Universidad del Valle de Guatemala
Facultad de ingeniería

Laboratorio 2: Deep Learning y Sistemas Inteligentes

Cayetano Molina 20211
Priscilla Gonzalez 20689

Guatemala 06 de julio del 2023

1. **El ancho (tamaño de la capa escondida) del algoritmo. Intenten con un tamaño de 200. ¿Cómo cambia la precisión de validación del modelo? ¿Cuánto tiempo tardó el algoritmo en entrenar? ¿Puede encontrar un tamaño de capa escondida que funcione mejor?**

   Tamaño de la capa = 50: Pérdida de prueba: 0.11. Precisión de prueba: 96.76%
   Tamaño de la capa = 200: Pérdida de prueba: 0.07. Precisión de prueba: 97.72%

   La precisión de la prueba con un tamaño de capa escondida de 200 mejoró en 1% más que el de tamaño de 50, lo que lo hace 1% mejor que la capa de 50. Con respecto al tiempo no varió para nada, ya que prácticamente se tardó lo mismo.

   El tamaño de 1000 ayuda a tener una precisión del 98.27% pero en este caso el tiempo de las épocas es de 17 segundos, por lo que sí afecta en el tiempo.

2. **La profundidad del algoritmo. Agreguen una capa escondida más al algoritmo. Este es un ejercicio extremadamente importante. ¿Cómo cambia la precisión de validación? ¿Qué hay del tiempo que se tarda en ejecutar? Pista: deben tener cuidado con las formas de los pesos y sesgos.**

   Tamaño de la capa = 200 y con una nueva capa escondida: Pérdida de prueba: 0.08. Precisión de prueba: 97.58%

   Comparando el resultado de la capa de 200 anterior y el nuevo resultado, se puede observar que bajó su porcentaje de manera poco significativa, ya que bajó .22%. Con respecto al tiempo que se tarda en ejecutar, no hay ningún cambio, ya que los tiempos siguen siendo los mismos.

3. **El ancho y la profundidad del algoritmo. Agregue cuantas capas sean necesarias para llegar a 5 capas escondidas. Es más, ajuste el ancho del algoritmo conforme lo encuentre más conveniente. ¿Cómo cambia la precisión de validación? ¿Qué hay del tiempo de ejecución?**

   Para poder agregar hasta cinco capas escondidas, lo que se realizó fue asignarle un tamaño de capa a cada capa, teniendo los siguientes valores: [1000, 800, 600, 400, 200]. Estos valores se fueron modificando hasta que la precisión se acercara a “lo mejor”. Con respecto al tiempo de ejecución, en la parte de las épocas, se tomó alrededor de 23 segundos para obtenerlas todas, algo que sí cambia bastante con respecto a las demás salidas.

4. **Experimente con las funciones de activación. Intenten aplicar una transformación sigmoidal a ambas capas. La activación sigmoidal se obtiene escribiendo "sigmoid".**

```
modelo = tf.keras.Sequential()

# Capa de entrada
modelo.add(tf.keras.layers.Flatten(input_shape=(28, 28, 1)))

# Capas escondidas con activación sigmoidal
for neuronas in neuronas_por_capa:
    modelo.add(tf.keras.layers.Dense(neuronas, activation='sigmoid'))

# Capa de salida
modelo.add(tf.keras.layers.Dense(tamanio_salida,
activation='softmax'))
```

De esta forma fue como se aplicó la función de sigmoidal. Al igual que el inciso anterior, el tiempo de ejecución de las épocas fue de alrededor de 23 segundos. Ahora bien, con respecto a la precisión, se obtuvo un 97.05%. Siendo la precisión más baja de todas las pruebas anteriores.

5. **Continúe experimentando con las funciones de activación. Intente aplicar un ReLu a la primera capa escondida y tanh a la segunda. La activación tanh se obtiene escribiendo "tanh".**

```
modelo = tf.keras.Sequential()

# Capa de entrada
modelo.add(tf.keras.layers.Flatten(input_shape=(28, 28, 1)))

# Primera capa escondida con activación ReLU
modelo.add(tf.keras.layers.Dense(neuronas_por_capa[0],
activation='relu'))

# Segunda capa escondida con activación tanh
modelo.add(tf.keras.layers.Dense(neuronas_por_capa[1],
activation='tanh'))

# Resto de capas escondidas con activación tanh
for neuronas in neuronas_por_capa[2:]:
    modelo.add(tf.keras.layers.Dense(neuronas, activation='tanh'))
```

```
# Capa de salida
modelo.add(tf.keras.layers.Dense(tamanio_salida,
activation='softmax'))
```

De esta manera fue como se utilizó las diferentes funciones de activación para las redes neuronales. En este caso, el tiempo de ejecución no varió tanto a como se venía trabajando con la función sigmoidal, pero la precisión fue de 97.67%, siendo un poco mejor que la función sigmoidal. Esto también depende mucho del tamaño de la capa escondida que se le esté pasando como parámetro.

6. **Ajuste el tamaño de la tanda. Pruebe con un tamaño de tanda de 10,000 ¿Cómo cambia el tiempo requerido? ¿Cómo cambia la precisión?**
   El tiempo de ejecución requerido para todo fue de 1 minuto y medio, aproximadamente se tardó 19 segundos para cada época. Sin embargo, en la última época, se tuvo un accuracy del 98.42%. Esto indica un aumento de 0.75% en el accuracy del entrenamiento. No obstante, cuando se obtiene la precisión del modelo en general, se sigue obteniendo una precisión de 97.67%.
7. **Ajuste el tamaño de la tanda a 1. Eso corresponde al SGD. ¿Cómo cambia el tiempo y la precisión? ¿Es el resultado coherente con la teoría?**
   El tiempo de la prueba disminuyó por aproximadamente 6 segundos, y el de cada una de las épocas disminuyó aproximadamente 2 segundos en promedio. Asimismo, la precisión disminuyó a 97.55% lo cual no es mucho para dar a entender que hay un cambio significativo. Sin embargo, si es acorde con la teoría ya que al entrenar solo con un dato a la vez, esto introduce más variabilidad al modelo. Sin embargo, el tiempo debió haber bajado.

8. **Ajuste la tasa de aprendizaje. Pruebe con un valor de 0.0001 ¿Hace alguna diferencia?**
   Si hubo una diferencia, ahora el nivel de precisión aumentó a un 98.52% aproximadamente 1% más del valor de precisión más alto hasta el momento.
9. **Ajuste la tasa de aprendizaje a 0.02 ¿Hay alguna diferencia?**
   Si hubo una diferencia abismal, se obtuvo una precisión de 9.8%. Es decir que el modelo realizado no es confiable y ni siquiera se puede decir que haya aprendido algo.
10. **Combine todos los métodos indicados arriba e intente llegar a una precisión de validación de 98.5% o más.**

```
    # Si se desea, se puede aplicar un formateo "bonito"
    print('Pérdida de prueba: {0:.2f}. Precisión de prueba: {1:.2f}%'.format(perdida_prueba, precision_prueba * 100.))
✓ 0.5s
Pérdida de prueba: 0.07. Precisión de prueba: 98.52%
```

Con una tasa de aprendizaje de 0.0001 se puede obtener un nivel de precisión del 98.52%. Por otro lado, se dejaron las mismas funciones de los anteriores incisos y se dejó el tamaño de tanda estándar de 100. No obstante, cabe recalcar que un learning

rate muy bajo puede llevar a encontrar mínimos locales por lo cual hay que tener cuidado con ello.