Tarea 2 de programación: Red Neuronal

1. Objetivo

Implementar en Python funciones complementarias a una red neuronal dada, para reconocer superficies terrestres. El uso de la red neuronal proporcionada es de carácter obligatorio.

2. Envío de la tarea

La tarea es de carácter individual y no se deben compartir entre los alumnos los códigos, sólo pueden establecer conversaciones respecto a los conceptos teóricos y prácticos asociados a la temática a ser desarrollada en la tarea.

Para desarrollar la tarea los alumnos tendrán un plazo de dos semanas a contar de la fecha de entrega, y debe enviarse antes de ingresar a la clase de IA. Los alumnos deberán hacer una breve presentación de resultados en la clase.

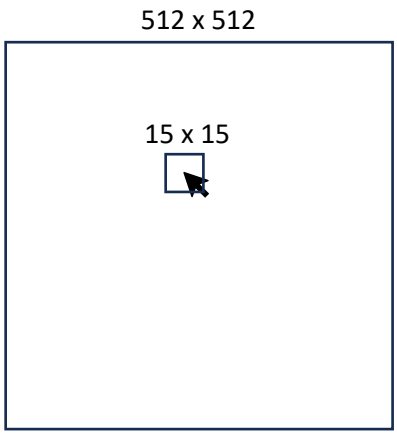
La tarea deberá ser enviada en un archivo comprimido con la siguiente estructura: `Tarea2_Nombre_Apellido.rar` y al interior del archivo comprimido, deben estar todos los códigos implementados y base de datos, señalando al código principal con el nombre de `main.py` o `main.ipynb` u otro compatible con Python.

Al usuario del código (profesor) no se le debe pedir que ingrese manualmente ningún dato de entrenamiento, solo en la prueba del modelo, se le debe solicitar el ingreso de un dato, para que el algoritmo haga la estimación. Se evaluará positivamente la usabilidad del código.

3. Generación de Base de Datos de Superficies Terrestres

Diseñar una interfaz gráfica de usuario, GUI (del inglés Graphical User Interface) en lenguaje de programación Python, para generar una base de datos de superficies terrestres. Su GUI deberá permitir abrir imágenes descargadas de Google Earth y usando el cursor del mouse realizar recortes de imágenes de tamaño 15 × 15 pixeles y almacenarlas en carpetas.

Las imágenes deberán extraerse de Google Earth a una altura entre 30 y 35km sobre el territorio chileno (incluyendo tierra y mar), y deberán ser convertidas a escala de grises antes de realizar cualquier tipo de procesamiento (se recomienda usar alguna función de Python, como por ejemplo `cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)`). Se deberán procesar al menos 10 imágenes de 512 × 512 píxeles por cada clase; 1) agua (mares, lagos, océanos), 2) vegetación (selva, bosques), 3) montañas (cordillera, cerros), 4) desiertos, 5) ríos y 6) ciudades. De cada una de estas imágenes deberá obtener 50 imágenes de 15 × 15 píxeles. Se recomienda descargar 12 imágenes de 512 × 512 píxeles: 10 para obtener las imágenes de prueba de 15 × 15 píxeles y 2 para ser usadas en la GUI de aplicación.



Nombre de carpetas: C1, C2, C3, C4, C5, C6

Nombre de imágenes de tamaño 15 × 15 píxeles: Ic_m ; c = 1, 2, 3, 4 , 5, 6

m = 001, 002, …, 500

Finalmente, en cada carpeta deberá haber 500 imágenes de tamaño 15 × 15 píxeles

Deberá crear la misma estructura de carpetas con las mismas imágenes, pero con aplicación de la Transformada Rápida de Fourier (FFT), tal como se indica: *imagen* ← *abs*(*fft*(Muestra)).

3. Archivos

Antes de comenzar la implementación:

Debe descargar, estudiar su funcionamiento y ejecutar el algoritmo de red neuronal base, que será utilizado (de carácter obligatorio):

`Neural Network_XOR.ipynb`

Además, debe descargar las imágenes asociadas al archivo:

`NN_XOR_Image1.png`

`Sigmoidal.png`

4. Desarrollo de la Tarea

Para el desarrollo de la tarea deberán realizar las siguientes etapas:

4.1 Cargar archivo de datos

Desde Python deberás cargar de forma vectorial todas las imágenes (con FFT) en una variable X y sus respectivas etiquetas y.

