Universidad del Valle de Guatemala
Deep learning y sistemas inteligentes
Sección 20

Laboratorio No. 1

Carlos Edgardo López Barrera 21666

Brian Anthony Carrillo Monzon - 21108

Guatemala, 29 de julio del 2024

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

   La inicialización de los pesos y sesgos es importante para el entrenamiento. Si los parámetros se inicializan en 0, cada neurona en la red aprende las mismas características durante el mismo, lo que lleva a una representación deficiente del problema y una convergencia lenta. Ahora bien, la inicialización aleatoria rompe la simetría y permite que diferentes neuronas aprendan diferentes características, lo que en la mayoría de ocasiones nos lleva a una mejor y más rápida convergencia.

```
Inicialización aleatoria:
Cost after iteration# 0: 1.052558
Cost after iteration# 100: 0.695402
Cost after iteration# 200: 0.693668
Cost after iteration# 300: 0.693206
Cost after iteration# 400: 0.692966
Cost after iteration# 500: 0.692779
Cost after iteration# 600: 0.692587
Cost after iteration# 700: 0.692352
Cost after iteration# 800: 0.692030
Cost after iteration# 900: 0.691539
Cost after iteration# 1000: 0.690679
Neural Network prediction for example (1, 1) is 1

Inicialización en 0:
Cost after iteration# 0: 0.693147
Cost after iteration# 100: 0.693147
Cost after iteration# 200: 0.693147
Cost after iteration# 300: 0.693147
Cost after iteration# 400: 0.693147
Cost after iteration# 500: 0.693147
Cost after iteration# 600: 0.693147
Cost after iteration# 700: 0.693147
Cost after iteration# 800: 0.693147
Cost after iteration# 900: 0.693147
Cost after iteration# 1000: 0.693147
Neural Network prediction for example (1, 1) is 1
```

   Inicialización aleatoria

   - **Iteración # 0: 1.052558**: El costo inicial es relativamente alto, lo cual es esperado dado que los parámetros son inicializados aleatoriamente y la red no ha aprendido nada aún.
   - **Iteración # 100: 0.695402**: El costo disminuye significativamente después de 100 iteraciones.
   - **Iteración # 1000: 0.690679**: El costo sigue disminuyendo gradualmente, indicando que la red está aprendiendo y ajustando sus parámetros para minimizar el error.

Inicialización en 0

- **Iteración # 0: 0.693147**: El costo inicial es más bajo que en el primer caso aleatorio.
- **Iteración # 100: 0.693147**: El costo no cambia después de 100 iteraciones.
- **Iteración # 1000: 0.693147**: El costo sigue siendo el mismo después de 1000 iteraciones, indicando que la red no está aprendiendo nuevas cosas.

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?**

   El learning rate determina el tamaño del paso que la red toma durante el entrenamiento. Un learning rate muy pequeño puede llevar a una convergencia muy lenta, mientras que uno muy grande puede hacer que esta no converja.

```
Testing with learning rate: 0.01
Cost after iteration# 0: 1.052558
Cost after iteration# 100: 0.948307
Cost after iteration# 200: 0.864690
Cost after iteration# 300: 0.803459
Cost after iteration# 400: 0.765007
Cost after iteration# 500: 0.742498
Cost after iteration# 600: 0.729047
Cost after iteration# 700: 0.720556
Cost after iteration# 800: 0.714882
Cost after iteration# 900: 0.710898
Cost after iteration# 1000: 0.707985
Neural Network prediction for example (1, 1) with learning rate 0.01 is 1

Testing with learning rate: 0.1
Cost after iteration# 0: 1.013592
Cost after iteration# 100: 0.674649
Cost after iteration# 200: 0.629729
Cost after iteration# 300: 0.590321
Cost after iteration# 400: 0.561134
Cost after iteration# 500: 0.540238
Cost after iteration# 600: 0.524827
Cost after iteration# 700: 0.512317
Cost after iteration# 800: 0.498693
Cost after iteration# 900: 0.452745
Cost after iteration# 1000: 0.312239
Neural Network prediction for example (1, 1) with learning rate 0.10 is 0

Testing with learning rate: 0.5
Cost after iteration# 0: 0.713866
Cost after iteration# 100: 0.637009
Cost after iteration# 200: 0.531485
Cost after iteration# 300: 0.504387
Cost after iteration# 400: 0.494727
Cost after iteration# 500: 0.489987
Cost after iteration# 600: 0.487218
Cost after iteration# 700: 0.485415
Cost after iteration# 800: 0.484154
Cost after iteration# 900: 0.483226
Cost after iteration# 1000: 0.482515
Neural Network prediction for example (1, 1) with learning rate 0.50 is 1
```

- **Learning Rate 0.01:** La red converge lentamente, lo cual es indicativo de un learning rate demasiado bajo. Aunque la red está aprendiendo, el proceso es muy lento.
- **Learning Rate 0.1:** La red converge de manera más eficiente y rápida en comparación con el learning rate de 0.01.vo de sobreajuste o de que la red no ha encontrado el óptimo global.
- **Learning Rate 0.5:** La red converge muy rápidamente y el costo alcanza un valor estable, lo que indica una buena convergencia.

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

   La Mean Squared Error (MSE) es otra función de costo que se utiliza en problemas de regresión. A diferencia de la entropía cruzada, que mide la distancia entre dos distribuciones probabilísticas, MSE mide la diferencia promedio al cuadrado entre las predicciones y los valores reales.

```
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
Neural Network prediction for example (1, 1) is 1
```

   Comportamiento del costo:

- **Costo inicial:** El costo inicial con MSE fue 0.357529.
- **Disminución del costo:** El costo disminuyó significativamente en las primeras iteraciones y se estabilizó cerca de 0.25 después de 1000 iteraciones.

Convergencia:

- **Convergencia más suave:** El uso de MSE resultó en una disminución más gradual y estable del costo sin grandes oscilaciones.
- **Estabilidad:** La función de costo MSE mostró una convergencia estable, manteniendo un descenso constante en el costo durante las iteraciones.

MSE: Proporcionó una convergencia estable y predicciones precisas, con un costo que decreció gradualmente.

Entropía Cruzada: Muestra una disminución más rápida del costo al principio, pero puede tener más oscilaciones.

Link repositorio de github: https://github.com/Carloslpez219/Lab01-DeepLearning.git