Clasificación binaria de rostros humanos

Alex Rodriguez 1, Bryan Astuyauri 2, Luis La Rosa 3, Samira Cueva 4, Leopoldo Paredes 5

Facultad de Ciencias 1, Universidad Nacional de Ingeniería 1, email: arodriguezv@uni.pe
Facultad de Ciencias 2, Universidad Nacional de Ingeniería 2, e-mail: bryan.astuyauri.r@uni.pe
Facultad de Ciencias 3, Universidad Nacional de Ingeniería 3, e-mail: luis.larosa.c@uni.pe
Facultad de Ciencias 4, Universidad Nacional de Ingeniería 4, e-mail: samira.cueva.b@uni.pe

Resúmen

Este trabajo brinda una completa explicación de la clasificación binaria usando redes neuronales

Palabras claves: Funciones reales, Vectores, Optimización, Redes neuronales

Abstract

This work gives complete explanations of binary classification using deep neural networks

Keywords Real valued functions, Vectors, Optimization, Neural networks

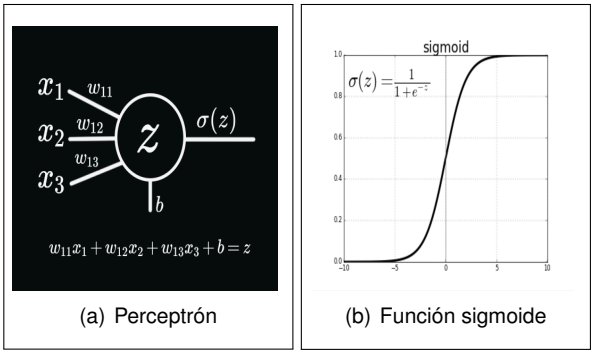
1. INTRODUCCIÓN

El Ministerio de Transportes y Comunicaciones del Perú desea conocer cuántos usuarios utilizan el tren eléctrico. Como parte de un proyecto gubernamental que promueva la ciencia y tecnología del país, el Estado peruano ha decidido trabajar con estudiantes de la Facultad de Ciencias de la UNI, y le ha asignado a un grupo la tarea de elaborar un módelo que permita identificar si hay un rostro humano en determinada imagen.

2. CONCEPTOS PREVIOS

Perceptrón

Los perceptrones son una herramienta matemática que nos servirá para crear un módelo. El perceptrón toma un conjunto de entradas numéricas bajo ciertos parámetros y nos brinda una salida. Considere una neurona con únicamente tres conexiones como vemos en la figura 1(a). En este caso, las entradas son x_1, x_2, x_3; las conexiones tienen los pesos w_{11}, w_{12}, w_{13}; hay un valor b adicional, denominado sesgo; y la suma ponderada z. Sobre este valor se aplica una función que devolverá un valor entre 1 y 0, a este tipo de funciones se les denomina función de activación. En nuestro módelo, como queremos que la función clasifique imágenes entre las etiquetas humano y no humano, nuestra salida podría ser un perceptrón. Por ejemplo, se puede utilizar la función sigmoide(ver figura 1(b)) y variar los parámetros $w_{11}, w_{12}, w_{13}, b$.

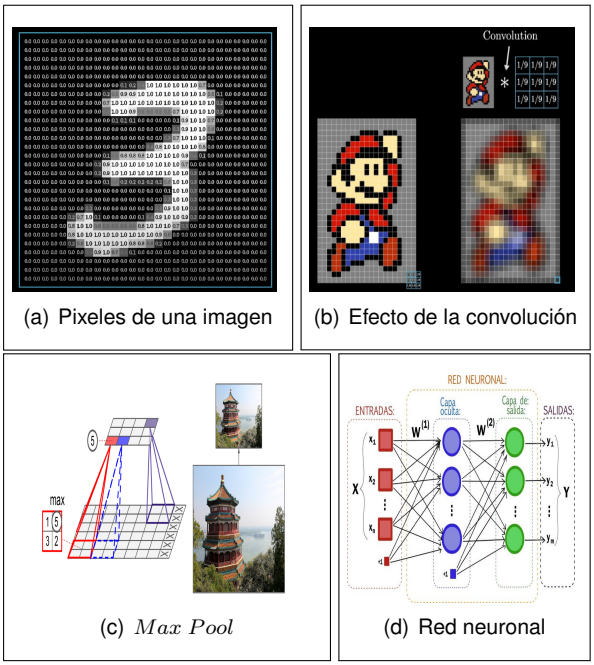


(a) Perceptrón (b) Función sigmoide

Figura 1: *Representación de una imagen en pixeles*

Convolución y redes neuronales

El estudio que se realiza sobre una determinada imagen utiliza como funte de informacion los pixeles de la misma. En nuestro caso, solo usaremos imágenes en blanco y negro; y los pixeles de estas imagenes cuentan con un valor entre 0 y 1, el cual indica la intensidad del blanco, ver figura 2(a). Luego, necesitamos extraer la mayor cantidad de informacion de la imagen disminuyendo el ruido y la cantidad de datos, este proceso se realiza bajo las conocidas convoluciones. Un tipo de convolución aplica una suma ponderada, con pesos arbitrarios, sobre cada pixel de la imagen, un ejemplo se puede visualizar en figura 2(b). Este nuevo paquete de información tambien se encuentra representado por números entre 0 y 1, un vector.



(a) Pixeles de una imagen (b) Efecto de la convolución

(c) $Max\ Pool$ (d) Red neuronal

Figura 2: *Representación de una imagen en pixeles*

Funciones de error

Una función de error o tambien denominada función de error, en el contexto de redes neuronales, brinda una estimación de la diferencia entre los valores reales de cada neurona(el valor obtenido luego de las convoluciones de cada imagen) y los valores obtenidos por el módelo. Supongamos que $\hat{y_i}$ es la salida de la neurona mediante cierto modelo y y_i el valor real, una de la funciones de error mas usadas, y la que usaremos, es la conocida función *cross entropy*, la cual matemáticamente se expresa mediante

$$C = -\sum_{i=1}^{N} y_i ln(\hat{y_i}) \tag{1}$$

Para nuestro modelo, los valores reales solo pueden ser dos valores: 1 para imágenes que contienen un rostro humano y 0 para imágenes que no contienen un rostro humano. usando la función de entropia cruzada se tiene

$$C = -yln(\hat{y_i}) - (1-y)ln(1-\hat{y}) \tag{2}$$

Veamos un ejemplo, considere tres imágenes y los respectivos parámetros dados en la Figure 3.

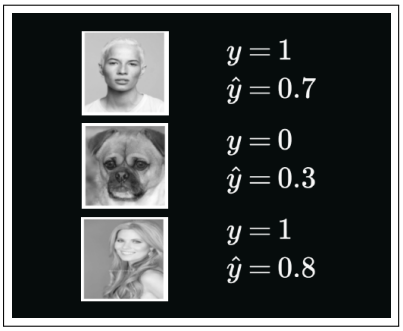


Figura 3: *Rostros clasificados*

Usando la Equation 2 obtenemos el siguiente costo promedio

$$C = -\frac{1}{3}[(1)ln(0.3) + (1-0)ln(1-0.3) + (1)ln(0.8)] \tag{3}$$

donde se divide entre 3 porque se tienen tres valores de entrada.

Descenso de gradiente

En todo módelo se busca minimizar la distancia entre el valor real y el valor obtenido mediante el módelo; con lo cual, se hace evidente la necesidad de métodos para minimizar esta distancia. Uno de estos métodos es el conocido descenso de gradiente, este consiste en ir actualizando el valor que queremos minimizar con un valor proporcional a $\frac{df}{dx}$ de la siguiente manera:

$$x_{n+1} = x_n - \alpha \frac{df}{dx} \tag{4}$$

donde α es un parámetro arbitrario. Por ejemplo, para la parábola $f = x^2$ nos movemos como se muestra en la Figure 4 alcanzando un mínimo bajo un número N de iteraciones.

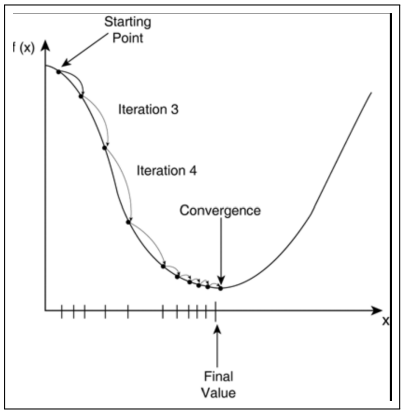


Figura 4: *Descenso de la gradiente*

En nuestro caso, el descenso de la gradiente se aplicará a una variable vectorial para minimizar la

función de costos; es decir, buscaremos actualizar los valores de los parámetros de la neurona para que nuestro valor de salida sea mas eficiente.

3. ANÁLISIS

Planteamiento del problema

Se busca hallar los parámetros del módelo, los pesos y los sesgos, para los cuales los resultados del módelo se acerquen más al valor real. Para esto, se brinda una imagen previamente calificada, con valor 0 o 1, y compararemos este valor con el obtenido mediante el módelo. Esto se verá reflejado en nuestra función de error, el cual se buscará minimizar mediante el método del descenso de gradiente. Un esquema de lo acotado se aprecia en la Figure 5.

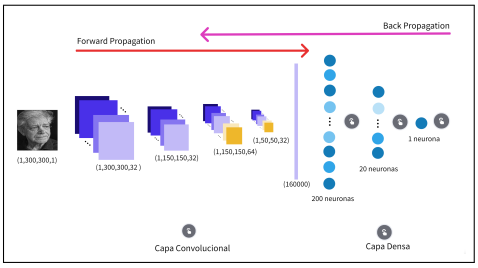


Figura 5: *Clasificación de imágenes*

Módelo matemático

Luego de aplicar las convoluciones a la imagen, obtenemos un vector de tamaño n_0

$$x^{(0)} = \begin{bmatrix} x_1^{(0)} \\ \vdots \\ x_{n_0}^{(0)} \end{bmatrix} \tag{5}$$

Aplicando una función afín sobre tal vector nos da

$$z^{(1)} = W^{(1)}x^{(0)} + b^{(1)} \tag{6}$$

donde $W^{(1)}$ es una matriz de dimensión $n_1 \times n_0$ que respresenta los pesos de cada conexión y $b^{(1)}$ es una matriz columna de n_1 columnas. Sobre este último vector se aplican las funciones de activación, la cual esta asignada a cada capa del módelo. En nuestro caso, usamos la función $ReLU$ que tiene la forma $\Gamma(\nu) = max(0, \nu)$; sin embargo, está función no es derivable en 0 y por lo cual usamos la siguiente aproximación

$$\Gamma(x) \approx f(x) = ln(1 + e^x) \tag{7}$$

cuya derivada es

$$f'(x) = \frac{1}{1 + e^{-x}} \tag{8}$$

La función $ReLU$ es usada en todas las capas exceptuando la última donde usamos una sigmoide, con la forma

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{9}$$

cuya derivada es

$$\sigma'(x) = \frac{e^{-x}}{(1 + e^{-x})^2} = \sigma(x)(1 - \sigma(x)) \tag{10}$$

Juntando lo mostrado obtenemos una sola función que es composición de $ReLU$s y al final una sigmoide, matemáticamente

$$\hat{y} = \sigma \circ z^{(L)} \circ f \circ z^{(L-1)} \circ ... \circ f \circ z^{(1)}(x^{(0)}) \tag{11}$$

El número de capas dependerá de cuánto se desee minimizar el error de las predicciones y del poder de cómputo. El error, o también llamado costo, lo obtendremos mediante la entropia cruzada dada por la Equation 2. El objetivo entonces será el de encontrar los pesos $W^{(i)}$ y los sesgos $b^{(i)}$ de cada neurona correspondiente a cada capa i que minimizan el error. Para ello, usaremos el método del descenso de gradiente, en nuestro contexto toma la forma

$$\begin{aligned} W_{n+1}^{(L)} &= W_n^{(L)} - \alpha \frac{\partial C}{\partial W^{(L)}} \\ b_{n+1}^{(L)} &= b_n^{(L)} - \alpha \frac{\partial C}{\partial b^{(L)}} \end{aligned} \tag{12}$$

donde α es un parámetro de entrenamiento arbitrario que será el paso con el que se actualizarán los pesos. En el presente proyecto se utiliza una red neuronal con tres capas y los valores $n_0 = 200$, $n_1 = 20$, $n_2 = 1$, entonces $\hat{y}$ toma la forma

$$\hat{y} = \sigma \circ z^{(3)} \circ f \circ z^{(2)} \circ f \circ z^{(1)}(x^{(0)}) \tag{13}$$

y las derivadas en la Equation 12 se obtienen bajo seguidas aplicaciones de la regla de la cadena. Empezando en la última capa y prosiguiendo hacia atrás (backpropagation) se tiene para la última capa, $L = 3$

$$\begin{aligned} \frac{\partial C}{\partial W^{(3)}} &= \frac{\partial C}{\partial \hat{y}} \frac{\partial \hat{y}}{\partial z^{(3)}} \frac{\partial z^{(3)}}{\partial W^{(3)}} \\ &= \left(\frac{1 - y}{1 - \hat{y}} - \frac{y}{\hat{y}} \right) \sigma'(z^{(3)}) x^{(2)} \\ &= \left(\frac{1 - y}{1 - \hat{y}} - \frac{y}{\hat{y}} \right) \sigma(z^{(3)})(1 - \sigma(z^{(3)})) x^{(2)} \\ &= \delta^3 x^{(2)} \end{aligned}$$

$$\begin{aligned}\frac{\partial C}{\partial b^{(3)}} &= \frac{\partial C}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial z^{(3)}}\frac{\partial z^{(3)}}{\partial b^{(3)}} \\ &= \left(\frac{1-y}{1-\hat{y}} - \frac{y}{\hat{y}}\right)\sigma'(z^{(3)})(1) \\ &= \left(\frac{1-y}{1-\hat{y}} - \frac{y}{\hat{y}}\right)\sigma(z^{(3)})(1-\sigma(z^{(3)})) \\ &= \delta^3\end{aligned}$$

donde δ^3 es una variable usual y por lo cual merece una notación propia. Similarmente, para la segunda se tiene

$$\begin{aligned}\frac{\partial C}{\partial W^{(2)}} &= \frac{\partial C}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial z^{(3)}}\frac{\partial z^{(3)}}{\partial x^{(2)}}\frac{\partial x^{(2)}}{\partial z^{(2)}}\frac{\partial z^{(2)}}{\partial W^{(3)}} \\ &= \delta^3\delta^2 x^{(1)} \\ \frac{\partial C}{\partial b^{(2)}} &= \frac{\partial C}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial z^{(3)}}\frac{\partial z^{(3)}}{\partial x^{(2)}}\frac{\partial x^{(2)}}{\partial z^{(2)}}\frac{\partial z^{(2)}}{\partial b^{(2)}} \\ &= \delta^3\delta^2\end{aligned}$$

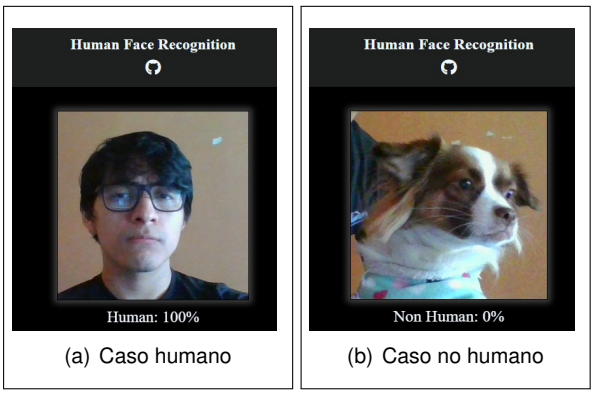
con $\delta^2 = W^{(3)}\frac{1}{1+e^{-z^{(2)}}}$ y para la primera capa

$$\begin{aligned}\frac{\partial C}{\partial W^{(1)}} &= \frac{\partial C}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial z^{(3)}}\frac{\partial z^{(3)}}{\partial x^{(2)}}\frac{\partial x^{(2)}}{\partial z^{(2)}}\frac{\partial z^{(2)}}{\partial x^{(1)}}\frac{\partial x^{(1)}}{\partial z^{(1)}}\frac{\partial z^{(1)}}{\partial W^{(1)}} \\ &= \delta^3\delta^2\delta^1 x^0 \\ \frac{\partial C}{\partial b^{(1)}} &= \frac{\partial C}{\partial \hat{y}}\frac{\partial \hat{y}}{\partial z^{(3)}}\frac{\partial z^{(3)}}{\partial x^{(2)}}\frac{\partial x^{(2)}}{\partial z^{(2)}}\frac{\partial z^{(2)}}{\partial x^{(1)}}\frac{\partial x^{(1)}}{\partial z^{(1)}}\frac{\partial z^{(1)}}{\partial b^{(1)}} \\ &= \delta^3\delta^2\delta^1\end{aligned}$$

con $\delta^1 = W^{(2)}\frac{1}{1+e^{-z^{(1)}}}$. Por último, reemplazamos siguiendo la Equation 12 iterativamente.

Resultados

EL módelo se emsambló a una página web($https://n9.cl/red_neuronal$), y se obtuvieron los resultados mostrados en la Figure 6



(a) Caso humano (b) Caso no humano

Figura 6: *Resultados del módelo*

4. OBSERVACIONES

Detalles técnicos

Se usaron 2 capas convolucionales para la extracción de características de las imágenes, con 32 y 64 filtros respectivamente; y reducción de dimensión por un factor de 2 y 3; es decir, capas de maxpooling con nucleos de 2×2 y 3×3. Seguidamente, se redimensionó el tensor a un vector para conectar las capas densas u ocultas, con 200 y 20 neuronas respectivamente. Finalmente, para la salida se usó una neurona con una función sigmoide limitando el resultado entre 0 y 1. Para el primer experimento se usó dos conjuntos de imágenes de 7219 cada una, etiquetadas con 1 para las que contienen sólo rostros humanos y 0 para cualquier imágen que no sea rostro humano. Luego del entrenamiento se logŕo un resultado del 97% en 14 épocas(iteraciones). En la Figure 7 se muestra el comportamiento del *accuracy* y la evolución de la función de pérdida a lo largo del entrenamiento.

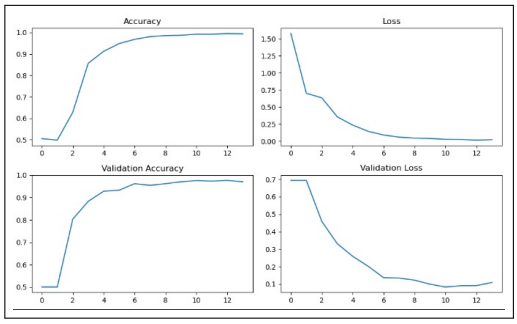


Figura 7: *Costo y exactitud del primer ensayo*

Problemas técnicos

- No existe una regla estricta para determinar la cantidad exacta de capas que debe tener un módelo, para esto la experimentación es imperativa. Se puede comenzar con una arquitectura simple y luego realizar experimentos iterativos agregando o eliminando capas para ver cómo afecta al rendimiento del módelo.

- Para entrenar el módelo, los datos se dividen en 3: datos de entrenamiento, lo utilizamos para actualizar los parámetros (pesos y sesgos); datos de validación, nos permite ajustar los hiperparámetros (como la tasa de aprendizaje, el tamaño del lote, el número de épocas, etc); datos de evaluación, una vez entrenado el modelo, usamos este conjunto de datos para evaluar el rendimiento de este.

Mejoramiento del módelo

Como se observa, el modelo matemático nos permite clasificar un rostro humano utilizando el descenso de gradiente; sin embargo, el módelo no es del todo preciso y se pude mejorar tomando las siguietes consideraciones

- Aumentar y diversificar el conjunto de datos para que el modelo sea más preciso y más general. Es decir, pueda reconocer rostros de seres humanos de diferentes edades en distintas condiciones.
- Variar el número de capas y/o filtros de redes neuronales para verificar si se puede llegar a una mayor exactitud. Tener en cuenta que a mayor cantidad de capas, mayor el costo computacional.
- Probar con diferentes tasas α y verificar si se acerca más al mínimo de costo y si se necesita más o menos iteraciones.
- Probar otros métodos para minimizar una función aparte del descenso de gradiente. Por ejemplo, existen el método Adam(Adaptive Moment Estimation), RMSprop, Descenso de gradiente estocástico, etc.

4. CONCLUSIONES

El módelo presentado es uno de los más básicos; no obstante, mostró gran eficacia en diferentes situaciones. Esto nos lleva a pensar que este campo de la ciencia tiene mucho por delante y para un completo entendimiento del mismo es necesario comprender temas como las funciones reales de variable vectorial y los métodos de optimización. Esto, además, corrobora la gran importancia de los temas de cálculo avanzado.

Referencias

[1] Murphy, K. (2012). *Machine Learning: A probabilistic perspective* (1st Edition)

[2] Barber, D. (2010). *Bayesian Reasoning and Machine Learning* (1st Edition)