Memoria
Práctica 4

Fundamentos de Aprendizaje Automático

Diego Pareja Serrano

INTRODUCCIÓN

Esta práctica se ha centrado principalmente en el diseño , entrenamiento y evaluación de las redes neuronales, uno de los temas más importante de la actualidad, en su totalidad consta de 4 ejercicios, que abordan diferentes aspectos sobre el tema.

Hemos podido trabajar con una biblioteca nn.py que simula el comportamiento básico de las redes neuronales. Esta herramienta nos ha permitido implementar manualmente operaciones como pueden ser funciones de activación, cálculo de perdidas, capas lineales... Favoreciendo así una comprensión en profundidad del funcionamiento interno del modelo sin recurrir a librerías externas.

En conjunto la práctica me ha ofrecido una visión progresiva y muy completa del aprendizaje profundo, yendo desde modelos simples hasta arquitecturas recurrentes, destacando la importancia de la arquitectura, los hiperparámetros y el criterio experimental.

DESARROLLO DE LA PRÁCTICA (Análisis propio)

Ejercicio 1: Perceptrón binario

Se generó un perceptrón clásico para que clasificara los datos linealmente separables. También se hizo uso de una capa lineal sin activación y se aplicó la regla de actualización estándar para modificar los pesos en función de errores de predicción. El entrenamiento finalizaba al lograr una clasificación correcta en los datos.

Ejercicio 2: Regresión no lineal

Se entrenó una red neuronal para realizar la aproximación una función no lineal. Para así lograr un ajuste preciso (error < 0.02), se probaron varias arquitecturas hasta encontrar una red de una capa oculta de 256 neuronas con activación ReLU, que ofrecía gran poder expresivo sin necesidad de aumentar la profundidad.

Ejercicio 3: Clasificación de dígitos MNIST

Se diseñó una red neuronal multicapa (MLP) para reconocer dígitos escritos a mano. Se utilizó una arquitectura con dos capas ocultas y activación ReLU, seguida de una capa final de 10 salidas con pérdida nn.SoftmaxLoss. Tras entrenamiento por batches y validación, se alcanzó una precisión superior al 97%, cumpliendo el requisito del ejercicio.

Ejercicio 4: Identificación de idiomas (RNN)

Se implementó una **red neuronal recurrente (RNN)** para clasificar palabras según su idioma. Cada palabra se representó como una secuencia de caracteres codificados como vectores one-hot. Se usó una estructura recurrente que procesaba letra por letra y generaba un estado oculto acumulativo, el cual se pasaba por una capa final de clasificación. El entrenamiento por epochs permitió alcanzar una precisión de validación ≥ 89%.

DIFICULTADES ENCONTRADAS Y SOLUCIONES APLICADAS

1.Comprender y ajustar arquitecturas

Problema: Definir una arquitectura que se ajustara bien a cada problema no fue algo que ocurriese al primerintento. En especial, la regresión requería un modelo lo bastante flexible para capturar una función no lineal.

Solución: Realice una exploración sistemática de combinaciones de capas y neuronas. Finalmente, se optó por una única capa oculta de 256 neuronas, lo que me sorprendió por su capacidad de ajuste, incluso mejor que redes más profundas que se intentaron también.

2.Implementar lógica recurrente en la RNN

Problema: En el ejercicio 4, la RNN debía procesar secuencias de longitud variable y compartir parámetros entre todos los pasos. Era fácil cometer errores al combinar correctamente x y h_{t-1}.
Solución: Se aplicó la estrategia recomendada: construir z = x @ W + h @ W_hidden + b usando nn.Add y nn.Linear, y reutilizar parámetros en cada paso. Se hizo mucho énfasis en validar la forma y consistencia de las dimensiones en cada paso.

3.Estabilidad en el entrenamiento

Problema: El entrenamiento de la red del ejercicio 3 (MNIST) a veces no convergía o lo hacía muy de manera muy lenta.
Solución: Ajuste los hiperparámetros, reduciendo el learning rate e introduciendo early stopping según la precisión de validación. También se eligió un tamaño de batch adecuado (32 o 64) para equilibrar rapidez y estabilidad.

4. Overfitting y generalización

Problema: En la regresión, modelos muy profundos (2 o más capas) terminaban sobreajustando y obteniendo malos resultados generales.
Solución: Se redujo la red a **una sola capa oculta con muchos nodos**, lo que permitió mantener capacidad de aprendizaje sin perder generalización. También se validó la pérdida tras cada epoch y se cortó el entrenamiento temprano al alcanzar el umbral requerido.

5.Entrenamiento excesivo en la RNN

Problema: Inicialmente, el entrenamiento de la RNN se prolongaba innecesariamente más allá del punto óptimo.
Solución: Se implementó una condición de parada temprana basada en la precisión del conjunto de validación (acc >= 0.89), lo que permitió ahorrar tiempo y evitar sobreentrenamiento.

CONCLUSION PERSONAL

La práctica 4 ha sido en mi opinión la que más me ha llamado la atención, por su cercanía con la realidad. La práctica ha ido desde implementar modelos básicos como el perceptrón, hasta construir redes neuronales profundas e incluso recurrentes. Se ha trabajado con diversos problemas de regresión, clasificación de imágenes y secuencias, todos muy representativos de aplicaciones reales de machine learning.

Desde mi experiencia, lo más interesante ha sido observar cómo pequeñas decisiones en el diseño de arquitectura, tamaño de capa, función de pérdida o tasa de aprendizaje afectan de una manera tan directa a la capacidad de aprendizaje y generalización del modelo. Esta práctica no solo ha mejorado mi comprensión técnica, sino que me ha permitido obtener de cierta manera criterio práctico a la hora de diseñar redes neuronales.

Además, destacar la parte de la RNN como la más complicada pero a la vez interesante, ya que permitió ver cómo se puede procesar información secuencial y representar el estado en el tiempo.