Guía Práctica: Redes Neuronales

Entrenamiento y optimización

Universidad Santiago de Cali

October 31, 2024

1 Guía Práctica de Optimización en Redes Neuronales

1.1 Objetivo

Aplicar y comparar algoritmos de optimización en redes neuronales mediante la implementación y entrenamiento de un modelo en `Python` usando `TensorFlow`. Los estudiantes observarán el impacto de los optimizadores y de diferentes tasas de aprendizaje en el rendimiento del modelo.

1.2 Instrucciones Generales

Trabajen en grupos de tres estudiantes y seleccionen uno de los siguientes datasets para la práctica:

- **Titanic**: Supervivencia de pasajeros (clasificación binaria).
- **Diabetes**: Diagnóstico de diabetes (clasificación binaria).
- **House Prices**: Predicción de precios de viviendas (regresión).
- **Heart Disease**: Diagnóstico de enfermedades cardíacas (clasificación binaria).
- **Customer Churn**: Predicción de abandono de clientes (clasificación binaria).

1.3 Paso a Paso

1.3.1 1. Selección y Carga del Dataset

- Seleccionen un dataset de la lista y cárguenlo en su entorno de trabajo en `Python`.
- Realicen una breve exploración y preprocesamiento básico del dataset según las necesidades del modelo (normalización, limpieza, etc.).

1.3.2 2. Implementación de la Red Neuronal

Implementen una red neuronal simple en `TensorFlow` que se adapte a las características del problema seleccionado (clasificación o regresión):

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

```
model = Sequential([
    Dense(16, activation='relu', input_shape=(input_dim,)),
    Dense(8, activation='relu'),
    Dense(1, activation='sigmoid') # Para clasificación binaria
])
```

1.3.3 3. Configuración del Optimizador SGD y Entrenamiento

Entrenen la red utilizando el optimizador `SGD` y una tasa de aprendizaje inicial. Luego, cambien la tasa de aprendizaje y observen cómo afecta la convergencia.

```
from tensorflow.keras.optimizers import SGD

# Configuración del optimizador SGD
optimizer_sgd = SGD(learning_rate=0.01)
model.compile(optimizer=optimizer_sgd, loss='binary_crossentropy', metrics=['accuracy'])

# Entrenamiento del modelo
history_sgd = model.fit(X_train, y_train, epochs=50, validation_data=(X_val, y_val))
```

1.3.4 4. Cambio al Optimizador Adam

Reemplacen el optimizador con `Adam` y repitan el proceso, probando distintas tasas de aprendizaje.

```
from tensorflow.keras.optimizers import Adam

# Configuración del optimizador Adam
optimizer_adam = Adam(learning_rate=0.01)
model.compile(optimizer=optimizer_adam, loss='binary_crossentropy', metrics=['accuracy'])

# Entrenamiento del modelo
history_adam = model.fit(X_train, y_train, epochs=50, validation_data=(X_val, y_val))
```

1.4 Análisis de Resultados

Para cada optimizador y tasa de aprendizaje, generen gráficos de las métricas de aprendizaje:

- Comparen las gráficas de **pérdida** y **precisión** en función de las épocas.
- Observen cómo la tasa de aprendizaje afecta la convergencia y estabilidad del modelo.

1.5 Conclusión

Con base en los gráficos obtenidos, cada grupo deberá responder las siguientes preguntas:

- ¿Qué optimizador mostró una convergencia más rápida?
- ¿Cómo afectaron las distintas tasas de aprendizaje al rendimiento del modelo?
- ¿Cuál fue el impacto de cada optimizador en la estabilidad de la convergencia?

Este análisis se utilizará como base para la discusión teórica final en clase, en donde se revisarán los principios de cada optimizador y su aplicación en problemas de redes neuronales.