Técnicas de Deep Learning Para Visión Artificial

Examen parcial 1
Septiembre 06, 2023

Nombre: Freddy Alejando Flórez Bohórquez ID: 1001057999

Nombre: Laura López Arbeláez ID: 1193221079

Tiempo: 3h Pueden trabajar en parejas.

1. ¿Cuál de las siguientes afirmaciones es cierta sobre una red neuronal feedforward?
a) Puede tener conexiones retroalimentadas.
b) Se utiliza principalmente para procesar secuencias de datos.
c) Las neuronas están organizadas en capas.
d) No necesita una función de activación.

2. ¿Qué función de activación comúnmente se utiliza en las capas ocultas de una red neuronal feedforward para introducir no linealidad?
a) Función de Identidad
b) Función Sigmoide
c) Función de Unidad Lineal Rectificada (ReLU)
d) Función Tangente Hiperbólica (tanh)

3. ¿Cuál es uno de los desafíos comunes al entrenar redes neuronales profundas?
a) El bajo consumo de memoria durante el entrenamiento.
b) La falta de flexibilidad en la arquitectura de la red.
c) El problema de la explosión del gradiente.
d) La rapidez en la convergencia del modelo.

4. ¿Qué término se utiliza para describir el proceso en el que una red neuronal ajusta sus pesos y sesgos para minimizar una función de pérdida durante el entrenamiento?
a) Regularización
b) Clasificación
c) Optimización
d) Normalización

5. ¿Cuál de las siguientes afirmaciones es cierta sobre el aprendizaje supervisado en deep learning?
a) Se requiere un conjunto de datos sin etiquetas.
b) El modelo aprende automáticamente las características de entrada.
c) El modelo se entrena utilizando ejemplos de entrada y salida conocidos.
d) El modelo no necesita ninguna función de activación.

6. ¿Cuál es el propósito principal de la capa de salida en una red neuronal feedforward?
a) Extraer características de entrada.
b) Realizar transformaciones no lineales en los datos de entrada.
c) Generar la salida final de la red para una tarea específica.
d) Realizar la retroalimentación de errores a las capas anteriores.

7. ¿Qué es el overfitting en el contexto del aprendizaje profundo?
a) Un proceso en el que el modelo se ajusta demasiado a los datos de entrenamiento y no generaliza bien a nuevos datos.
b) Un método para aumentar el tamaño del conjunto de entrenamiento.
c) Un tipo de función de activación comúnmente utilizado en redes neuronales.
d) La cantidad de capas ocultas en una red neuronal profunda.

8. ¿Cuál es el objetivo de la función de pérdida en el entrenamiento de una red neuronal?
a) Minimizar el número de capas en la red.
b) Maximizar la tasa de aprendizaje.
c) Minimizar la diferencia entre las predicciones del modelo y los valores reales (etiquetas).
d) Maximizar el número de neuronas en cada capa.

Parte práctica

En este ejercicio, vamos a crear una red neuronal feedforward para clasificar dígitos manuscritos del conjunto de datos MNIST. El objetivo es entrenar el modelo para reconocer dígitos escritos a mano y predecir a qué número corresponde cada imagen.

1. Preparación de datos:
- Descarga el conjunto de datos MNIST, que consta de imágenes en escala de grises de dígitos escritos a mano.
- cargar datos de entrenamiento y prueba en variables separadas

```
import numpy as np
import matplotlib.pyplot       as plt
from mlxtend.data import loadlocal_mnist
from sklearn.metrics import confusion_matrix

X_train, y_train = loadlocal_mnist(
        images_path='MNIST/train-images-idx3-ubyte',
        labels_path='MNIST/train-labels-idx1-ubyte')

X_test, y_test = loadlocal_mnist(
        images_path='MNIST/t10k-images-idx3-ubyte',
        labels_path='MNIST/t10k-labels-idx1-ubyte')

labels,count_class = np.unique(y_train,return_counts=True)

print('[INFO] \nlabels: \n %s \ncount per class \n %s' % (labels,count_class))
print('Training set dimensions: %s x %s' % (X_train.shape[0],X_train.shape[1]))
print('Test set dimensions: %s x %s' % (X_test.shape[0], X_test.shape[1]))
```

