**Universidad Autónoma de Occidente**

**Redes neuronales artificiales y deep learning**

**Ana Maria Pinto 2140133**

**Miguel Ángel Saavedra 2140156**

**Ingeniería Mecatrónica**

**Problemas de clasificación**

1. **Reconocimiento de letras usando una red multicapa superficial**

Para el desarrollo de esta red neuronal se trabaja con el nombre “Angel”, se hace su respectiva conversión a una matriz 7x5 por cada letra del nombre. Posteriormente, se convierte cada matriz a un vector columna de 35 posiciones, que a su vez se convierten en 35 características de entrada para la red neuronal y así hacer posible que la red lea cada carácter contenido en la cuadrícula original 7x5 (Figura 1). Con ayuda de Python y la librería Matplotlib se decide graficarlas letras de entrada a la red, los resultados arrojados fueron los siguientes (Figura 2).

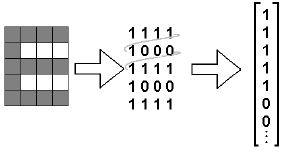


Figura 1. Conversión de matriz 7x5 a vector de 35 posiciones.

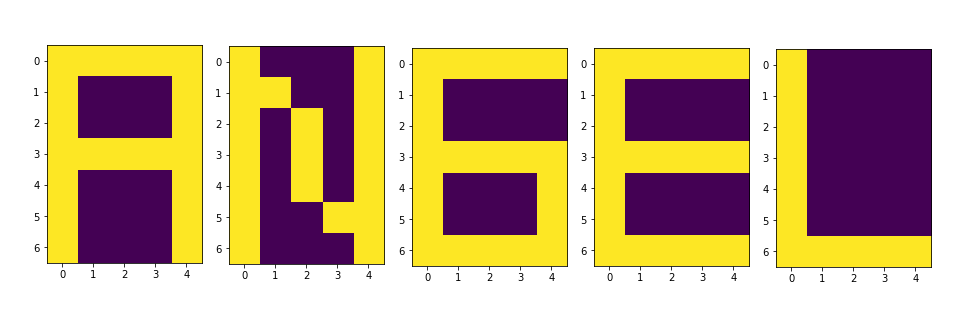


Figura 2.Gráfica de los vectores de las letras usadas.

Una vez obtenidos los datos de entrada, se construye la arquitectura de la red neuronal superficial con 35 neuronas en la capa de entrada, 20 nodos en la capa oculta con función de activación tanh y 5 neuronas de salida con función de activación sigmoidal, obteniendo en esta última capa una salida codificada por cada letra. En el caso de que la Salida deseada sea una A, el vector de salida correspondiente será [1 0 0 0 0].

Se deciden utilizar función de activación tanh para generar una mayor dispersión de los datos en la capa oculta y a la salida sigmoidal debido a que los datos de salida solo estarán comprendidos entre 0 y 1. Por otra parte, se utiliza el optimizador Adagrad para realizar el entrenamiento de la red, con un Alfa de 0.1 y con un valor inicial de acumulador de 0.001. Junto a esto se entrena la red con un total de 20000 épocas, generando en variadas pruebas los mejores resultados por parte de la red con una pérdida de 4.0888062e-06. Las matrices de confusión generadas en la fase de entrenamiento son las siguientes.

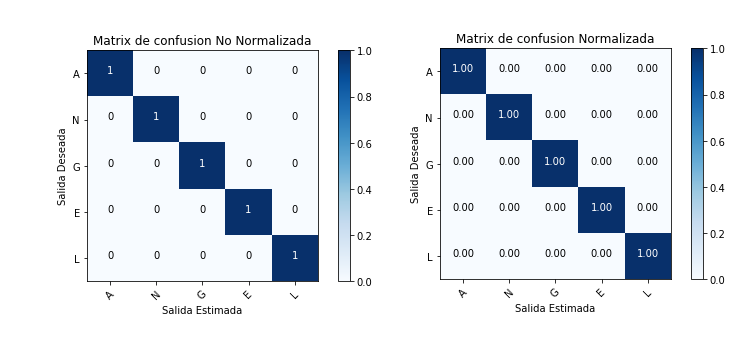


Figura 3. Matrices de confusión de la fase de entrenamiento.

En la fase de validación se genera un nuevo conjunto de datos similar al de entrenamiento, sin embargo estos cuentan con algo de ruido, como se muestra a continuación.

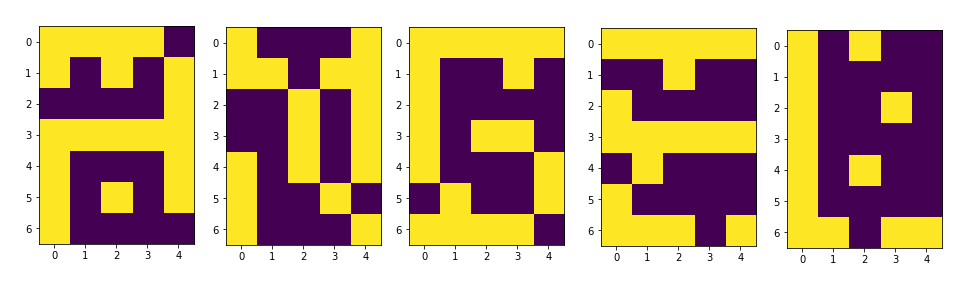


Figura 4. Datos usados para la validación.

Finalmente se probó que la red opera de manera adecuada, arrojando un loss de 0,0012194876 y una precisión de 1.0. A continuación se muestran las matrices de confusión del proceso de validación.

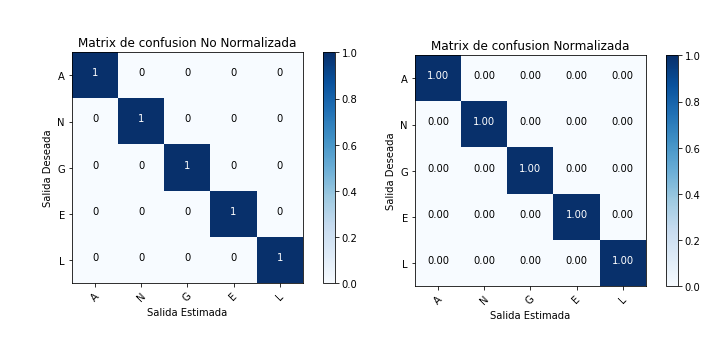


Figura 5. Matrices de confusión del proceso de validación.

1. **Problema de clasificación de vinos**

Para resolver el problema de clasificación de vinos se proponen dos soluciones una red superficial y una red profunda. Inicialmente, se realiza la lecura y codificación de los datos de entrada organizando las 13 características y la salida en un vector de tres posiciones, una por cada tipo de vino (clase A, B y C). Cabe destacar, que el Dataset cuenta con un total de 178 datos en total, de los cuales se deciden utilizar 125 para el entrenamiento y 53 para la validación de la red.

