“AÑO DEL FORTALECIMIENTO DE LA SOBERANÍA NACIONAL”

UNIVERSIDAD NACIONAL DEL ALTIPLANO - PUNO

ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS

TRABAJO:

[ACTIVIDAD] HANDWRITTEN DIGITS RECOGNITION

CURSO:

INTELIGENCIA COMPUTACIONAL

DOCENTE:

ACEITUNO ROJAS MIGUEL ROMILIO

PRESENTADO POR:

- *CONDORI GUTIERREZ, Rodrigo Bernardo*

PERÚ, 2022 - II

CÓDIGO:

Link de Github: [RodrigoCG1281/Tarea_inteligencia_computacional (github.com)]

In [1]:
```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data
```

In [2]:
```
mnist=input_data.read_data_sets("MNIST_data/", one_hot=True) #La imagenes tienen dimension de 28x28
```

```
Extracting MNIST_data/train-images-idx3-ubyte.gz
Extracting MNIST_data/train-labels-idx1-ubyte.gz
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
```

In [3]:
```
x=tf.placeholder(tf.float32,[None,784]) #imagen del numero descompuesta a un vector
P=tf.Variable(tf.zeros([784,10])) #Matriz de pesos, 784 para recibir la imagen, 10 por las posible salidas
b=tf.Variable(tf.zeros([10])) #Vector con bias
y=tf.matmul(x,P)+b #La operacion que se hara en los nodos que reciben entradas
yR=tf.placeholder(tf.float32,[None,10]) # Matriz con las etiquetas REALES del set de datos
```

Definir la función de costo entropia cruzada (Cross Entropy) para poder medir el error. La salida será con Softmax

In [4]:
```
softmax=tf.nn.softmax_cross_entropy_with_logits(labels=yR,logits=y)
costo=tf.reduce_mean(softmax)
optimizador=tf.train.GradientDescentOptimizer(0.5).minimize(costo)
```

Correr la gráfica computacional

In [5]:
```
prediccion = tf.equal(tf.argmax(y, 1), tf.argmax(yR, 1)) #Nos da arreglo de booleanos para decirnos
                                                         #cuales estan bien y cuales no
accuracy = tf.reduce_mean(tf.cast(prediccion, tf.float32))#Nos da el porcentaje sobre el arreglo de prediccion
Produccion = tf.argmax(y,1)
init=tf.global_variables_initializer()
```

Entrenar algoritmo

In [6]:
```
#Funcion que usaremos para ver que tan bien va a aprendiendo nuestro modelo
def avance(epoca_i, sess, last_features, last_labels):
    costoActual = sess.run(costo,feed_dict={x: last_features, yR: last_labels})
    Certeza = sess.run(accuracy,feed_dict={x:mnist.validation.images,yR: mnist.validation.labels})
    print('Epoca: {:<4} - Costo: {:<8.3} Certeza: {:<5.3}'.format(epoca_i,costoActual,Certeza))
```

In [7]:
```
with tf.Session() as sess:
    sess.run(init)
    for epoca_i in range(1000):
        lotex, lotey = mnist.train.next_batch(100)
        sess.run(optimizador, feed_dict={x: lotex, yR: lotey})
        if (epoca_i%50==0):
            avance(epoca_i, sess, lotex, lotey)
    print('RESULTADO FINAL: ',sess.run(accuracy, feed_dict={x: mnist.test.images,yR: mnist.test.labels}))
    print ('Resultado de una imagen',sess.run(Produccion,feed_dict={x: mnist.test.images[5].reshape(1,784)}))
```

```
Epoca: 0    - Costo: 1.8      Certeza: 0.111
Epoca: 50   - Costo: 0.382    Certeza: 0.875
Epoca: 100  - Costo: 0.407    Certeza: 0.898
Epoca: 150  - Costo: 0.316    Certeza: 0.896
Epoca: 200  - Costo: 0.34     Certeza: 0.908
Epoca: 250  - Costo: 0.313    Certeza: 0.912
Epoca: 300  - Costo: 0.286    Certeza: 0.907
```

```
        lotex, lotey = mnist.train.next_batch(100)
        sess.run(optimizador, feed_dict={x: lotex, yR: lotey})
        if (epoca_i%50==0):
            avance(epoca_i, sess, lotex, lotey)
    print('RESULTADO FINAL: ',sess.run(accuracy, feed_dict={x: mnist.test.images,yR: mnist.test.labels}))
    print ('Resultado de una imagen',sess.run(Produccion,feed_dict={x: mnist.test.images[5].reshape(1,784)}))
```

```
Epoca: 0    - Costo: 1.8      Certeza: 0.111
Epoca: 50   - Costo: 0.382    Certeza: 0.875
Epoca: 100  - Costo: 0.407    Certeza: 0.898
Epoca: 150  - Costo: 0.316    Certeza: 0.896
Epoca: 200  - Costo: 0.34     Certeza: 0.908
Epoca: 250  - Costo: 0.313    Certeza: 0.912
Epoca: 300  - Costo: 0.286    Certeza: 0.907
Epoca: 350  - Costo: 0.391    Certeza: 0.914
Epoca: 400  - Costo: 0.509    Certeza: 0.909
Epoca: 450  - Costo: 0.478    Certeza: 0.914
Epoca: 500  - Costo: 0.506    Certeza: 0.914
Epoca: 550  - Costo: 0.215    Certeza: 0.917
Epoca: 600  - Costo: 0.273    Certeza: 0.914
Epoca: 650  - Costo: 0.227    Certeza: 0.917
Epoca: 700  - Costo: 0.224    Certeza: 0.919
Epoca: 750  - Costo: 0.177    Certeza: 0.922
Epoca: 800  - Costo: 0.324    Certeza: 0.919
Epoca: 850  - Costo: 0.248    Certeza: 0.924
Epoca: 900  - Costo: 0.258    Certeza: 0.923
Epoca: 950  - Costo: 0.162    Certeza: 0.918
RESULTADO FINAL:  0.92
Resultado de una imagen [1]
```

In [8]:
```
mnist.test.labels[5]
```

Out[8]:
```
array([ 0.,  1.,  0.,  0.,  0.,  0.,  0.,  0.,  0.,  0.])
```

In [ ]:
```
```