In [1]:
```
import autograd.numpy as np
from autograd import grad
```

In [16]:
```
y = np.array([[2],[5]])
t = np.array([[1],[1]])
A = np.array([[2,1],[1,1]])
x = np.array([[4],[3]])
```

In [54]:
```
t = np.array([[1],[1]])
y = np.array([[2],[5]])

x = np.random.random((2,1))
def f(x,y,t):
    return np.dot(np.transpose(y),x) + np.dot(np.transpose(x),t)
grad_foo = grad(f)

print('linear function: Autogen Gradient: \n', grad_foo(x,y,t))
print('Linear function: Theoretical Gradient: \n', np.array([[3],[6]]))
```

```
linear function: Autogen Gradient:
 [[3.]
 [6.]]
Linear function: Theoretical Gradient:
 [[3]
 [6]]
```

In [49]:
```
A = np.array([[2,1],[1,1]])

x = np.random.random((2,1))
def f(x,A):
    return np.dot(np.dot(np.transpose(x),A),x)
grad_foo = grad(f)

print('Quadratic Function: Autogen Gradient: \n', grad_foo(x,A))
print('Quadratic Function: Theoretical Gradient: \n', np.dot(np.array([[4,2],[
```

```
Quadratic Function: Autogen Gradient:
 [[1.9139124 ]
 [1.12669622]]
Quadratic Function: Theoretical Gradient:
 [[1.9139124 ]
 [1.12669622]]
```

In [45]:
```
A = np.array([[4,0],[1,1]])
x = np.random.random((2,1))
def f(x,A):
    return np.exp(np.dot(np.dot(np.transpose(x),A),x))
grad_foo = grad(f)

print('Exponential Function: Autogen Gradient: \n', grad_foo(x,A))
print('Exponential Function: Theoretical Gradient: \n', np.dot(np.dot(np.array
```

```
Exponential Function: Autogen Gradient:
 [[12.85300117]
 [ 4.41344879]]
Exponential Function: Theoretical Gradient:
 [[12.85300117]
 [ 4.41344879]]
```

In [48]:
```
w = np.array([[7],[8]])
x = np.random.random((2,1))

def f(w,x):
    return 1/(1+np.exp(-(np.dot(np.transpose(w),x))))
grad_foo = grad(f)

print('Logistic Regression Function: Autogen Gradient: \n', grad_foo(w,x))
print('Logistic Regression Function: Theoretical Gradient: \n', np.dot(-1*(1+n
```

In [ ]:

In [ ]: