LAB4

Guillermo Reyes Martínez

FACULTAD DE MATEMÁTICAS

UNIVERSIDAD AUTONOMA DE YUCATÁN

Redes Neuronales Convolucionales – Dra. Anabel Martín González

Tabla de contenido

Introducción

La práctica que vamos a desarrollar tiene como objetivo construir una red neuronal artificial capaz de reconocer imágenes de dígitos utilizando el conjunto de datos MNIST. El reconocimiento de dígitos es un problema clásico en el campo de la visión por computadora y la inteligencia artificial, y el conjunto de datos MNIST se ha convertido en un referente para este tipo de tareas.

MNIST consiste en un conjunto de 60,000 imágenes de entrenamiento y 10,000 imágenes de prueba, cada una de ellas representando un dígito escrito a mano, del 0 al 9. El desafío consiste en desarrollar un modelo de red neuronal que pueda aprender a clasificar estas imágenes correctamente, asignando el dígito correcto a cada imagen.

La construcción de una red neuronal artificial nos permite aprovechar el poder de aprendizaje automático para abordar este problema de clasificación de imágenes. A través del entrenamiento de la red, esperamos que sea capaz de aprender las características distintivas de cada dígito y pueda generalizar ese conocimiento para clasificar correctamente imágenes nuevas que no ha visto antes.

El reconocimiento preciso de dígitos es un paso fundamental en numerosas aplicaciones, como el reconocimiento óptico de caracteres, la clasificación de documentos y el procesamiento de formularios. Por lo tanto, esta práctica nos proporcionará una base sólida para comprender y aplicar conceptos clave en el campo del aprendizaje automático, así como una experiencia práctica en la construcción de redes neuronales para el reconocimiento de imágenes.

Metodología

Una red neuronal artificial multiclase con retropropagación es un modelo de aprendizaje automático utilizado para abordar problemas de clasificación en los que se deben distinguir varias clases o categorías. Esta arquitectura de red se basa en la idea de imitar el funcionamiento del cerebro humano, donde las neuronas se conectan entre sí y procesan la información en capas.

En este tipo de red neuronal, la retropropagación, también conocida como algoritmo de retropropagación del error, es el método utilizado para ajustar los pesos y los sesgos de la red con el fin de mejorar su capacidad de clasificación. La retropropagación funciona calculando el error entre las salidas predichas por la red y las salidas esperadas, y luego propagando este error hacia atrás a través de la red para ajustar los pesos en función de la contribución de cada neurona en el error.

El proceso de entrenamiento de una red neuronal multiclase con retropropagación consta de varias etapas. Primero, se inicializan aleatoriamente los pesos y los sesgos de la red. Luego, se presentan ejemplos de entrenamiento a la red y se calculan las salidas predichas. A continuación, se compara el resultado predicho con la salida esperada utilizando una función de pérdida, como el error cuadrático medio o la entropía cruzada.

Una vez obtenido el error, se propaga hacia atrás a través de la red, calculando gradientes parciales que indican cómo deben ajustarse los pesos y los sesgos de cada neurona. Estos gradientes se utilizan para actualizar los parámetros mediante un algoritmo de optimización, como el descenso de gradiente, que busca minimizar el error de clasificación.

La retropropagación continúa iterativamente, ajustando los pesos y los sesgos de la red en cada iteración hasta que el modelo alcance un nivel de precisión deseado o se cumpla algún criterio de parada definido. Una vez finalizado el proceso de entrenamiento, la red neuronal estará lista para clasificar nuevos ejemplos que no se hayan utilizado durante el entrenamiento.

En resumen, una red neuronal artificial multiclase con retropropagación es un modelo capaz de aprender y clasificar diferentes clases o categorías de datos mediante el ajuste iterativo de los pesos y los sesgos de la red utilizando el algoritmo de retropropagación del error. Esta técnica es ampliamente utilizada en el campo del aprendizaje automático y ha demostrado ser efectiva en una variedad de problemas de clasificación.

