Taller 2 aprendizaje de máquina.

Integrantes: Luis Frontuso, Miguel Zúñiga.

Este documento aborda un problema de clasificación de imágenes que contienen números escritos a mano. Estas imágenes corresponden a 10 clases: 0, 1, 2, 3, 4, 5, 6, 7, 8 y 9. El conjunto de datos consta de 70,000 observaciones, divididas en un conjunto de entrenamiento (60,000 85%) y un conjunto de prueba (10,000 15%). Las clases están balanceadas en ambos conjuntos, con una distribución cercana al 10% para cada una.

Por último, con el fin de mejorar las métricas de precisión del modelo —sensibilidad, especificidad, precisión, y F1-score— se llevan a cabo una serie de experimentos sobre la arquitectura de la red, incluyendo la cantidad de capas, y el número de neuronas en cada capa, cambiando dichos componentes de forma aleatoria. De esta forma se proponen los siguientes experimentos:

	Input layer	**Hidden Layer 01**		**Hidden Layer 02**		**Hidden Layer 03**		**Output layer**	
		Units	Act. Func.	Units	Act. Func.	Units	Act. Func.	Units	Act. Func.
Model_00	784	90	ReLU	100	ReLU		ReLU	10	softmax
Model_01	784	97	ReLU	52	ReLU	47	ReLU	10	softmax
Model_02	784	95	ReLU	50	ReLU	43	ReLU	10	softmax
Model_03	784	93	ReLU	54	ReLU	38	ReLU	10	softmax
Model_04	784	92	ReLU	52	ReLU	36	ReLU	10	softmax

En los que se obtuvieron los siguientes resultados con el training stop por defecto de la librería. Para el cálculo de las métricas generales: sensibility, specificity, F1 se hizo mediante el promedio de métrica para cada clase. Además el drift se calculó como la diferencia entre la métrica train menos la métrica en test.

Los experimentos obtuvieron los siguientes resultados:

	Validation				**Drift**			
	sensibility	specificity	accuracy	F1	sensibility	specificity	accuracy	F1
Model_00	61.7679%	95.8478%	63.0350%	63.3159%	0.5309%	0.0499%	0.0000%	0.8461%
Model_01	90.1553%	98.9182%	90.2600%	90.1604%	8.1490%	0.8951%	8.0583%	9.6529%
Model_02	88.2690%	98.7108%	88.3900%	88.2482%	3.6915%	0.4094%	3.6883%	10.8720%
Model_03	26.3948%	91.9755%	27.6200%	20.9052%	0.1883%	0.0232%	0.2667%	0.3455%
Model_04	51.2620%	94.6824%	52.0400%	52.9558%	0.0692%	0.0171%	0.1917%	0.2394%

Con base en los resultados, el modelo Model_02 presenta un mejor desempeño general en las métricas de precisión. Además, la diferencia en estos indicadores entre los conjuntos de entrenamiento y prueba es inferior al 4 %, excepto en la métrica F1, donde la diferencia del 10 % sugiere un posible sobreajuste. Para mejorar Model_02, se podrían aplicar técnicas de regularización. Asimismo, estas técnicas también podrían implementarse en Model_01 para reducir el sobreajuste y mejorar su desempeño.

Cambios en la librería

Según lo indicado en el enunciado del taller, es necesario realizar algunos ajustes en la librería para que funcione correctamente. Por esto, se llevaron a cabo los siguientes cambios:

1. **ActivationFunctions.py:**
    a. En la función de activación softmax se encontró que se indeterminaba al calcular $e^{Z_{ij}}$ cuando el valor de Z_{ij} es relativamente grande, es decir, tiende a infinito y la representación de infinito no existe. Para ajustarlo se encontró la solución en un artículo de Medium que propone:

$$\text{softmax}(\boldsymbol{Z}) = \frac{e^{\boldsymbol{Z}-max(\boldsymbol{Z})}}{\sum_{k=1}^{n_L} e^{Z_k-max(\boldsymbol{Z})}}$$

    b. También se incluyen el parámetro keepdims=True en la función sum() para que las dimensiones sean consistentes y las operaciones estén definidas, esto hace que dimensión del vector que retorna dicha función sea (m,1) en lugar de (m,).
2. **FeedForward.py:**
    a. La función cost_gradient y cost está mal definida debido a que la definición de la formula de costo que implementa es binary cross-entropy, por este motivo se implementa adicional a esta se implementa multi-class cross entropy:

$$E(\{\boldsymbol{w_i}, \boldsymbol{w_{i0}}\}_{\boldsymbol{i}}|X) = -\sum_t \sum_i r_i^t log(y_i^t + \varepsilon)$$

3. **Adam.py:**
    a. La función _fit() tiene una inconsistencia al definir mt y vt pues en el proceso de actualización de los parámetros y los sesgos se genera un matriz de (n,n) y debería generar un vector de (1,n). Para solucionarlo en la definición de mt y vt se definen como las transpuestas de las definiciones anteriores.

   b. También se cambió la multiplicación $G \cdot G$ por $G \cdot G^T$ en la definición del stop.

4. **Helpers.py**
   a. También se encontró que la matriz de confusión y las métricas de precisión sensibilidad, especificidad, precisión, y F1 no se calculaban bien porque la función estaba pensada para clasificación binaria.