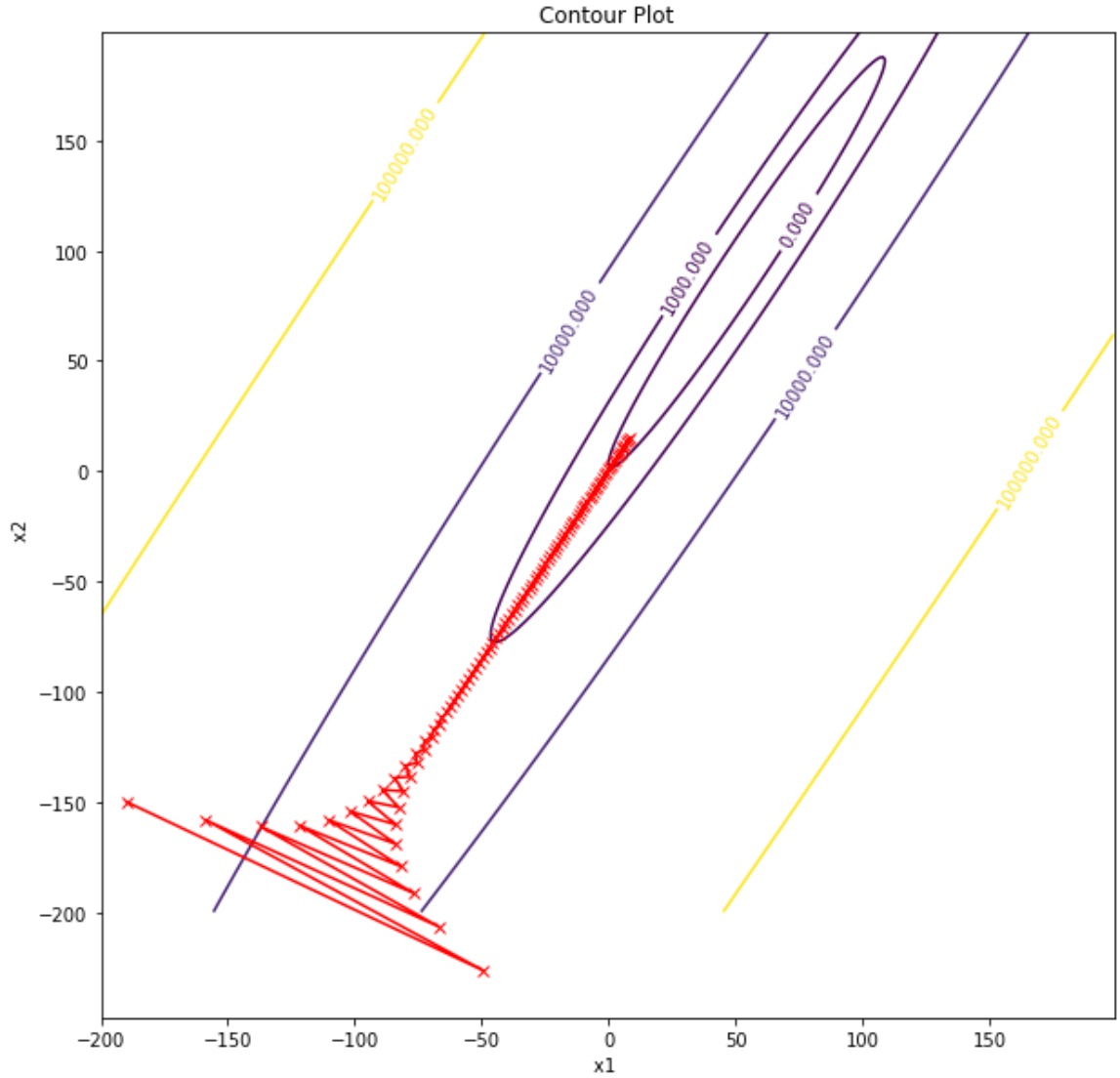
```
# Explain and propose a better solution to compute the variance of the numpy arr
ay x
import numpy.random as rand
x = np.array([10000 + rand.random() for i in range(10)]).astype('float32')
variance = np.mean(np.power(x,2,dtype='float32'),dtype='float32') - np.power(np.
mean(x, dtype='float32'),2,dtype='float32')
print (variance)
stddev = np.sqrt(variance)
```

```
-24.0

/Users/mohammadnassar/anaconda3/lib/python3.7/site-packages/ipykerne
l_launcher.py:6: RuntimeWarning: invalid value encountered in sqrt
```