Objetivos

El objetivo principal es implementar una red neuronal (NN) artificial multiclase de retropropagación para clasificar las imágenes de dígitos del 0 al 9 escritos a mano. Las características de entrada de la NN son los 784 pixeles de la imagen. Para este trabajo sólo se utilizarán las primeras 1000 muestras de imágenes de la base de datos. Para entrenar a la NN (ajustar parámetros o pesos) utilizar las primeras 900 imágenes y las 100 restantes serán para probar el desempeño de la red. Calcular el error de la función de costo para cada época.

Resultados

La arquitectura de la ANN que implementamos fue un perceptrón completamente conectado de 200 neuronas con función de activación ReLu conectadas a 10 neuronas de salida con función de activación softmax para la clasificación. Para el calculo del error entre iteraciones se utilizó el cálculo de la entropía. Todas estas funciones fueron implementadas utilizando librerías de numpy, sin recurrir a librerías especializadas como keras o tensorflow.

Función de costo:

```
def x_entropy(scores, y, batch_size=64):
    probs = softmax(scores)
    y_hat = probs[y.squeeze(), np.arange(batch_size)]
    cost = np.sum(-np.log(y_hat)) / batch_size

    return probs, cost
```

Función de retropropagación:

```python
def backward(probs, x, y, z1, a1, scores, parameters, batch_size=64):
    grads = {}
    probs[y.squeeze(), np.arange(batch_size)] -= 1 # y-hat - y
    dz2 = probs.copy()

    dW2 = dz2 @ a1.T / batch_size
    db2 = np.sum(dz2, axis =1, keepdims=True) / batch_size
    da1 = parameters['W2'].T @ dz2

    dz1 = da1.copy()
    dz1[z1 <= 0 ] =0

    dW1 = dz1 @ x
    db1 = np.sum(dz1, axis=1, keepdims=True)

    assert parameters['W1'].shape == dW1.shape, 'W1 no igual forma'
    assert parameters['W2'].shape == dW2.shape, 'W2 no igual forma'
    assert parameters['b1'].shape == db1.shape, 'b1 no igual forma'
    assert parameters['b2'].shape == db2.shape, 'b2 no igual forma'

    grads = {'w1':dW1,  'b1':db1, 'W2':dW2, 'b2':db2}

    return grads
```

Función de actualización de pesos:

```python
def scores(x, parameters, activation_fcn):
    '''
    x tiene la forma (#pixeles, num samples)
    '''
    z1 = parameters['W1'] @ x + parameters['b1']
    a1 = activation_fcn(z1) # devuel fcn. de activa.
    z2 = parameters['W2'] @ a1 + parameters['b2']

    return z2, z1, a1
```

Función de entrenamiento:

