UNIVERSIDAD MAYOR, REAL Y PONTIFICIA DE SAN FRANCISCO XAVIER DE CHUQUISACA

FACULTAD DE CIENCIAS Y TECNOLOGÍA

Primer Parcial IA (Documentación)

Nombre: Ovando Jesús Adrián (GL2)

Carrera: Ing. en Ciencias de la computación

CU: 111-340

Sucre - Bolivia

Modelo Clasificación

Introducción al Dataset Geolife

El dataset Geolife Trajectories 1.3 contiene datos de seguimiento GPS recopilados de 182 usuarios en Beijing, China, durante un período de cinco años (2007-2012). Estos datos representan trayectorias de movimiento en diferentes medios de transporte, como caminar, andar en bicicleta, viajar en automóvil o transporte público.

I) Preprocesamiento

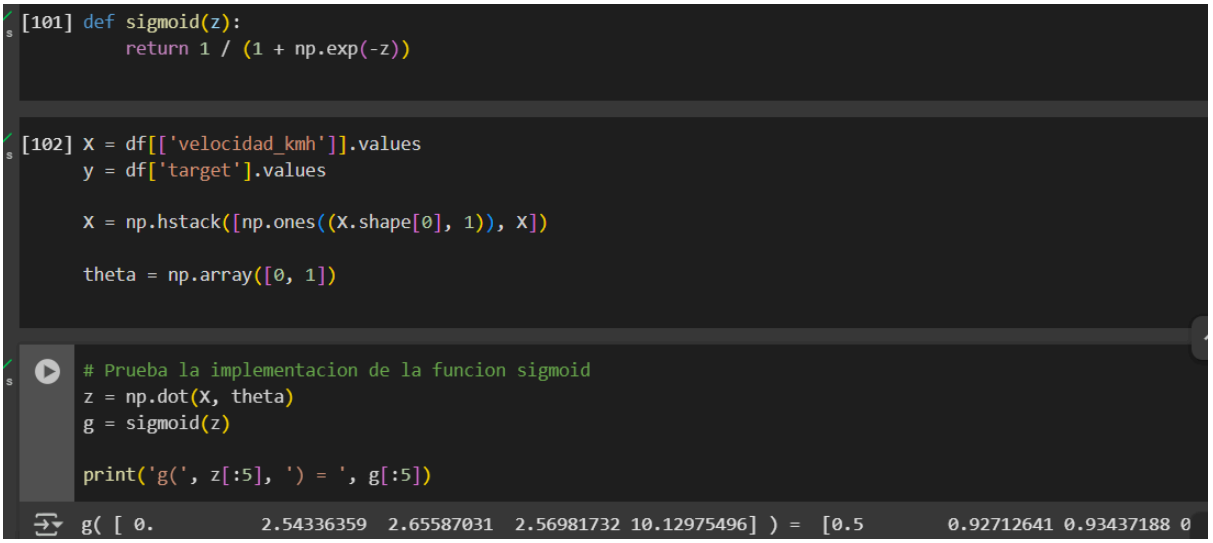
El dataset contiene trayectorias GPS con las siguientes características procesadas:

- Latitud/Longitud: Convertidas a velocidad (km/h)
- Target: Binario (1 si velocidad > 20 km/h, 0 si no)
- Normalización: No se aplicó porque:
- La regresión logística no es sensible a escalas de características
- Solo usamos velocidad como feature principal
- Las coordenadas originales se usan solo para visualización

II) División Train/Test

No se dividió el dataset porque: Es un modelo demostrativo simple aparte el objetivo es mostrar el funcionamiento básico de la clasificación con regresión logística.