Posteriormente, se construye la arquitectura de la red donde se decide utilizar 13 neuronas en la capa de entrada (Una por cada característica de los vinos), 20 nodos en la capa oculta con función de activación sigmoidal y 3 nodos en la capa oculta con función de activación sigmoidal. Se decide utilizar el optimizar RMSprop y un total de 10000 épocas para realizar el entrenamiento de la red. Realizando el entrenamiento de la red, se obtiene un loss de 0.010555179 y las siguientes matrices de confusión.

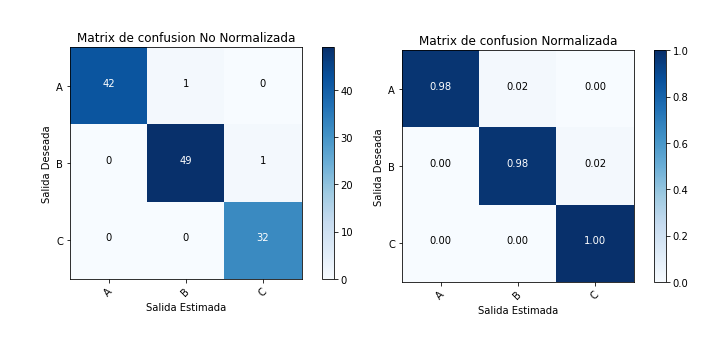


Figura 6.Matrices de confusión del proceso de entrenamiento, red superficial.

Para la etapa de validación, se corre la red previamente entrenada y se obtiene un loss de 0.059798617 junto a un accuracy de 0.9056604. Las matrices de confusión generadas son las siguientes.

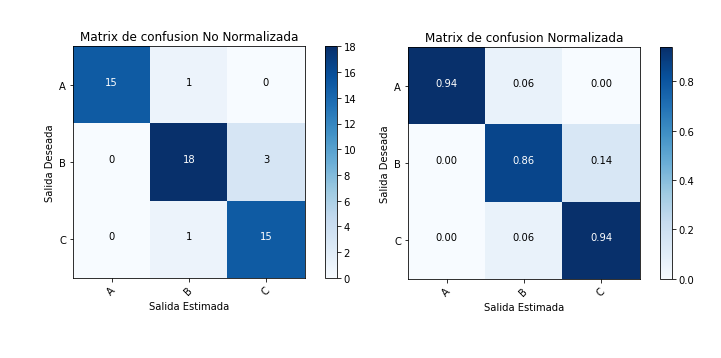


Figura 7. Matrices de confusión del proceso de validación, red superficial.

Aunque los resultados son satisfactorios, la red tiene un comportamiento atípico debido que a la hora de utilizar otros optimizadores la red queda entrenada de manera errónea, arrojando todos los vinos a una única clase como se muestra a continuación.

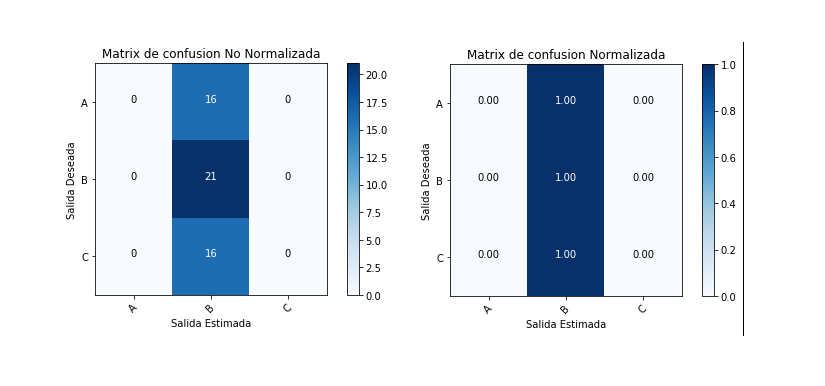


Figura 8. Matrices de confusión con resultados erróneos.

Es por estos comportamientos que finalmente se usó el optimizador RMSprop con un número de 10000 épocas, ya que al aumentarlo mucho también solía generar sobre entrenamiento en la red y un mal comportamiento en la etapa de validación.

Para la red profunda, se decide utilizar el mismo número de nodos en la capa de entrada que la red superficial con la diferencia que ahora se adicionan 3 capas ocultas más, donde las 2 primeras cuentan con 15 nodos ocultos, la tercera con 10 y la última con 3 nodos para codificar la salida. Dicha red se entrena con el optimizador RMSprop con un número de 15000 épocas, ya que, después de varias pruebas se comprobó que el mejor resultado se obtenía con este número de épocas y este optimizador.

Por otra parte, se decide utilizar función de activación RELU en todas las capas ocultas y función Softmax a la salida de la red neuronal, arrojando estas los mejores resultados ya que, a la hora de cambiar estas funciones por tanh o sigmoid, el resultado no fue mejor que el obtenido con las funciones de activación nombradas inicialmente. El loss obtenido en la fase de entrenamiento fue de 0.0055100564, arrojando los siguientes resultados en las matrices de confusión.

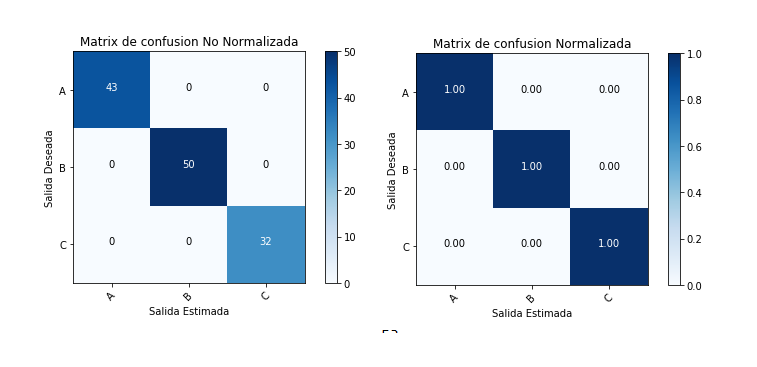


Figura 9. Matrices de confusión del proceso de entrenamiento, red profunda.

Asimismo, la fase de validación generó un loss de 0.024863956 y un accuracy de 0.9811321, arrojando las siguientes matrices de confusión.