2. Diseño de la Red Neuronal:
- Crea una red neuronal feedforward utilizando una biblioteca de aprendizaje profundo como TensorFlow o Keras.
- Diseña la capa de entrada para recibir imágenes en escala de grises de 28x28 píxeles (784 píxeles en total).
- Agrega una o varias capas ocultas con unidades/neuronas ajustables.
- Utiliza una capa de salida con 10 neuronas para representar las clases de dígitos (0-9).
- Aplica funciones de activación apropiadas en las capas ocultas, como la función ReLU, y una función de activación softmax en la capa de salida.

```python
##importamos librerias para la construcción del modelo
#multilayer feedforward
from keras.models import Sequential
from keras.layers import Dense


##construccion del modelo
num_classes = labels.size ##tamaño de la clases
nu_hl1 = 400    #neuronas en la capa 1
nu_hl2 = 200    #neuronas en la capa 2
nu_hl3 = 100    #neuronas en la capa 3
nu_hl4 = num_classes    #total de clases MNIST


model = Sequential()
model.add(Dense(nu_hl1, input_dim=dim1*dim2, activation='relu'))
model.add(Dense(nu_hl2, activation='relu'))
model.add(Dense(nu_hl3, activation='relu'))
model.add(Dense(nu_hl4, activation='softmax'))  #sigmoid
model.summary()         #estructura del modelo
```

Resultados:

```
labels:
 [0 1 2 3 4 5 6 7 8 9]
count per class
 [5923 6742 5958 6131 5842 5421 5918 6265 5851 5949]
Training set dimensions: 60000 x 784
Test set dimensions: 10000 x 784
5
2023-09-06 20:49:45.614023: I tensorflow/core/platform/cpu_feature_guard.cc:193] This TensorFlow binary is optimized wit
h oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critical operations:
 AVX AVX2
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 400)               314000

 dense_1 (Dense)             (None, 200)               80200

 dense_2 (Dense)             (None, 100)               20100

 dense_3 (Dense)             (None, 10)                1010

=================================================================
Total params: 415,310
Trainable params: 415,310
Non-trainable params: 0
_________________________________________________________________
```

3. Entrenamiento de la Red:
- Utiliza el conjunto de entrenamiento para entrenar la red neuronal.
- Define una función de pérdida (como la entropía cruzada) y un algoritmo de optimización (como el descenso de gradiente estocástico).
- Entrena el modelo durante varias épocas hasta que la pérdida converja o alcance un nivel aceptable.

```python
#lo que hace es coger una categoria volver las otra una sola para clasificar una a una
from keras.utils import to_categorical
train_labels = to_categorical(y_train, num_classes=num_classes)
test_labels = to_categorical(y_test, num_classes=num_classes)

print('convertir: ',y_train[0],' a one hot encoding : ',train_labels[0])


#'categorical_crossentropy' debido a que es una clasificacion multiclase

model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X_train, train_labels, epochs=5, batch_size=10)

#prediccion del modelo y de un dato de prueba
predictions = model.predict(X_test)
predictions[0]


print("etiqueta 10 primeras imagenes de prueba:  ",y_test[:10],\
      "\nprediccion 10 primeras imagenes de prueba:",predictions[:10])

_, accuracy = model.evaluate(X_test, test_labels)
print('Accuracy: %.2f' % (accuracy*100))
```

Resultados:

```
convertir:  5  a one hot encoding :  [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
Epoch 1/5
6000/6000 [==============================] - 10s 2ms/step - loss: 0.7879 - accuracy: 0.8914
Epoch 2/5
6000/6000 [==============================] - 11s 2ms/step - loss: 0.2240 - accuracy: 0.9401
Epoch 3/5
6000/6000 [==============================] - 11s 2ms/step - loss: 0.1737 - accuracy: 0.9559
Epoch 4/5
6000/6000 [==============================] - 11s 2ms/step - loss: 0.1553 - accuracy: 0.9632
Epoch 5/5
6000/6000 [==============================] - 11s 2ms/step - loss: 0.1412 - accuracy: 0.9680
313/313 [==============================] - 1s 1ms/step
etiqueta 10 primeras imagenes de prueba:   [7 2 1 0 4 1 4 9 5 9]
```

```
prediccion 10 primeras imagenes de prueba: [[5.86433072e-12 1.62062780e-14 1.14094667e-09 2.52803556e-08
  1.53827423e-10 2.01259731e-13 1.04446330e-11 9.99994516e-01
  1.37498021e-12 5.49852211e-06]
 [3.15530446e-09 1.97780611e-10 9.99999762e-01 1.88304874e-08
  4.93754072e-13 3.20391602e-10 2.44196726e-13 4.42139511e-12
  2.23543253e-07 2.37569274e-12]
 [3.14010695e-37 1.00000000e+00 1.71046840e-35 4.24158676e-25
  2.86584658e-08 2.86726307e-34 6.41908614e-30 0.00000000e+00
  2.40250291e-14 3.60122441e-13]
 [9.99999642e-01 1.84908361e-27 2.21809404e-10 5.97441341e-09
  5.60933732e-13 4.54595632e-11 3.60449448e-09 3.99993468e-07
  1.16853896e-14 4.29937934e-08]
 [8.10460321e-08 9.25268523e-06 8.40430919e-11 5.04995273e-16
  9.99563634e-01 3.01326741e-09 1.32979258e-04 1.43506832e-05
  1.41913063e-06 2.78270745e-04]
 [7.46028383e-31 1.00000000e+00 7.83385816e-25 6.03253101e-19
  1.69424421e-08 2.78541547e-26 5.71299378e-25 5.61554064e-27
  5.68353437e-12 1.56033502e-11]
 [4.06803395e-07 2.83583468e-05 8.75526207e-10 1.77179099e-14
  9.98954773e-01 2.23128804e-08 3.26220354e-04 4.28525855e-05
  5.96381642e-06 6.41471299e-04]
 [1.72209726e-14 1.55523523e-08 1.41277242e-11 3.51479315e-07
  9.59325803e-07 5.53916209e-07 0.00000000e+00 6.78686956e-06
  1.35760945e-06 9.99989986e-01]
 [7.65912555e-05 6.84408238e-03 9.92099755e-04 1.58082433e-02
  1.90722831e-02 2.06153691e-02 2.23532781e-12 2.17604693e-02
  4.33701016e-02 8.71460736e-01]
```

```
 [4.27124718e-18 4.86583884e-10 4.97944920e-15 7.82597875e-11
  3.49860613e-07 3.13740744e-09 0.00000000e+00 3.33409190e-07
  2.98740517e-08 9.99999285e-01]]
313/313 [==============================] - 1s 1ms/step - loss: 0.1856 - accuracy: 0.9567
Accuracy: 95.67
```