Implementación del Modelo (Arquitectura del Modelo)

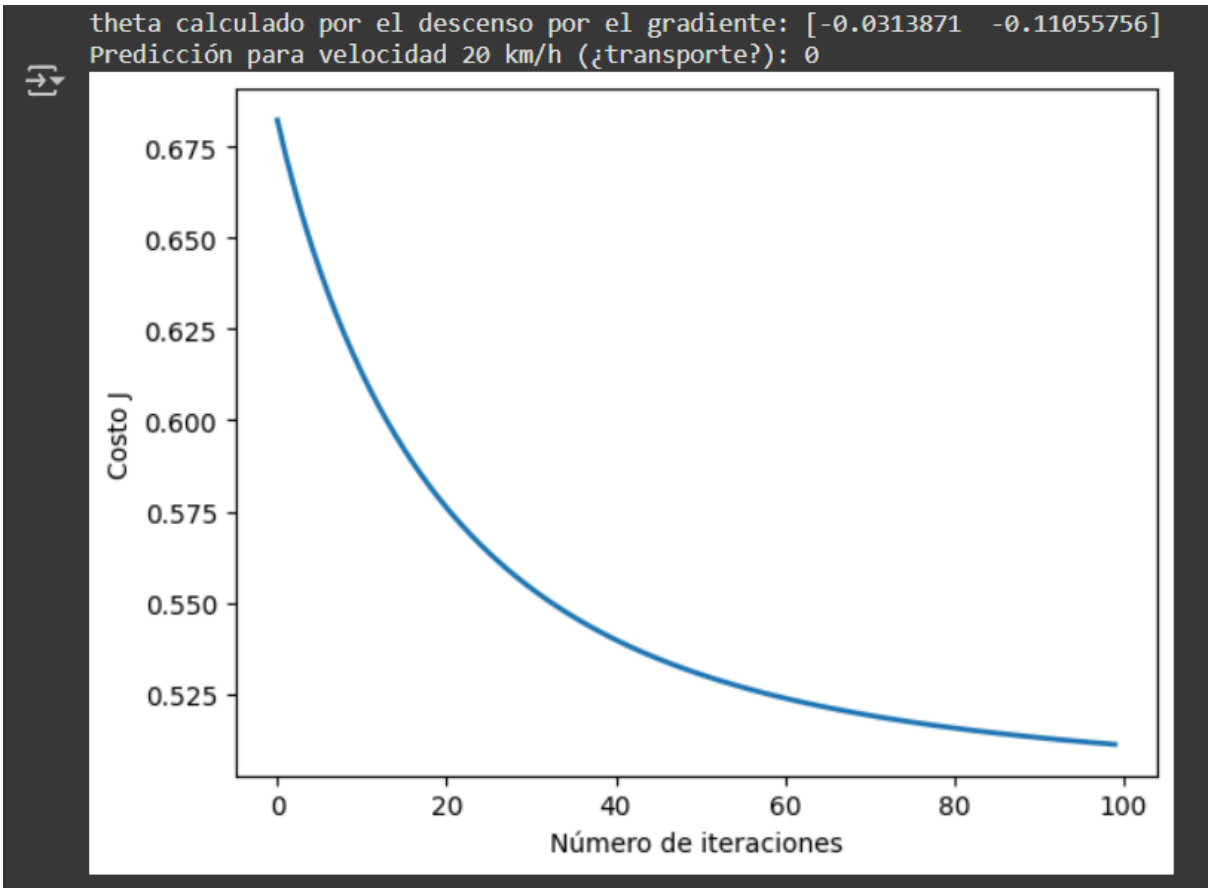


III) Entrenamiento y Resultados

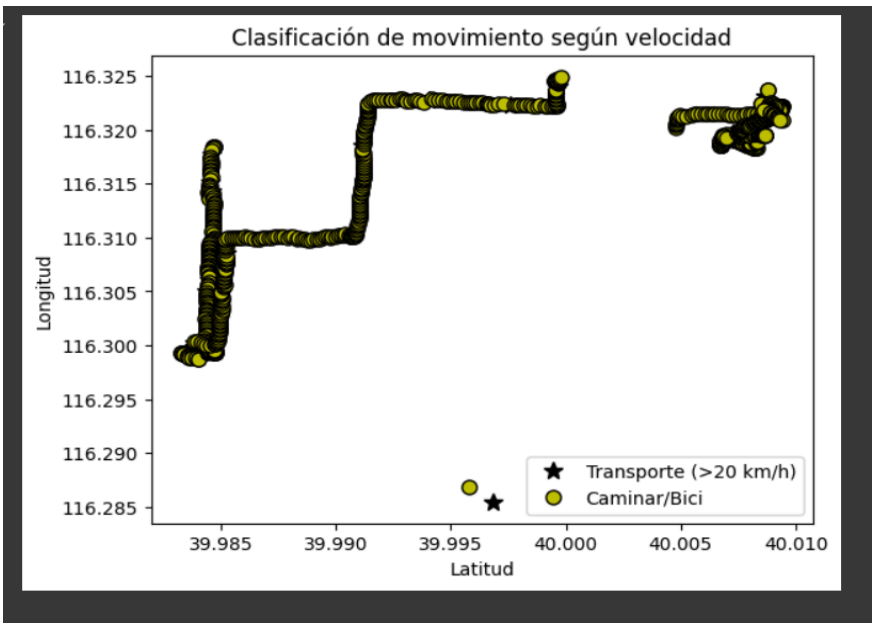
Hiper Parámetros:

