UNLaM - Elementos de Inteligencia Artificial

Trabajo Práctico - ML y Redes Neuronales

Objetivo:

Experimentar con la aplicación de distintos algoritmos de aprendizaje de máquina.
Familiarizarse con al menos una biblioteca de código para redes neuronales.
Experimentar sobre el flujo completo de entrenamiento a testing y entender los efectos de los parámetros básicos que caracterizan a una red feedforward.

Descripción:

En este TP vamos a utilizar una aplicación de ML y una biblioteca gratuita de redes neuronales para entrenar y testear una red neuronal de tipo feedforward-backpropagation (perceptron) de 1 y más capas para la clasificación de 2 clases en datasets didácticos, y otra para reconocimiento de dígitos numéricos escritos a mano (ocr). Los datos de prueba pertenecen al set público MNIST, compuesto por imágenes de 28x28 píxeles en escala de grises.
Luego evaluaremos el efecto de variar los hiper-parámetros de la red, tanto en la performance como en la calidad del resultado y finalmente confeccionaremos una serie de árboles de decisión.

Preparación:

1. Asegurarse de tener instalado Java JDK7 o superior.
2. Descargar código inicial de la web de la cátedra.
3. Importar el proyecto Maven (archivo .pom) en Eclipse, Idea o IDE favorito.

Tareas:

Parte A: NN

1.

Leer:

https://en.wikipedia.org/wiki/Precision_and_recall

http://web.archive.org/web/20180724100313/http://deeplearning4j.org/mnist-for-beginners

y entender el código de inicio previamente descargado.

Leer TODA la consigna antes de empezar a resolver.

2.

a. Alterar el código en NNStarterCode.java para cargar el dataset, uniendo el archivo de negativos que es común a todos con el caso de positivos que les corresponde según su numero de orden de alumno de la materia en Miel. (pueden juntar los archivos a mano, no hace falta hacerlo por código). Alterar el código para colocar en las variables clave los valores correspondientes a los datos del dataset.

b. Correr la red en la clase NNStarterCode y registrar los 4 valores del resultado de la evaluación, la matriz de confusión y medir el tiempo de entrenamiento.

c. Correr la red con una capa oculta en las siguientes configuraciones: 2, 5, 20, 100 y finalmente 10000 neuronas, registrar resultados en cada caso.

d. Resolver:

i. Graficar el F1 en función de la cantidad de neuronas y el tiempo en función de la cantidad de neuronas.

ii. Por qué observamos estos patrones en los gráficos? (no explicar los gráficos, explicar el por qué)

iii. Graficar el dataset asignado en forma de puntos dispersos, utilizar el valor de los pesos y el bias para graficar sobre lo anterior la recta de cada neurona de la capa oculta cuando la red tiene 2 y 5 neuronas en dicha capa. (En libreoffice pueden usar gráficos con fondo transparentes superpuestos o agregar cada elemento como una serie aparte en el mismo gráfico. Usen lo que usen, que quede legible.)

3.

a. Correr la red en MNISTStarterCode.java y registrar resultados.

b. Agregar una segunda capa oculta manteniendo el total de neuronas entre TODAS las capas ocultas en 20, correr y registrar resultados.

c. Agregar una tercera capa oculta manteniendo el total de neuronas entre TODAS las capas ocultas en 20, correr y registrar resultados.

d. Resolver:

i. Graficar el F1 en función de la cantidad de capas ocultas y el tiempo en función de la cantidad de capas ocultas.

ii. Calcular el número de conexiones en cada caso y graficarlo en relación a la cantidad de capas ocultas.

iii. Explicar por qué el F1 reacciona de esa manera al modificar la red en este caso.

4.

a. Correr la red de la clase LenetMnistExample, esta es una red convolucional (CNN).

b. Resolver:

i. Identificar la cantidad de parámetros (pesos) en esta red y compararlos, junto con sus resultados a la red anterior (todas las variantes de las del punto 3).

5.

a. Buscar un dataset en internet que nadie ya haya tomado, indicando de donde se obtuvo y publicar en el foro de Miel el nombre del mismo y la url para que otro alumno no lo tome. Explicar brevemente de que se trata y que se va a predecir o clasificar.
A modo de ejemplo pero sin limitarse a los mismos, se puede buscar en:

- Kaggle:
  - https://www.kaggle.com/datasets
- Argentina Unida:
  - https://datos.gob.ar/dataset
- Data.World:
  - https://data.world/search?q=machine+learning+datasets
- Johns Hopkins COVID-19 Case Tracker:
  - https://data.world/associatedpress/johns-hopkins-coronavirus-case-tracker
- Wine Quality:
  - https://data.world/food/wine-quality

b. Hacer las modificaciones necesarias al código NNStarterCode.java para realizar los puntos 2.b, 2.c, 2.d.i y pero con su dataset buscado, utilizando todas las variables (columnas) que se puedan después de descartar las inapropiadas de acuerdo a los criterios vistos.
c. Realizar el punto 2.d.iii seleccionando 2 de las variables (columnas) de su dataset y corriendo la red nuevamente para los casos de 2 y 5 neuronas.

Consideraciones:

- El data set buscado por uds debe ser incluido en la entrega.
- En TODOS los casos donde haya una variable SEED o similar deben usar su numero de orden en la lista de miel (provista como csv en la carpeta del TP).
- TODOS los resultados informados deben ser reproducibles por los profesores utilizando lo entregado. Resultado que no pueda ser reproducido sera considerado invalido.
- Los resultados se esperan en un informe PROFESIONAL, no es necesario exagerar en formalismos ni detalles pero debe ser claro, conciso, completo y ordenado.
- “Tabular” los datos significa hacer una tablita o similar.

- Registrar significa que tomen nota, se incluyen en el informe cuando se pide tabularlos.

Parte B: ML

1. Abrir el archivo diabetes.csv y entender la estructura de los datos.
2. En NNTraining.java. editar la ruta y nombre de archivo donde se guardará la red, modificarla utilizando la cantidad de capas ocultas y neuronas deseadas (si lo dejan como esta les repruebo el TP -.-) y correr el código. Documentar los valores de F1.
3. En REPTreeTraining.java

a. Alterar el archivo para filtrar (no utilizar) entre un tercio y la mitad de las columnas del dataset de diabetes, elegidas al azar. (Documentandolo)

b. Repetir pasos a 4 veces alterando las columnas filtradas y los nombres de archivo.

c. Resolver:

    i. Para cada árbol entrenado, calcular con el output (o directamente alterando el código java si prefieren) la precisión, el recall y el F1.

    ii. Tabular los datos en el informe, incluyendo los de la red.