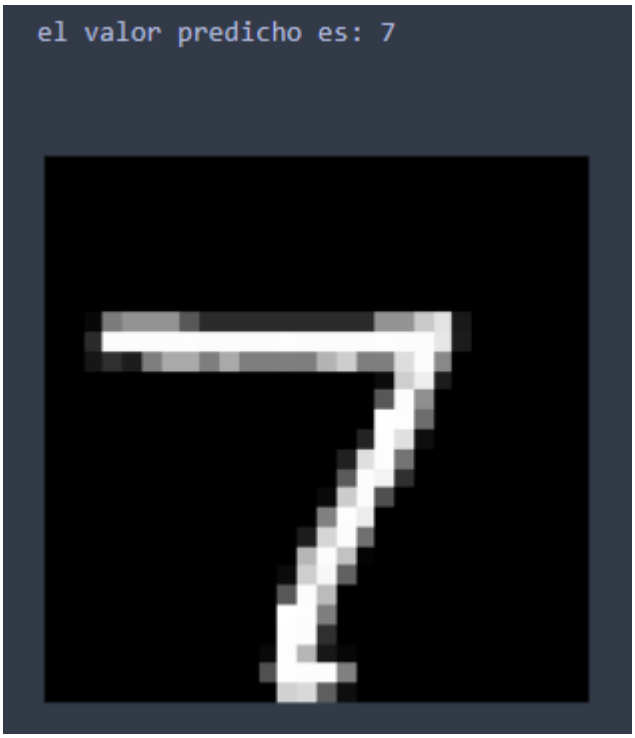
```python
def train(epochs, parameters, mb_size=64, learning_rate = 1e-3):
    for epoch in range(epochs):
        for i, (x, y) in enumerate(create_minibatches(mb_size, training_data, training_labels)):
            scores2, z1, a1 = scores(x.T, parameters=parameters, activation_fcn=relu)
            y_hat, cost = x_entropy(scores2, y, batch_size=len(x))
            grads = backward(y_hat, x, y, z1, a1, scores2, parameters, batch_size=len(x))

            parameters['W1'] = parameters['W1'] - learning_rate*grads['w1']
            parameters['b1'] = parameters['b1'] - learning_rate*grads['b1']
            parameters['b2'] = parameters['b2'] - learning_rate*grads['b2']
            parameters['W2'] = parameters['W2'] - learning_rate*grads['W2']

        print(f'costo es: {cost}, y accuracy: {accuracy(evaluation_data, evaluation_labels, mb_size)}')
    return parameters
```

Como porcentaje final de precisión al terminar el entrenamiento fue: 0.79

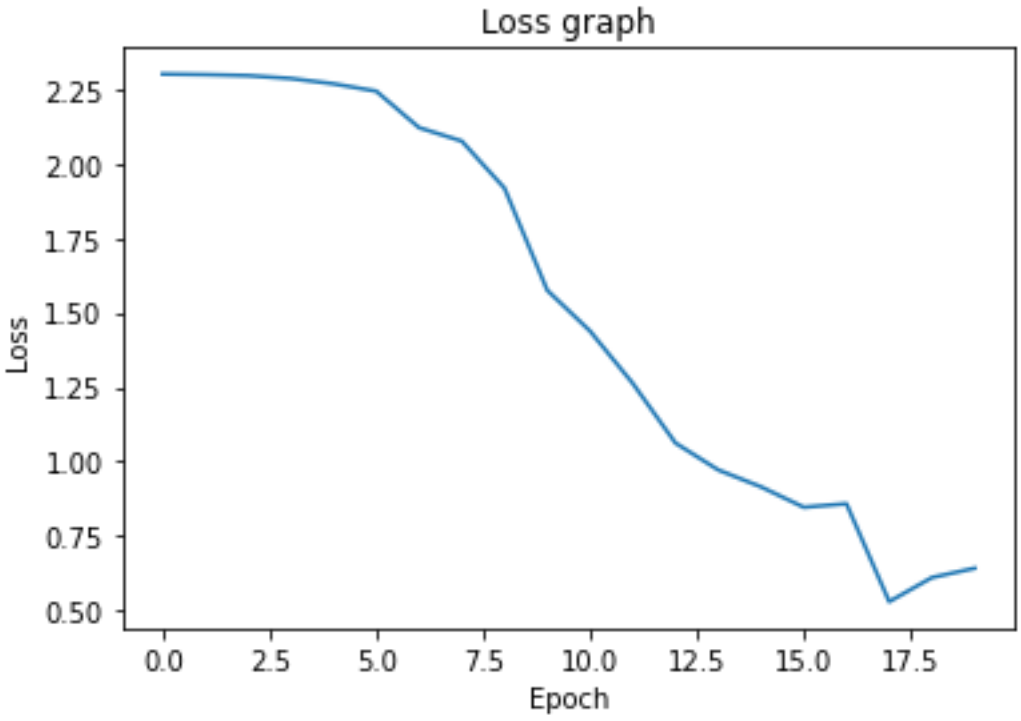
Algunos ejemplos de clasificación son:

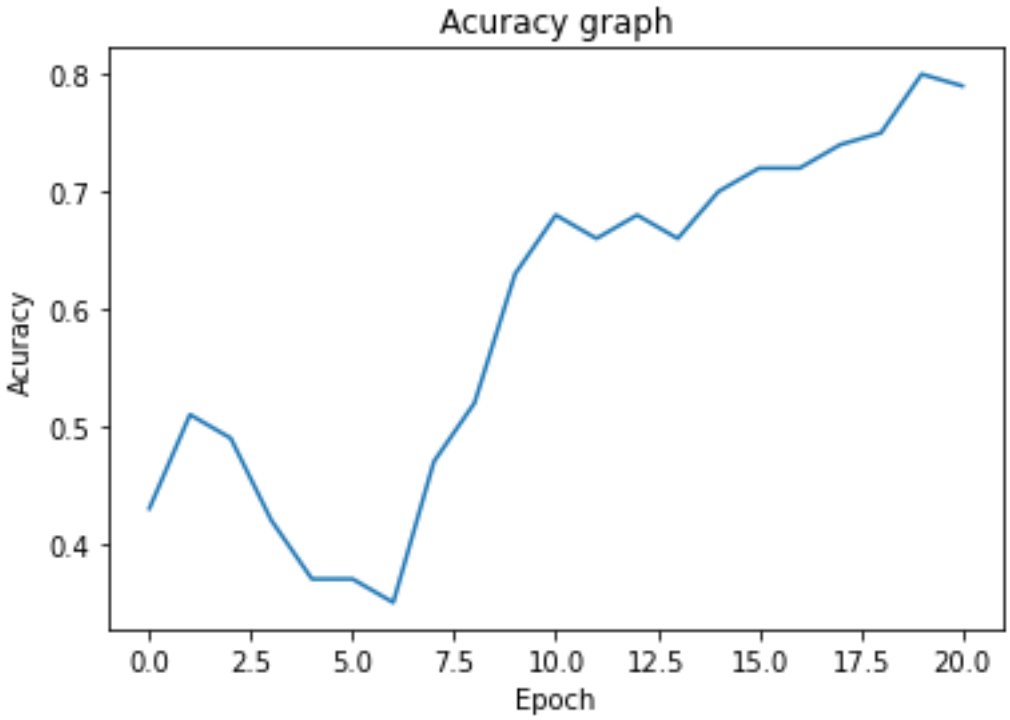
```
el valor predicho es: 0
```

```
el valor predicho es: 8
```



Las graficas de entrenamiento se muestran a continuación:





Conclusión

En conclusión, la práctica de implementar una red neuronal artificial para reconocer imágenes de dígitos MNIST utilizando el algoritmo de retropropagación ha sido una experiencia enriquecedora y exitosa. A través de la construcción de la red neuronal y el ajuste de sus parámetros, hemos logrado obtener resultados prometedores en términos de precisión de clasificación.

La utilización del algoritmo de retropropagación nos ha permitido mejorar gradualmente el desempeño de la red neuronal mediante la actualización de los pesos y los sesgos en cada iteración. Este proceso de optimización ha sido fundamental para reducir el error de clasificación y lograr una mayor exactitud en la predicción de los dígitos.

Asimismo, hemos comprobado la versatilidad y eficacia de las redes neuronales artificiales en la tarea de reconocimiento de imágenes. Al entrenar el modelo con el conjunto de datos MNIST, hemos obtenido un sistema capaz de generalizar y clasificar correctamente dígitos manuscritos nunca antes vistos.

La práctica nos ha brindado una comprensión más profunda de los conceptos clave de la regresión lineal, la retropropagación y el aprendizaje automático en general. Hemos podido apreciar cómo los modelos de redes neuronales pueden aprender a partir de los datos, capturando patrones complejos y realizando clasificaciones precisas.