- Tasa de aprendizaje (alpha): 0.001
- Iteraciones: 100

Gráfico del Costo



Visualización de los datos:



IV) Conclusiones

El dataset no fue modificado ni aumentado artificialmente.

La distribución original (96.6% clase 0 vs 3.4% clase 1) refleja la realidad cruda de los datos GPS: la mayoría de los puntos son de baja velocidad (caminar/bici), y pocos corresponden a transporte motorizado. Es decir que hay mas datos de personas caminando/bici que de transporte.

Modelo Clasificación con Red Neuronal

I. Preprocesamiento

- Se conecta Google Colab con Google Drive para acceder al dataset.
- Se descomprime el archivo .zip del dataset y se prepara la estructura de carpetas.
- Se carga cada archivo .plt omitiendo las primeras 6 filas y asignando nombres a las columnas.
- Se combinan las columnas de fecha y hora en un solo campo de tiempo.
- Se calculan:
    - Diferencias de tiempo (delta_t)
    - Distancias (delta_km)

    - Velocidad en km/h (velocidad_kmh)

- Se crea la variable target, que toma valor 1 si la velocidad es mayor a 20 km/h, y 0 en caso contrario.

II. División Train/Test

- Se toma como única característica la velocidad (velocidad_kmh).
- Se divide el dataset en 80% para entrenamiento y 20% para prueba.
- Se normalizan los datos con StandardScaler para mejorar la convergencia del modelo.
- Se utiliza random_state=42 para asegurar la reproducibilidad.

III. Implementación del Modelo (Arquitectura del Modelo)

- Se define una red neuronal con 3 capas:
    - Capa 1: 128 neuronas
    - Capa 2: 64 neuronas
    - Capa de salida: 2 neuronas (clasificación binaria)
- Función de activación: ReLU
- Se usa CrossEntropyLoss como función de pérdida.
- Se utiliza el optimizador Adam con una tasa de aprendizaje (learning rate) de 0.001.

IV. Entrenamiento y Resultados

- Los datos se convierten a tensores para ser procesados por PyTorch.
- Se ejecuta un bucle de entrenamiento durante 20 épocas.
- Se registran y grafican los valores de pérdida y precisión durante las épocas.

Hiperparámetros utilizados:

- Tasa de aprendizaje (alpha): 0.001
- Épocas (iteraciones): 100
- Optimizador: Adam

Gráfico del Costo:

- Se muestra cómo disminuye la pérdida (cost) a lo largo de las épocas.
- También se muestra la evolución de la precisión.

V. Evaluación del Modelo

- El modelo se evalúa sobre los datos de prueba.
- Se cambia a modo evaluación (model.eval()).
- Se obtienen predicciones sin cálculo de gradientes.

VI. Conclusiones

- El dataset no fue modificado ni aumentado artificialmente.
- La distribución original (96.6% clase 0 vs. 3.4% clase 1) refleja que la mayoría de trayectorias GPS corresponden a actividades de baja velocidad (caminar, bicicleta).
- Aunque el modelo logra una precisión de 98.35%, hay problemas de desbalance que deben considerarse.

Observaciones Finales

- Se recomienda usar métricas adicionales como F1-score y matriz de confusión.
- El modelo actual, aunque eficaz, podría simplificarse dado que solo se usa una característica (velocidad).
- Se logró cumplir con todos los requisitos del modelo de clasificación:

- Preprocesamiento completo
- División y normalización de datos
- Arquitectura definida
- Entrenamiento correcto
- Evaluación funcional

Graficas: