Tarea 4 EL4106 - Semestre Otoño 2021

Profesor: Javier Ruiz del Solar
Auxiliar: Patricio Loncomilla

Fecha enunciado: Jueves 13 de Mayo
Plazo entrega tarea: Miércoles 2 de Junio

El objetivo de esta tarea es implementar un clasificador de fallas de motores basado en análisis de señales de corriente, mediante redes neuronales artificiales. Se utilizará la base de datos Dataset for Sensorless Drive Diagnosis Data Set, que es una base de datos tomadas del UC Irvine Machine Learning Repository. Contiene 48 características extraídas de la señal de corriente que alimenta a un motor síncrono. Hay 11 clases. Se seleccionó un subconjunto de la base de datos original debido a su gran tamaño. Para mayor información revise la página del repositorio:
https://archive.ics.uci.edu/ml/datasets/Dataset+for+Sensorless+Drive+Diagnosis

La base de datos estará disponible en u-cursos.

Se usará pytorch para entrenar y validar la red neuronal que implementa el clasificador de fallas. Se analizará los efectos de cambiar el tamaño de la red (número de capas ocultas y de neuronas en estas capas) y la función de activación.

En esta tarea se entregará un código base, el cual debe ser usado para realizar las actividades pedidas en la tarea.

Se pide:

1) Ejecute el código base entregado e indique si se ejecutó adecuadamente. Este código carga los archivos “sensorless_tarea4_train.txt” y “sensorless_tarea4_test.txt” en dos *DataFrames*, los cuales son usados a continuación para generar dos objetos *Dataset*. Los conjuntos de entrenamiento, validación y prueba tienen *DataLoader*s, los cuales permiten acceder a los elementos de los *Dataset*s mediante *batches*. Además, los datos se normalizan. Finalmente, el código base permite entrenar una red neuronal, aunque se deberán implementar las funcionalidades que eviten el sobreajuste (*overfitting*) de la misma.

   Las redes a ser entrenadas tienen la siguiente estructura: capa de entrada de dimensionalidad 48 (correspondiente a los datos de entrada), capas ocultas (una o dos) y capa de salida con 11 neuronas y función de activación *softmax*. La función de *loss* (pérdida) es entropía cruzada. El optimizador que se debe usar es *Adam*. La función *softmax* está implícita al usar la función de pérdida *CrossEntropyLoss* de pytorch (no se debe agregar *softmax* a la salida de la red).

2) Entrene una red neuronal para cada uno de los siguientes casos:
   a) 10 neuronas en la capa oculta, usando función de activación ReLU y 1000 épocas.
   b) 40 neuronas en la capa oculta y función de activación ReLU, y 1000 épocas
   c) 10 neuronas en la capa oculta y función de activación Tanh, y 1000 épocas
   d) 40 neuronas en la capa oculta y función de activación Tanh, y 1000 épocas
   e) 2 capas ocultas con 10 y 10 neuronas cada una y función de activación ReLU, y 1000 épocas
   f) 2 capas ocultas con 40 y 40 neuronas cada una y función de activación ReLU, y 1000 épocas

Se debe evitar el sobreajuste (*overfitting*) de la red. Para esto se debe detener el entrenamiento cuando el *loss* de validación comience a aumentar, mientras el de entrenamiento siga bajando. Como las mediciones de *loss* pueden tener fluctuaciones (ruido), en lapsos cortos de tiempo, la implementación de esta funcionalidad debe realizarse en forma robusta. El código guarda *checkpoints* para poder recuperar las redes correspondientes a distintas épocas de entrenamiento.

Para cada red a ser entrenada:

i) Calcule el *loss* de entrenamiento y el de validación, así como el tiempo de procesamiento requerido. El entrenamiento se debe finalizar cuando el *loss* de validación comience a subir, para evitar sobreajuste de la red.
ii) Grafique el *loss* de entrenamiento y el de validación en función del tiempo. Ambas curvas deben mostrarse en un mismo gráfico, colocando las leyendas adecuadas.
iii) Genere la matriz de confusión normalizada y el *accuracy* normalizado, usando el conjunto de entrenamiento. Se debe mostrar la matriz de confusión con distintos colores o tonos de gris según el valor del *accuracy*. La matriz se debe calcular usando *scikit-learn*, usando la opción `normalize = "true"`.
iv) Genere la matriz de confusión normalizada y el *accuracy* normalizado, usando el conjunto de validación. Se debe mostrar la matriz de confusión con distintos colores según el valor del *accuracy*.

3) Usando la mejor red encontrada en validación (aquella con mayor *accuracy* en validación), calcule la matriz de confusión normalizada y el *accuracy* normalizado, **usando el conjunto de prueba**.

4) Análisis de los resultados obtenidos:
   a) Explique los efectos de variar la cantidad de neuronas en la capa oculta y como esto afecta el desempeño de la red.
   b) Explique los efectos de variar la cantidad de capas ocultas y como esto afecta el desempeño de la red.
   c) Explique el efecto de la función de activación y como esto afecta el desempeño de la red.
   d) Analice los siguientes puntos: tiempos de entrenamiento, matrices de confusión y *accuracies* de las arquitecturas probadas en validación.
   e) Analice la matriz de confusión y el *accuracy* en el conjunto de prueba, respecto a los obtenidos en el conjunto de validación.

Los informes y códigos deben ser subidos a u-cursos a más tardar a las 23:59 del día Miércoles 2 de Junio. Incluir un corto archivo de texto explicando cómo se utiliza su programa. Las tareas atrasadas serán penalizadas con un punto base más un punto de descuento adicional por cada día de atraso, incluyendo los fines de semana.

Importante:

- La evaluación de la tarea considerará el correcto funcionamiento del programa, la inclusión de los resultados de los pasos pedidos en el informe, la calidad de los experimentos realizados y de su análisis, la inclusión de las partes importantes del código en el informe, así como la forma, prolijidad y calidad del mismo. Se debe seguir la estructura indicada en material docente.
- Las partes importantes del código (sobre todo las modificadas/programadas) así como los resultados deben ser incluidas en el informe.
- Incluya el código completo como un anexo al final del informe.
- Cada uno de los puntos pedidos debe estar documentado en el informe
- El informe se debe entregar en formato PDF, y los códigos dentro del informe se deben agregar como texto (no como imágenes obtenidas usando un pantallazo). El formato básico de los informes se indica en el material docente de u-cursos.
- El código se debe entregar como un notebook con extensión .ipynb
- Se debe activar el uso de GPU en colaboratory para acelerar el proceso de entrenamiento.
- El informe se debe subir a la plataforma turnitin, en formato PDF. Tanto el contenido como los códigos del informe deben estar en formato de texto (no pantallazos).