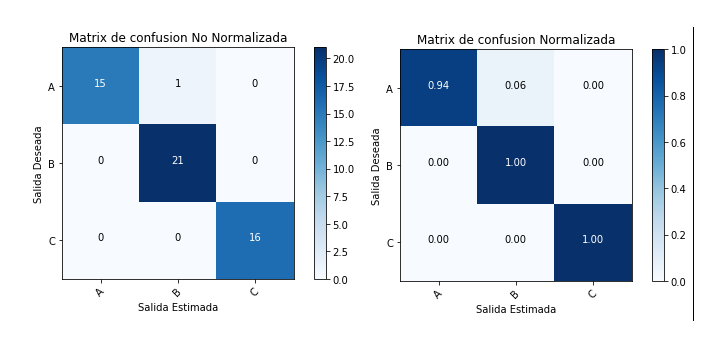


Figura 10. Matrices de confusión del proceso de validación, red profunda.

De todo lo anterior se puede concluir que el mejor desempeño se logró en la red profunda, evadiendo el comportamiento atípico previamente nombrado en la red superficial y siempre consiguiendo una exactitud mayor al 96%.

1. **Problema de clasificación de carros**

En este inciso, se decide trabajar el problema de clasificación de los automóviles, donde se cuenta con un total de 6 características de entrada para decir si el carro pertenece a 4 posibles categorías ('Unacc', 'Acc', 'Good' o 'V-Good'). Teniendo esto en mente, se resuelve el problema con una red superficial y otra red profunda con las mismas arquitecturas del inciso anterior, con la diferencia que ahora el número de neuronas en la capa de entrada son 6 y se necesitan 4 neuronas en la capa de salida para codificar de manera adecuada la salida de la red. Asimismo, como el Dataset cuenta con un total de 1728 datos en total, se utilizan 1210 datos para el entrenamiento y 518 para su validación.

Para la red superficial, se varía el número de nodos en la capa oculta a 15 con una función de activación sigmoidal y 4 nodos en la capa de salida con función de activación sigmoidal debido a que la salida estará comprendida entre 0 y 1. Cabe resaltar que se decide utilizar el mismo optimizador y número de épocas que en la capa superficial del punto anterior, arrojando un loss de 0.0074087693 con las siguientes matrices de confusión.

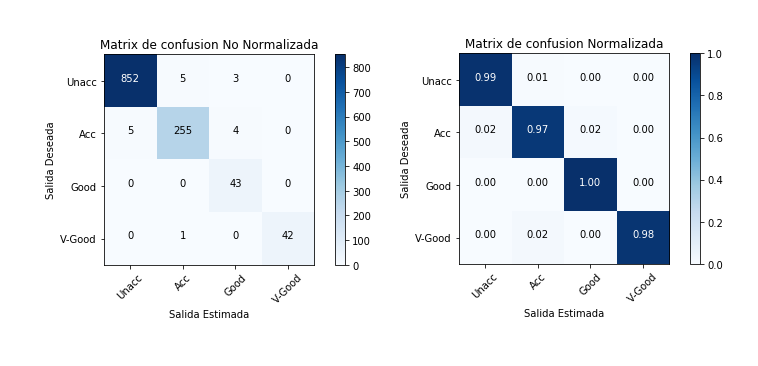


Figura 11. Matrices de confusión del proceso de entrenamiento, red superficial.

En la etapa de validación se obtuvo un loss de 0.010387761 y un accuracy de 0.976834, junto a las siguientes matrices de confusión.

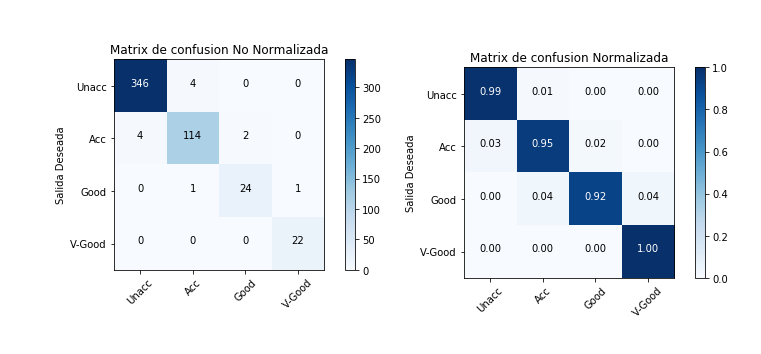


Figura 12. Matrices de confusión del proceso de validación, red superficial.

Para la red profunda, se utiliza la misma arquitectura variando únicamente el número de nodos en la capa de entrada y de salida de la red, el optimizador, el número de épocas y funciones de activación. Arrojando un loss de 0.0011870537 y las siguientes matrices de confusión.

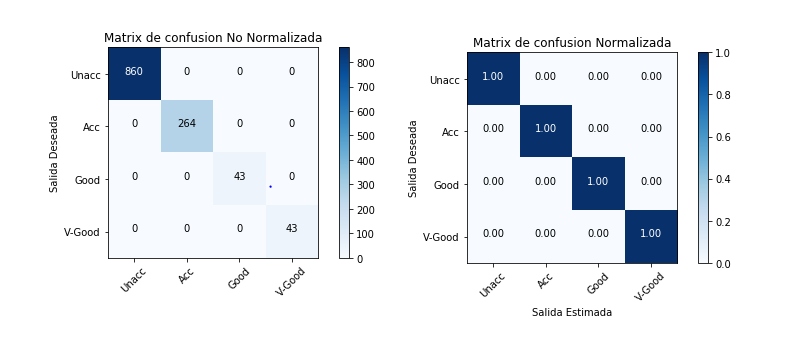


Figura 13. Matrices de confusión del proceso de entrenamiento, red profunda.

La fase de validación arroja un loss de 0.01833057 y un accuracy de 0.9555985 con las siguientes matrices de confusión.

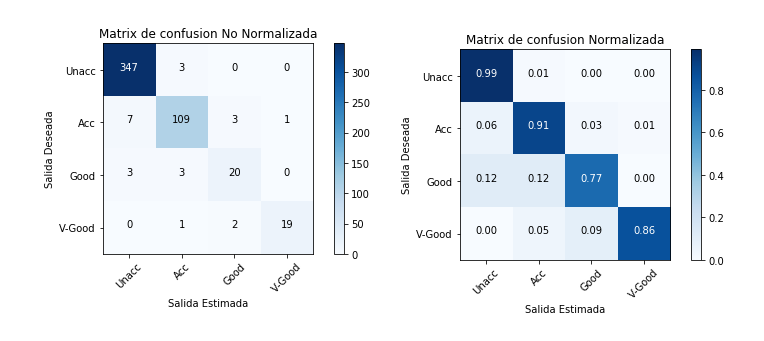


Figura 14. Matrices de confusión del proceso de validación, red profunda.

Lo anterior nos lleva a concluir, que en esta ocasión la mejor clasificación la realizó la red superficial obteniendo mejores valores de loss y accuracy en múltiples pruebas realizadas a pesar de variar el optimizador. Este comportamiento puede ser debido a que la red no cuenta con tanta dispersión en los datos de entrada como era el problema de los vinos, siendo solucionable con una arquitectura de red mucho más simple. No obstante, ambas redes realizar su trabajo de manera adecuada y obtiene una exactitud mayor al 95%.

1. **Red neuronal para detección temprana de enfermedades**

Mediante un dispositivo vestible se desea realizar un diagnóstico temprano de posibles enfermedades que sufra una persona. Las enfermedades que se han seleccionado para su diagnóstico son la diabetes, la presión alta, la presión baja y arritmias cardiacas. El dispositivo vestible permite tener información de la frecuencia cardiaca, la frecuencia respiratoria, la temperatura corporal, la concentración de azúcar y de oxígeno en la sangre.

Teniendo en cuenta la información necesaria para realizar el diagnostico se plantea el uso de una red neuronal con dos capas ocultas cada una con diez nodos y función de activación Relu, la capa de salida tendrá cuatro nodos (uno por cada enfermedad) y se usará como función de activación Softmax. Adicional a lo anterior, se utilizará un optimizador Ada o RMS siendo este el mejor en términos de resultados. El número de épocas del algoritmo variará según los datos que se tengan de entrada.

Es importante resaltar que las entradas a la red corresponden a la información obtenida del paciente y las salidas corresponden a las posibles enfermedades a diagnosticar. En la siguiente tabla se muestran cada una de las entradas y salidas del sistema.

|  |  |  |  |
| --- | --- | --- | --- |
| Entradas | | Salidas | |
|  | Frecuencia cardiaca |  | Diabetes |
|  | Frecuencia respiratoria |  | Presión alta |
|  | Temperatura corporal |  | Presión baja |
|  | Concentración de azúcar en sangre |  | Arritmias cardiacas |
|  | Concentración de oxígeno en sangre |  |  |

Tabla 1. Entradas y salidas del sistema

A continuación, se muestran el diagrama de bloques del sistema que será implementado (figura 15) y el esquema de la arquitectura de la red neuronal que planteada para este caso (figura 16). Finalmente, se plantea la codificación de las salidas para este problema en la tabla 2.

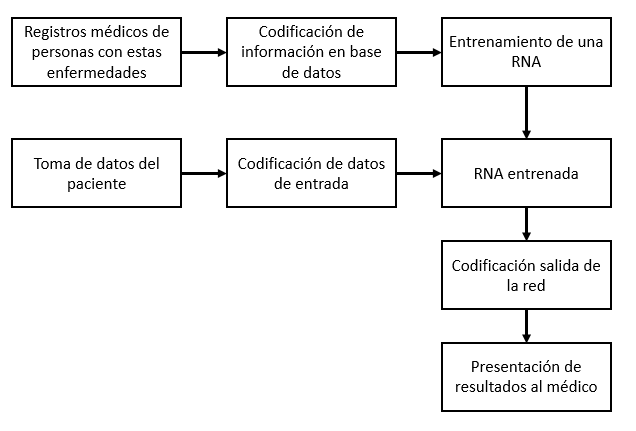


Figura 15. Diagrama de bloques del sistema.

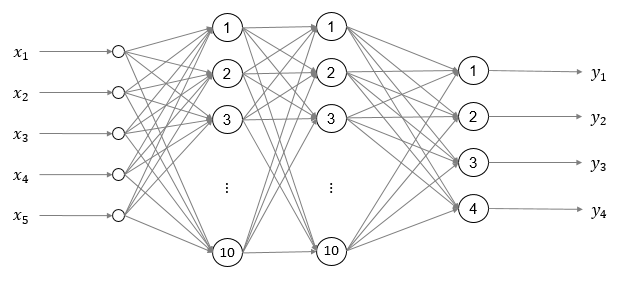
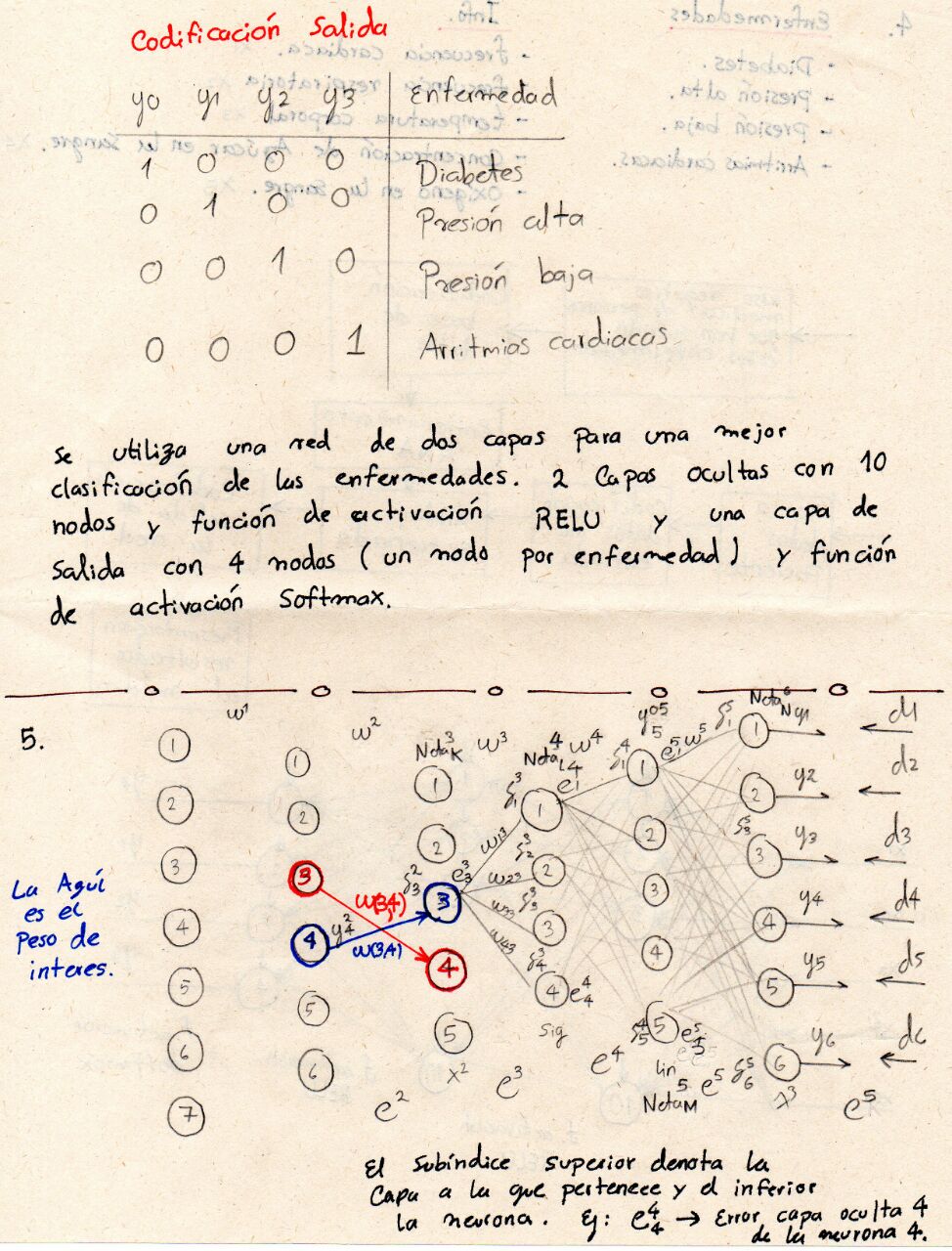