Ex 3

In [32]:

```
# plug in the formula for the gradient of z
# Take learning rate = 0.18, 0.01, 0.1, 0.2 and explain what's happening when we
perform gradient descent
# Why learning rate = 0.1 performs so nicely at the begining of the descent. Jus
tify.
%matplotlib inline
import matplotlib.pyplot as plt


x1_vals = np.arange(-200, 200, 1)
x2_vals = np.arange(-200, 200 , 1)

x1, x2 = np.meshgrid(x1_vals , x2_vals)

z = 7.525/2 * x1**2  + 2.575/2 * x2**2 + -4.32 * x1 * x2 +  -9 * x2 + 15

fig = plt.figure(figsize=(10,10))
ax = plt.axes()
cp = ax.contour(x1, x2, z, [0, 1000, 10000, 100000])
ax.clabel(cp, inline=True, fontsize=10)
ax.set_title('Contour Plot')
ax.set_xlabel('x1 ')
ax.set_ylabel('x2 ')


# gradient descent

# starting point
x1, x2 = -190, -150

# learning rate
eps = 0.18


pts_x1  = [x1]
pts_x2 = [x2]

for i in range (100 ):

  g= # implement the gradient here
  (x1, x2)   =  (x1 - eps * g[0] ,  x2 - eps * g[1] )
  pts_x1.append(x1)
  pts_x2.append(x2)

plt.plot(pts_x1, pts_x2, 'r-x')
plt.show()
```


Contour Plot
100000.000
10000.000
1000.000
0.000
10000.000
100000.000
x1
x2

Ex 4

In [33]:

```
# explain what is going wrong, propose a fix
# n.b. you cannot change the hard coded numbers

def softmax (x):
  return np.exp(x)/np.sum(np.exp(x))

def logloss ( probs, y ):
  return -np.log (np.sum( probs * y))

logits = np.array([89, 50, 60]).astype('float32')
probs = softmax(logits)
y = np.array([1, 0, 0])
loss = logloss ( probs, y )

print (loss)
```

```
nan

/Users/mohammadnassar/anaconda3/lib/python3.7/site-packages/ipykerne
l_launcher.py:5: RuntimeWarning: overflow encountered in exp
  """
/Users/mohammadnassar/anaconda3/lib/python3.7/site-packages/ipykerne
l_launcher.py:5: RuntimeWarning: invalid value encountered in true_d
ivide
  """
```

Ex 5

In [34]:

```
# explain what is going wrong, propose a fix

def sigmoid(x):
  return (1/(1+ np.exp(-x)))

def logloss ( prob, y ):
  return -np.log (prob * y)

logit = np.float32(-89)
prob = sigmoid(logit)
y = 1
loss = logloss ( prob, y )

print (loss)
```

```
inf

/Users/mohammadnassar/anaconda3/lib/python3.7/site-packages/ipykerne
l_launcher.py:4: RuntimeWarning: overflow encountered in exp
  after removing the cwd from sys.path.
/Users/mohammadnassar/anaconda3/lib/python3.7/site-packages/ipykerne
l_launcher.py:7: RuntimeWarning: divide by zero encountered in log
  import sys
```

Ex 6

Propose an example of your choice to show why it is worth keeping an eye on numerical computations issues when implementing machine learning algorithms