Práctica 3 sobre Redes Neuronales y Deep Learning

Parte A: Perceptrón multicapa

En esta parte de la práctica vamos a aprender a usar redes neuronales con Scikit-learn.

Instalación

Se necesita la Scikit-learn que ya se instaló en las anteriores prácticas

https://scikit-learn.org/

Aprendizaje

Mira con detalle el apartado de clasificación, busca el apartado de Neural network models supervised. En concreto, céntrate en cómo se puede aplicar la red neuronal Multi-layer Perceptron (MLP):

https://scikit-learn.org/stable/modules/neural_networks_supervised.html#multi-layer-perceptron

Usa la clase MLPClassifier, y observa el siguiente ejemplo sobre entrenamiento:

```
>>> from sklearn.neural_network import MLPClassifier
>>> X = [[0., 0.], [1., 1.]]
>>> y = [0, 1]
>>> clf = MLPClassifier(solver='lbfgs', alpha=1e-5,
...                     hidden_layer_sizes=(5, 2), random_state=1)
...
>>> clf.fit(X, y)
MLPClassifier(alpha=1e-05, hidden_layer_sizes=(5, 2), random_state=1,
              solver='lbfgs')
```

En general, observa todos los atributos y operaciones que tiene dicha clase. En concreto aprende el significado de atributos tales como “hidden_layer_sizes”, “activation”, “solver” y “learning_rate”.

Observa el siguiente ejemplo sobre predicción de clases con MLP previamente entrenada.

```
>>> clf.predict([[2., 2.], [-1., -2.]])

array([1, 0])
```

Práctica

Esta práctica usa el mismo contexto que la práctica anterior sobre presión arterial y el colesterol.

Hay que generar unos datos sobre conjuntos de puntos con valores de presión arterial y colesterol, clasificando entre tener riesgo cardiovascular (1) o no (0), como hicimos en la práctica anterior.

1. Genera una gran cantidad de puntos aleatorios (presión arterial y colesterol) y muestra los resultados en una gráfica, utilizando diferentes dibujos para cada punto.
2. Divide el conjunto de datos en entrenamiento y evaluación como consideres.
3. Entrena una red neuronal con el conjunto de entrenamiento.

4. Aplica el modelo entrenado sobre el conjunto de evaluación
5. Muestra los resultados aplicados sobre el conjunto de evaluación. Se puede mostrar una gráfica como la anterior: puntos (presión arterial y colesterol) y su clasificación con MLP pintado de diversa manera (e.g. “x” o círculos).
6. Usa las métricas precisión y recall estudiadas en clase para evaluar la clasificación.
7. Para evitar “overfitting” habría que entrenar y validar los datos con conjuntos diferentes y con configuraciones de la red distintas. Prueba a mejorar la precisión de la red neuronal cambiando algunos de sus parámetros que has aprendido y haciendo distintas particiones del dataset. Se deben probar al menos dos configuraciones distintas.
8. Calcula la precisión para cada una de las configuraciones realizadas en el punto anterior. Dibuja también sus gráficas.
9. Discute los resultados y determina cuál es la mejor configuración de todas las estudiadas.

Parte B: Redes Neuronales Convolucionales

Instalación

En esta parte de la práctica vamos a usar la librería PyTorch. Aquí está disponible la documentación de la librería:

https://pytorch.org/docs/stable/index.html

Para instalar pytorch, podéis intentar los siguientes comandos de instalación en la consola de anaconda:

- *pip install torch torchvision matplotlib*
- *conda install pytorch torchvision torchaudio cpuonly -c pytorch*

Si ninguna opción hace que torch se importe, probad a ejecutar el primer comando en la primera celda del notebook.

Aprendizaje

Sigue el tutorial de PyTorch para entrenar una red CNN sobre el dataset de imágenes CIFAR10 (imágenes a color RGB y de 32x32 píxeles), donde encontrarás imágenes de 10 clases distintas. En este tutorial, veremos cómo se carga un dataset con Pytorch, cómo se crea una red CNN, definiendo la función de pérdida y el optimizador, cómo se entrena la red y se evalúa el modelo calculando el accuracy total y por clases. Revisa cada línea y entiende su significado.

https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

Responde a las siguientes preguntas (podéis incluir la respuestas en markdown en el notebook):

1. ¿Qué significa que el batch_size sea 4? Explica cómo afectaría al cálculo de los pesos en cada epoch.
2. ¿Por qué al dividir el conjunto de entrenamiento el parámetro shuffle está a True, mientras que en el conjunto de test está a False?
3. Revisa la constructora de la clase Net:

   a. En el constructor, describe cada línea y explica la configuración de cada capa creada. Importante: hay que describir cada parámetro de cada línea. Ten en cuenta que hay algunos parámetros que se pueden personalizar al gusto del programador, pero hay otros que dependen del tamaño de las imágenes (32x32) y del número de clases que tenemos.
   b. En la función forward, explica cada línea de código describiendo el proceso de forward propagation.
4. Indica la línea de código donde se realiza el cálculo de los gradientes para determinar cómo modificar los pesos (los filtros kernel), e indica la línea de código donde se recalculan los pesos. ¿Qué se ha tenido que realizar antes para poder hacer el cálculo de gradientes y la actualización de pesos?
5. ¿Cuál es la estructura de los outputs que se obtienen en outputs = net(images)?

Práctica

En esta parte vamos a trabajar con el dataset MNIST es un conjunto de datos en el que hay imágenes de números escritos a mano (del 0 al 9). Las dimensiones de las imágenes son de 28x28 píxeles y las imágenes están en escala de grises (1 canal). Hay diez posibles clases (los dígitos del 0 al 9).

Entrena y evalúa una red CNN con dos configuraciones distintas para el dataset MNIST, siguiendo el ejemplo visto en el tutorial. Las dos configuraciones tienen que tener:

- Un batch_size distinto
- Distinto número de capas convolucionales
- Distinto número de capas conectadas
- Distinto tamaño de kernel
- Distinto número de filtros (kernels)
- Distintos tamaños para reducir las matrices en la capa de pooling
- Distintos números de neuronas en las capas ocultas de las capas conectadas
- Una tasa de aprendizaje distinta
- Distinto número de epochs

Consejo: primero indica explícitamente (en markdown) estos hiperpárametros elegidos para las dos configuraciones. Después, haz el código. Explica el código en las partes donde hagas estas configuraciones.

Calcula el accuracy total y por clases para cada una de las configuraciones. Puedes dibujar una gráfica de barras para ayudarte a comparar los resultados. Explica las diferencias en las accuracies calculadas para ambas configuraciones y discute las razones de dichos resultados y posibles mejoras si es necesario.

Normas de Entrega

La práctica debe entregarse utilizando el mecanismo de entregas del campus virtual. Se entregarán dos notebooks (uno por cada parte) con el código en Python explicando el análisis de resultados y el código realizado. Si las explicaciones del código realizado y el análisis de los resultados obtenidos, la práctica será no apta.