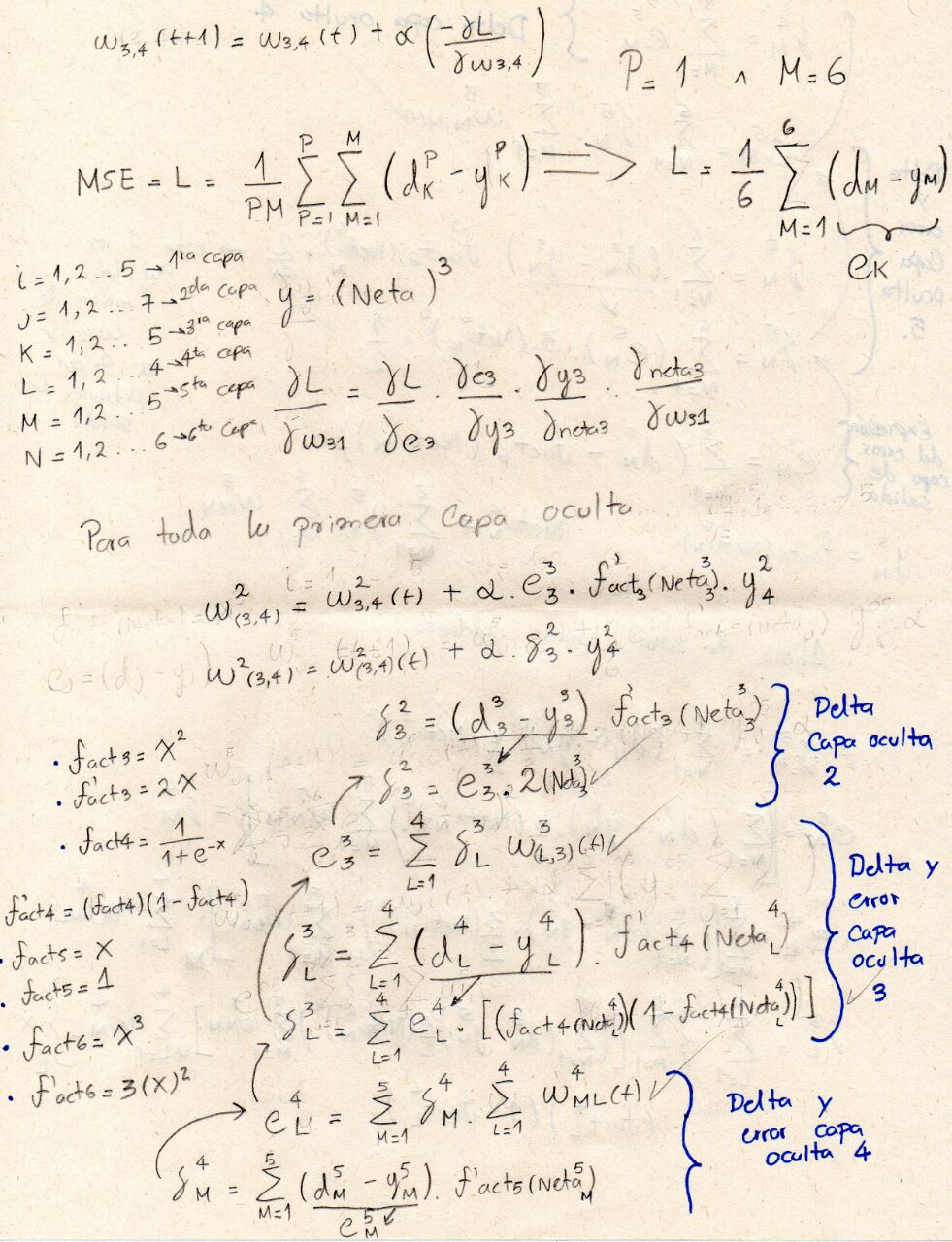
Figura 16. Esquema de la red neuronal.

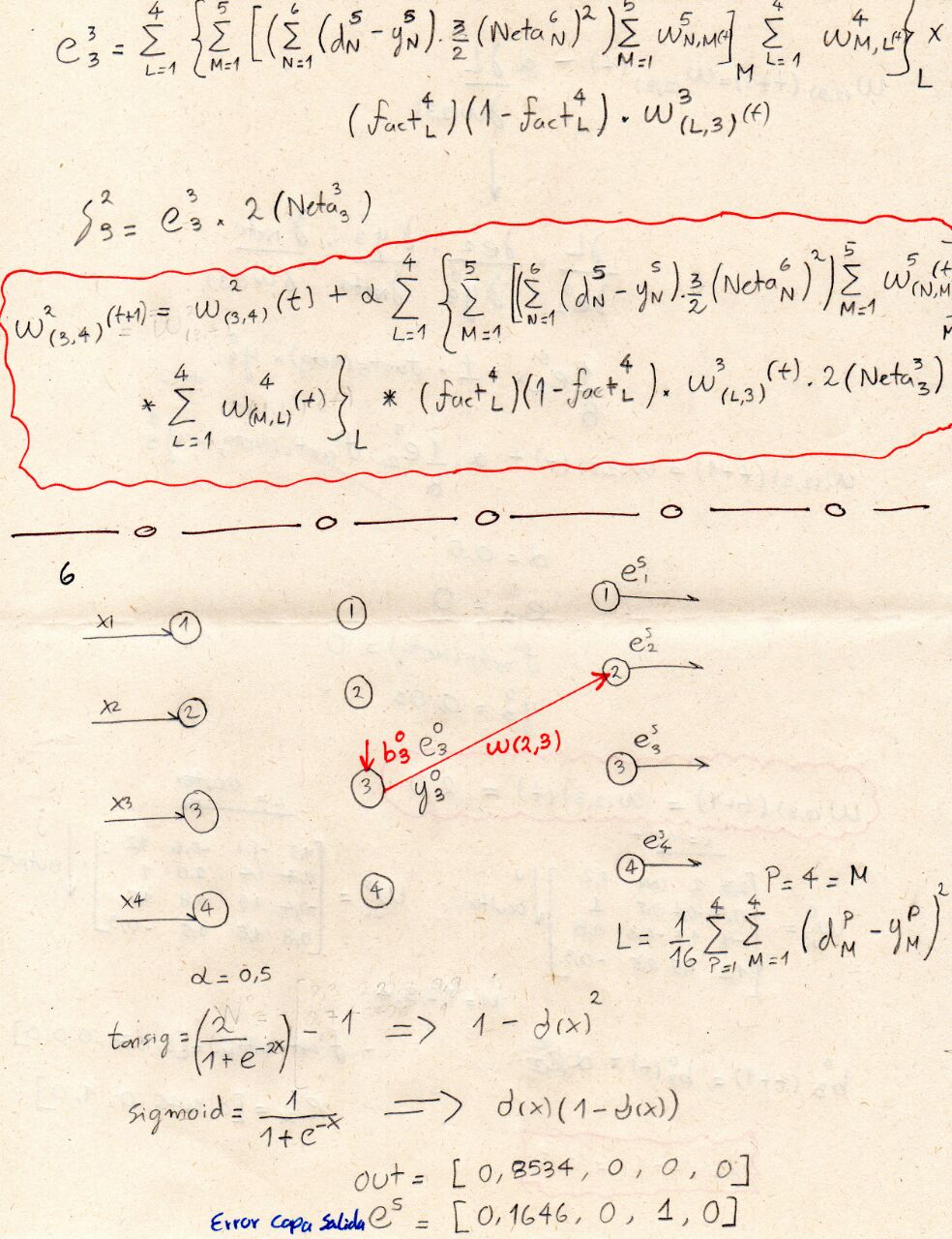
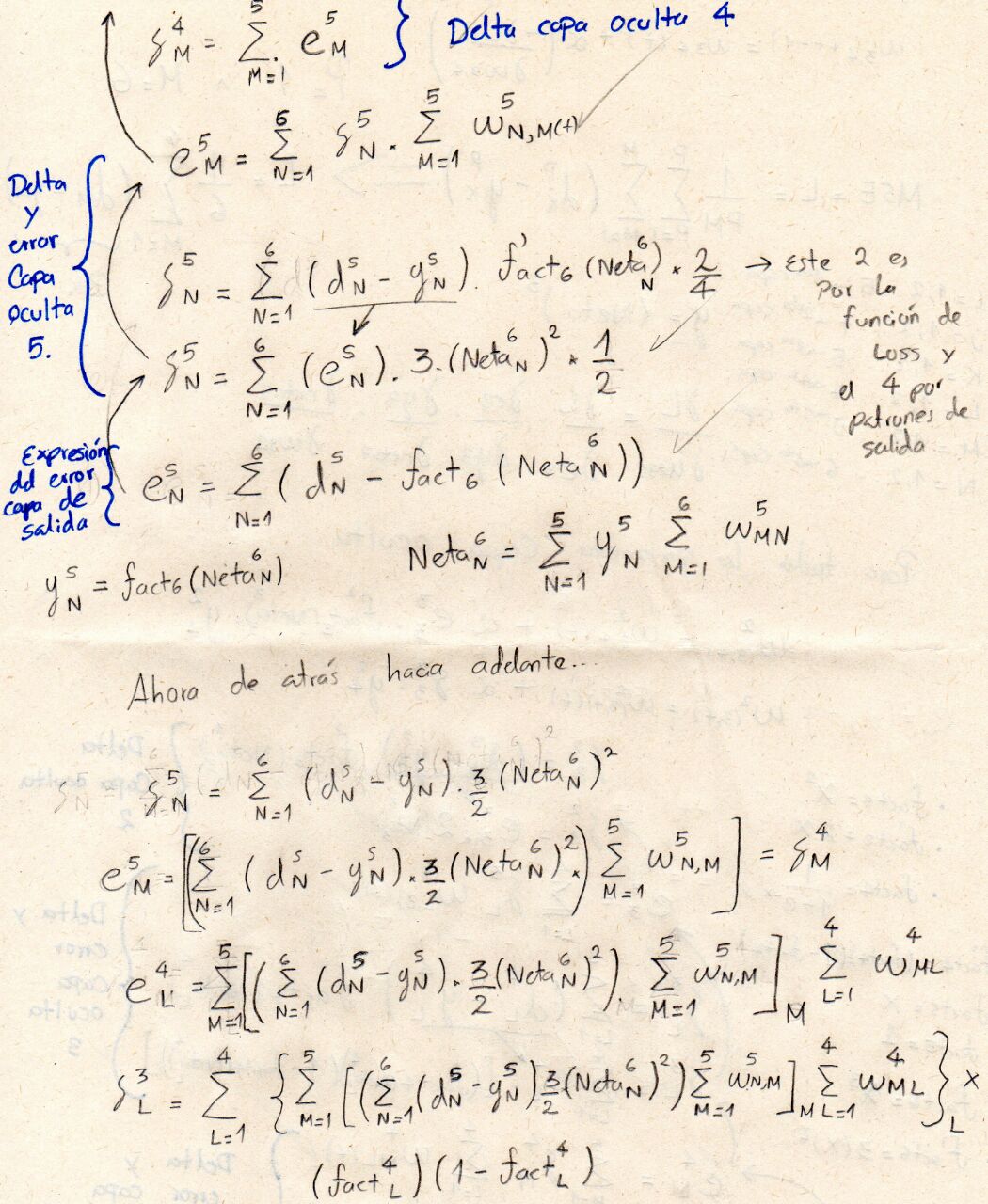
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  |  |  | **Enfermedades** |
|  |  |  |  | Diabetes |
|  |  |  |  | Presión alta |
|  |  |  |  | Presión baja |
|  |  |  |  | Arritmia cardiaca |
|  |  |  |  |  |

Tabla 2. Codificación de las salidas

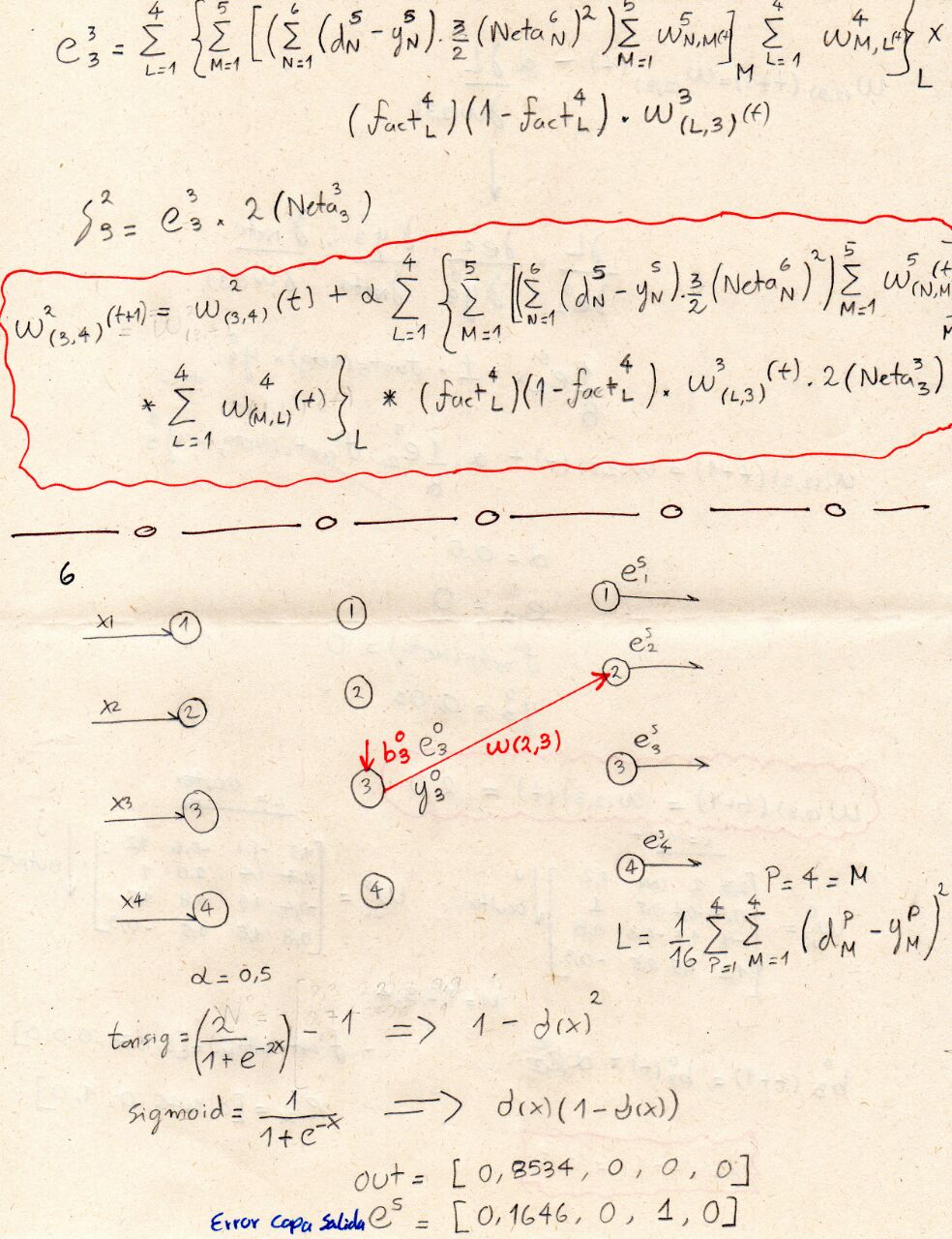
1. En este punto se halla la expresión para actualizar el valor del peso sináptico entre las neuronas 3 y 4 de la capa oculta 2, el proceso se muestra a continuación.

