Manual de Usuario para el Códido de Preentrenamiento de TabNet No Supervisado

Grupo de Control y Procesamiento Digital de Señales
Universidad Nacional de Colombia
CHEC

April 28, 2025

Contents

1 Descripción General

Este código implementa un modelo de aprendizaje automático no supervisado basado en la arquitectura TabNet. TabNet es un modelo de red neuronal diseñado específicamente para datos tabulares, que combina la capacidad de aprendizaje profundo con la interpretabilidad de los modelos basados en árboles. El objetivo principal de este código es preentrenar un modelo TabNet en un conjunto de datos no etiquetado, lo que permite al modelo aprender representaciones útiles de los datos sin la necesidad de etiquetas de entrenamiento.

El código está diseñado para ser ejecutado en un entorno de Google Colab, aunque también puede ser adaptado para ejecutarse en otros entornos de desarrollo de Python. El modelo preentrenado puede luego ser utilizado para tareas de aprendizaje supervisado o no supervisado, como la clasificación, la regresión o la reducción de dimensionalidad.

2 Funcionalidades Principales

El código proporcionado implementa las siguientes funcionalidades principales:

- **Instalación de Dependencias:** El código instala automáticamente las bibliotecas necesarias, como `pytorch-tabnet` y `wget`, para asegurar que todas las dependencias estén disponibles.
- **Definición del Modelo TabNetPretraining:** Se define una clase `TabNetPretraining` que implementa la arquitectura TabNet para el preentrenamiento no supervisado. Esta clase incluye métodos para el entrenamiento, la predicción y la explicación del modelo.
- **Entrenamiento del Modelo:** El código proporciona una clase `TabNetPretrainer` que encapsula el proceso de entrenamiento del modelo. Incluye métodos para ajustar los hiperparámetros, entrenar el modelo y evaluar su rendimiento en conjuntos de datos de validación.
- **Visualización de Resultados:** Se incluyen funciones para visualizar los resultados del preentrenamiento, como la reconstrucción de datos, las máscaras de atención y las representaciones aprendidas mediante t-SNE.
- **Carga y Preprocesamiento de Datos:** El código incluye un ejemplo de cómo cargar y preprocesar un conjunto de datos de fenotipificación, incluyendo la codificación de variables categóricas y el manejo de valores faltantes.

3 Bibliotecas Necesarias

Para ejecutar este código, se necesitan las siguientes bibliotecas de Python:

- `torch`: Biblioteca de aprendizaje profundo de PyTorch.
- `numpy`: Biblioteca para operaciones numéricas.
- `pandas`: Biblioteca para manipulación de datos tabulares.
- `pytorch-tabnet`: Implementación de la arquitectura TabNet para PyTorch.
- `scipy`: Biblioteca para operaciones científicas.
- `matplotlib`: Biblioteca para visualización de datos.
- `seaborn`: Biblioteca para visualización de datos estadísticos.
- `sklearn`: Biblioteca de aprendizaje automático de scikit-learn.
- `wget`: Biblioteca para descargar archivos desde la web.

4 Archivos Necesarios

Para ejecutar este código, se necesitan los siguientes archivos:

- `PhenotypingData.csv`: Archivo de datos de fenotipificación. Este archivo debe estar en el mismo directorio que el código o en una ruta accesible.

- $\texttt{Datasets}_b iplot.zip$: *Archivo comprimido que contiene el conjunto de datos de fenotipificación. Este archivo se descarga auto*

5 Instrucciones de Uso

5.1 Entorno de Ejecución

Este código está diseñado para ser ejecutado en un entorno de Google Colab. Si se ejecuta en otro entorno, asegúrese de instalar las bibliotecas necesarias utilizando los siguientes comandos:

```
!pip install pytorch-tabnet --quiet
!pip install wget --quiet
```

5.2 Carga de Datos

El código carga un conjunto de datos de fenotipificación desde un archivo CSV. Asegúrese de que el archivo `PhenotypingData.csv` esté en el mismo directorio que el código o en una ruta accesible. Si el archivo no está disponible, el código intentará descargarlo automáticamente desde un repositorio de GitHub.

5.3 Preprocesamiento de Datos

El código realiza el siguiente preprocesamiento de datos:

- Codificación de variables categóricas utilizando `LabelEncoder`.
- Manejo de valores faltantes reemplazándolos con el promedio de la columna correspondiente.
- Normalización de datos numéricos.

5.4 Entrenamiento del Modelo

Para entrenar el modelo, ejecute el siguiente código:

```
unsupervised_model = TabNetPretrainer(
    cat_idxs=cat_idxs,
    cat_dims=cat_dims,
    cat_emb_dim=3,
    gamma=1.6,
    n_d=10,
    n_a=10,
    n_steps=5,
    lambda_sparse=0.01,
    optimizer_fn=torch.optim.Adam,
    optimizer_params=dict(lr=2e-2),
    mask_type="sparsemax",
    n_shared_decoder=2,
    n_indep_decoder=2,
    verbose=5,
)

unsupervised_model.fit(
    X_train=X_train,
    eval_set=[X_valid],
    max_epochs=100,
```

```
    patience=5,
    batch_size=500,
    virtual_batch_size=125,
    num_workers=0,
    drop_last=False,
    pretraining_ratio=0.5,
)
```

5.5 Visualización de Resultados

Después de entrenar el modelo, puede visualizar los resultados utilizando las siguientes funciones:

- **Reconstrucción de Datos:** Visualice la reconstrucción de datos utilizando t-SNE.
- **Máscaras de Atención:** Visualice las máscaras de atención aprendidas por el modelo.
- **Matriz de Explicación:** Visualice la matriz de explicación del modelo.

Ejemplo de código para visualizar los resultados:

```
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE

# Reconstrucci n de datos
reconstructed_X, embedded_X, encoder = unsupervised_model.predict(X_test)
x_embedded = TSNE(n_components=2).fit_transform(encoder)

plt.figure(figsize=(8, 4))
plt.scatter(x_embedded[:, 0], x_embedded[:, 1])
plt.title("Reconstrucci n de Datos con t-SNE")
plt.show()

# M scaras de atenci n
unsupervised_explain_matrix, unsupervised_masks = unsupervised_model.explain(X_test)

fig, axs = plt.subplots(1, 3, figsize=(20, 20))
for i in range(3):
    axs[i].imshow(unsupervised_masks[i][:50])
    axs[i].set_title(f"M scara {i}")
plt.show()

# Matriz de explicaci n
import seaborn as sns

plt.figure(figsize=(22, 15))
sns.heatmap(pd.DataFrame(unsupervised_explain_matrix, columns=features), cmap="viridis")
plt.title("Matriz de Explicaci n")
plt.show()
```

6 Consideraciones Adicionales

- Asegúrese de que el archivo `PhenotypingData.csv` esté en el mismo directorio que el código o en una ruta accesible.
- Si el archivo `PhenotypingData.csv` no está disponible, el código intentará descargarlo automáticamente desde un repositorio de GitHub.
- Ajuste los hiperparámetros del modelo según sea necesario para obtener mejores resultados.

- El entrenamiento del modelo puede ser computacionalmente intensivo. Asegúrese de tener suficiente capacidad de procesamiento y memoria.
- El código proporciona un ejemplo de cómo visualizar los resultados del preentrenamiento. Puede adaptar estas funciones para visualizar otros aspectos del modelo.