Universidad del Valle de Guatemala
Deep Learning
Sección 20

Josué Isaac Morales González 21116
Marco Vinicio Ramirez Centes21032
21-07-2024

Laboratorio 1

1. *Existe diferencia entre la convergencia de los parámetros (pesos y sesgos) si estos son inicializados en 0 o como números aleatorios?*

```
Inicialización aleatoria de los parámetros de la red neuronal
Costo después de la iteración# 0: 1.052558
Costo después de la iteración# 100: 0.695402
Costo después de la iteración# 200: 0.693668
Costo después de la iteración# 300: 0.693206
Costo después de la iteración# 400: 0.692966
Costo después de la iteración# 500: 0.692779
Costo después de la iteración# 600: 0.692587
Costo después de la iteración# 700: 0.692352
Costo después de la iteración# 800: 0.692030
Costo después de la iteración# 900: 0.691539
Costo después de la iteración# 1000: 0.690679
Predicción de la red neuronal para el ejemplo (1, 1) is 1

Inicialización en 0 de los parámetros de la red neuronal
Costo después de la iteración# 0: 0.693147
Costo después de la iteración# 100: 0.693147
Costo después de la iteración# 200: 0.693147
Costo después de la iteración# 300: 0.693147
Costo después de la iteración# 400: 0.693147
Costo después de la iteración# 500: 0.693147
Costo después de la iteración# 600: 0.693147
Costo después de la iteración# 700: 0.693147
Costo después de la iteración# 800: 0.693147
Costo después de la iteración# 900: 0.693147
Costo después de la iteración# 1000: 0.693147
Predicción de la red neuronal para el ejemplo (1, 1) is 1
```

Sí, existe una diferencia significativa. Si los parámetros se inicializan en cero, cada neurona en la red aprende de manera idéntica, lo que lleva a que todas las neuronas reciban las mismas actualizaciones durante el entrenamiento. Esto resulta en una simetría que impide que la red aprenda de manera efectiva, ya que no se produce una reducción significativa en la función de costo y los pesos y sesgos permanecen inmutables. Por otro lado, la inicialización aleatoria de los parámetros rompe esta simetría, permitiendo que las neuronas aprendan diferentes características desde el inicio. Esto generalmente lleva a una mejor representación del problema y a una convergencia más rápida, como se puede observar en la disminución gradual del costo a lo largo de las iteraciones.

2. *¿Qué diferencia en la convergencia de la función de costo y los parámetros existe si el learning rate del código es 0.01? 0.1? 0.5?*

```
Probando con tasa de aprendizaje: 0.01
Costo después de la iteración #0: 1.052558
Costo después de la iteración #100: 0.948307
Costo después de la iteración #200: 0.864690
Costo después de la iteración #300: 0.803459
Costo después de la iteración #400: 0.765007
Costo después de la iteración #500: 0.742498
Costo después de la iteración #600: 0.729047
Costo después de la iteración #700: 0.720556
Costo después de la iteración #800: 0.714882
Costo después de la iteración #900: 0.710898
Costo después de la iteración #1000: 0.707985
Predicción de la red neuronal para el ejemplo (1, 1) con tasa de aprendizaje 0.01 es 1

Probando con tasa de aprendizaje: 0.1
Costo después de la iteración #0: 1.013592
Costo después de la iteración #100: 0.674649
Costo después de la iteración #200: 0.629729
Costo después de la iteración #300: 0.590321
Costo después de la iteración #400: 0.561134
Costo después de la iteración #500: 0.540238
Costo después de la iteración #600: 0.524827
Costo después de la iteración #700: 0.512317
Costo después de la iteración #800: 0.498693
Costo después de la iteración #900: 0.452745
Costo después de la iteración #1000: 0.312239
Predicción de la red neuronal para el ejemplo (1, 1) con tasa de aprendizaje 0.10 es 0

Probando con tasa de aprendizaje: 0.5
Costo después de la iteración #0: 0.713866
Costo después de la iteración #100: 0.637009
Costo después de la iteración #200: 0.531485
Costo después de la iteración #300: 0.504387
Costo después de la iteración #400: 0.494727
Costo después de la iteración #500: 0.489987
Costo después de la iteración #600: 0.487218
Costo después de la iteración #700: 0.485415
Costo después de la iteración #800: 0.484154
Costo después de la iteración #900: 0.483226
Costo después de la iteración #1000: 0.482515
Predicción de la red neuronal para el ejemplo (1, 1) con tasa de aprendizaje 0.50 es 1
```

El learning rate tiene un impacto significativo en la convergencia de la función de costo y de los parámetros. Con un learning rate de 0.01, la convergencia es muy lenta, lo que indica que la red está aprendiendo, pero el proceso es ineficiente. Con un learning rate de 0.1, la red converge de manera más eficiente y rápida, alcanzando un buen equilibrio entre la velocidad de aprendizaje y la estabilidad de las actualizaciones. Sin embargo, con un

learning rate de 0.5, aunque la red converge rápidamente al principio, existe el riesgo de que el modelo salte alrededor del mínimo de la función de costo, lo que puede llevar a una convergencia menos estable y a predicciones inconsistentes. Por lo tanto, un learning rate intermedio como 0.1 suele ser más adecuado para lograr una buena convergencia sin comprometer la estabilidad del modelo.

3. *Implemente MSE como función de costo y propague los cambios en las funciones que lo requieran. ¿Qué cambios observa?*

```
Entrenamiento con MSE:
Cost after iteration #0: 0.357529
Cost after iteration #100: 0.251127
Cost after iteration #200: 0.250260
Cost after iteration #300: 0.250029
Cost after iteration #400: 0.249910
Cost after iteration #500: 0.249816
Cost after iteration #600: 0.249720
Cost after iteration #700: 0.249602
Cost after iteration #800: 0.249441
Cost after iteration #900: 0.249196
Cost after iteration #1000: 0.248766
Predicción de la red neuronal para el ejemplo (1, 1) is 1
```

Inicialmente, el costo con MSE puede ser diferente, pero lo más notable es el comportamiento de la convergencia. Con MSE, el costo disminuye de manera más gradual y estable sin grandes oscilaciones, lo que resulta en una convergencia más suave. Esto puede ser ventajoso para problemas de regresión donde se busca minimizar la diferencia promedio al cuadrado entre las predicciones y los valores reales. En general, la función de costo MSE proporciona una convergencia estable y una reducción constante del costo durante las iteraciones, manteniendo predicciones precisas y un comportamiento estable del modelo.

Link al repositorio: https://github.com/mvrcentes/deep-learning_2024.git