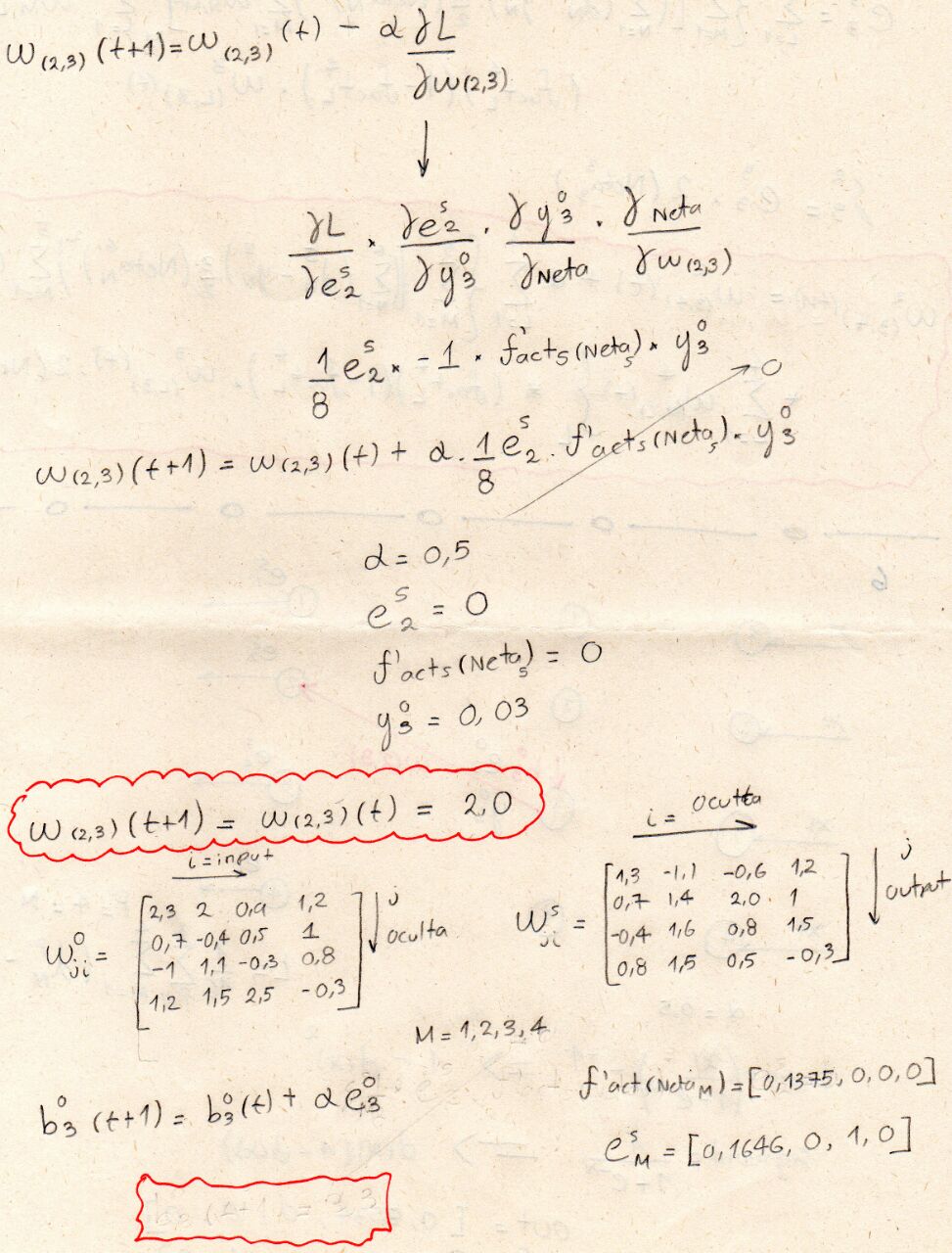


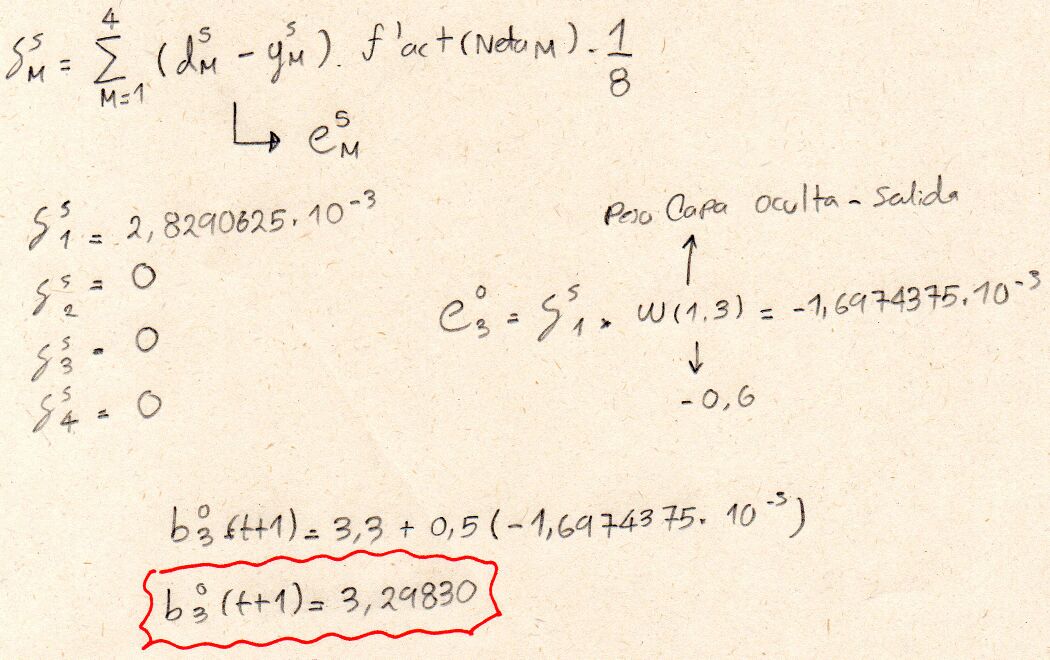




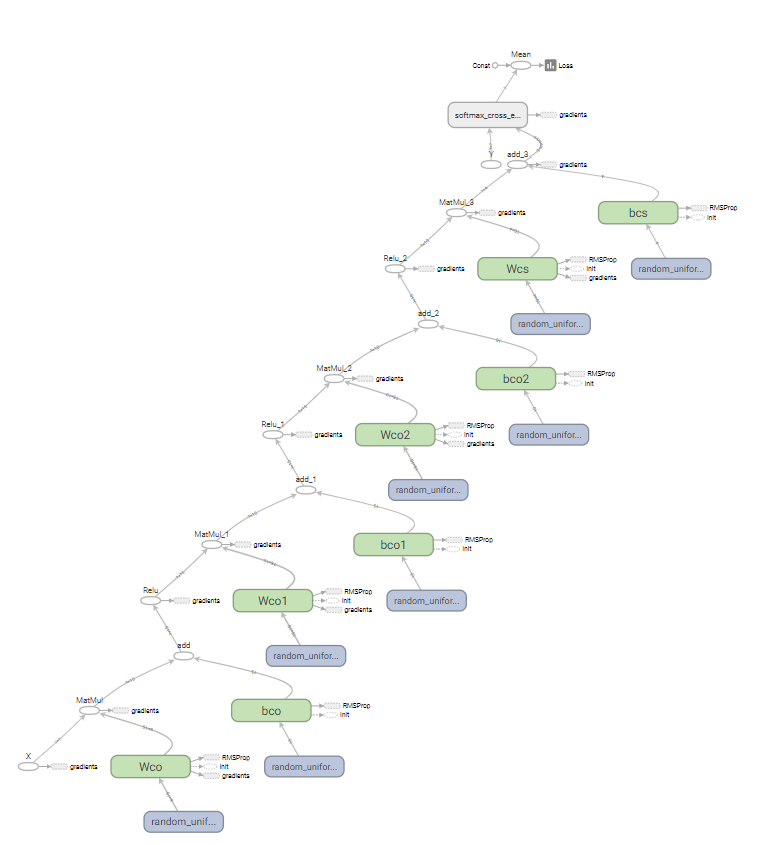
1. En este punto se halla la expresión para actualizar el valor del peso sináptico entre las neuronas 2 y 3 de la capa de salida junto al respectivo bias de la neurona 3 de la capa oculta, el proceso se muestra a continuación.



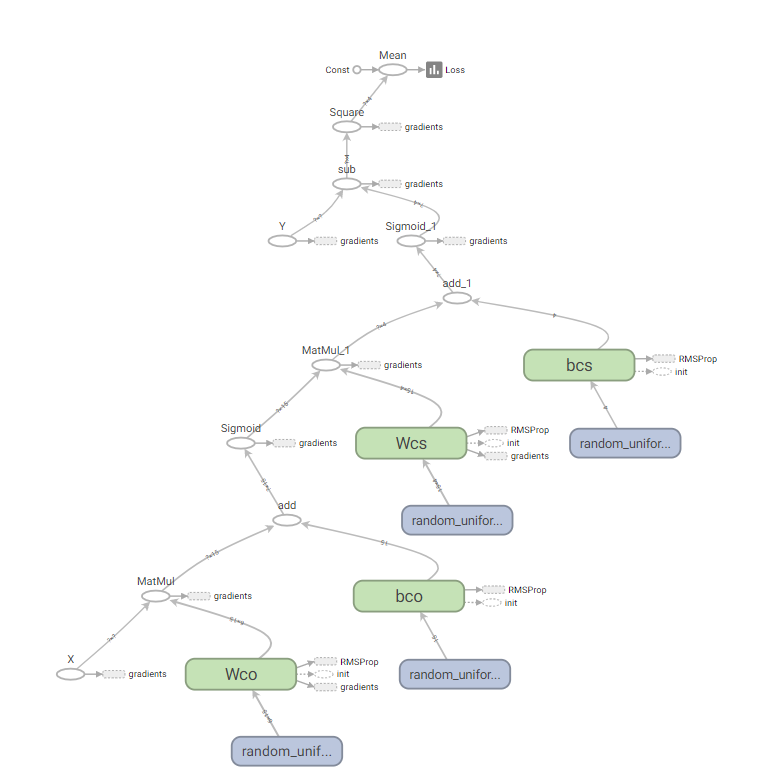




Anexos.



Anexo 1. Grafo de la red neuronal profunda utilizada para el punto 2 y 3



Anexo 2. Grafo de la red neuronal superficial utilizada en los puntos 1, 2 y 3.