Cada ejemplo de entrenamiento es una imagen en escala de grises de 15 × 15 píxeles. Cada píxel es representado por un número de punto flotante que indica la intensidad de la escala de grises en esa ubicación. La cuadrícula de 15 × 15 píxeles se "desenrolla" en un vector de dimensión 225. Cada uno de estos ejemplos de entrenamiento se convierte en una sola fila en nuestra matriz de datos X. Esto nos da una matriz X de dimensión 3000 × 225, donde cada fila es un ejemplo entrenamiento de alguna superficie terrestre.

La segunda parte del conjunto de entrenamiento es un vector y de dimensión 3000 que contiene etiquetas para el conjunto de entrenamiento, se identificarán las clases como sigue: *i*) agua (mares, lagos, océanos) con "1", *ii*) vegetación (selva, bosques) con "2", *iii*) montañas (cordillera, cerros) con "3", *iv*) desiertos con "4", *v*) ríos con "5" y *vi*) ciudades con "6".

$$X = \begin{bmatrix} -- \left(x^{(1)}\right)^{\top} -- \\ -- \left(x^{(2)}\right)^{\top} -- \\ \vdots \\ -- \left(x^{(m)}\right)^{\top} -- \end{bmatrix} \qquad y = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \cdots, \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$$

4.2 Conjunto de entrenamiento y prueba

Para el entrenamiento y prueba de la red neuronal, usted deberá aleatorizar X en las filas, y guardar el índice de aleatorización, y usando ese mismo índice deberá re-ordenanar las etiquetas y.

El **conjunto de entrenamiento** corresponderá al 80% de los datos aleatorizados X con sus respectivas etiquetas y. El **conjunto de prueba** será el 20% de datos restantes aleatorizados de X, con sus respectivas etiquetas y, aunque debe tener presente, que para probar la red neuronal entrenada no se requiere de la etiqueta y, la cual solo es útil para evaluar el desempeño de la red neuronal.

4.3 Visualización de datos

Deberá implementar un código que le permita visualizar un subconjunto del conjunto de X. Esta función asigna cada fila a una imagen en escala de grises de 15 × 15 píxeles y muestra las imágenes juntas, debiendo mostrar una imagen similar a la Figura 1.

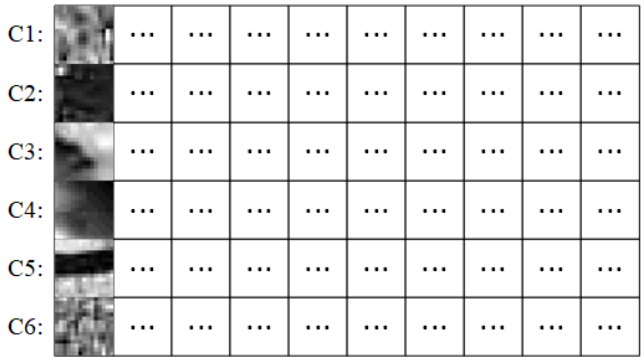


Figura 1: Ejemplos del conjunto de datos.

4.4 Configuración de la red neuronal

La red neuronal que usted deberá definir tendrá la siguiente estructura:

- Una capa de entrada de 225 unidades (imagen de texturas terrestres de 15 × 15 pixeles vectorizada),
- Una capa oculta de 25 unidades, y
- Una capa de salida de 6 unidades (6 etiquetas, desde el 1 al 6).

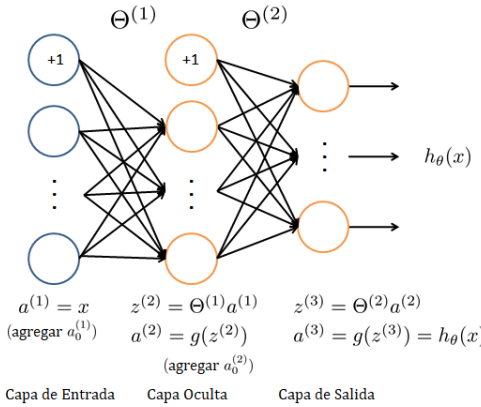


Figura 2: Modelo de red neuronal.

Notar que $\Theta^{(1)}$ será de tamaño 25 × 226 y $\Theta^{(2)}$ será de tamaño 6 × 26.

4.5 Función de costo con y sin regularización

Deberá probar su red neuronal con y sin regularización, colocando una opción que permita decidir si la función de costo contiene -o no- la regularización.

La función de costo para redes neuronales con regularización viene dada por:

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(h_\Theta(x^{(i)})\right)_k + \left(1 - y_k^{(i)}\right)\log\left(1 - h_\Theta(x^{(i)})\right)_k\right] + \frac{\lambda}{2m}\left[\sum_{i=1}^{400}\sum_{j=1}^{25}\left(\Theta_{ji}^{(1)}\right)^2 + \sum_{i=1}^{25}\sum_{j=1}^{10}\left(\Theta_{ji}^{(2)}\right)^2\right]$$

donde K es el número de unidades de salida.

Puede asumir que la red neuronal sólo tendrá 3 capas: una capa de entrada, una capa oculta y una capa de salida. Sin embargo, **su código debería contemplar cualquier número de capas ocultas y funcionar para cualquier número de unidades de entrada, unidades ocultas y unidades de salida**. En esta tarea, y sólo a modo de ejemplo, se han enumerado explícitamente los índices de la función de costo para $\Theta^{(1)}$ y $\Theta^{(2)}$ para mayor claridad, tenga en cuenta que su código debería funcionar en general con $\Theta^{(1)}$, $\Theta^{(2)}$, …, $\Theta^{(L-1)}$ de cualquier tamaño. De forma genérica se puede escribir la función de costo sigue:

$$J(\Theta) = -\frac{1}{m}\left[\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log\left(h_\Theta(x^{(i)})\right)_k + \left(1 - y_k^{(i)}\right)\log\left(1 - h_\Theta(x^{(i)})\right)_k\right] + \frac{\lambda}{2m}\sum_{\ell=1}^{L-1}\sum_{i=1}^{s_\ell}\sum_{j=1}^{s_{\ell+1}}\left(\Theta_{ji}^{(\ell)}\right)^2$$

donde, λ es el parámetro de regularización, L es el número de capas en la red, s_ℓ es el número de unidades en la capa ℓ (no se contabiliza la unidad de sesgo. Tenga en cuenta que no debe regularizar los términos que corresponden al sesgo. Para las matrices $\Theta^{(1)}$ y $\Theta^{(2)}$, esto corresponde a la primera columna de cada matriz (o primera fila según el ordenamiento que usted haya definido). Ahora debe agregar regularización a su función de costo. Observe que primero puede calcular la función de costo no regularizada $J(\Theta)$ y luego agregar a la función de costo el término de regularización. Use el parámetro de regularización $\lambda = 1$, y pruebe con un par de valores más.

4.6 Gradiente sigmoidal

Para ayudarlo a comenzar con esta parte de la tarea, ya está implementada la función de gradiente sigmoidal. El gradiente de la función sigmoidal puede ser calculada como:

$$g'(z) = \frac{d}{dz}g(z) = g(z)(1 - g(z))$$

donde

$$\text{sigmoide}(z) = g(z) = \frac{1}{1 + e^{-z}}$$

Para valores grandes (tanto positivos como negativos) de z, el gradiente debe estar cerca de 0. Cuando $z = 0$, el gradiente debe ser exactamente 0.25. El código también debería funcionar con vectores y matrices. Para matriz, su función debe realizar la función de gradiente sigmoide en cada elemento.

4.7 Inicialización aleatoria

Al entrenar las redes neuronales, es importante inicializar aleatoriamente los parámetros Θ para romper la simetría. Una estrategia eficaz para la inicialización aleatoria es seleccionar al azar valores para $\Theta^{(\ell)}$ uniformemente en el rango $[-\epsilon, +\epsilon]$. Debe usar $\epsilon = 0.12$[1]. Este rango de valores asegura que los parámetros se mantengan pequeños y hacen que el aprendizaje sea más eficiente.

---

[1] Una estrategia efectiva para elegir ϵ se basa en el número de unidades de la red. Una buena elección de ϵ es $\epsilon = \frac{\sqrt{6}}{\sqrt{L_{in} + L_{out}}}$ donde $L_{in} = s_\ell$ y $L_{out} = s_{\ell+1}$ son el número de unidades en las capas adyacentes a $\Theta^{(\ell)}$.

4.8 Uso de la red neuronal entrenada para hacer predicciones

Una vez entrenada la red neuronal, deberá almacenar en un archivo los parámetros $\Theta^{(\ell)}$, considerando que la estructura de la red puede ser definida a partir del número de matrices $\Theta^{(\ell)}$, el índice ℓ y el tamaño de dichas matrices.

Ahora **debe implementar una función que realice la propagación hacia adelante para la red neuronal**. Se recomienda re-utilizar parte del código entregado (la parte que realiza la propagación hacia adelante). La función que debe implementar se debe llamar: `p = predict(Theta1, Theta2,…, X)`, considere implementar una función `predict` genérica, es decir, que permita $L-1$ matrices `Theta`. Aquí X corresponde al conjunto de prueba (el 20% de X e y). En concreto, debe implementar la propagación hacia adelante que calcule $h_\Theta(x^{(i)})$ para cada ejemplo i y devuelva las predicciones asociadas. la predicción de la red neuronal será la etiqueta que tenga la salida más grande $(h_\Theta(x))_k$.

4.9 Desempeño de la red neuronal

Para determinar el desempeño de la red neuronal deberá implementar una función llamada `[R, P, Fm] = performance(p, y)`, que determine el Recall (`R`), la Precisión (`P`) y el F-measure (`Fm`). En esta función `p` corresponde a la predicción e `y` corresponde a la etiqueta asociada al conjunto X de prueba. Los valores de desempeños se calculan de la siguiente forma:

$$\text{Recall} = \frac{TP}{nP}$$

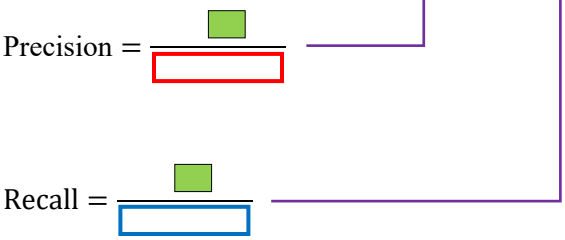
$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{F-measure} = \frac{2 \cdot \text{Recall} \cdot \text{Precision}}{\text{Recall} + \text{Precision}}$$

donde, TP es el número de verdaderos positivos, FP es el número de falsos positivos, y nP es el número total de positivos en el conjunto de prueba.

Deberá construir la matriz de confusión de la siguiente manera:

Clasificado Como / Es de la Clase	1	2	3	4	5	6	Precision	Recall	F-measure
1									
2									
3									
4									
5									
6									
Promedio Total:									



4.10 Estimación in situ

Deberá construir una GUI de aplicación que permita abrir imágenes de 512 × 512 píxeles o de cualquier tamaño (alguna de las dos imágenes por clase que fueron reservadas para este propósito, de la etapa de generación de la Base de Datos) de tal forma de poder realizar una estimación *in situ* de la superficie donde se haga "clic" con el mouse, tal como se muestra en la Figura 2.

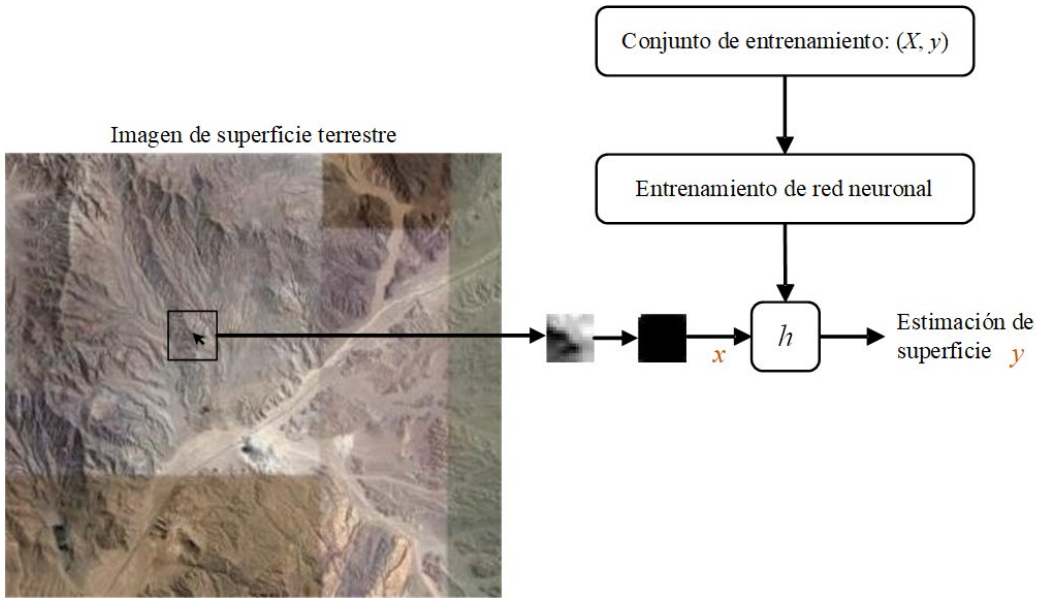


Figura 2: Entrenamiento de una red neuronal y uso para la estimación de superficie.