Reconocimiento de patrones y aprendizaje automático
Tarea 3: Redes neuronales y árboles de decisión

Fecha de entrega: Lunes 24 de enero de 2022

Profesor: Andrés Aldana Gonzáles
Ayudante: Felipe Navarrete Córdova

1. Investigación

Realiza un breve resumen donde contestes las siguientes preguntas:

- ¿Qué son las redes neuronales recurrentes (RNN's)?
- Describe los diferentes tipos de RNN's
- ¿Qué tipo de problemas se pueden resolver a través de RNN's?
- ¿Qué son las redes de tipo *Long Short-Term Memory* (LSTM)?
- Describe la arquitectura principal de una red LSTM
- ¿Cuál es la ventaja de usar LSTM en la predicción de series de tiempo sobre otros tipos de redes neuronales?

2. Implementación de redes neuronales

El objetivo de este ejercicio es construir una red neuronal para clasificar el tipo de actividad que realiza una persona, usando como entrada los datos de un acelerómetro, que se encuentran en el archivo **dataset2.zip**.

Los datos fueron recolectados de un acelerómetro puesto en el pecho de 15 participantes que realizan siete actividades. Los datos están separados por participante, por lo que hay 15 diferentes archivos, uno por cada participante con sus diferentes actividades.

Estos datos fueron grabados con una frecuencia de 52 hz en tres direcciones x,y,z, por lo que cada segundo de registro del acelerómetro contiene 52 datos en tres dimensiones de la actividad que la persona está realizando.

Cada columna es un atributo de la medición que representa:

- Número serial o identificador
- Aceleración x
- Aceleración y
- Aceleración z
- Actividad

La actividad está codificada por un número del 1-7:

1. Trabajo en computadora
2. Pararse, caminar y subir/bajar escaleras
3. Mantenerse en pie
4. Caminar
5. Subir/bajar escaleras
6. Caminar y hablar con alguien
7. Hablar estando de pie

2.1. Ejercicios

1. Crea una red neuronal capaz de realizar la clasificación de la actividad. Recuerda dividir los datos en subconjuntos de entrenamiento y prueba.
2. Determina el número óptimo de capas y neuronas por capa para este problema. Reporta tus resultados
3. Prueba con diferentes funciones de activación: paso, sigmoide, tangente hiperbólica y relu. Reporta tus resultados
4. Elabora un reporte en el que se indique:
    - El preprocesamiento efectuado sobre los datos.
    - El número de elementos utilizados como conjunto de entrenamiento y prueba.
    - Matriz de confusión e interpretación.
    - Reporta el valor de las métricas precisión, exhaustividad (recall), exactitud (accuracy) y valor F1 (F1-Score) de este clasificador.
    - ¿Cómo utilizarías funciones de activación continuas (como la sigmoide) para calibrar el rendimiento de la red neuronal?

2.2. Software recomendado

Por la simplicidad, facilidad de uso, y compatibilidad se recomienda utilizar *python* como lenguage de programación y las siguientes bibliotecas:

- numpy
- pandas
- tensorflow + keras
- scikit-learn

3. Árboles de decisión

El archivo **creditcard.csv** contiene las transacciones realizadas con tarjeta de crédito durante dos días en noviembre de 2013. Algunas de estas transacciones fueron fraudes bancarios (Class=1). Por motivos de confidencialidad, cada registro contiene valores numéricos de las 28 componentes resultantes de un procesamiento por PCA (V1 ... V28). Además, se presenta el tiempo en segundos entre la transacción actual y la primer transacción (Time).

El conjunto de datos está altamente desbalanceado, contiene 492 fraudes (Class=1) y 284,315 transacciones legítimas (Class=0). Tu objetivo será construir un sistema de clasificación basado en árboles de decisión para detectar transacciones con alta probabilidad de fraude.

Los detalles de este conjunto de datos pueden ser encontrados en `https://www.kaggle.com/mlg-ulb/creditcardfraud`

3.1. Ejercicios

1. Describe la diferencia entre un árbol de decisión y un bosque aleatorio.
2. Describe qué es el *GINI Index* y cómo se utiliza en árboles de decisión.
3. Describe brevemente al menos dos técnicas para evitar sobre entrenamiento en árboles de decisión.
4. Separa el conjunto en datos de prueba (30 %) y datos de entrenamiento (70 %) utilizando el algoritmo de muestreo de tu preferencia.
5. Entrena un árbol de decisión con el conjunto de entrenamiento y reporta la precisión obtenida en ambos conjuntos.
6. Suponiendo que en el conjunto de entrenamiento E existen T datos totales, construye $N = T/F$ subconjuntos de entrenamiento balanceados $E_1, ..., E_N$, en dónde F es el número de fraudes en el conjunto de entrenamiento. Cada subconjunto E_i esta formado con los siguientes componentes:
    - Los F registros de fraudes en E

- F registros legítimos en E de forma que los registros legítimos de E_i sean distintos de los registros de E_j para todos i, j.

En términos simples, construimos diversos subconjuntos de entrenamiento completamente balanceados. Cada uno de ellos contiene todos los registros de fraudes y un subconjunto de los registros legítimos.

7. Entrena un árbol de decisión para cada subconjunto balanceado de entrenamiento

8. Para cada registro en el conjunto de prueba, todos los árboles asignarán una clasificación a la transacción, basados en el aprendizaje de su conjunto de prueba particular. De esta forma, cada árbol "votará"la clasificación del registro en cuestión. La clase asignada al registro será entonces la más votada por el conjunto de árboles.

9. Utiliza el sistema de árboles votantes para clasificar los datos en el conjunto de prueba. Reporta la precisión de este sistema en ambos conjuntos. Compara los resultados con el árbol entrenado con el conjunto desbalanceado de entrenamiento.

10. Suponiendo que los casos positivos son los fraudes, ¿cómo utilizarías una curva ROC para controlar el compromiso entre falsos positivos y verdaderos positivos?

11. ¿Qué es más importante en la calibración de esta curva ROC, maximizar la tasa de verdaderos positivos o minimizar la tasa de falsos positivos?

12. ¿Cómo manejarías un conjunto de datos desbalanceado con más de dos clases para utilizarlo en árboles de decisión?

3.2. Software recomendado

Por la simplicidad, facilidad de uso, y compatibilidad se recomienda utilizar *python* como lenguaje de programación y las siguientes bibliotecas:

- numpy
- pandas
- scikit-learn

4. Entregables

La tarea se debe entregar en un notebook de Jupyter con los resultados y las gráficas correctamente discutidos. Los archivos de datos deben estar en el mismo nivel de directorio del notebook para facilitar la ejecución de los programas.

5. Bibliografía recomendada

- Aurelien Geron - Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow-O'reilly (2019). Chapter 6 - Decision Trees, Chapter 10 - Introduction to Artificial Neural Networks with Keras
- Tom Mitchell - Machine Learning. Chapter 3 - Decision Tree Learning, Chapter 4 - Artifitial Neural Networks
- Christopher M. Bishop - Pattern Recognition and Machine Learning-Springer (2011) - Chapter 5 - Neural Networks