REDES NEURONALES 2023

Trabajo Final Integrador

Autoencoder y clasificador convolucional sobre Fashion-MNIST

Nota:

- Entreguen el trabajo final integrador **sólo** en formato `.pdf`. Si desean pueden enviar la notebook, pero por separado.
- El `.pdf` no puede tener más de cuatro (4) páginas.

PARTE 1) Red neuronal autoencoder convolucional de varias capas

1.1) Defina y cree una red neuronal autoenconder convolucional. El **encoder** tienen que tener las siguientes capas.

Primero una capa convolucional 2D compuesta por:

- Una capa `Conv2d` de entrada de dimensiones $(1, 28, 28)$ (i.e. 1 canal de imagenes de dimensiones $(28, 28)$) y de salida de dimensiones $(16, 26, 26)$ (i.e. 16 canales de imagenes de dimensiones $(26, 26)$). Para ello, utilizar un kernel de dimensiones $(3, 3)$ y el resto de los parámetros con los valores por defecto.
- Una capa `ReLU`.
- Una capa `Dropout`.
- Una capa `MaxPool` con un kernel de dimensiones $(2, 2)$, lo cual mapea dimensiones $(16, 26, 26)$ a dimensiones $(16, 13, 13)$.

Luego otra capa convolucional 2D compuesta por:

- Una capa 'Conv2d' de entrada de dimensiones $(16, 13, 13)$ y de salida de dimensiones $(32, 11, 11)$. Para ello, utilizar un kernel de dimensiones $(3, 3)$ y el resto de los parámetros con los valores por defecto.
- Una capa `ReLU`.
- Una capa `Dropout`.
- Una capa `MaxPool` con un kernel de dimensiones $(2, 2)$, lo cual mapea dimensiones $(32, 11, 11)$ a dimensiones $(32, 5, 5)$.

Una capa lineal compuesta por:

- Una capa `Flatten` que mapea dimensiones $(32, 5, 5)$ a dimensión $(32 * 5 * 5,)$.
- Una capa `Linear` que mapea dimensión $(32 * 5 * 5)$ a dimensión n.
- Una capa `ReLU`.
- Una capa `Dropout`.

El **decoder** tiene que tener las siguientes capas:

Una capa lineal compuesta por:

- Una capa `Linear` que mapea dimensión $(n,)$ a dimensión $(32 * 5 * 5,)$.
- Una capa `ReLU`.
- Una capa `Dropout`.
- Una capa `Unflatten` que mapea dimensión $(32 * 5 * 5,)$ a dimensiones $(32, 5, 5)$.

Una capa convolucional 2D transpuesta (de la segunda convolucional) compuesta por:

- Una capa `ConvTranspose2d` que mapea dimensiones $(32, 5, 5)$ a dimensiones $(16, 13, 13)$. Para ello utilizar un `kernel_size` de $(4, 4)$, un `stride` de $(2, 2)$ y un `output_padding` de $(1, 1)$.
- Una capa `ReLU`.
- Una capa `Dropout`.

Luego otra capa convolucional 2D transpuesta (de la primera convolucional) compuesta por:

- Otra capa `ConvTranspose2d` que mapea dimensiones $(16, 13, 13)$ a dimensiones $(1, 28, 28)$. Para ello utilizar un `kernel_size` de $(3, 3)$, un `stride` de $(2, 2)$ y un `output_padding` de $(1, 1)$.
- Una capa `Sigmoid`.
- Una capa `Dropout`.

1.2) Grafique, comparativamente, las imagenes a predecir vs las imagenes predichas por el modelo sin entrenar para $n = 64$ y dropout $p = 0{,}2$.

PARTE 2) Entrenando el autoencoder

2.1) Implemente, en una función, un loop de entrenamiento que recorra los **batchs** (lotes).

2.2) Implemente, en una función, un loop de validación (o prueba) que recorra los batchs.

2.3) Inicialize dos `DataLoader`s llamados `train_loader` y `valid_loader` que estén definidos sobre el `train_set` (conjunto de entranmiento) y el `valid_set` (conjunto de validación) de **Fashion-MNIST**, respectivamente, y que usen **batchs** de 100 ejemplos.

2.4) Cree una función de pérdida usando el **Error Cuadrático Medio (ECM)**.

2.5) Cree un optimizador con un learning rate igual a 10^{-3}. Pruebe con **ADAM**.

2.6) Cree una instancia del modelo con $n = 64$ neuronas en la capa intermedia y dropout $p = 0{,}2$.

2.7) Especifique en que dispositivo (`device`) va a trabajar: en una **CPU** o en una **GPU**.

2.8) Implemente un loop de entrenamiento y validación que trabaje con el `train_loader` y el `valid_loader` respectivamente, usando un número arbitrario de épocas. Este loop debe guardar en dos listas los valores de los promedios del **ECM** sobre el conjunto de entrenamiento y el de validación, respectivamente.

IMPORTANTE: No olvide copiar los batchs al dispositivo de trabajo.

2.9) Entrene y valide el modelo.

2.10) Use las listas del inciso anterior para graficar en función de las épocas el **ECM** de **entrenamiento** y **validación**. Discuta y comente, cual es el número óptimo de épocas de entrenamiento?

2.11) Grafique, comparativamente, algunas de las imagenes a predecir vs las imagenes predichas por el modelo entrenado.

2.12) Repita para otros valores de n, el optimizador **GPU**, otros valores de **dropout**, otros valores de **learning-rate**, otros tamaños de **batchs** y cambiando las **Sigmoid**s por **ReLU**s en la capa de salida. Que valores de estos hiperparámetros considera los más convenientes? Porqué?

PARTE 3) Definiendo y entrenando un clasificador convolucional reutilizando el encoder

3.1) Defina y cree un clasificador convolucional reutilizando el encoder del autoencoder convolucional entrenado anteriormente. Más precisamente, el clasificador convolucional tiene que tener las siguientes capas:

Primero, el **encoder** del **autoencoder** entrenado anteriormente. Este mapea una entrada de dimensiones $(1, 28, 28)$ a una salida de dimensión n.

Luego una capa lineal de clasificación compuesta:

- Una capa `Linear` que mapea dimensión $(n,)$ a dimensión $(10,)$.
- Una capa `ReLU`.
- Una capa `Dropout`.

3.2) Reimplemente las funciones con los loop de entrenamiento y validación adaptados para el problema de clasificación (i.e. hay que incorporarles el cálculo de la precisión).

3.3) Cree una función de pérdida usando la **Cross Entropy Loss (CEL)**.

3.4) Cree un optimizador con un learning rate igual a 10^{-3}. Pruebe con **ADAM**.

3.5) Cree una instancia del modelo con $n = 64$ neuronas en la capa intermedia y dropout $p = 0{,}2$.

3.6) Especifique en que dispositivo (`device`) va a trabajar: en una **CPU** o en una **GPU**.

3.7) Implemente un loop que itere sobre épocas de entrenamiento y validación, y que correspondientemente guarde en cuatro listas los valores de los promedios de la **CEL** y la **Precisión** en función de las épocas para cada conjunto.

3.8) Entrene y valide el modelo.

3.9) Use las listas del inciso **3.7)** para graficar en función de las épocas la **CEL** y la **Precisión** de **entrenamiento** y **validación**, respectivamente. Discuta y comente, cual es el número óptimo de épocas de entrenamiento?

3.10) Utilice el conjunto de validación para calcular una **Matriz de confusión**. Grafíquela y comente los resultados.

3.11) Repita para otros valores de n, el optimizador **SGD**, otros valores de **dropout**, otros valores de **learning-rate**, otros tamaños de **batchs** y cambiando las **Sigmoid**s por **ReLU**s en la capa de salida. Pruebe, además, modificando el optimizador para que sólo reentrene los parámetros de la capa clasificadora, dejando los parámetros de la capa codificadora tal como los deja el autoencoder convolucional. Que valores/opciones de estos hiperparámetros considera los más convenientes? Porqué?