4. Evaluación del Modelo:
- Utiliza el conjunto de prueba para evaluar el rendimiento del modelo.
- Calcula métricas de evaluación, como la precisión, el puntaje F1 o la matriz de confusión.

```python
y_test=np.argmax(predictions, axis = 1)
test_labels=np.argmax(test_labels, axis = 1)

# Calculo de la matriz de confusión
cnf_matrix = confusion_matrix(test_labels, y_test)
print(cnf_matrix)
```

Resultados:

```
[[ 973    0    0    0    0    0    1    1    5    0]
 [   0 1113    1    1    1    0    7    0   12    0]
 [  12    0  997    1    0    0    1   11   10    0]
 [   3    0    9  967    0    3    0    3   20    5]
 [   2    0    2    0  941    0    4    0    5   28]
 [   5    0    0   26    0  790    7    2   40   22]
 [   7    2    2    0    5    4  926    0   11    1]
 [  16    2    7    0    3    0    1  978    4   17]
 [  10    0    2    4    7    4    4    2  921   20]
 [   8    3    0    5   13    0    1    5   13  961]]
```

5. Ajustes y Optimización:
   - Experimenta con diferentes arquitecturas de red, números de capas ocultas, unidades en las capas ocultas, tasas de aprendizaje y funciones de activación para optimizar el rendimiento de la red.

Se realizaron los siguientes cambios:

```
##construccion del modelo
num_classes = labels.size ##tamaño de la clases
nu_hl1 = 50    #neuronas en la capa 1
nu_hl2 = 100    #neuronas en la capa 2
nu_hl3 = 130    #neuronas en la capa 3
nu_hl4 = num_classes    #total de clases MNIST


model = Sequential()
model.add(Dense(nu_hl1, input_dim=dim1*dim2, activation='relu'))
model.add(Dense(nu_hl2, activation='relu'))
model.add(Dense(nu_hl3, activation='relu'))
model.add(Dense(nu_hl4, activation='sigmoid'))  #sigmoid
model.summary()         #estructura del modelo
```

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X_train, train_labels, epochs=9, batch_size=15)
```

Donde obtuvimos los siguientes resultados:

```
313/313 [==============================] - 0s 775us/step - loss: 0.1797 - accuracy: 0.9636
Accuracy: 96.36
[[ 958    0    0    1    0    0    5    8    7    1]
 [   0 1124    2    1    0    0    2    0    6    0]
 [   4   32  957    5    2    0    3   16   10    3]
 [   0    0    7  975    0    6    0    6    3   13]
 [   1    1    6    0  930    0    7   10    1   26]
 [   3    3    1   24    1  829   10    1   12    8]
 [   6    4    0    0    6    6  929    0    7    0]
 [   0    3    4    0    1    0    0 1015    1    4]
 [   7    0    4    8    7    4    3   11  921    9]
 [   1    5    1    2   12    1    1   15    3  968]]
```

Presentar los resultados de forma clara y compacta describiendo los modelos usados, y los resultados que obtuvieron para cada uno.

Adjuntar el código generado para el experimento.